Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

Assisted Road Remarking System

A thesis presented in partial fulfilment of the requirements for the degree of

Master of Engineering

in

Mechatronics

at Massey University, Turitea,
New Zealand

Mark Cameron

2012

Abstract

Roads are used in every part of the world and where there are roads, there are almost always road markings guiding the users of the roads in what they can and cannot do. The process of road marking involves first painting the lines then covering them with small glass beads. This process need to be repeated frequently, depending on the wear of the markings. The remarking of existing lines is complicated and difficult to become proficient in.

What is wanted is an assisted road remarking system that will help road markers in their remarking duties and combat some of the problems they face. Creating an assisted remarking system will make it easier to operate a truck. This means it will be easier to move from one truck to another and also it will be easier to train someone up to do the job, eliminating the long teaching period. It will potentially increase the accuracy of the markings as well. In addition it allows the operators to concentrate on the road ahead improving safety and also the speed that remarking can occur. All of these add up to a significant cost saving.

This project was split up into two major sections: sensing, and control and actuation. Research was carried out into existing technologies and processes that may be useful. An algorithm was developed using the radon transform to detect existing lane markings by analysing images acquired via a CCD camera. Also developed was a control system for the actuation of the spray nozzle, this was created by modelling the entire system.

Through testing the overall system can position an actuator over a line accurately and fast. It can detect a line in a variety of conditions and at various qualities. The actuator moves smoothly and quickly to where it needs to be without any sudden changes in direction. The system that was designed would be of great aid to a road marker and would improve the quality and speed that remarking can be carried out at. It would also allow faster training, improved safety, and cost savings.

Acknowledgements

I wish to thank my supervisor Dr Ibrahim Al-Bahadly for his help and assistance during this project. For guiding me through the process and all the valuable feedback he has given me during this time.

A special thank you to Ross Arnopp from Roadmarking Services Ltd. who gave me a valuable insight into road marking and all that it entails and shared the problems that they face.

Finally, I would like to thank my parents, Clive and Alison for their patience and encouragement while I completed this thesis and my grandmother Grace for all her proofreading skills.

Table of Contents

Abstract iii
Acknowledgements v
Table of Contents vii
List of Figures ix
List of Tables xiii
1 Introduction 1
1.1 Thesis Overview 6
2 Literature Review 7
2.1 Line Detection Methods 7
2.1.1 Sensors 7
2.1.2 Algorithms 9
2.2 Control Methods 15
3 Proposed System 17
3.1 Sensing 18
3.1.1 Sensor 18
3.1.2 Line Detection Method 20
3.2 Control 22
3.2.1 Hardware 22
3.2.2 Method 23
3.3 Actuation 24
3.3.1 Linear Actuator 24
3.3.2 Motor 27
3.4 Design Details 30
3.4.1 Sensors: 30
3.4.2 Control: 31
3.4.3 Actuation 31
4 Algorithm Development 35
4.1 Image Processing Algorithm 35

4.1.1 Pre-processing 35
4.1.2 Radon Transform 38
4.1.3 Algorithm Development 39
4.2 Conclusion 51
5 Control and Actuation Development 53
5.1 Actuation 53
5.2 Control 55
5.2.1 Model Creation 55
5.2.2 Fuzzy Logic 58
5.3 Conclusion 66
6 Results and Analysis 67
6.1 Image Processing 67
6.1.1 Results 67
6.1.2 Analysis 73
6.2 Control and Actuation 74
6.2.1 Results 74
6.2.2 Analysis 77
6.3 Overall System 77
6.4 Conclusion 80
7 Conclusion 81
7.1 Further Work 83
8 References 85
Appendix 1 – Main Algorithm 89
Appendix 2 – Subroutines 93
Appendix 3 – Testing Program 105

List of Figures

Figure 1.1 - Trolley mounted sprayer [1] 2
Figure 1.2 – Driving over a painted line 3
Figure 1.3 – Setup of existing road marking apparatus including spray nozzle, compressed air blower and glass bead gun. 3
Figure 1.4 – This shows the view out of the truck window with the Marker Pole (oval), and position of the spraying apparatus (rectangle). 4
Figure 2.1 – Vehicle and road model used in [4] for driver assistance. 7
Figure 2.2 – Four different images are merged together to form a single view of the road ahead. . 8
Figure 2.3 – View of a road before and after perspective mapping. 10
Figure 2.4 – Top hat filter for line detection [17]. 11
Figure 2.5 – Canny filter edge image with the correctly selected outputs based on the Hough transform. 12
Figure 2.6 – Geometry of the radon transform. 14
Figure 2.7 – PID control of a system. 15
Figure 3.1 – Sections of this project 17
Figure 3.2 – Typical electro mechanical actuator. 24
Figure 3.3 – Linear motion stage. 25
Figure 3.4 – Hydraulic actuator. 25
Figure 3.5 – Various belt driven stages. 26
Figure 4.1 – Image of a line along with the corresponding intensity histogram. 35
Figure 4.2 – Normal intensity histogram (left), Contrast expanded histogram of the same information (right). 36
Figure 4.3 – Custom lookup table function. 36
Figure 4.4 – This shows the original image and its histogram (left), the resulting image and histogram for the lookup table (centre), and the resulting image from the contrast expansion (right) 37
Figure 4.5 – Geometry of the radon transform [34]. 38
Figure 4.6 – The original image (top), the radon transform (left), and the radon transform at θ = 8°. 39
Figure 4.7– The radon transform at 13° along with the lines coresponding to the detected edges (top), and the angled line at which the radon transform took place along with the lines extrapolated from the points found in the radon transform (bottom). 40

Figure 4.8 – Normal and zoomed in views of the number of points above the threshold over each angle in the radon transform. 41
Figure 4.9 – Result from using the improved algorithm, angle found to be 12°. 42
Figure 4.10 – Plots of the smoothed row, the averages for each side and the derivation of the averages. The red lines correspond to the found edges of the line. 44
Figure 4.11 – Horizontal line with left (red) and right (green) averages and their threshold value plotted together. 45
Figure 4.12 – The algorithms progress up an image. (a) Moving up in large steps, (b) when no line is found take small steps backward, (c) once a line is found continue with the large steps until the top of the image. 47
Figure 4.13 – A line modelled by quadrilaterals with the edge regions of interest to either side marked out. 48
Figure 4.14 – Basic Algorithm Flowchart 49
Figure 5.1 – Example of a stepper motor torque-speed curve [35] 55
Figure 5.2 – Model of the mechanical system. 56
Figure 5.3 – Model of the computer control. 57
Figure 5.4 – Entire simple motor control model. 57
Figure 5.5 – Full motor control model. 58
Figure 5.6 – An inference table showing the subjective terms and what they correspond to relating to speed. Values can be in more than one section, for example if the speed is 75 km/h then it can be partly medium and partly fast. 59
Figure 5.7 – Membership functions. 60
Figure 5.8 – Results from initial Fuzzy Logic Controller. Top is the set point (purple) and actual position (yellow). Bottom is the frequency output of the fuzzy logic controller. 61
Figure 5.9 – Improved fuzzy logic controller with a smoothed output. 62
Figure 5.10 – Fuzzy logic controller with improved membership functions. 63
Figure 5.11 – Improved rate of change membership function. 63
Figure 5.12 – Improved error membership function with increased overlap. 64
Figure 5.13 – Addition of fuzzy sets to the rate of change membership function. 64
Figure 5.14 – Resulting output from the improved membership functions. 65
Figure 5.15 – Results of test with the complex stepper motor model. 65
Figure 6.1 – Images of normal road marking situations. 67
Figure 6.2 - Difficult situations for road marking recognition 68
Figure 6.3 – Quadrilateral models superimposed on original images. 68
Figure 6.4 – Ignoring small holes in the line. 69

Figure 6.5 - Cracked Lines. 69
Figure 6.6 – Result on a shaded worn line. 70
Figure 6.7 – Result on a highly worn line. 71
Figure 6.8 – Result of video running at 19.5 frames per second. 72
Figure 6.9 – Smoothly swerving output position of the actuator along with its set point. 74
Figure 6.10 – Histogram of the distance from the set point to the actuator. 75
Figure 6.11 – Step response of the control and actuator. 76
Figure 6.12 – Actuator set point and position (above) and load torque applied to the motor (below). 76
Figure 6.13 – Road marking equipment layout. 78
Figure 6.14 - Position of components on the actuator. 79

List of Tables

Table 3.1 – Advantages and disadvantages of line detection methods. 22
Table 3.2 – Advantages and disadvantages of actuators. 27
Table 3.3 – Advantages and disadvantages of motors. 29
Table 5.1 – Inference table for deciding the speed of the motor. 61

1 Introduction

Roads are used in every part of the world and where there are roads there are almost always road markings guiding the users of the roads in what they can and cannot do. In our country markings are defined and policed by the New Zealand Transportation Authority. The process of road marking involves first painting the lines with approved paint then covering it with small glass beads. This process need to be repeated frequently, depending on the wear of the markings. Remarking occurs as often as twice a year for some places and others it occurs only once every three years.

This job seems relatively straightforward but it needs to be carried out with a high degree of accuracy, in a short period of time when the weather is suitable. Initial marking of roads is straightforward in that there is no line to follow and it does not matter if the line being painted is not perfectly where it should be. Remarking on the other hand requires the line to be painted exactly over the existing line otherwise there will be errors and the road will start to look messy. This thesis aims to look into the possibility of speeding up and making easier the process of remarking roads.

Road markings can be split up into three categories: A Markings, B markings and C Markings. A markings include most lines painted on the road such as edge markings, centre lines etc. B Markings include all the thick diagonal strips of paint in median strips, the thick lines a stop signs and other thick lines. C markings are all the graphics that are painted on the roads such as arrows, diamonds, cycle lane markings and words such as school and give way. They also include car parking and other non road markings. Each of these marking types is done a different way. C markings use stencils and either a hand spray gun or a trolley mounted sprayer, figure 1.1. B markings all use a trolley mounted sprayer although the trolley can be connected to a truck rather than being self contained. A type markings are all painted by trucks and make up most of the markings on the road. This project will concentrate on this type of marking as it is the most common marking and the easiest to automate.

Figure 1.1 - Trolley mounted sprayer [1]

The setup of the trucks for marking depends on the owner/operator. Each company does things very differently, for example where they place the nozzle, how they mark the lines, the equipment on the trucks and its position. This results in very different truck setups and methods of painting. By mounting the nozzle in front of the rear axle of the vehicle it is closer to the centre of the truck and becomes easier to accurately control its position. Being nearer to the centre means that there is less sensitivity when turning and hence more control. This does create some problems, if for instance a wide line is being painted. The nozzle would need to be a large distance away from the side of the truck so the wheel doesn't go over the freshly painted lines. Trucks can paint unto 200 mm wide depending on the height of the nozzle and the type of tip it has. Also when painting very sharp corners it can be very difficult to pain them without the trucks rear wheel going over the lines, figure 1.2. Having the nozzle behind the rear wheel eliminates any issue with sharp corners or wide lines but the nozzle becomes very sensitive to any changes in direction of the vehicle.

There are many items that need to be present on the vehicle for road marking, these include tanks for both colours of paint and the glass beads, hydraulic motor and air compressor and the nozzles and spraying devices. On the vehicles that were looked at hydraulics are used to extend and retract the nozzles and also set the height of the nozzle above the road surface. Pneumatics are used to help propel the glass beads onto the road and air is also blown onto the road between the spraying of the paint and the glass bead application to remove any dust or dirt that may have settled. The setup in figure 1.3 shows the nozzle for paint spraying and the two hoses for each different paint colour, two rectangular shafts used for directing compressed air over the freshly painted line and the glass bead gun. These are all mounted in a linear fashion.

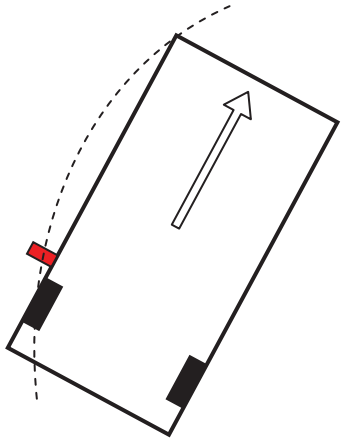

Figure 1.2 – Driving over a painted line

Figure 1.3 – Setup of existing road marking apparatus including spray nozzle, compressed air blower and glass bead gun.

Lines that are marked by trucks are predominately 100 mm wide although vary from 75 to 200 mm. Centre lines that are dashed required to be 3000 mm long with a 7000 mm gap between them. The requirement for the width is that they are within -5 mm and +10 mm of the specified width [2]. Government requirements for lines are quite specific but there is currently little or no policing of these standards. Road markers have currently, in their trucks, electronics controlling the nozzle when painting these lines. These are hooded up to the wheels and know the speed of the truck, they can then pain dashed lines very easily without much help from the operator. There is no real control as there is currently no feedback in this system as it starts and stops the spray

nozzle based on the distance the truck has travelled and not on whether there is a line underneath. As there is little or no policing of road marking specifications markings can often end up being longer or shorter than the requirement. This can lead to an offset in the where the new paint is applied blurring where the line actually is.

Current road marking systems are almost completely manual in their application. The driver/operator is responsible for making sure that the nozzle is positioned exactly over the current line as well as for driving the vehicle. This means that they must be concentrating on no only where the truck is in relation to the line but also watch where they are going. This means that the vehicle will move at a slow speed and also that the operators focus will not be entirely on what is in front of the truck. Experienced drivers currently drive at anywhere from 10 to 20 km/h while maintaining the nozzle over the line to be painted depending on circumstances. A variety of different tools are used to aid them in keeping the truck straight and over the line. Some use large poles or markers so that when they are driving they can look out the window and see where they should be lining up the truck. Others just use a small stick to aid them in the lining up. Figure 1.4 shows the view from the left hand window including the marker pole and rear vision mirror which would have the spraying nozzle in view over the line.

Figure 1.4 – This shows the view out of the truck window with the Marker Pole (oval), and position of the spraying apparatus (rectangle).

Other than the potential increase in speed, accuracy and safety there is one other benefit to owners of road marking companies. To properly train a road marker to a high standard requires about three years of tuition and practice. Even then it is not guaranteed that the person being trained will have the required skill to drive and paint simultaneously. Sometime they will spend a significant amount of time training, only to find that the person is unsuitable for the job. When selecting people to train they have to look for stable people without any issues, who will be likely to stay with the company once they have been trained. As it takes a long time to train and can be a demanding job, road marking operators can command a high salary meaning that to some extent the managers have to keep the markers happy so they will stay with the company.

Each operator gets used to their own truck and has it setup in their own specific way, to road mark in another truck can be difficult as everything is very slightly different. This presents a problem if the road marker wants a holiday or is sick. There is no one who is able to use their truck and so it sits unused during this time. Consequently road markers often work when sick and during the busy time they don't have holidays while working six day weeks and 12 hour days.

The long training periods, truck setups and high wages all contribute to the high cost of road remarking. By creating an assisted remarking system many of the problems that cause these high costs would be removed. Training would be much faster, each operator would not need their own truck, and wages would be lower.

What is wanted is an assisted road remarking system that will help road markers in their remarking duties and combat some of the problems explained above. Creating an assisted remarking system will make it easier to operate a truck. This means it will be easier to move from one truck to another and also it will be easier to train someone up to do the job, eliminating the long teaching period and uncertainty. It will potentially increase the accuracy of the markings as well. In addition it allows the operators to concentrate on the road ahead improving safety and also the speed that remarking can occur. The cost benefits would also be advantageous.

1.1 Thesis Overview

These problems are widespread and the fastest way to improve them would be to modify and improve the process of remarking. This thesis will look at the development of the automation of some aspects of the remarking process, namely using a nozzle that can follow the line being repainted autonomously. The aims include developing a system that can detect and follow a line to a high degree of accuracy (± 5 mm) and at a high speed. This needs to be coupled with smooth control so that the line will appear visually smooth.

The system that will be developed will use sensors, controllers and actuators to adjust the position of the paint nozzle so it will be able to move back and forth to follow the previously painted line. This has the advantage over a traditional system in that there is some variability in the possible positions of the nozzle meaning that the position of the truck in relation to the line is not so vital when compared to the current method involving a fixed nozzle. The driver can now mark the lines faster and pay more attention to the road without sacrificing quality. The quality and accuracy of the painted lines will also be improved as the nozzle will only be turned on when it is appropriate.

The first part of this thesis consists of chapters 1 – 3. This section introduces road marking and remarking and the problems it faces. It investigates technologies and processes that have been developed and used for similar situations that might be useful for the purposes of this project. The various sections of the system are looked at in detail and basic design decisions are made based on the advantages and disadvantages of each of the components.

The second part of the thesis is chapters 4 – 7. This section looks at the development of the line finding algorithms, the choices around the actuation and the development of the motor control system. In chapter 6 these are modelled and tested to see how they respond in various conditions and to various situations that might arise. They are compared to the desired characteristics of the system. Chapter 7 concludes this thesis by drawing conclusions from the results and looks at any further development that may have arisen from this thesis.

2 Literature Review

This is a review of technologies that may aid in the design and implementation of an Assisted Road Remarking System. The system that will be developed will use sensors, controllers and actuators to adjust the position of the paint nozzle so it will be able to move back and forth to follow the previously painted line.

2.1 Line Detection Methods

There is almost no prior research in this specific area. It is limited to [3] where faded lines are assessed for their parameters (dashed, double etc.) and even this is not performed in real time. The closest area is that of lane and marking tracking for the purposes of driver assistance and autonomous driving.

2.1.1 Sensors

The area of lane tracking and detection has been the subject of a wide variety of research projects. The number will continue to grow as the areas of automobile driver warning, assisted driving and autonomous driving systems are increasingly being investigated and developed. Using various sensors and processing algorithms the end result of the majority of these projects is usually the creation of a computational model involving both the vehicle and the road it is travelling on [4] [5], see figure 2.1.

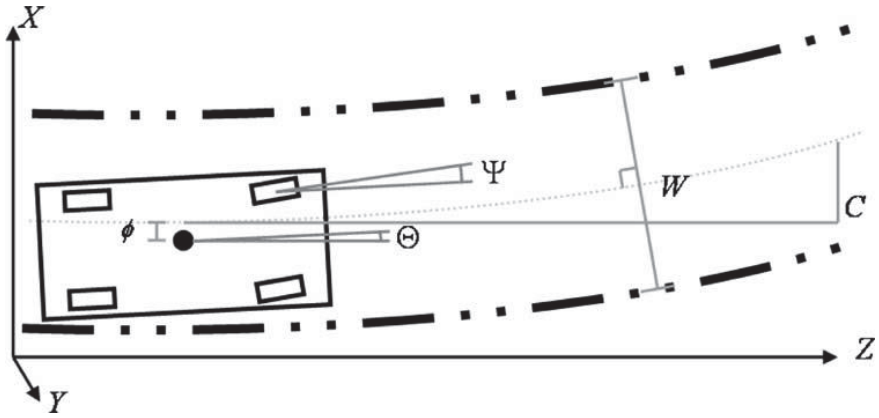

Figure 2.1 – Vehicle and road model used in [4] for driver assistance.

Information in these models may include the position and angle of the vehicle within the lane, the shape of the road ahead and to the side of the vehicle and the makeup of the road such as the number of lanes. It also usually includes additional information such as other vehicles and their positions and any significant road markings such as arrows. These models are then used to perform various functions depending on the desired application. For example they may warn the driver in the cases of driver assistance or correct the direction and/or speed of the vehicle under control.

The majority of these systems use a Charge Couple Device (CCD) camera as one of the main sensors to obtain the information required to create the model. These are organised and set up in a variety of different ways depending on the application required. For autonomous vehicles the cameras are focused primarily forward so as to analyse what is happening on the road in front of the vehicle and also for information about any concerns that may be in the distance and may need to be responded to. For example in [6] a single CCD camera is attached to the rear vision mirror looking out of the windscreen. The more cameras there are the more data there is available and the better the model will be and hence the control. [7] has three wide angle cameras looking at front left, front centre and front right of the vehicle and a narrow angle camera giving a much higher quality picture of the most important section of the road ahead. When these images are merged together it allows for a more accurate mapping of the road ahead in comparison to a single camera, figure 2.2.

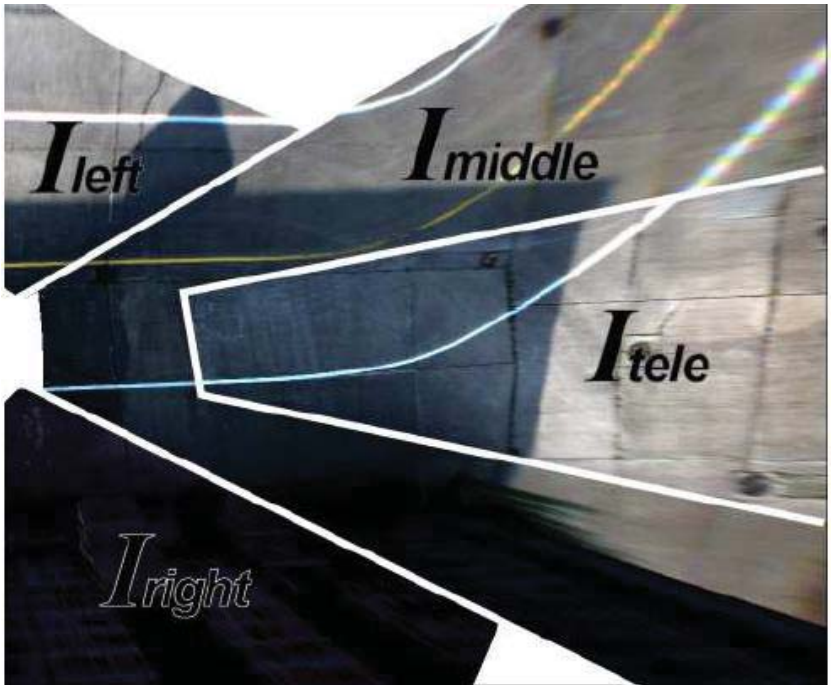

Figure 2.2 – Four different images are merged together to form a single view of the road ahead.

An application requiring only lane position and tracking has its cameras focused downward for higher accuracy at the cost of any prediction capabilities. [8] has two cameras mounted either on the rear bumper or under the wing mirrors looking toward the ground allowing the vehicle to know very well its current position within the lane.

It was found in [8] that CCD cameras are slow to adjust to changes in light intensity when the cameras were mounted on the rear bumper. Reflections of sunlight off the bumper along with headlights of following vehicles were some of the issues encountered. A custom complementary metal oxide semiconductor (CMOS) camera has been proposed in [8] to combat this issue. It has a higher dynamic range than a CCD camera allowing it to cope more effectively with sudden changes in light such as at dawn or dusk or from reflections from other vehicles. Other methods of detection used to create these computational models include laser RADAR and line sensors. The RADAR was found to be very good at finding road boundaries but not road markings while line sensors performed well at detecting lines that were correctly marked but had no way to look ahead or predict where the lane was going [4]. In this case vision sensors were found to perform well in the widest variety of situations and are the most versatile.

2.1.2 Algorithms

Many different approaches have been taken in the area of image processing and computer based lane tracking. Most of these approaches work on greyscale images as the processing time to analyse a greyscale image with one channel of information is significantly less than a corresponding colour image, typically with three channels [6]. If colour information is used, it is usually used to supplement the greyscale image in finding the markings [10]. In this case if any of the Red, Green or Blue values fall outside the accepted variance of the image then it is flagged as a feature. Another colour space that has been used is the HSV colour space [6] as road markings show a change of saturation and/or intensity and as HSV encodes colour and saturation in different channels this means that only one channel needs to be worked on instead of three.

Perspective mapping has also been utilised pre-processing in a variety of situations [11]. It is used to convert the images the cameras see into a bird's eye view of the road ahead. It is quite possible to merge several cameras into one top down view giving increased detail where needed [7]. To accurately use perspective mapping many details of the camera and its position must be known such as viewpoint, viewing direction, its aperture, resolution etc. The perspective mapping is a function of these values and is carried out on the received image before it is processed for line information [12], figure 2.3.

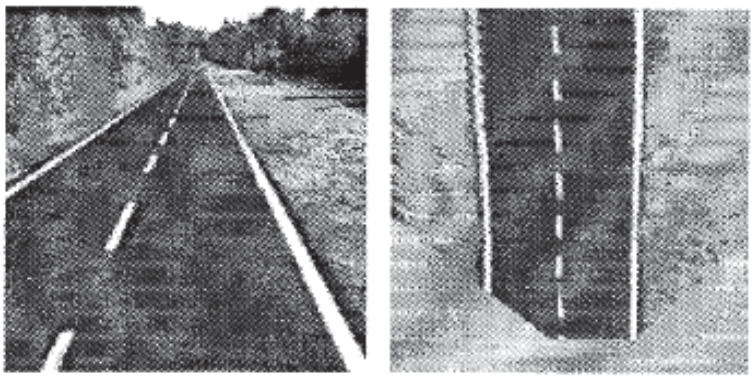

Figure 2.3 – View of a road before and after perspective mapping.

Some systems use Single Instruction, Multiple Data (SIMD) architecture to process images. These perform exactly the same function on many different pixels at the same time and so require each pixel to have the same amount of information. As an image of the road from a camera has more information contained in the pixels closest to the camera this sort of architecture cannot be used unless the image is perspective mapped first giving each pixel the same amount of information [13].

Along with the use of SIMD architecture the advantages of using perspective mapping include allowing the filter and program to use various assumptions to better find the markings. From the top view, markings appear as long, uniform width variations in the roads surface and can be classified this way. It also allows the measuring of various distances more easily. The only issue with perspective mapping is the assumption that the image being mapped is uniformly flat and in the case of roads that is not always true. By using some form of RADAR a height map can be generated and a more accurate perspective mapping can occur [7].

As the images that are being analysed are not independent of one another and rather form a sequence it can be safely assumed that the position of the lane markings will not have significantly moved from one frame to the next and to exploit this fact regions of interest (ROI) are used. These are selected areas of the images that are searched in place of the whole image and are chosen based on where the markings were in the previous image. This approach is used in several applications [14] [15] [16] and allows the processing to be sped up. [8] and [16] both use short horizontal lines of pixels spaced out over the projected location. These are at specific distances corresponding to distances in the real world. [17] uses increments of 0.2m once perspective mapping has been carried out. [15] on the other hand uses both a trapezoidal window based on the shape of the road that is derived from a perspective transformed image, and also small windows not unlike the scan lines but with a height component added. This approach is also found in [14] and allows some vertical averaging to take place. Rather than have the entire model

resting on the results of a single line of pixels which could include a white fleck giving a false positive or a worn segment falsely indicating no line is present. If an area is sampled it is much more likely that these cases would be picked up as false and a better result would be obtained.

There are a large variety of methods used to detect lane markings, some with only subtle differences between each other and others very different. The most common of the methods is that of positive negative gradients found in [5] and [17]. The positive negative gradient method scans horizontally for dark-bright-dark gradient changes where the changes are greater than a threshold. It also checks that the space between the gradient changes is within a specific range of values. Both these conditions must be met for a line to be declared present. This method works well for vertical and curved lines but doesn't function for horizontal lines which do not feature significantly in road markings. There are varieties of this method such as [8] which scans for dark bright dark transitions where the two gradient changes are equal in magnitude but mirrored in appearance.

Top hat filters perform a similar process to the positive negative gradient method. Horizontal lines of the image are convolved with a top hat shaped wave of varying widths, figure 2.4. When the top hat meets a line of the same width, the positive sections combine with the line and the negative with the road colour and give a maximum value. Maxima in the widths vs. horizontal position plot are accepted as features then compared to a threshold [17].

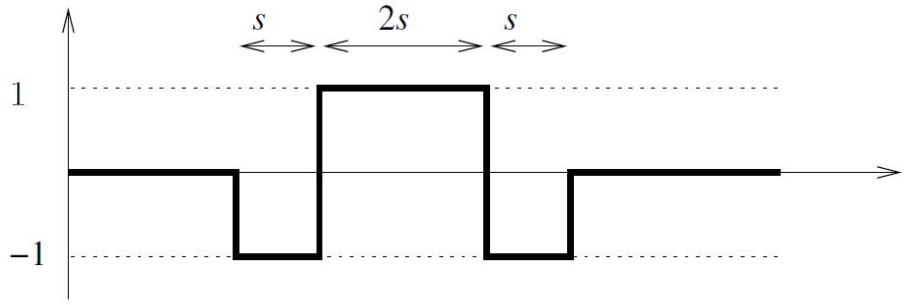

Figure 2.4 – Top hat filter for line detection [17].

[18] is similar where it convolves horizontal lines of pixels with a set matrix and then those values are put through a threshold to find the edges. This second method only works for a line of a specific width. These filters only work with vertical lane markings but are quick and simple to perform.

Parallel features or symmetrical local thresholds are also similar to positive negative gradients in that along each horizontal line of pixels it computes the averages to the left and right of each pixel

and if the intensity of the current pixel is above both of these by a certain amount then it is counted as a feature [17]. [10] has a more analogue approach where it checks pixels a certain distance each side of the specified pixel and depending on the amount of difference assigns a value to the pixel corresponding to that difference. This two dimensional array of values is then analysed to find features. This method has the advantage that despite different brightness of road markings, such as shadows, the difference between the road marking and the surroundings will still be maintained.

[6] and [9] both use a combination of Canny Edge Detection and then the Hough transform to detect lane markings, figure 2.5. Canny edge detection is used as it has a very low error rate and it outputs a single pixel connected edge. Hough transform is used in conjunction with the Canny Edge Detection as it is able to detect lanes whether they are dashed or continuous. This approach finds the lane boundaries reasonably well although based on visible results the accuracy of the position of the boundaries of the lane markings are not very precise.

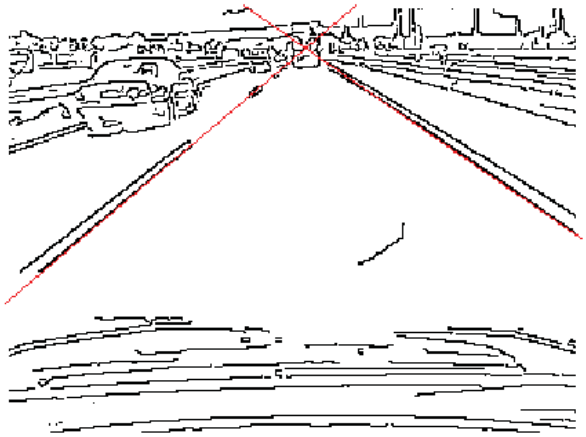

Figure 2.5 – Canny filter edge image with the correctly selected outputs based on the Hough transform.

Local thresholding is looked at by [17] where local averages are used to decide whether a feature is present or not. If the current pixel is more than a certain amount above the average of the surrounding pixels then it is considered a feature. The size of the averaging shrinks as the horizontal lines shrink to cope with perspective and a reducing amount of information in each pixel. The extracted features are then put into an extraction map and horizontal connected features that are wider than a certain value are selected as the final features.

Different filters have also been used to detect lane markings, steerable being the most common of those. Steerable filters are a combination of rotations of a certain filter through which the

response of that filter can be interpolated at any orientation [4] [19]. Reasons for using steerable filters include that the filters can be separated into X and Y components and so the processing can be sped up and the fact that only a few angles of the specific filter are needed to get the whole range of angles and so finding the angle with the highest response is the direction of the lane marking. These are used to find angles at which minimum and maximum responses are or the response at a certain angle. This method works well with lines that are under shadows.

[17] presented different methods and assessed their ability to detect road markings and also the number of false positives that were generated. It looked at the Global Threshold, Positive Negative Gradient, Top Hat, Local Threshold, Symmetrical Local Threshold and Steerable Filters. The results showed that the least successful was global thresholding and the positive negative gradient method also performed poorly. The other four tested were similar, but were in the following order: Fourth was the steerable filter, top hat filter was third and local thresholding was second. First was the symmetrical local filter. It has a relatively low processing requirement but it does not detect significant curves very well.

[17] also proposes using two functions $S_m(x)$ and $S_M(x)$ to keep the line width information in. As the camera will be looking forward the width of the line will change depending on its distance from the camera. It is proposed that there be a function that corresponds to the current row x and can give the minimum expected width of any markings $S_m(x)$ and the maximum expected width of any markings $S_M(x)$ at any row x of the image. There may not need to be an entire set of values for this application as depending on the position of the camera the width may not change significantly. It is worth calculating the expected width and possible having a single minimum and maximum value instead of a function.

All of the above methods rely on lines that are well defined and have a contrast ratio that is large enough to detect. The problem is that when remarking the lines they are not likely to be in good condition, they are going to be faded and/or dirty from use. It will be much harder to detect lines under these conditions. [3] developed a method involving symmetrical local thresholding several horizontal lines throughout the image then using a radon transform between these lines to decide where the road marking is. The radon transform is an integral transform that takes a two dimensional function and breaks it down into a one dimensional view over a set angle or range of angles. Integrating along the length of the line will result in a spike in the output corresponding to the line even if it is somewhat faded or dirty. The line will be made up of small rectangular segments once this is complete.

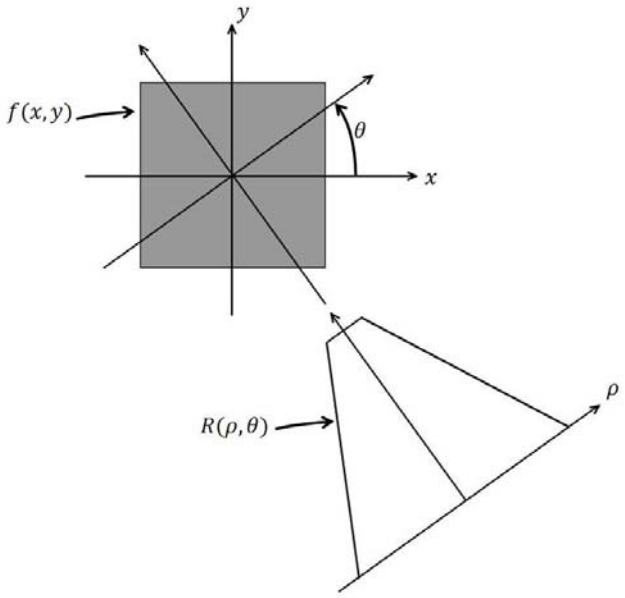

Figure 2.6 – Geometry of the radon transform.

The full radon transform integrates the pixels values of the image over 180° and there are some potential issues that arise from this when looking for lines in an image. A biased peak within a radon transform can have repercussions such as having the centre of the line incorrectly identified and/or the angle of the line incorrectly identified. This occurs because the line appears as a region of values with a greater value than those surrounding it. The highest single value in this region may not be at the centre because these values are an integration of the image and rely on the overall quality of the line. If a line has degraded then in the output of the radon transform it will not appear as a uniform square pulse with a flat top but rather peaks and troughs. Hence if the highest point is taken then it may not actually be the centre of the line. A similar effect can occur if, when looking for the angle of the line, the highest point is sought. For example, if a radon transform was performed on a line the highest point of the resulting output would occur not in the direction of the line but the angle that corresponds to a line connecting one corner to the opposing one.

[20] manages to solve these problems by using a low pass filter on the radon values to smooth them out both in the angular and linear directions. It was suggested that a filter just smaller than the size of the width of the line be used to get the best results and it could be either a mean, Gaussian or median filter. There is also a boundary problem that occurs when a line is on the boundary of when the filter starts and finishes (0°). As the road markings will be mostly vertical this has the potential to occur. To remedy this it is suggested that the radon transform be used not from 0° – 180° but from -90° – 90°. Other problems that are not able to be solved include the

radon transforms inability to define start and end points of a line or to detect particularly short lines. Also if the line has some curvature clear results may not be discernible.

Lane tracking has been achieved in the past at speeds of greater than 80km/h [21] [22] and so in theory to detect road markings to paint over at lower speeds should not be a problem. These speeds may not be the best estimate of the speeds obtainable as they are for a vehicle where the accuracy is not as critical as with the repainting. A graph in [19] shows a variation in the position of the lane by as much as 0.2 of the lane's width which is not near to the accuracy that is required for this project.

2.2 Control Methods

The most common control method currently in practice is PID control. It is widely used throughout the world in many different industries including industrial automation and process control. The reasons for its popularity include its robustness and simplicity of design [23] [24], a clear and predictable relationship between the tuning parameters and the response of the system [23] [24] [25], and many techniques to help with tuning and its flexibility [24]; for example, some modern systems incorporate auto-tuning of the parameters [24] [25]. The equation governing PID control, equation (1) is straight forward and the process can be seen in figure 2.7.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t) \qquad (1)$$

Where $u(t)$ is the controller output, K_p is the proportional gain, K_i is the integral gain, K_d is the differential gain, e is the error (set point – current value) and t is time.

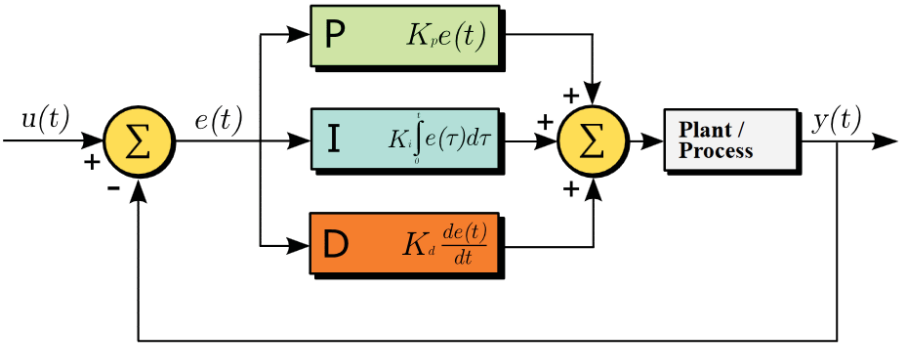

Figure 2.7 – PID control of a system.

Small advances such as partial derivative action help to make the PID controller more robust by eliminating derivative kick. This kick occurs when there is a set point change and the error will jump instantly giving a very high/low derivative component for a small amount of time. To

eliminate this derivative action is applied to the process variable instead eliminating the possibility of sudden changes. Trapezoidal integration is also used in [26] to help avoid large changes in the amount of integral action when there is a set point change. It can cause overshoot to occur if it is ignored but by smoothing out the integral action this will not be as significant a problem.

PID works very well on linear systems but doesn't function well with nonlinear systems. This is because to use a PID controller a mathematical model of the system is needed and for many systems this can be too complicated or vague to model [27] [28]. Time delays in the model also pose a problem for PID controllers. When using motors the systems can be modelled but if there is potential for the torque to change the model may not be accurate all of the time. It was found in [29] that if a motor is controlled while under varying loads, the overshoot and oscillations of the system will increases. It was also found that the size of the set point change had an adverse effect on the control. This means that the PID controller is sensitive to inertia variation and sensitive to variation of the range of reference speed alteration.

Fuzzy Logic is also used in a large variety of areas although not to the extent of PID. Its major advantage is that there is no mathematical model needed for this control method to work. The rules of the controller are based on the system behaviour and also the knowledge and intuitiveness of the designer. As a Fuzzy Logic Controller (FLC) in its simplest form is equivalent to a PI controller these two were compared and the FLC worked better on nonlinear and time delayed systems than the PI controller [30] confirming that FLCs can and do work well for nonlinear systems. This also means that varying loads and set points are also not a problem [28] [29].

The major difficulty with FLCs is the ability to tune them. As there is no model and it is all done by the designer it can be very difficult to get it working without trial and error. Tuning a PID controller only involves adjusting three variables and it has been widely documented as to how they will affect the overall system (response, oscillations, overshoot) whereas a FLC has many membership functions and an inference section to contend with. Methods can be used for initial tuning that involve describing the controller as a classic PID controller and setting some initial parameters that will allow for qualitative tuning of the FLC to a more precise level [24]. Automatic tuning can also be applied to FLCs although these are PID auto tuning methods adapted for FLC [25].

3 Proposed System

Current road marking systems are almost completely manual in their application. The driver/operator is responsible for making sure that the nozzle is positioned exactly over the current line as well as for driving the vehicle.

In some vehicles there is a limited amount of computer use in controlling the nozzle for painting different patterns of lines such as long white dashed lines or the short yellow dashed lines. There is no real control as there is currently no feedback in this system as it starts and stops the spray nozzle based on the distance the truck has travelled and not on whether there is a line underneath.

Currently road markings in New Zealand are repainted every 12 months and due to this there is a need for speed to enable the maximum number of lines to be painted. This project aims to speed up the task of remarking by introducing some automation into the process. The remarking system can be split up into three main sections: Sensing, Control and Actuation. Sensing will look at how the line is detected, right from the sensors used to the algorithms and processes used to determine the makeup and position of the line. Control will involve receiving the sensing information and determining how that information is to be used. It will also investigate the best method of control for the actuation so as to keep the nozzle correct and keep the line smooth. Actuation will focus on the movement of the nozzle and how it will follow the line. From the various actuator types that can be used to the methods of powering those actuators.

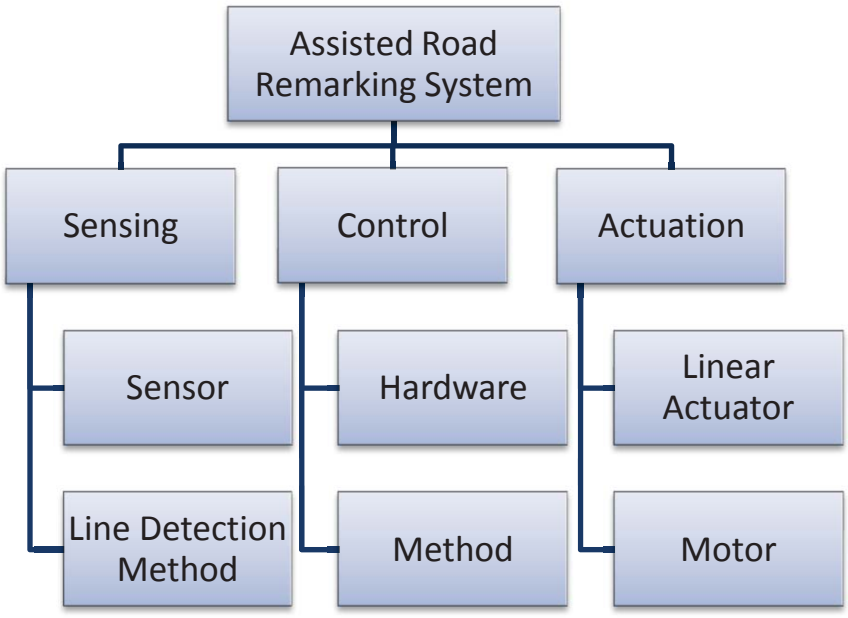

Figure 3.1 – Sections of this project

3.1 Sensing

3.1.1 Sensor

3.1.1.1 Type

After looking at various journal articles, the most common sensor used for line following and detection is a camera. There are two main types of cameras: Charge-Coupled Device (CCD) and Complementary Metal–Oxide–Semiconductor (CMOS). CCD cameras are the most common of the cameras available today. They make up most compact digital cameras, DSLR cameras and video cameras. CMOS cameras are mainly used in webcams, mobile phones and some DSLR cameras.

When light is captured in each sensor of a CCD camera it is stored as an electrical charge. These charges are then changed into a voltage one pixel at a time then an external analogue to digital converter changes the voltages into digital values corresponding to each pixel. This method of image acquisition has been used for longer than CMOS and so tends to have larger pixel counts. It is also somewhat energy intensive and consumes a large amount of power. A CCD sensor has a large dynamic range and a good response time. The method used to get the information off can result in vertical smear if there is a bright light and the sensor gets overloaded.

CMOS cameras work by using a series of photo sensors, each individually connected to circuitry that converts the charge from the light into a voltage. It is converted to digital information elsewhere on the chip. CMOS sensors have a lower sensitivity than CCD sensors as more photons hit the additional circuitry required for each pixel than the actual sensors themselves. The large degree of integration of functions within the sensor means that it is slightly faster, smaller and requires much less power than the CCD sensor. CMOS sensors often employ what is called a rolling shutter where the information is collected sequentially from one edge to the other however long that takes, rather than from a single point in time. The advantage to this is that images can be captured more frequently as parts of the sensor can be capturing information while others can be sending it off but side effects include partial exposure, wobble and aliasing.

Photodiodes are also occasionally used in some simple line following applications. From previous work done in similar fields of autonomous/assisted driving it has been found that the photodiodes may be good at picking up where the line is but they can only see immediately below themselves. There is no ability to look forward to where the line will be. They are also only able to determine if the area they are looking at has a line present, there is no ability to infer that a line should be there if at that position the line has worn away. The photodiode will just assume that there is not a line there and this could create issues with a particularly worn line. They do have the advantage

of immediate response as there is no need to process or figure out where the line is, their information can be used immediately.

3.1.1.2 Connection

CCD cameras come with either a USB or FireWire connection. Both have their advantages and disadvantages. USB is a very common connection; almost every computer has a connector of this type allowing versatility in the hardware to interpret the data from the camera. USB works in a master-slave configuration and requires some processing ability on the host side to enable the communication to work slowing the actual transfer speed. So if more than one camera is used the transfer rates of the cameras would be slowed down even more. FireWire on the other hand uses a peer to peer method of data transfer so that if more than one camera is used then there will be negligible impact on the overall speed.

Another consideration is that of power. USB is powered via a 5V connection with a maximum of 500mA giving a total of about 2.5W of power while FireWire can deliver voltages up to 30V with 16W of power. This amount of power means that using USB will most likely require an external power supply for the camera but that the use of FireWire can provide adequate power for most cameras itself.

The biggest advantage of USB is the fact that the connection is universal and very common and the disadvantage is the requirement for extra power and also the fact that with multiple cameras the transmission of the data may be slower than FireWire. FireWire's biggest advantage is that it can achieve high data transfer rates perfect for video transmission but the connector is relatively uncommon [31].

3.1.1.3 Location

The position of the sensor relative to the nozzle is very important for the system. It will affect the control, stability and overall performance of the system. The actual position will depend on the sensor used but there are some issues that can be clarified now.

Having any device mounted a distance in front of where the nozzle is spraying does not work, as the vehicle is still being driven by an operator. Unless the speed and steering direction of the vehicle are known, then the line cannot be assumed to be where the sensor senses it. If the operator has increased speed or changed direction of travel then the line will not be in the same place relative to the nozzle and any predefined positioning of the nozzle will not be effective.

Positioning of the sensor presents a challenge as it needs to be able to see where the nozzle is spraying to make adjustments but it also needs some look ahead so be able to predict where the line is going and where it will need to move the nozzle to in the future.

3.1.2 Line Detection Method

Either colour or greyscale images are both viable possibilities for use in lane detection. Using colour has the advantage over greyscale in that more information is captured in the image and potentially a more reliable result can be obtained. The downside to the use of colour images is the time taken to process those images. It takes roughly three times a long to process colour images when compared to greyscale images and in applications where speed is a factor greyscale images are better suited.

Regions of interest are also widely used in the area of lane detection in videos. These are areas of the images that are searched instead of the whole image and are based on where the markings were in the previous image. This approach allows the processing to be sped up. Various shapes of ROIs have been used in a number of processes. Short horizontal lines of pixels spaced out over the projected location as specific distances corresponding to distances in the real world [9] [16] are one example. Another is a trapezoidal window based on the shape of the road that is derived from [15]. There are also small windows not unlike the scan lines but with a height component added to them.

Through research many different methods of line detection have been proposed and used in various previous applications. The most common methods being Global Threshold, Positive Negative Gradient, Top Hat, Local Threshold, Symmetrical Local Threshold and Steerable Filters. Each performs slightly differently and requires different amounts of time to complete.

The positive negative gradient method scans horizontally for dark-bright-dark gradient changes where the changes are greater than a threshold. It also checks the space between the gradient changes and this must be within a specific range to be allowed. This method works well for vertical and large radius curved lines but doesn't function for horizontal lines.

Top hat filters perform a similar process to the positive negative gradient method. Horizontal lines of the image are convolved with a top hat shaped matrix. The resulting values are put through a threshold to find the edges [18]. This second method only works for a line of a specific width. With additional processing time the width of the line can be taken into consideration by varying the width of the convolution matrix [17].

Parallel features or symmetrical local thresholds ask whether the current pixel is a line by looking at the pixels a small distance to each side and then checking the intensity difference. Both single pixels [12] and averages of pixels either side [17] have been used but it depends on processing speed and also the road surface. This process is also width dependant although by making it a large distance the possibilities of ignoring a line are diminished.

There are two types of thresholding, global and local. Global creates a threshold based on the entire image and then anything over that threshold is counted as a feature. This has a large number of false positives and is not very accurate [17]. Local thresholding uses local averages to decide whether a feature is present or not. If the current pixel is more than a certain amount above the average then it is a feature. The size of the averaging shrinks as the horizontal lines shrink to cope with perspective.

Different filters have also been used to detect lane markings, steerable being the most common of those. Steerable filters are a combination of rotations of a certain filter through which the response of that filter can be interpolated at any orientation [4] [19]. Reasons for using steerable filters include that the filters can be separated into X and Y components and so the processing can be sped up and the fact that only a few angles of the specific filter are needed to get the whole range of angles and so finding the angle with the highest response is the direction of the lane marking. These filters are used to find angles at which minimum and maximum responses occur or the response at a certain angle. This method works well with lines that are under shadows.

All of the above methods rely on lines that are well defined and have a contrast ratio that is large enough to detect. The problem is that when remarking the lines they are not likely to be in good condition as they are going to be faded and/or dirty from use. It will be much harder to detect lines under these conditions. The radon transform is an integral transform that takes a two dimensional function and breaks it down into a one dimensional view over a set angle or range of angles. Integrating along the length of the line will result in a spike in the output corresponding to the line even if it is somewhat faded or dirty [3].

Table 3.1 – Advantages and disadvantages of line detection methods.

Line Detection Method	Advantages	Disadvantages
Global threshold	Quick Simple	Large number of false positives
Positive-Negative Gradient	Fast	Doesn't work for significantly curved lines
Top Hat	Works for different thickness lines	Somewhat resource intensive
Local Threshold	Quick Simple	Medium number of false positives
Symmetrical Local Threshold	Fast Most accurate	Doesn't work for significantly curved lines
Steerable Filters	Wide range of data to work from Works with shadows	Complex Resource intensive Only works for a set width of line
Radon Transform	Eliminates issues of dirty/missing lines	Moderately complicated

3.2 Control

3.2.1 Hardware

The various tasks that need to be performed include the decoding of the video stream from the camera/cameras analysing where and by how much the nozzle needs to move and moving it. To be doing this at a fast enough speed to have high enough accuracy and to be able to do it at a fast enough speed will require a fast processor.

Microcontrollers are very versatile and can perform many tasks but image processing is a complex process and may be too complicated to be handled via microcontrollers. Microcontrollers may be able to be used for other tasks that do not require as much complexity such as motor control or sensor analysis, only forwarding the final processed signal back to the main processor allowing the main processor to focus on the main task of image analysis.

A PC or laptop can be customised to the required specs in regards to processing power, memory etc. They support a wide variety of programming languages and have a host of connectors that can be used to control and drive peripherals. The issues resolve around the robustness of a computer and the size that it will take up. It is required to work consistently and any failure on its part will be visible.

To combat the size of a computer a single board PC can be used which incorporates all the significant workings of a computer onto a single small board and any peripherals such as a keyboard or screen can be plugged in or attached when needed. They also incorporate a variety of I/O ports more suited to an industrial application such as ADCs, DACs and analogue and digital I/O ports built onto the board instead of requiring an external module.

3.2.2 Method

To accurately position the paint head over the line some method of control will be needed to achieve this. The two main methods of control are Proportional, Integral and Differential (PID) control and Fuzzy Logic Control.

PID controllers work by attempting to remove the difference between a set point and the current point by calculating and implementing the corrective action required to keep the difference minimal. The way it calculates the corrective action is through three different parameters; proportional, integral and differential values. The proportional value determines the response based on the current error, the integral value based on the build up of the recent errors and the differential value based on the rate of change of the error. By adjusting the importance of each of these values the response of the system will change. The response can be described in the following terms, the responsiveness to an error, the degree of overshoot of the set point and the amount of system oscillation. It is possible to only use one or two of the parameters if the specifications so require. Every application and system is different and has different requirements and so will have different tuning parameters. To accurately determine the parameters of the system it is required that the mathematical model of the system is known or else various experiments be carried out.

Fuzzy logic works through reasoning that is approximate rather than precise. First it takes absolute inputs and quantifies them into subjective terms (e.g. large, warm, big). The value can be in more than one category as something that is large may also be very large. The designer defines what input values correspond to in the subjective way and how much it corresponds to that description, these are the membership functions. Once the inputs have been fuzzified the correct subjective outputs are inferred from a table or set of statements based on the input values. Once there is a subjective output, it is possible through the reverse process to defuzzify it back into an absolute output. The fuzzification, defuzzification and inference are all chosen by the designer based on knowledge of the system and its application.

3.3 Actuation

3.3.1 Linear Actuator

3.3.1.1 Electro Mechanical Actuators

Electro mechanical actuators are actuators that convert the rotational movement of an electric motor into linear movement. The motor is used to turn a threaded rod inside a shaft. This threaded rod is wound through another shaft inside the outer shaft that is rotationally fixed so it cannot rotate. The rotation of the threaded rod in turn causes the shaft to move in and out depending on the speed of the rotation and also the gauge of the thread on the shaft.

Electro mechanical actuators come in a wide variety of types but the most common configuration can be seen in figure 3.2. The sizes range from 5 – 1000 mm and can accommodate what is being looked for in this application. Speeds vary depending on the make and motor used but are generally around 35mm/s. The accuracy of these systems should be good provided there is adequate sensing of the motor's rotation and knowledge about the threaded rod's characteristics.

Figure 3.2 – Typical electro mechanical actuator.

The only major possible problem is the length of the actuator when it is compressed. It is longer than the stroke when compressed and double that when extended. As road marking needs to be within one lane when working the size of this actuator and where it needs to be placed to actuate properly, may pose a problem.

3.3.1.2 Linear Motion Stages

A similar principal to the electro mechanical actuators is employed with linear motion stages where rotational is converted to linear via a threaded rod. In this case though a shaft isn't pushed out the end, rather a table travels along the threaded rod between two ends of the stage.

Figure 3.3 – Linear motion stage.

The only major differences between the linear motion stage and the electro mechanical actuators are the lengths and support. The length of the linear motion stage is fixed as the table moves back and forth along it and so the overall length is comparable to the compressed state of the electro mechanical actuator and much shorter than the extended state. The linear motion stage is also able to support the weight of what it is moving (to a certain extent) anywhere along the stage whereas the electro mechanical actuator would need another mechanism to support the weight of the object while it moved it back and forth.

3.3.1.3 Pneumatic/Hydraulic Actuators

Air/hydraulic fluid under pressure is used to move pistons/actuators forward and backwards creating linear motion.

Pistons are able to move large loads back and forth in a very similar manor to electro mechanical actuators. There are a number of problems and issues with hydraulic/pneumatic actuators relating to the control of the actuators. The motors are able to stop the actuators at specific points to a high degree of accuracy whereas using a compressed fluid/gas allows for simple control such as close open but not the precise control needed. They also suffer from the problem of needing a large amount of room to work in a linear manner and also will require another mechanism for supporting the weight of the object being moved back and forth like the electromechanical actuator.

Figure 3.4 – Hydraulic actuator.

The support system for either one of these systems is also fairly large when compressor, storage tank/reservoir and all the fittings are taken into account. There is the possibility that one of these systems is currently on a truck being used for some other purpose and it may be possible to utilise that system and save the room required for another setup.

3.3.1.4 Belt Driven Stages

A motor is used to drive a belt (usually toothed for accuracy) around two pulleys. Connected to this belt is a face plate capable of holding the required item. It is held in place via a set of tracks and the belt is used solely as the means of movement and not stability. It has lower accuracy when compared to other methods but not significantly so for road marking purposes. (Accuracy is approx. ± 0.1 mm compared to 0.01mm for linear stages) The only complication may arise when replacement for the belt is required and also in an environment such as road marking whether its lifespan will be long enough to justify using it.

Figure 3.5 – Various belt driven stages.

Table 3.2 – Advantages and disadvantages of actuators.

Type of Actuator	Advantages	Disadvantages
Linear Actuator	Accurate	Would need additional supports so it can move back and forth Higher Cost
Linear Slide	Accurate Able to carry weight without additional support	The weight it can carry is limited Higher Cost
Pneumatic/Hydraulic	Move large loads	High number of additional components needed for only a single actuator Can only push and pull and not hold The accuracy is not guaranteed
Belt Drive	Fast moving Cheap	Lower accuracy(±0.1 mm) Belt replacement could be a problem

3.3.2 Motor

3.3.2.1 Brushed DC

Brushed DC motors are the most common of electric motors. They are quite simple and come in a large variety of sizes and power ratings. DC power is connected to the motor and this is transferred into torque through the use of commutation and stationary magnets. Generally the speed of the motor is proportional to the voltage applied and the torque is proportional to the current. Brushes are needed to transfer the power from the source to the commutator which rotates inside the magnets. The rubbing of the brushes on the commutator causes the brushes to wear out over time and creates a fine metal dust. This necessitates frequent maintenance in replacing the brushes and springs and the cleaning of the commutator. It is also not the most efficient motor when compared to other types.

For the purposes of this project it will need to have an external sensor fitted so as to determine the position and rotational speed of the motor.

3.3.2.2 Brushless DC

The difference between a brushed and brushless motor is that in a brushless motor the permanent magnets rotate instead of the commutator. The problems with the brushes connected to the commutator are not found in this setup as there is no need to connect power to the rotating component. Instead a timing circuit is required to energise various sections as needed. This gives a slightly more efficient motor that does not require servicing and hence a longer lifespan. The negatives include the fact that it is less robust and requires complex control to

manage the timing. Its peak torque is when it is stationary and it linearly decreases with speed after that. It is also expensive when compared to a brushed DC motor especially when the control electronics are included.

Like the bushed DC motor, this will also require an external sensor fitted so as to determine the position and rotational speed of the motor.

3.3.2.3 Servo Motor

Servo motors can be DC brushed, brushless or AC controlled. They involve a motor usually paired with a gearbox and a positioning tool. The inbuilt sensor gives very precise control over the position of the motor that is continuous. Whereas a stepper motor assumes that every step has resulted in a slight turn and it needs to count these steps to know its position, the servo motor can know straight away its position very similar to an absolute encoder. Due to the more complex control required for the use of the servo motors, sometimes they are not designed to rotate more that a certain amount, and to get them to operate an actuator of some sort would not be what they were designed for.

3.3.2.4 Stepper Motor

Stepper motors are very similar in operation to a brushless DC motor. A core is made up of toothed magnets which are surrounded by carefully placed electromagnets. These electromagnets are energised in a specific order which causes the core to rotate. Each time a magnet is energised constitutes one step and depending on the motor is usually 3.6° or 1.8°. These steps occur at very high speed and so the motor appears to travel with a constant rotation rather than steps. These motors are used fairly widely and so despite the need for additional circuitry to control the motors they are not expensive.

If a coil is left energised it will dynamically brake the rotor. This is also useful when the motor is stationary. Up to a certain torque it will act as a brake and not turn, which is very different to the other motors which have no mechanism stopping them from turning when not in use. The accuracy of the step size allows the controller to know how far the motor has turned without any sort of external feedback but if the motor fails to make a step there is no way to know this and so it is advised to home the motor to a known location occasionally to keep it accurate. One disadvantage to the stepper motor is that because the movement is not uniform but stepped, the motor introduces a small amount of vibration into the system.

3.3.2.5 AC Induction Motor

The AC induction motor works similarly to the other motor types except there is no permanent magnets. The AC power supply is split into its different phases and then used to create a revolving magnetic field in the stator. The changing magnetic fields inside create currents inside the rotor wires and these in turn create magnetic fields which interact with the original fields creating torque. The biggest issue with these motors is their size and that they use AC power, which on a truck may be hard to source. They do have excellent power outputs and a high torque output.

Table 3.3 – Advantages and disadvantages of motors.

Type of Motor	Advantages	Disadvantages
Brushed DC	Simple to control Cheap	High maintenance requirement (brushes) Short lifespan Sensor required for position feedback
Brushless DC	Long lifespan Low maintenance	High cost Complex control Sensor required for position feedback
Servo Motor	Inbuilt feedback Variety of types (DC Brushed/Brushless, AC)	Complex control Usually intended for small movements (<360°)
Stepper Motor	No need for feedback Low cost High holding torque	High loads = poor performance Complex Control May cause vibration in system.
AC Motor	High power and torque	No source for AC aboard a truck Slips in some circumstances

3.3.2.6 Feedback

If a motor is used that doesn't have the ability to know where it is like a stepper motor then an encoder is needed to accurately find the position of the motor and how far it has turned. The best sensor for this job would be an incremental encoder. It would require an initial calibration each time it is turned on to find its location but that is no different to a stepper motor.

3.4 Design Details

3.4.1 Sensors:

The most robust sensor to use would be a CCD camera. These come in a variety of sizes and shapes and can be easily acquired. Other options include a CMOS camera or some form of photodiode. From previous work done in similar fields of autonomous/assisted driving it has been found that the photodiodes may be good at picking up where the line is but they offer no look ahead for the line.

A CCD camera has several advantages over a CMOS one, it has a larger dynamic range and a better response meaning it will give a better representation of the overall image. The shutter is also operated independently to the pixels giving uniform shuttering. The CMOS camera has individual pixels which are responsible for sending their information which means that the pixels do not send the information about the same period of time. The difference might only be slight but in fast moving applications it can be a problem. CMOS overall is slightly faster at acquiring and sending off the images though [32].

Connection via FireWire will provide the most benefits as it is designed to transmit high volume traffic such as video. Initially one camera will be used for the purposes of proof of concept and testing. This should also suffice for low speeds and with straight lines that are not dashed. For higher speed applications it may be necessary for two cameras to be used; one aimed down and slightly forward and the other looking forward so that it is possible to plan ahead and forecast where the spray head may need to move to so it is in the path of the next dash. Comparisons carried out by [17] show that there is minimal difference when using greyscale compared to full colour images.

From existing applications of road marking detection it can be determined that no method is perfect and each has their own advantages and drawbacks. The biggest factor in choosing a method of line detection is finding segments of the line that are missing and only the radon transform really does this. As each process is not usually used in isolation and each application is unique more than one method can be combined to create a more robust algorithm.

3.4.2 Control:

The control of the position of the nozzle is very important as it will determine the success of this project. It is important that the movement is smooth and controlled so the painted line will appear straight and not jagged. It will need to get to where it is going without any sudden changes in speed or direction but it will need to get there as fast as possible.

The best method of controlling the position of the nozzle would be a Fuzzy Logic Controller. It doesn't require a mathematical model for this control method to work. It functions well for various different motor related applications and they don't need to be linear. The ability to have a varying load such as extra paint guns or a glass bead dispenser is taken into account and so doesn't need to be compensated for each time it is changed. Varying the change in set point also doesn't cause it to go unstable allowing for smoother control.

In comparisons done between Fuzzy Logic Control and PID control the fuzzy control performs equally well or better. In [4] where a DC motor's position is being controlled FLC had less overshoot for an equivalent rise time and a lower settling time. [7] also found that the settling time for the FLC was 30% shorter than PID. [9] tested a motor under varying loads and set point changes and found that FLC was able to consistently achieve control compared to the PID which struggled.

PID has some advantages and these are its robustness and simplicity of its design, a clear and predictable relationship between the tuning parameters and the response of the system and many techniques to help with tuning. It is also very common and thoroughly tested and certain tweaks can make it more responsive such as partial derivative action, trapezoidal integration and also some methods of automatic tuning.

3.4.3 Actuation

For this project an actuator is needed that is fast, accurate and be able to move a spray head of a paint gun. Having looked at linear actuators, linear slides, belt drives and pneumatic and hydraulic actuators, there is a large variety of actuator shapes, sizes and methods of operation. All types are able to be fast depending on the motor, gearbox or pressure of the fluid. All are also capable of moving the weight of the nozzle back and forth. Only the linear slide and belt drive are capable of carrying this weight unaided with only support at each end. The other methods require a moveable support and are only there to provide the linear motion. Keeping in mind that this structure will be attached to the side of a truck that will be driving on the road with other automobiles size is also an important factor. The overall lengths of each of these actuators will depend on the required stroke length. Linear, hydraulic and pneumatic actuators are all slightly

longer than twice the stroke of the actuator and so when attached to the side of a truck may extend too far. Linear slides on the other hand are almost the same length as their strokes and so pose much less of a problem in this regard. There is also some difficulty/complexity in the accurate positioning of a hydraulic or pneumatic actuator. It would require complex feedback and control methods to accurately determine the position of the nozzle head and to correct it in comparison to that of a motor.

Based on the length of each of the actuators, the ease of control and the ability to carry weight it can be determined that for the purposes of this project the belt drive as a method of linear action would be best suited. For actual applications the linear slide may be more applicable.

To drive the actuator a motor is needed that can be controlled precisely so as to ensure the correct location of the spray nozzle is achieved. Only servo and stepper motors are able to accurately assess how far they have turned without external sensors although servo motors usually have a DC brushed or brushless motor inside. The question is whether to use a stepper motor or a servo motor and if it is a servo motor then which out of brushed or brushless is better suited.

The major factor in comparing brushed to brushless DC motors is their maintenance requirements. The brushes in the brushed motor are always in contact with the commutator and so over time wear down due to friction and so have to be replaced frequently. This rubbing by the brushes also leads to particles being released within the motor and also a power loss due to the friction. The brushless doesn't suffer from any of these issues although it is significantly more complicated to control. It simply needs a DC voltage to be applied across two terminals and it will run where the brushless motor needs a complicated controller to make it turn. Even though a brushless DC motor may be more expensive and require more complicated control but due to the fact that it has a much higher reliability and longer lifespan it would be more applicable to this project.

The major difference between the brushless DC motor and the stepper motor is the feedback. Stepper motors do not need any form of a feedback loop to know where they are as their design allows for precise control over where they are. DC motors would require an external sensor such as an incremental encoder to accurately determine the position and will require an entire control loop to be set up. The stepper operates at a lower maximum speed due to the number of poles (50) in comparison to a brushless DC motor (4-8) as switching can only go so fast. The stepper does have a higher overall torque which decreases as the speed goes up. The inertia of a stepper

is much higher than that of a DC motor (>5 times), this means that the acceleration of the stepper is somewhat slower than that of the DC motor.

The speeds required for this project are not great but accuracy is the critical factor and with a stepper motor much of the issue of control loops is not needed. This will decrease the complexity of the overall system and give it a greater reliability. The added advantage of a stepper motor is the ability to lock it in place so that any knocks or sudden movements by a truck will not cause the nozzle unit to move to either side.

4 Algorithm Development

What is wanted is a line detection method that is fast, accurate and robust. This section goes into the theory behind the radon transform and why it works well. It then looks at the development of the algorithm for line detecting. The desired outcome is an algorithm that gives the required distance the nozzle will need to move.

4.1 Image Processing Algorithm

As a basic line detection method the radon transform will form the basis of the method being developed. The main reason for this is its ability to infer where the line should be if the line is worn away, dirty or covered. It is an integral projection which integrates an image along straight lines.

4.1.1 Pre-processing

The information required from the image is the painted line. This line differs from the background in that it has higher intensity values. To take advantage of this and to make the processing easier some pre-processing of the images should be done. This is to maximise the difference between the line and the background. A histogram for a typical image is shown in figure 4.1.

The higher peak corresponds to the background and the smaller peak to the line. What is wanted

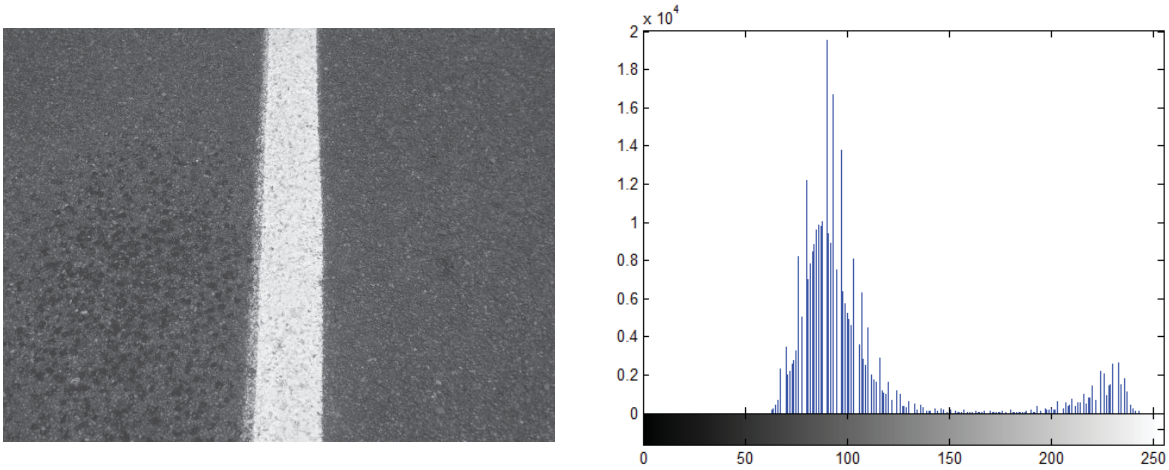

Figure 4.1 – Image of a line along with the corresponding intensity histogram.

is a larger distance between the peaks as this is where the algorithm will be looking for line edges. If the difference between the road and markings is easier to see then the processing does not have to be so complex. The simplest method is a contrast expansion where both edges of the histogram are moved to the minimum and maximum possible pixel values, stretching out the histogram and increasing the contrast, figure 4.2. This works well and is very fast (less than 5ms) and adapts depending on the image with no user input necessary.

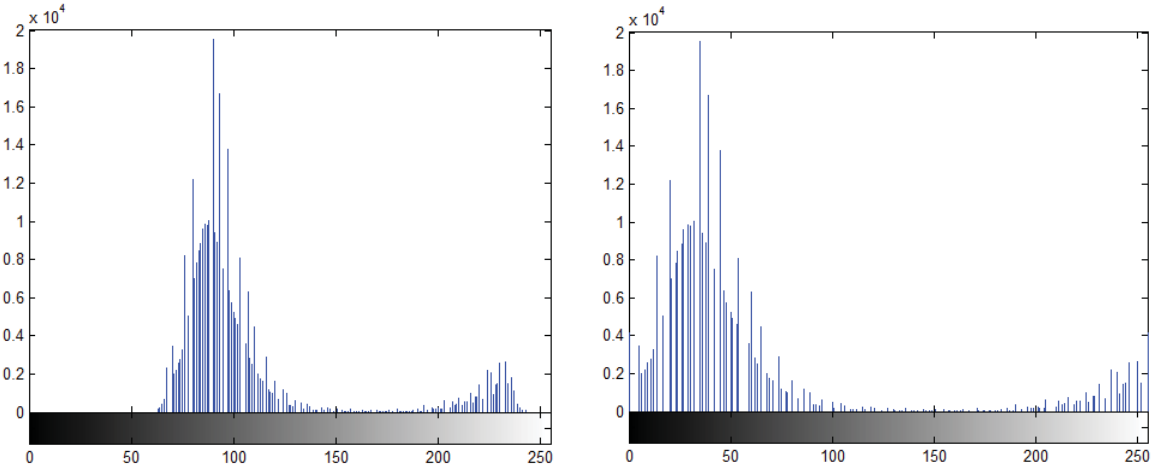

Figure 4.2 – Normal intensity histogram (left), Contrast expanded histogram of the same information (right).

A custom lookup table was designed to determine if any further improvement could be achieved over the contrast expansion. The function can be seen in figure 4.3. This is designed to maximise the distance between each of the peaks by moving everything above or below the peaks very close to either end.

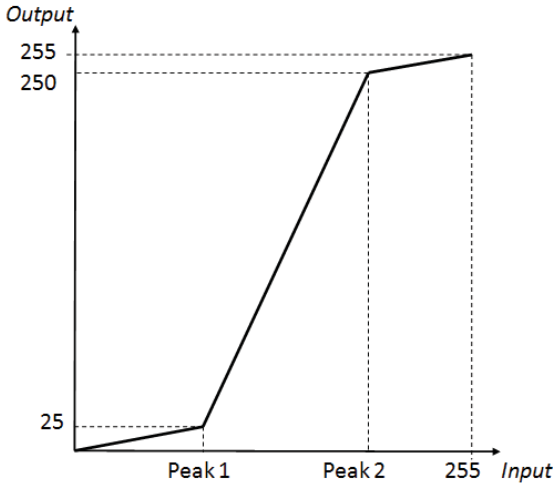

Figure 4.3 – Custom lookup table function.

Looking at results from the custom lookup table and comparing them to the histogram expansion, there is no significant difference other than the time taken to perform this process, figure 4.4. This process took very much longer (1 second) than the contrast expansion for a very similar result. This is partly due to Matlabs inbuilt optimisation of its own functions and partly to do with the algorithm method. This was not investigated further as the improvements gained from the optimisation of the lookup table would not be worth the development time required.

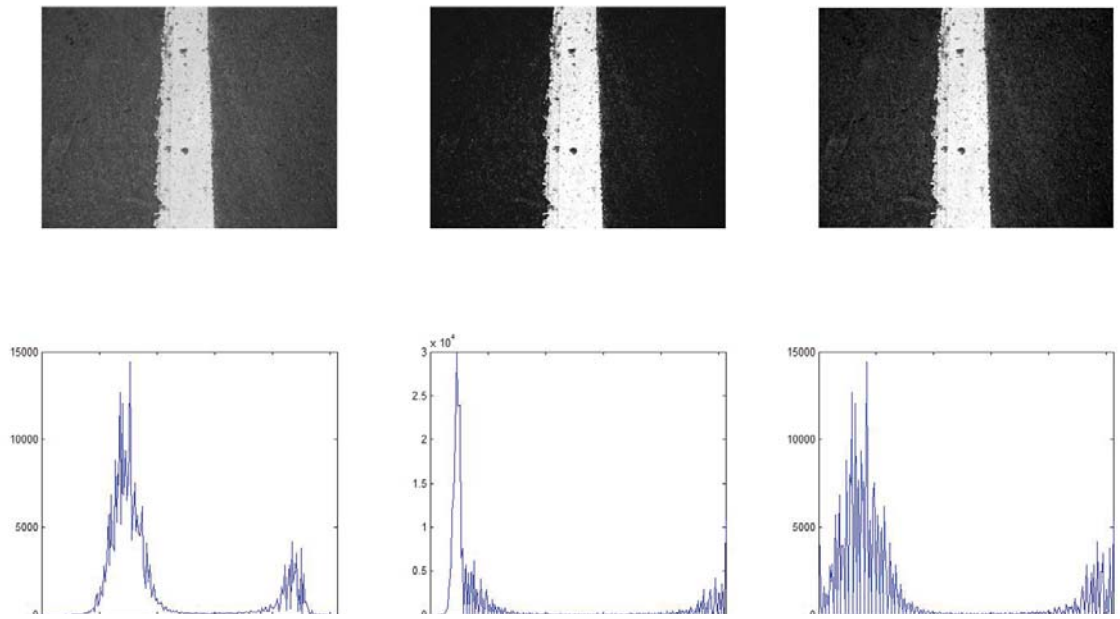

Figure 4.4 – This shows the original image and its histogram (left), the resulting image and histogram for the lookup table (centre), and the resulting image from the contrast expansion (right).

4.1.2 Radon Transform

The radon transform of a function $f(x, y)$ can be defined by [33]:

$$R(\rho, \theta) = \iint_{-\infty}^{\infty} f(x, y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy \qquad (2)$$

$$\text{where } \delta(r) = \begin{cases} \infty, & x = 0 \\ 0, & x \neq 0 \end{cases}.$$

As $\delta(r) = \infty$ it integrates to one, the reason for the inclusion of $\delta(x\cos\theta + y\sin\theta - \rho)$ in the definition is that it causes the integration only along the line defined by:

$$x\cos\theta + y\sin\theta - \rho = 0. \qquad (3)$$

An illustration of the radon transform is shown in figure 4.5.

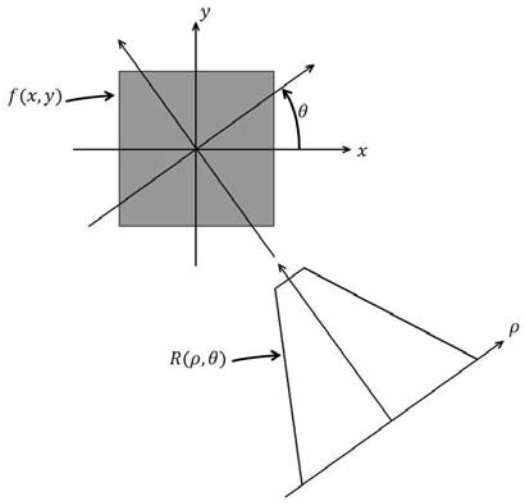

Figure 4.5 – Geometry of the radon transform [34].

The result of the radon transform $R(\rho, \theta)$ is a 2-D array of intensity values corresponding to integrations at various angles θ and offsets ρ which are relative to a parallel line passing through the centre of the image [20]. Based on this definition the result of a radon transform should show peaks where there are lines that are brighter than their surroundings and troughs for dark lines. This means that the detection of road markings can be simplified to detecting the peaks in the transform. Figure 4.6 shows a road marking image and the corresponding radon transform of that image. From the image the peak can be detected and then the width and angle of the line can be deduced.

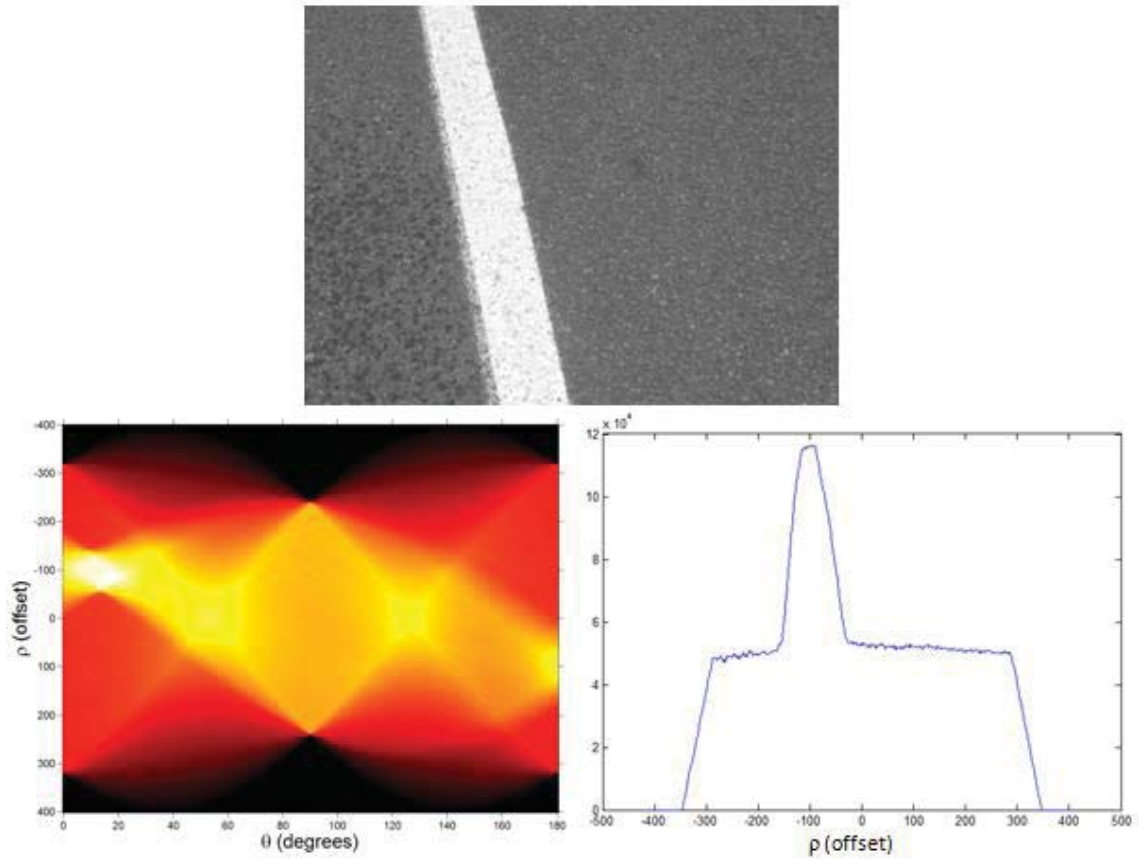

Figure 4.6 – The original image (top), the radon transform (left), and the radon transform at $\vartheta = 8°$.

4.1.3 Algorithm Development

4.1.3.1 Step 1 – Radon Development

Initial methods used a single radon transform over the entire acquired image, over a range of angles, to look, try and acquire the line in the fastest and most accurate way. As was noted in [20] there is the high possibility of local peaks giving false positives in this situation along with the possibility of getting a peak in the radon domain not in the actual angle of the line but in the angle that intersects the corners of the line which has a greater length than the actual line and hence a greater value when integrated. The suggested solution to this problem was to apply a filter (median, Gaussian etc) across the output of the radon transform to smooth out these peaks. If the filter is just smaller than the width of the line then it will take the surrounding pixels into account and the highest point will no longer be from corner to corner.

Once the point with the highest value has been found the data from that angle can be used to find the line edges. The edges of the line were found by using the characteristic shape of the output. The edges of the line were determined to be when the radon transform dropped below

75% of the maximum value in that angle of the transform. These points will give two different offsets which are the distances along the line away from the centre of the image that the line edges are.

The offsets can be translated into actual coordinates and then they can be extrapolated up and down to find the edges of the line at the top and bottom of the image. This can be seen below in figure 4.7, theta in this case is 13°.

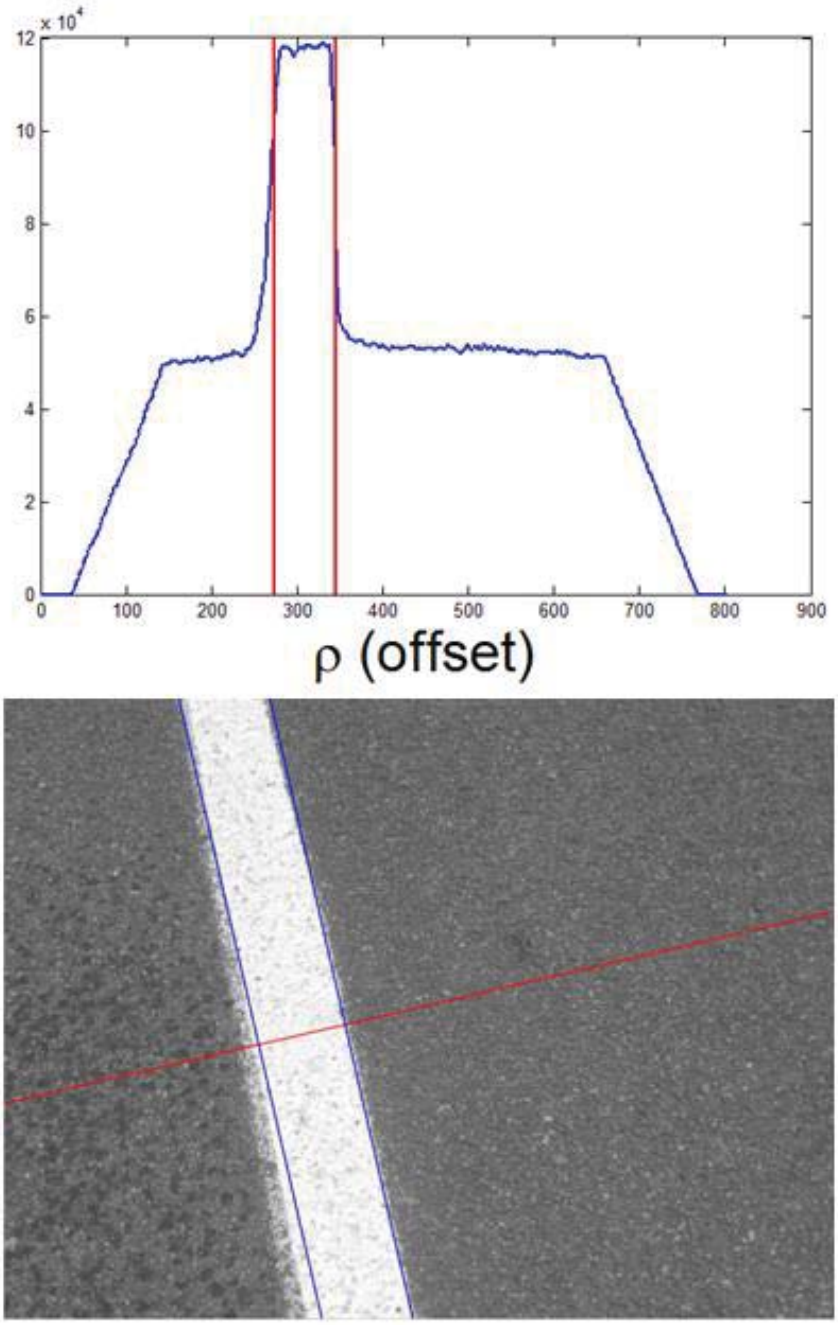

Figure 4.7– The radon transform at 13° along with the lines coresponding to the detected edges (top), and the angled line at which the radon transform took place along with the lines extrapolated from the points found in the radon transform (bottom).

This method worked reasonably well over a number of images that were tested. The radon transform successfully found the line and was able to determine the angle of the line to within two degrees. It is difficult to accurately determine the correct angle of the line due to perspective which causes the line to thin as it approaches the top of the image giving each side of the line a different angle.

One advantage of using the radon transform is the fact that it integrates over the whole line and so the whole width of the line should have very high values and to just look for a single high value is not taking advantage of this fact. Instead of just looking for the high point in the output of the radon transform which could occur over a variety of angles, the algorithm should look at the numbers of "high" values for each angle. The angle that has the highest number of these high values would be the best angle to approximate the line by as the radon transform will be widest when it is perfectly in line with the road markings. Figure 4.8 shows the number of points for each angle that are above 90% of the maximum value in the transform.

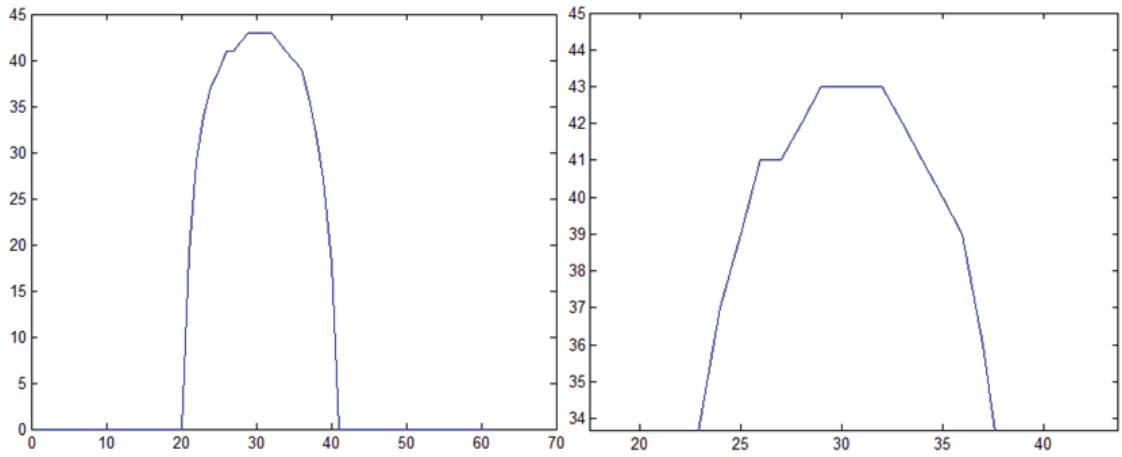

Figure 4.8 – Normal and zoomed in views of the number of points above the threshold over each angle in the radon transform.

As is expected it loosely follows a parabolic shape with the most likely angles at the top. It was seen that usually there was one angle that had highest number but then there were often one or two other angles on either side that were very similar in number. The number of high points would then drop off. The fact that these angles all have a similar result points to the issue with the perspective and the non-uniform thickness of the line. Averaging these angles would counter some of the perspective issues and give a balanced angle in between the two extremes. This gave a more accurate reading but still did not eliminate the issue with the perspective, figure 4.9.

Applying a single radon transform to an entire image doesn't work as well as anticipated. Problems include non-uniform width of the line over the image and also the time taken to process

the frame. This change in width occurs due to the perspective issue and the degree to which it occurs depends on the angle of the camera in relation to the road. The line appears fatter at the bottom of the image where it is closer to the camera than at the top. This means that the line edges are not actually parallel; they are in actuality getting closer together as the line moves up the image. To approximate them as parallel would not give the best representation of the line, either the bottom or top of the image would have incorrect placements or else one side of the line would be correct and the other would be off. It also takes a relatively long period of time to perform a radon transform of an entire image over a reasonably large range of angles. Matlab takes about 0.5 seconds per 640 x 480 greyscale image. This time is far too slow for the high frame rate needed for this project.

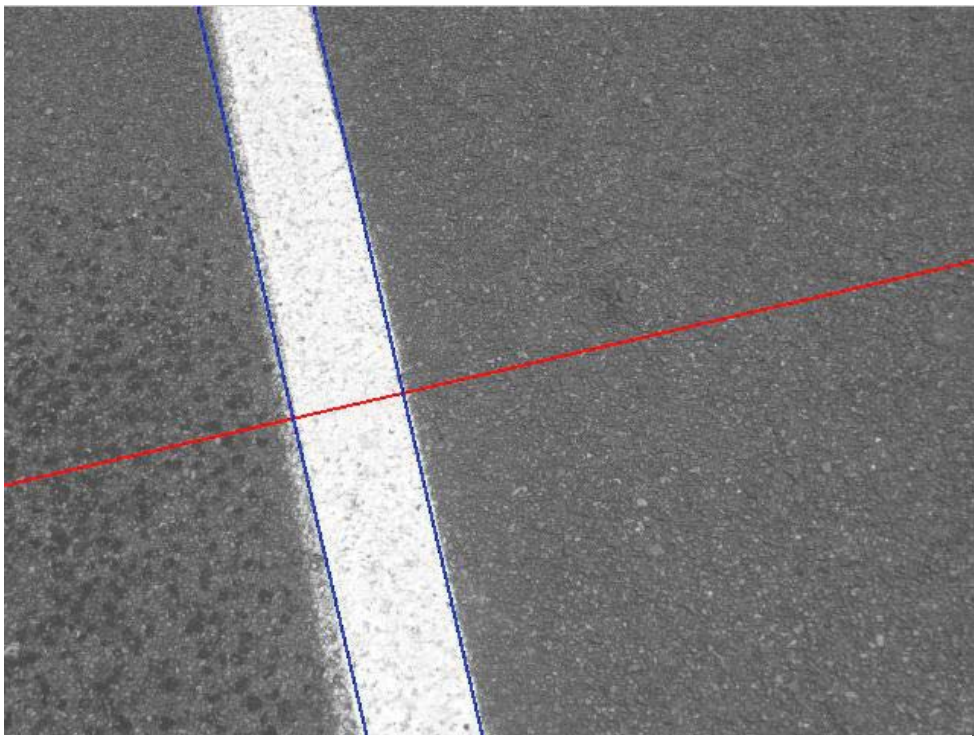

Figure 4.9 – Result from using the improved algorithm, angle found to be 12°.

One solution of this is to cut the image into horizontal bands and perform the radon transform on each one individually which will allow for a more accurate representation of the lines. This will not solve the issue of non parallel lines, but should improve accuracy as there will be a smaller change in width over the vertical distance. This could also potentially impact on the algorithms ability to detect significantly worn lines if there is less line to integrate over.

4.1.3.2 Step 2 – Algorithm Development

As the radon transform is a time and resource intensive transform it is best to use it as little as possible. By modelling the line on a set of quadrilaterals as proposed in [3] the perspective issue will be negligible and if the radon transform is only used to determine if a line is present between

the top and bottom of a quadrilateral then its computational time will be greatly reduced. By using it to check to see if the section is actually a line it will only be used over a single angle and a small portion of the image greatly reducing the time taken to process. Local thresholding will be used first as this scored well in a test of various line finding algorithms [17].

To form a basis for the overall algorithm, an algorithm was developed that looked at a single row of pixels every 20^{th} row. The process uses the average of entire row and finds the longest section that is above the average, it concludes that this is the section that is the line. It will link two parts together if they are only separated by a small trough to counter dark spots or variations in the colour of the paint. The only way this process detects whether a line is present or not is to check the length of the found "line" and if it is under a specified length then there is no line present. This algorithm works well when used with images of well marked lines and is a good initial step in finding the line. Some of the problems with this method include false positives such as detecting the beginning or end of the line to soon or late making the line wider than it actually is. This method only looks at individual pixels in correlation to their horizontal row where actually there is more significant correlation vertically (or near vertically) than horizontally. A better method of thresholding/edge detection is needed.

The next method trialled involved convolving the rows with a 1/5[1 1 1 1 1] matrix to smooth them out. Once some smoothing had been carried out the process of finding the edges involved moving along each of the rows calculating the average to the left and right of the current pixel. When these averages are plotted it can be seen that where the line is present there is an almost uniform increase/decrease in the average and the two values cross over. Based on that observation the averages were differentiated and the results were assessed. It looks for the section that had the longest period in the positives or negatives depending on the averaging side. Where these two meet and cross zero is defined as the edge of the line, figure 4.10.

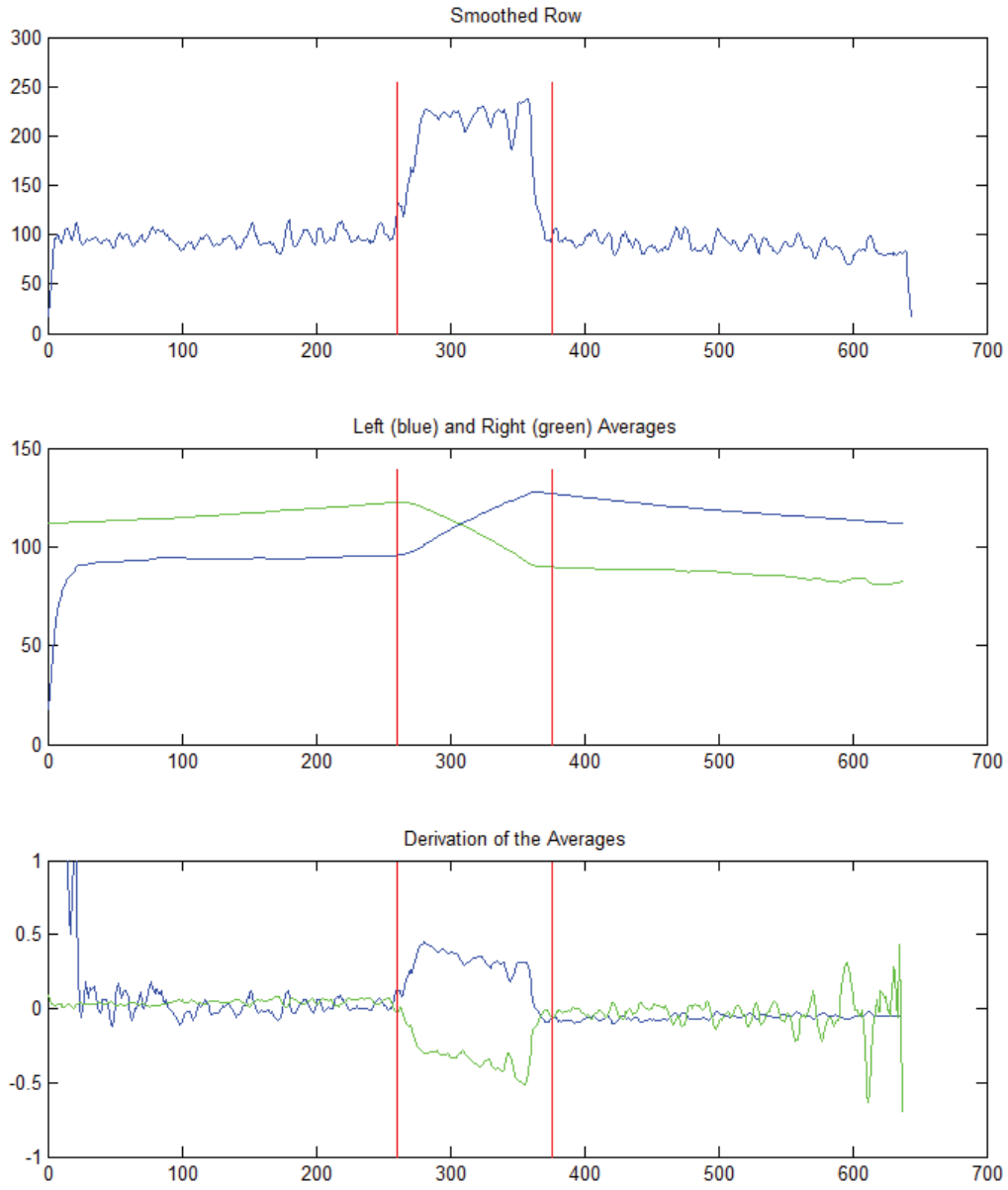

Figure 4.10 – Plots of the smoothed row, the averages for each side and the derivation of the averages. The red lines correspond to the found edges of the line.

This performed better than the previous method; it found the lines to a higher accuracy than a simple averaging. Differentiating the averaging signal did work to some extent although as the averaging value is changing in a way that is directly related to the change in values from one pixel to another, the output resulting from differentiation is very similar to the original. It is still not accurate enough as the resulting images seem to be finding the line too early and stopping it too late. It is still also resource intensive performing this on every few rows of the image which with a 640 x 480 pixel image means 24 times.

An elegant and simpler method was needed and was developed by combining parts of the two previous methods. By adding a set amount or threshold to each of the averages and then comparing them to the current pixel it was found that the edges could be detected much more

accurately, figure 4.11. A pixel is considered part of the line if the averages on either side are more than a specified amount less than that pixel. A line is considered present if there are enough of these pixels present together. This works very well in finding the line edges accurately and doesn't cause any false positives, the elimination if the derivation speeds the process up as well.

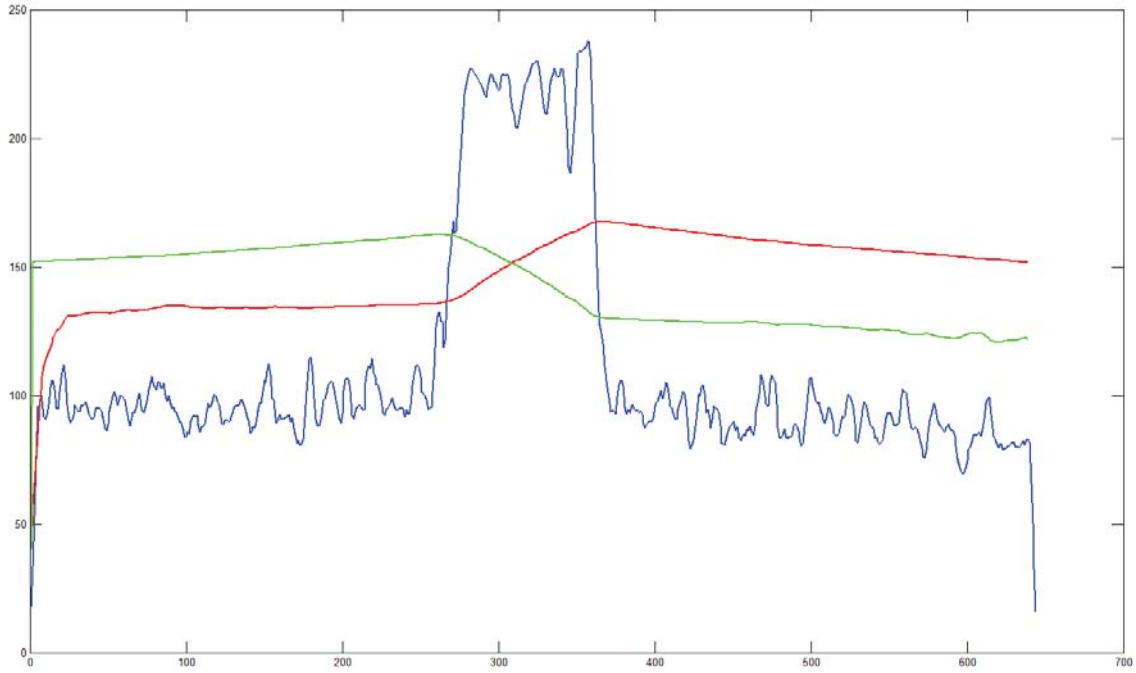

Figure 4.11 – Horizontal line with left (red) and right (green) averages and their threshold value plotted together.

Through testing on different images and rewriting of the algorithm various other improvements have been made. It was found that it was not necessary to have a smooth line if the averaging was to be used without the derivation and so the convolution was removed. It was found that instead of average the entire distance either side it was only necessary to average an amount twice the width of the line on either side, this depends on the value of the threshold used. This makes the processing faster still.

Regions of interest were implemented as well. Utilising the similarity between frames in a video, only the position of the previous line plus a certain distance either side were even looked at in the algorithm. This gives a much smaller distance to be processing over without the chance of missing the line. If an isolated image or the first frame in a video is looked at and there is no ROI to go from then the entire width is scanned and nothing is lost but it is sped up when there is.

The process was also changed in how the algorithm deals with multiple lines. Initially it found the longest segment and connected up any small sections separated via a trough. The algorithm now tries to connect sections together if they are close enough, if not, and there are still multiple

sections of line and they still meet the minimum and maximum width requirements then they are passed back as both being part of the line.

4.1.3.3 Step 3 - Quadrilateral Creation and Main Process

The basis of the process used in [3] can be used for this application but the view of the lines is quite different. In [3] the lines are viewed from above a vehicle and multiple lanes are in view for a long distance. This project is looking more at a closer detail image of the road. The sections that are being looked at in this algorithm will be larger and may take more time to process.

With tweaking and some development a line detection process was developed using the local thresholding process previously investigated. The algorithm will start at the bottom of the image and work its way up as the most important information is toward the bottom. It will look at a row and, using the line finding algorithm, decide if a line is present. If it is it will move up a set number of rows corresponding to the desired quadrilateral height and look again at the row. If a line is also found here then this is considered a line segment or quadrilateral. This process is repeated until the top of the image is reached. If after the move up no line is found then it will make smaller steps back toward the initial starting row checking each time until either a line is found or it reaches the row it first moved from. If this happens, it means that there is no line within that space and so the algorithm will begin looking at rows again from above the highest row previously looked at and start a forward search from there. The algorithm is shown in figure 4.12.

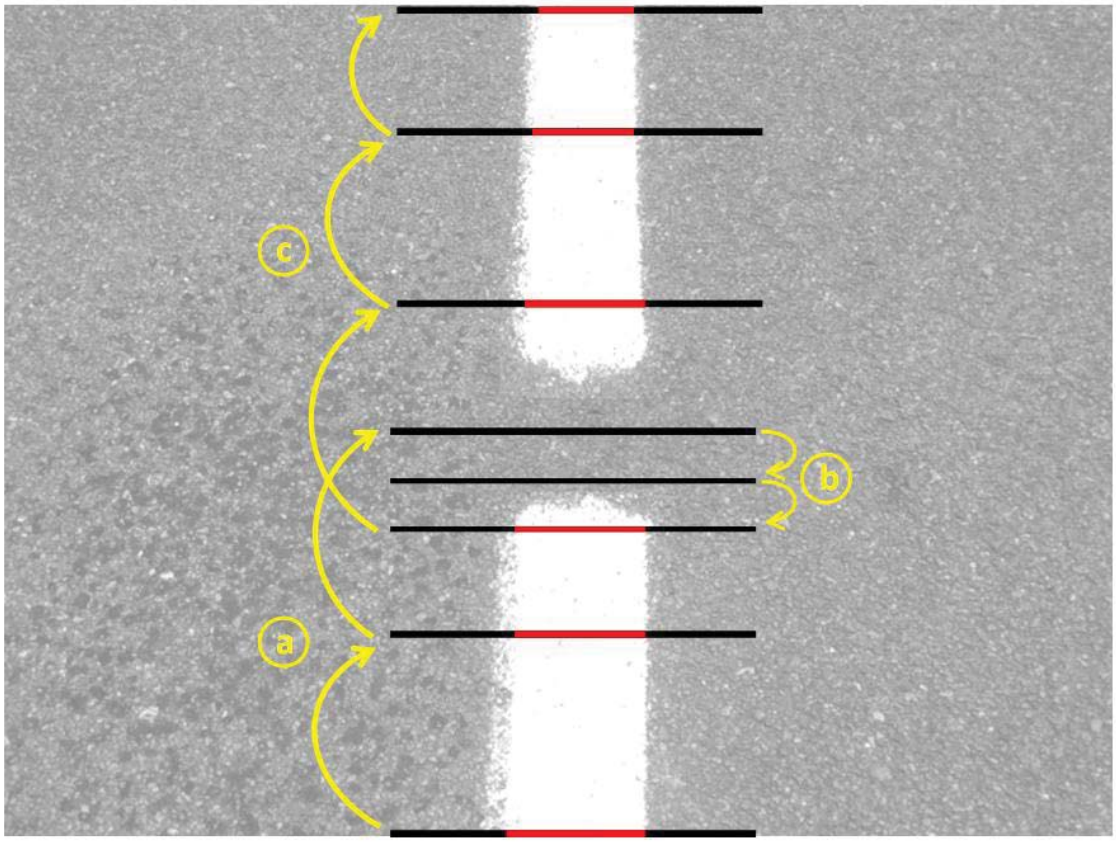

Figure 4.12 – The algorithms progress up an image. (a) Moving up in large steps, (b) when no line is found take small steps backward, (c) once a line is found continue with the large steps until the top of the image.

Once the image has been processed and the line/quadrilaterals have been found the region of interest can be calculated. The region of interest is an area to each side of the image that will be searched the next time if in a video/camera setting is used. It is calculated based on the width of the line and is around one width of the line on each side of the current line position. The ROI is stored as a matrix of two gradient and zero crossing values which create the equation that defines the lines for each of the two sides of the ROI. The ROI lines can be seen in figure 4.13.

To give a better check of whether a line is present or not when a quadrilateral has been formed, it is put through a radon transform. The process followed is: first the quadrilateral and an area each side of it are cut out of the image. Any pixels that are over one line width away from the line are set to zero so as to make the radon transform clearer. The radon transform is then taken of the resulting segment of the image at the angle calculated from the line. The resulting output is split into three segments; left of the line, right of the line and the line. The average of these segments is calculated and a ratio is generated comparing the line value and the highest of either the left or right segments. If this ratio is significant enough then it recognises that a line is present.

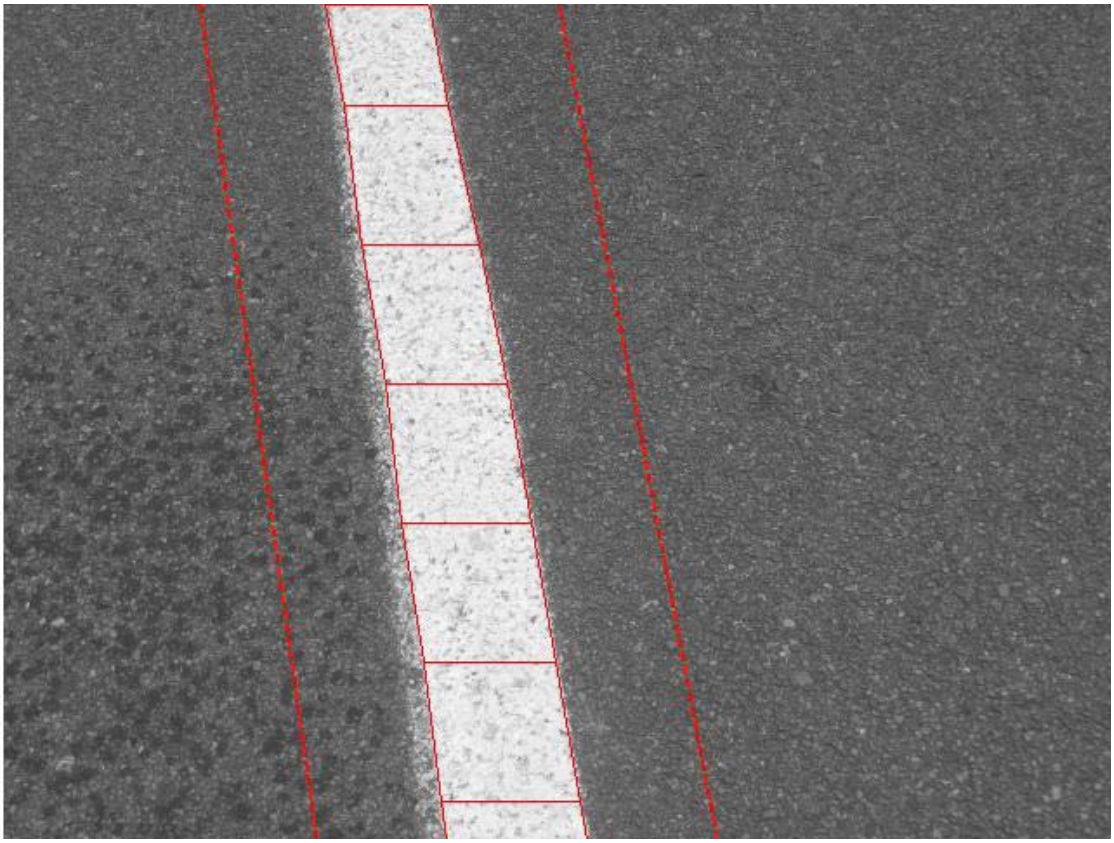

Figure 4.13 – A line modelled by quadrilaterals with the edge regions of interest to either side marked out.

The final output of the line is a set of quadrilaterals that describe the line. From this the current deviation of the nozzle can be calculated along with the intended position of the line when the image is next taken. Looking at the bottom edge of the image will correspond to where the line is now and after the image processing has happened the truck will have moved forward slightly meaning that the bottom of the image is no longer where the nozzle is after processing. Finding the entire line allows the algorithm to select any point on the line to take these measurements from to eliminate this delay. A simplified flowchart of the algorithm can be seen in figure 4.14 and the source code can be found in the appendix.

Things within the algorithm need to be developed to cope with a variety of different scenarios when it comes to line detection. These could be any number of things including the line only making its way half way up the image or vice versa, multiple lines, diverging lines, line thinning in width, sections of the line missing giving a false positive for a gap in the line.

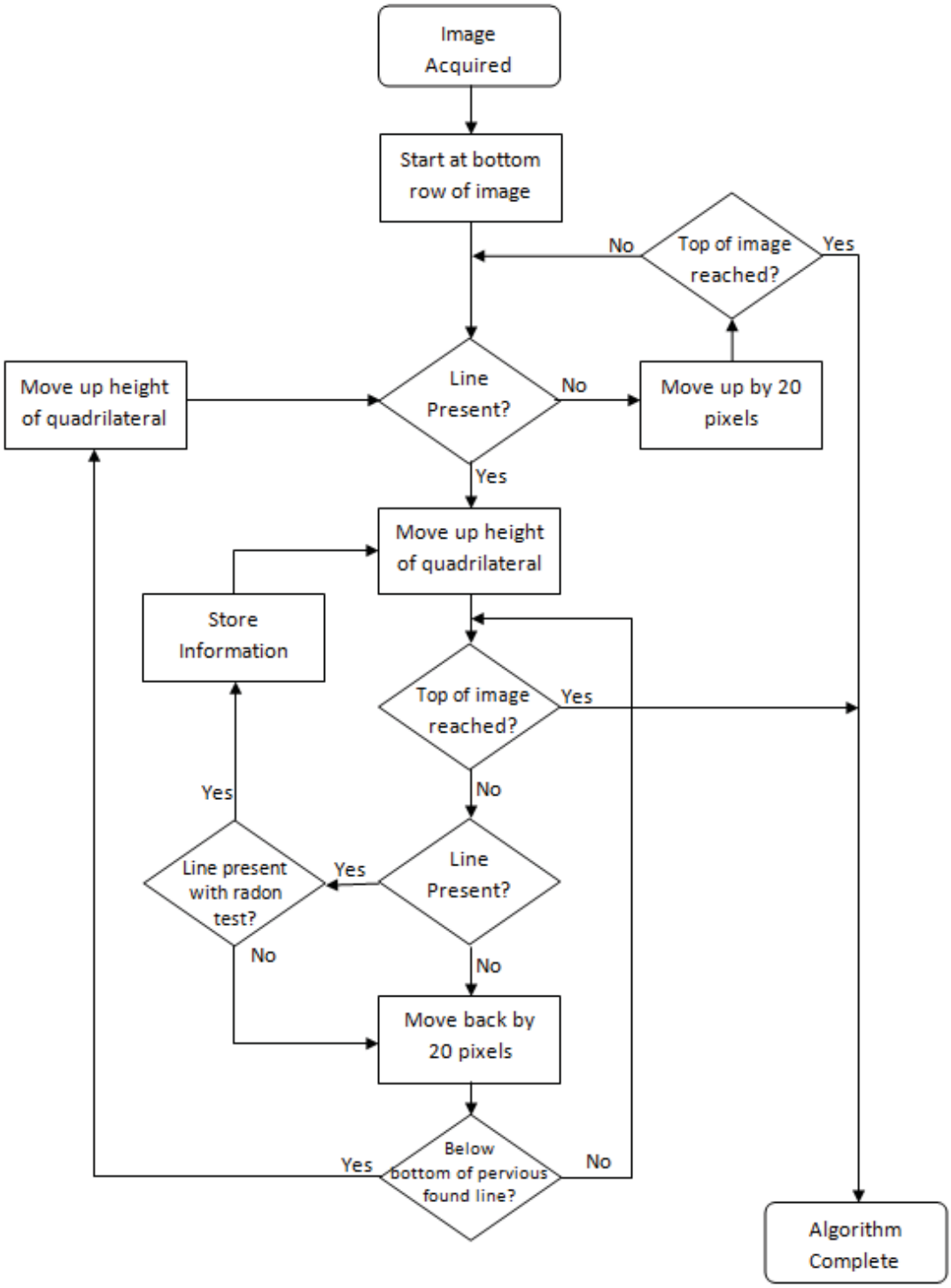

Figure 4.14 – Basic Algorithm Flowchart

Various additions to the line finding algorithm have been implemented as well which do slow down image processing speeds but give a greater approximation of the actual position of the line and can help deal with different issues that may arise when viewing the lines. As the image is scanned every 20^{th} row of pixels there is a chance that a line may start between the scan lines in this case the approximation of where the line starts will be incorrect and would cause the line spraying to start late. To remedy this, a side radon transform is taken of the line from part way up the line to where it is anticipated that the line would meet the bottom of the image. From this transform it is relatively easy to determine where the start of the line is even if it has a rough or odd shaped beginning. This technique can also be used if it appears if there is a segment missing from the line but instead the line is only dirty or weirdly shaped. This may occur with an increased frequency if the heights of the quadrilaterals are too low. Making them smaller might give a better representation of the line in curved situations but it reduces the effectiveness of the radon transform as a process for determining whether a line is present. Faded or dirty lines may register as no line present if the height is too small.

To counter the change in line width over the image and the effect that this has on false positive detection etc. a function similar to the one described in [17] was created. This allows a preset width at the top and bottom of the image to be set and the thickness at any line is then calculated based on either of those two values.

As the information required from the algorithm is obtainable from the bottom of the image, it is not necessary to perform the full algorithm and model the entire line, if all that is required is in the bottom section. How far up the image the algorithm will need to go will depend on the speed of marking, by decreasing the amount the algorithm will need to model, the algorithm will be sped up. The algorithm is not limited to a lower section of the image; rather it will stop looking for the line once it finds a section of line longer than a predetermined length [36]. In this way dashed lines will, still be picked up ahead of time and the nozzle will be in the correct position when remarking should begin.

4.2 Conclusion

The image processing algorithm takes an image obtained from a camera, pre-processes it before determining the location of the painted line. The result is a set of values corresponding to the quadrilaterals that make up the line. From this information and the known width of the line, a distance in mm can be found that corresponds to the distance from the centre of the image. This measurement can be taken from anywhere, height wise, in the image. This is so the speed of the vehicle can be taken into account when deciding where to retrieve the measurement from.

5 Control and Actuation Development

What is wanted is a process to retrieve the information sent from the line finding section and utilise that to correctly position the road marking equipment over the line. This will involve choosing an actuator and motor specifications and designing the control of the motor. The system will be modelled to see how well the control works.

5.1 Actuation

Before the control can be developed details of the actuation need to be determined. Based on the length of each of the potential actuators, the ease of control and the ability to carry weight it was decided that for the purposes of this project the belt drive as a method of linear action would be best suited.

To drive the actuator a motor is needed that can be controlled precisely so as to ensure the correct location of the spray nozzle is achieved. The speeds required for this project are not great but accuracy is a critical factor and with a stepper motor the requirement for a control loop is not there. This will decrease the complexity of the overall system and give it greater reliability.

Design and development of a method of actuation is not necessary as there is a vast array of actuators in the market today that can satisfy almost any application requirements and can be sourced easily. The travel of the actuator needs to be chosen, it cannot be too large as the truck cannot be too wide but it also should not be too small as the advantages of this system would not be realised. 400 millimetres was chosen as it is not too wide but also gives the driver some space to work with.

The size of the stepper motor to be used for this application is of some importance. If the motor does not have enough torque then the actuator cannot move properly and the accuracy will not be reliable. Also if it is too large and overpowered it may draw too much current, so a balance is required. It also needs to have a certain holding torque so as to keep the nozzle in the required location despite any bumps or jarring that may occur through normal road use.

The apparatus that will be held on the actuator will vary depending on the application and setup of the individual companies. It will always include a spray gun and a glass bead dispenser, also the related hoses and supporting structure. As the actuator is moving in a horizontal direction and runs on a rail most of the weight of the guns and hoses will be supported by the rail. The main force to overcome will be the static friction of the system as this is usually higher than the kinetic

friction in any situation. The fact that stepper motors have higher torques at lower speeds means that they are ideally suited for this application.

The choice of stepper motor depends on a number of factors and these are investigated below.

The weight of the apparatus being moved will not exceed 1 kg and so the motor will need to have enough torque to move this weight when required. The actuator being used will take the weight of the apparatus and so all the motor needs to do is propel it back and forth along a horizontal track with no vertical movement. Looking at an appropriate actuator specification, the unloaded starting torque required is in the region of 0.2 Nm and will be slightly increased when loaded.

The voltage of the motor is also a consideration. Being that it is located on a vehicle and most vehicle systems run on 12 volts, it makes sense to have the motor running on the same voltage.

The stepping mode, along with the gearbox, are also factors that need to be taken into account. These two are closely related as they deal with how far the actuator will move when a step is taken. A good step size is needed but it is also important to have it moving at a reasonable speed while maintaining the required torque. Half stepping and micro stepping reduce the torque by around 30% and so are not ideal to use unless absolutely necessary. The actuator chosen will move 85mm linearly with each revolution and this value will be similar for other belt driven stages. This equates to 0.425 mm per step without a gearbox. If there is a required speed of 150 mm per second then without a gearbox the stepper motor need to be able to run at around 105 RPM. This is a very low speed.

0.425mm may seem like a large distance but when dealing with tolerances of -5 +10 mm it is not large. It would only be just visible to the human eye if it was drawn as a line but as it is being sprayed as paint onto a road with a very rough surface this step size could be small enough. A gearbox would decrease the size of the step distance at the cost of an increase in the required speed of operation. 150 mm per second is not large and with a small gearbox of 3.6:1 the speed would not need to be larger than 380 RPM. This is still relatively low for a stepper motor and means that the motor will still put out a decent amount of torque, figure 5.1. The addition of a gearbox will also change the size of the step to 0.118 mm which, for accuracy, is smaller and hence better.

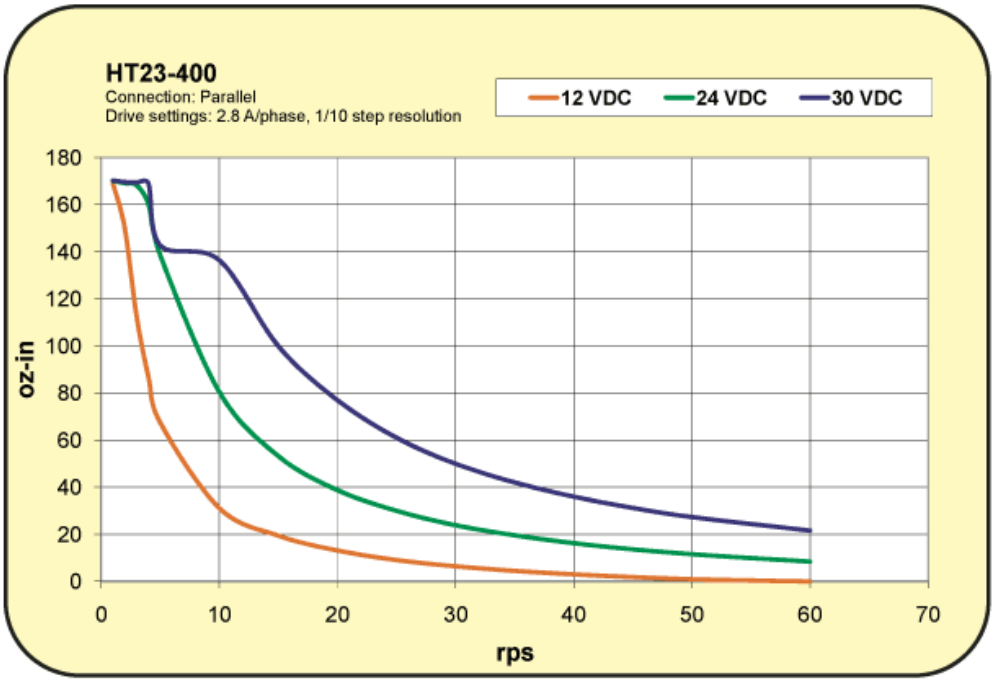

Figure 5.1 – Example of a stepper motor torque-speed curve [35]

A 12 volt stepper motor with a maximum torque rating of 0.5 N, connected to a gearbox with a ratio of 3.6:1 and attached to a belt driven actuator will be adequate for the purposes of this application.

5.2 Control

The best method of controlling the position of the nozzle is a Fuzzy Logic Controller. It doesn't require a mathematical model for this control method to work. It functions well for various different motor related applications and they don't need to be linear. The ability to have a varying load such as extra paint nozzles or a glass bead dispenser is taken into account and so the model does not need to be compensated for each time the load is changed. Varying the change in set point will also not cause it to go unstable allowing for smoother control. A model will be used to simulate the process to fine tune the control. The control that is required will ramp smoothly up to the maximum desired speed before slowing down again as it approaches the set point. Overshoot is not desired at all, as it is preferred that it takes a longer time reaching the set point. It does not need to be exactly over the set point because when it gets close it is dealing in tenths of a millimetre and the human eye will not be able to pick up any error under two millimetres.

5.2.1 Model Creation

The first step in creating a method of control is to build a model of the desired system to be controlled. Simulink was used for this purpose. Two models were created, the first was a very

simple approximation of a stepper motor and driver and the second used the provided hybrid stepper motor model for more accurate modelling. As the actual stepper motor model models voltages, currents and torque output along with the rotational output it runs very slowly. For this reason the simple stepper motor model will be used for creating the fuzzy logic control and then the actual model will be used for testing and analysis.

There are three parts to the model: the image processing signal, the computer control and the mechanical system.

The mechanical system consists of the stepper motor and driver, gearbox and actuator. The inputs are what is expected for a driver and consist of an on, a direction and a wave signal. The on signal decides whether the motor should turn irrespective of the other inputs. The direction specifies which way the motor should step and the motor will step on each rising edge of the wave that is sent into wave. This is all translated into a degree value corresponding to the rotation of the motor. That goes through a gearbox and then is translated into linear movement by the actuator, the value of which is calculated by equation 4. This section can be seen in figure 5.2.

$$1° = \frac{85mm}{360°} = 0.236mm \qquad (4)$$

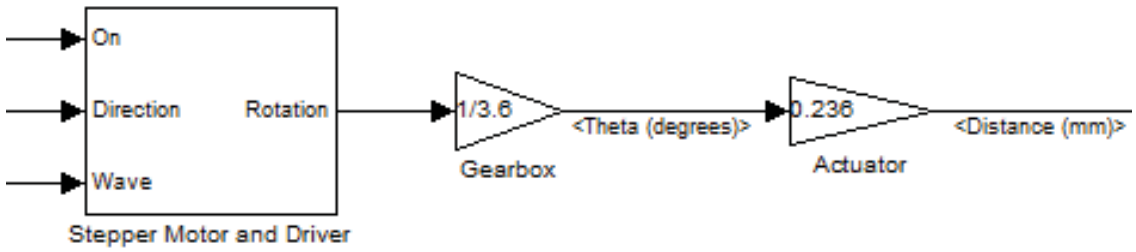

Figure 5.2 – Model of the mechanical system.

The computer control consists of the fuzzy logic controller and the computer output. The fuzzy logic controller has two inputs, the error from the required location and the current speed of the motor. The current speed is required as there can be issues with the stepper motor if it is not accelerated and decelerated. It seems like they are able to step at any speed straight away but this can lead to miss steps which is very undesirable in this situation. As the stepper motors position is also known from the signals sent to it, the error can be calculated at the same time the signals are being sent to the motor driver. The output from the image processing algorithm is a distance that the actuator needs to move. These changes are added to the current position and become the new set point. It may seem like the current position is added to the error only to be

taken away again just before going into the fuzzy logic controller. This is not redundant as the error is only updated every time an image is processed, whereas the distance travelled is updated in real-time and so the error will change between image processing updates. The setup can be seen in figure 5.3.

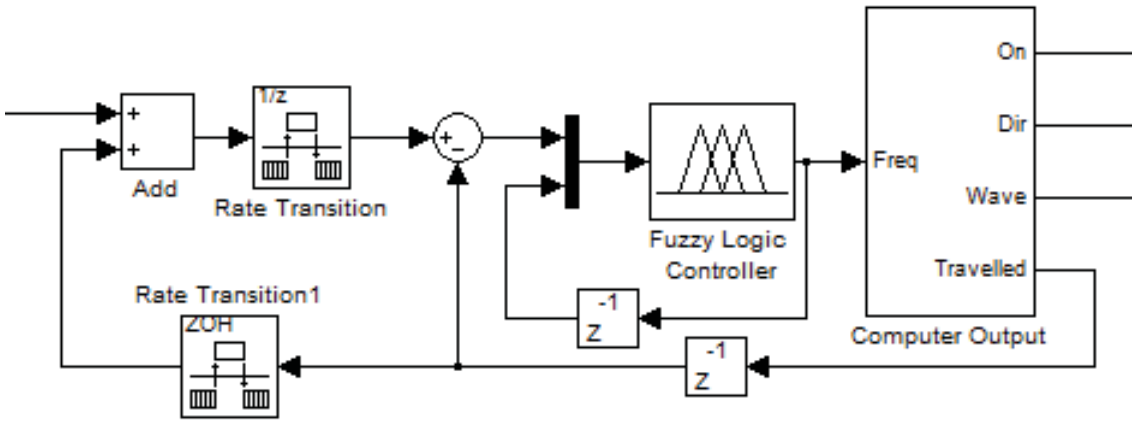

Figure 5.3 – Model of the computer control.

A random number generator was used to simulate the image processing algorithm, giving distances to move every 0.03 seconds. This is 33 times per second and is more than the speed required for this project. If the model can cope with this then the actual outputs should not be a problem. The outputs form a normally distributed random signal with its mean at 0 and a variance of 30. The entire model can be seen in figure 5.4.

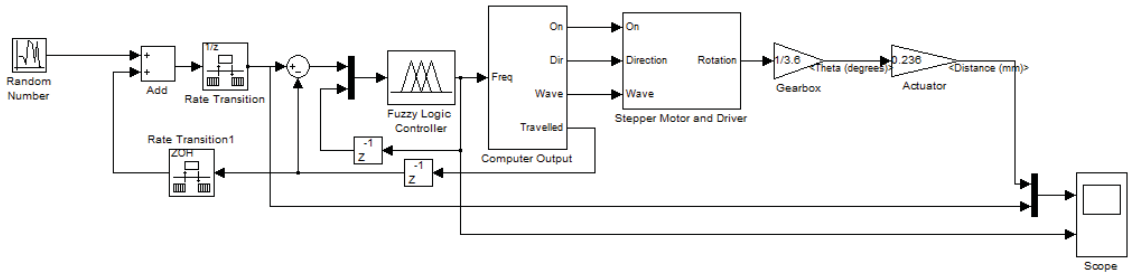

Figure 5.4 – Entire simple motor control model.

The full model can be seen in figure 5.5. The only differences are the driver and motor approximation. The outputs are also more detailed and include the voltage in the windings, the current in the windings, the torque output, the speed of the motor, and the total rotation of the motor. There is also the ability to put a load torque on the motor to see what effect that has on its operation.

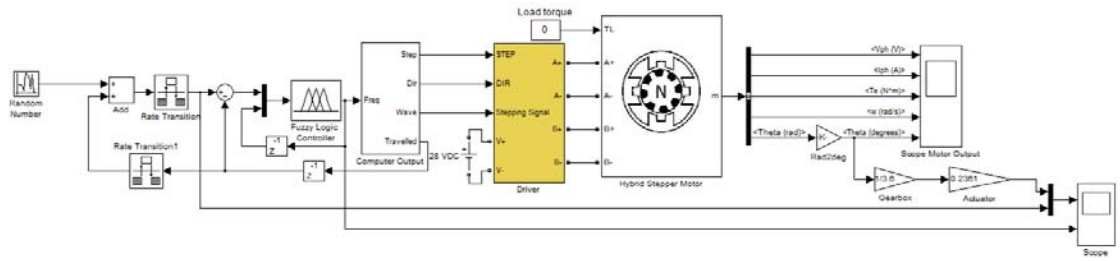

Figure 5.5 – Full motor control model.

5.2.2 Fuzzy Logic

Fuzzy logic uses approximations and inferences similar to how the human brain works to decide the outcomes for certain situations as opposed to specific values and binary information. Data that may be hard to quantify and categorise can be used to make decisions and affect outcomes that would be difficult to solve otherwise. The process followed in obtaining an outcome is as follows.

First absolute inputs are quantified into subjective terms such as far, close or very close. The value can be in more than one category as something that is large may also be very large. The designer defines what input values correspond to in the subjective language and how much that value corresponds to that description, these are the membership functions, figure 5.6. Once the inputs have been fuzzified the correct subjective outputs are inferred from a table or set of statements based on the input values that were created by the designer. The statements are logical statements such as “IF large AND big THEN fast.” Once there is a subjective output, it is possible through the reverse process to defuzzify it back into an absolute output. To choose its position, along with the defuzzification table there is the need for a mathematical operation, as the fuzzy region is not specific. The two most common of these operations are centroid and height methods. The centroid method defines the value to be the centre of mass of all the resulting outputs while the height method takes the value of the biggest contributor as the result. The fuzzification, defuzzification and inference tables are all chosen by the designer based on knowledge of the system and its application.

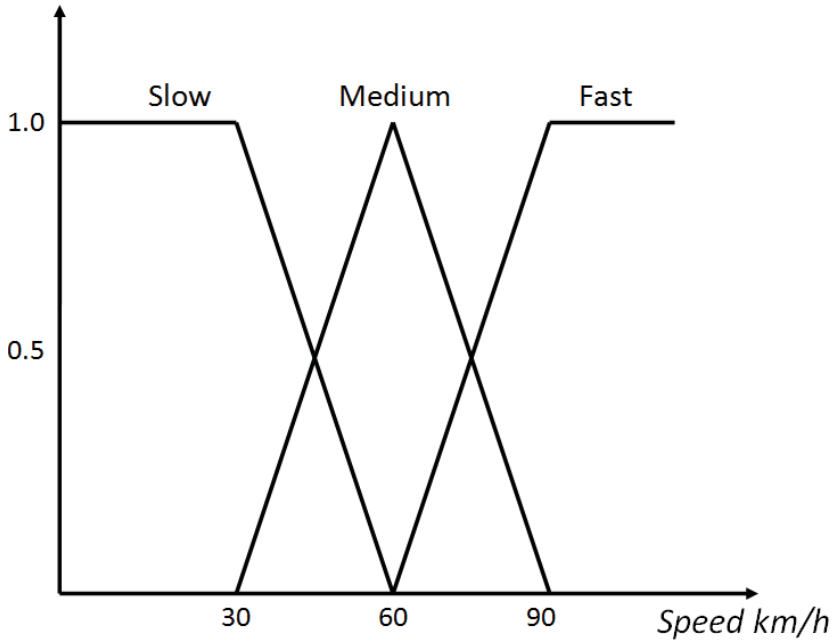

Figure 5.6 – An inference table showing the subjective terms and what they correspond to relating to speed. Values can be in more than one section, for example if the speed is 75 km/h then it can be partly medium and partly fast.

5.2.2.1 Fuzzy Logic for this system

The fuzzy logic controller has two inputs, the error and the current frequency of steps. The output is a desired frequency of steps. Traditionally the second input is a rate of change but when using a stepper motor the rate of change should always be directly proportional to the frequency of steps and so this was used instead. The error is calculated as

$$e(k) = D(k) - P(k) \qquad (5)$$

where k is a sample point, P is the actuator position and d is the desired position. The two inputs were initially converted into fuzzy variables using the membership functions shown in figure 5.7. These initial membership functions were taken and modified from [27].The fuzzy sets were chosen to have five categories each giving a total of 25 control rules. The actual range of the error extends from -400 to +400 mm and the step frequency from -2000 to 2000 Hz.

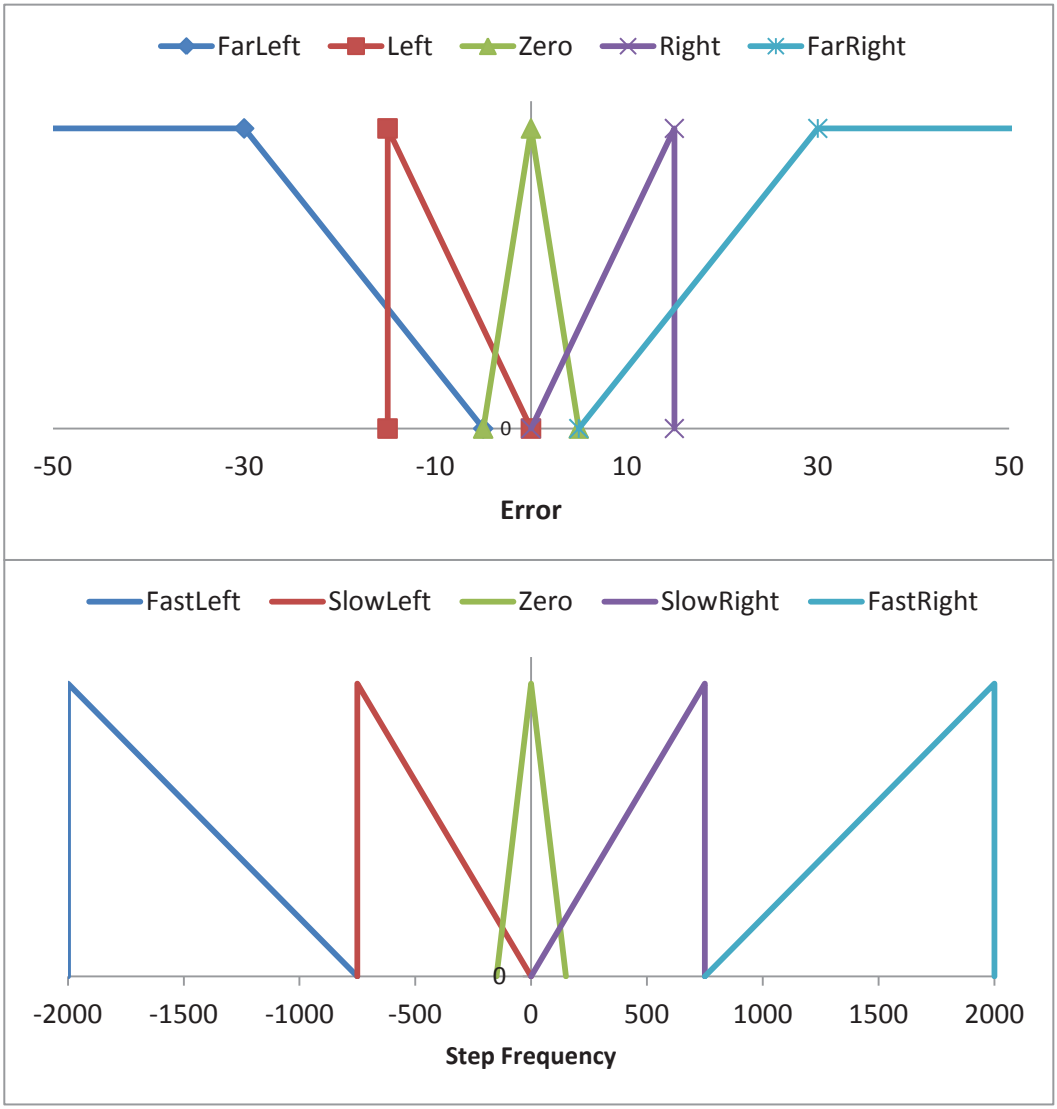

Figure 5.7 – Membership functions.

The rule evaluation occurs based on the results of the fuzzyification. An inference table was created based on the knowledge of how the system operates and programmer experience, table 5.1. The output shares the same membership function as the step frequency. The position along the output was calculated using the centroid method as it takes into account all the results.

Table 5.1 – Inference table for deciding the speed of the motor.

Step Frequency \ Error	FarLeft	Left	Zero	Right	FarRight
FastLeft	FastLeft	FastLeft	SlowLeft	SlowLeft	SlowLeft
SlowLeft	FastLeft	FastLeft	Zero	Zero	Zero
Zero	SlowLeft	SlowLeft	Zero	SlowRight	SlowRight
SlowRight	Zero	Zero	Zero	FastRight	FastRight
FastRight	SlowRight	SlowRight	SlowRight	FastRight	FastRight

Figure 5.8 shows the results from the initial fuzzy logic controller output. The biggest issue with this is the sudden changes in direction of the motor. The actual position zigzags from one set point to another which will be visible to the naked eye and not look good. The other problem with this is that the motor is jumping from very fast in one direction to very fast in another. This is not possible and is not desired either, as it can cause missed steps and ruin the motor. This is despite the rules stating if going fast in one direction slow down, stop then speed up in the other direction. This is occurring as the fuzzy logic controller is updating itself very fast, much faster than the stepper motor is stepping. All of these rules are being observed but they are going through the changes in frequency quicker than the motor can step. The easiest way to stop this is to slow down the updating of the frequency. Whatever is coming out of the fuzzy logic controller is fed straight back into it 0.00005 seconds later. By slowing down the sampling of this feedback to every 0.005 seconds it allows the stepper motor to operate at the new speed for a period of time before changing again.

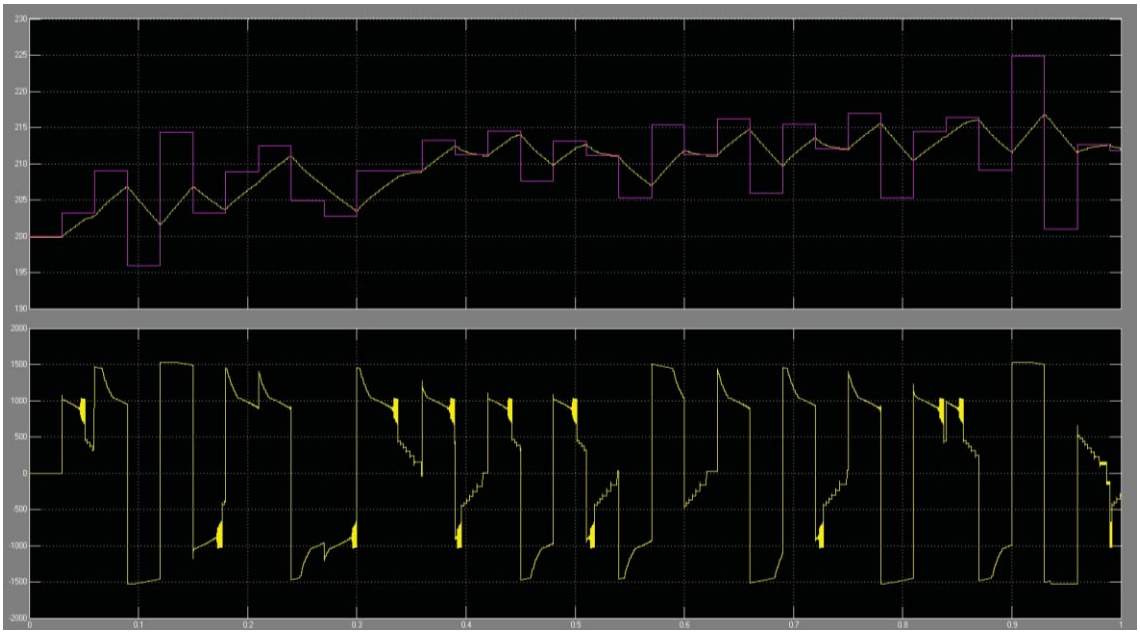

Figure 5.8 – Results from initial Fuzzy Logic Controller. Top is the set point (purple) and actual position (yellow). Bottom is the frequency output of the fuzzy logic controller.

Figure 5.9 shows the output with the previous issue resolved to an extent. The output is now much smoother but the frequency output is still somewhat stepped. What is wanted is a smooth increase, and while the steps here are an improvement they are not smooth and were not the result of a change in control rather a slowing in the updating of the speed. This is not ideal and limits the speed at which the controller can react, so it is better to change the fuzzy control. This will need to be changed in the membership functions and rule sets of the fuzzy logic controller.

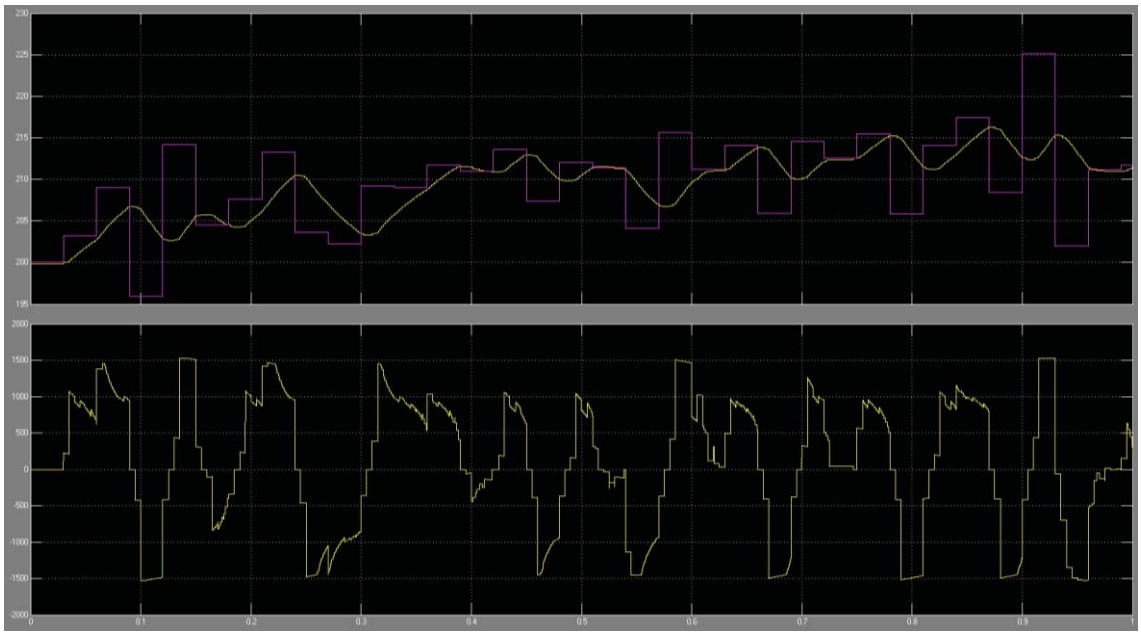

Figure 5.9 – Improved fuzzy logic controller with a smoothed output.

There is some concern over the ability of the actuator to follow the line properly. The speed it is running at when going its fastest is 1500 steps per second, this equates to 177mm per second and is faster than was initially decided upon. This problem was looked into and when investigated it was found that the errors put out by the random number generator to simulate line errors were too large. It is very unlikely when looking at the road line 30 times per second that the line will have moved 15mm within 0.03 of a second. It was decided the variance should be changed to 10 to make this simulation more realistic.

Figure 5.10 shows the resulting output after changing the membership functions of the two inputs and the output to make them overlap more, figure 5.11 and 15.12. The slow sampling of the frequency was also removed to allow the controller to determine the pace at which the output was changed. This shows an improvement in the output without the need for an increase in complexity.

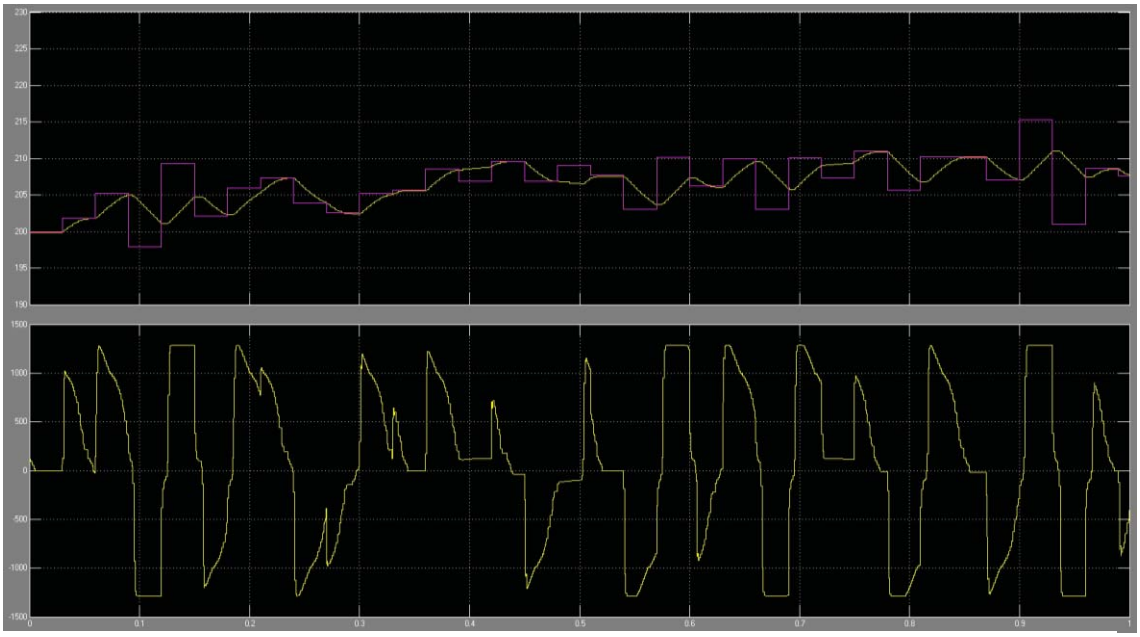

Figure 5.10 – Fuzzy logic controller with improved membership functions.

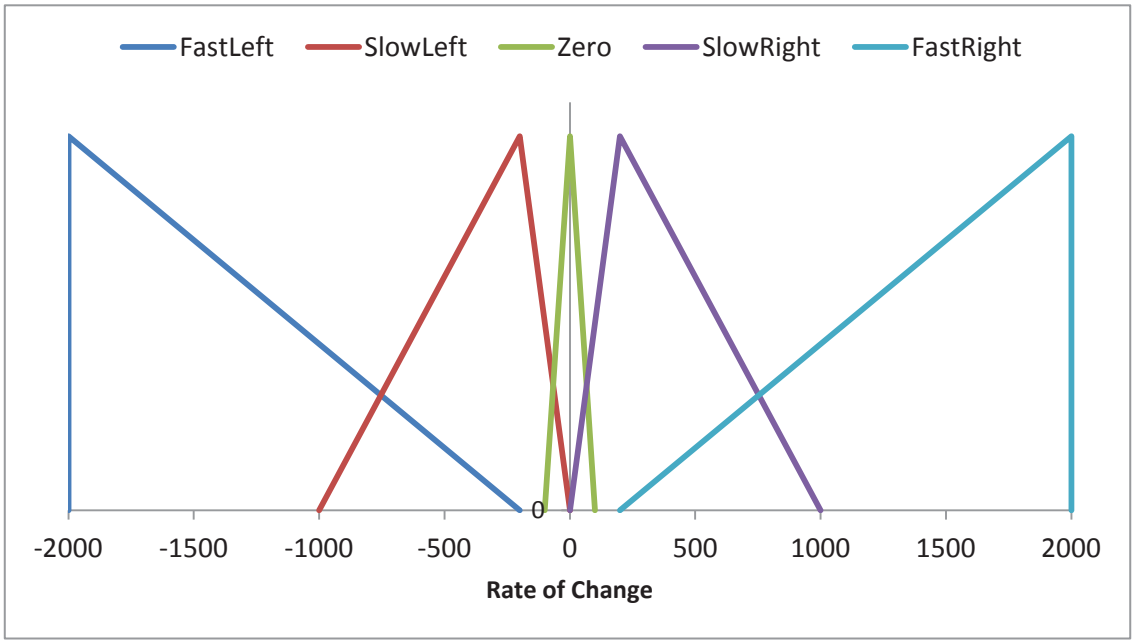

Figure 5.11 – Improved rate of change membership function.

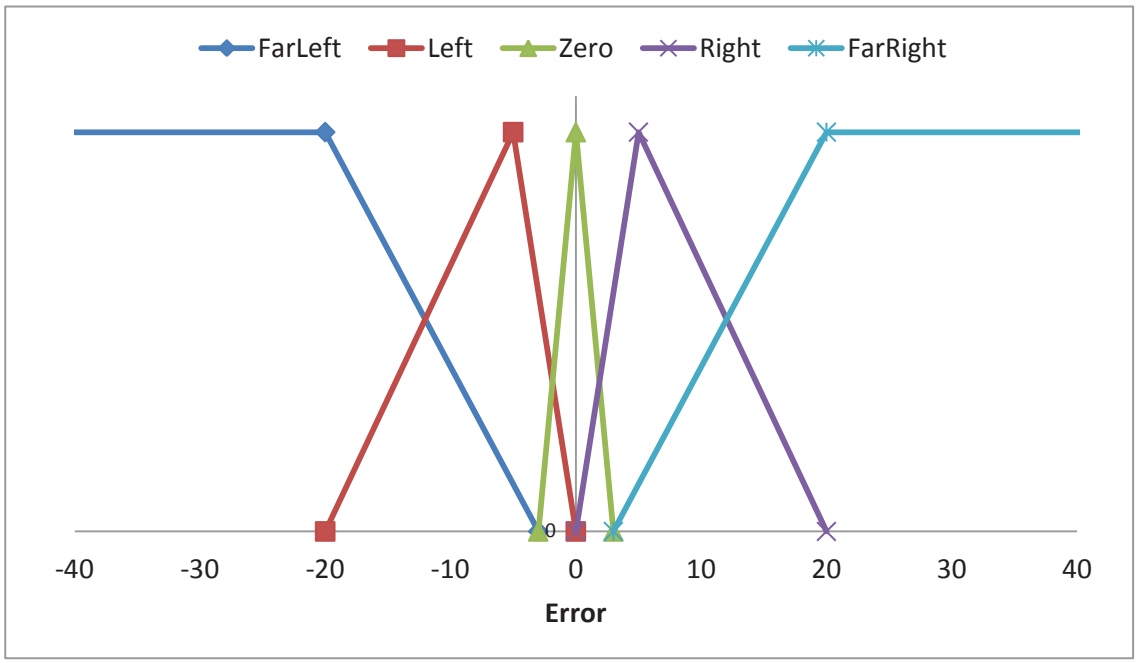

Figure 5.12 – Improved error membership function with increased overlap.

Although this is adequate further changes can be made for a possible improvement in control. These were the addition of two fuzzy sets to the rate of change input and output membership function and the modifying of the rule sets. These were both implemented to smooth the speed transition of the motor. Figure 5.13 shows the new membership functions trialled and the results figure 5.14.

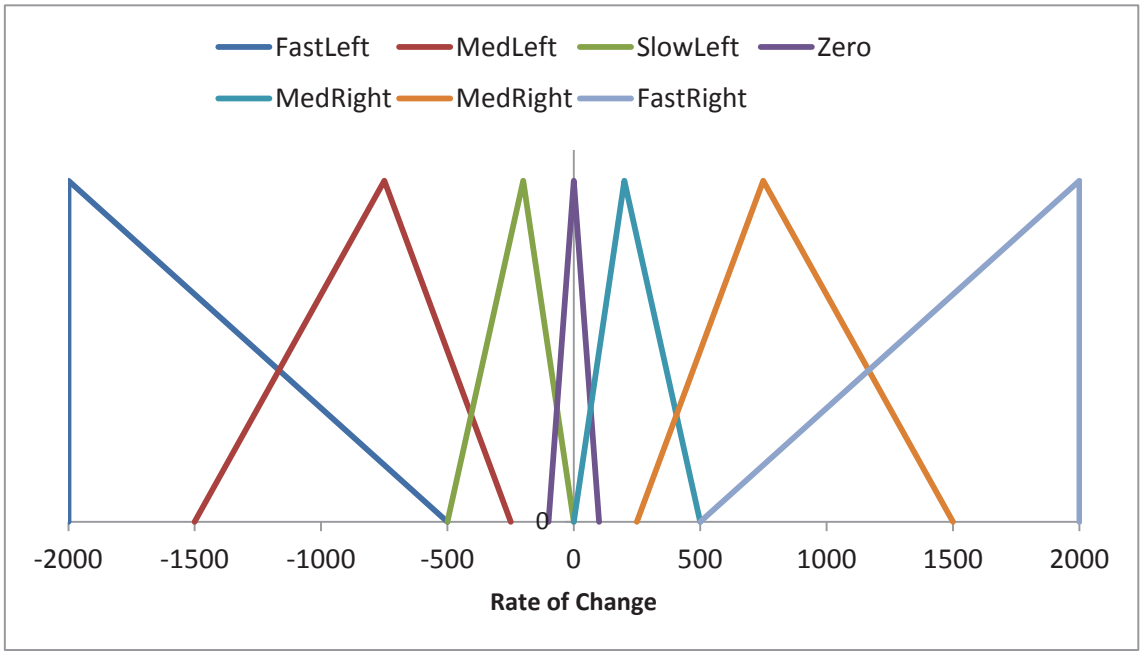

Figure 5.13 – Addition of fuzzy sets to the rate of change membership function.

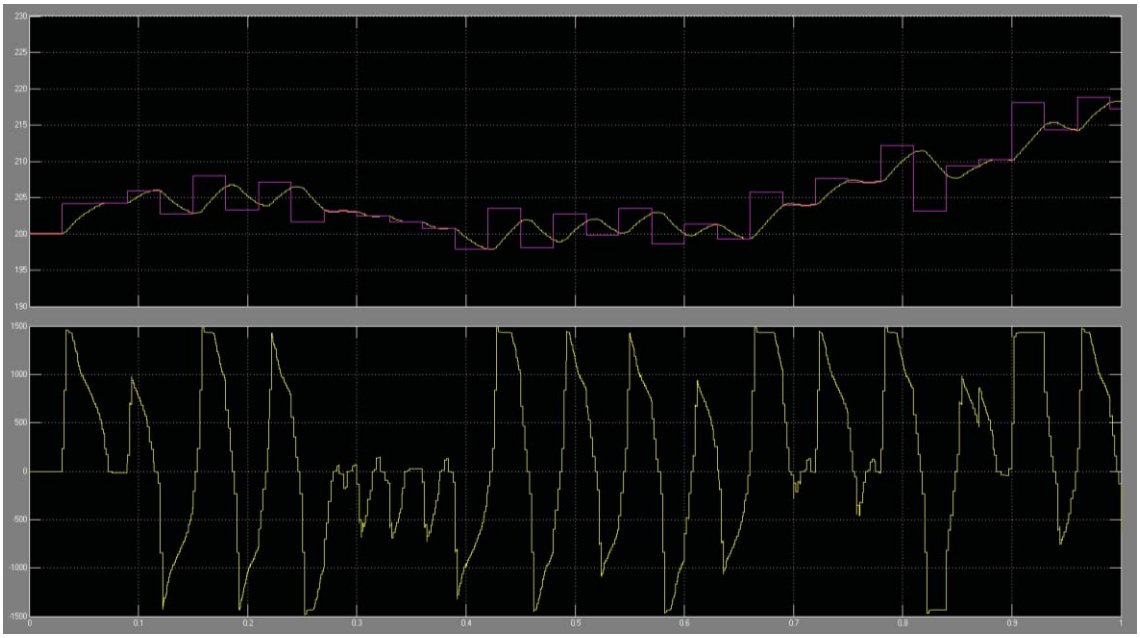

Figure 5.14 – Resulting output from the improved membership functions.

This fuzzy logic controller setup meets all the requirements of a controller for the positioning of the nozzle and all that is required now is to test it with the actual model of the stepper motor to see if it responds the same.

The test shown with the actual stepper motor model, figure 15.15, shows no difference in control from the previous model and although this is without load, this setup for control of the actuator meets all the requirements and the desired operating conditions.

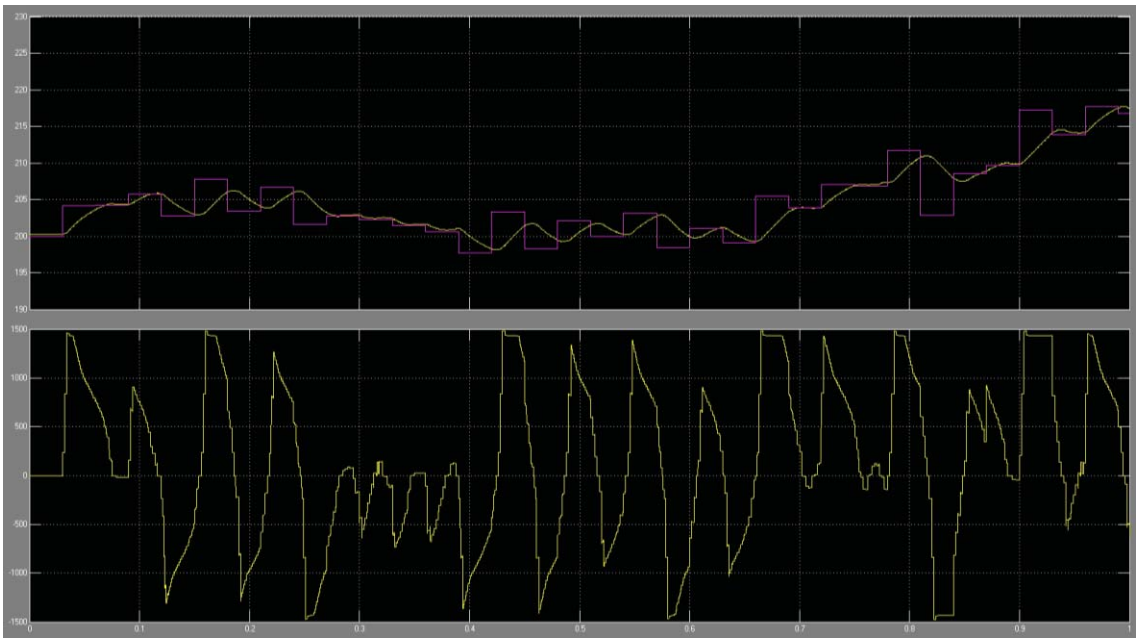

Figure 5.15 – Results of test with the complex stepper motor model.

5.3 Conclusion

A belt driven stage with travel of 400 millimetres will be used alongside a 12 volt stepper motor with a maximum torque rating of 0.5 N, connected to a gearbox with a ratio of 3.6:1 will be adequate for the purposes of this application. It will be controlled via a fuzzy logic controller with set membership functions and inference table. The output shows smooth transitions and fast responses to changes in set point.

6 Results and Analysis

Having designed a system to detect worn markings, find how much error there is and to control an actuator to minimise that error. This section will look at how each of these parts performs individually and together to see how well they work in situations they might encounter.

6.1 Image Processing

The image processing algorithms ability to detect road markings will be assessed in two ways: using images for a variety of different marking situations and video to determine speed and accuracy.

6.1.1 Results

As truck setups are all different and the methods or marking are slightly different a variety of different images were chosen to test the algorithm on. Slightly different angles and zooms were used to simulate these different setups along with a variety of different images of lines including worn, dirty, and cracked lines, and partly shaded images along with images of lines in various positions. Figure 6.1 shows some of the images used for testing normal situations and figure 6.2 shows some of the more difficult situations.

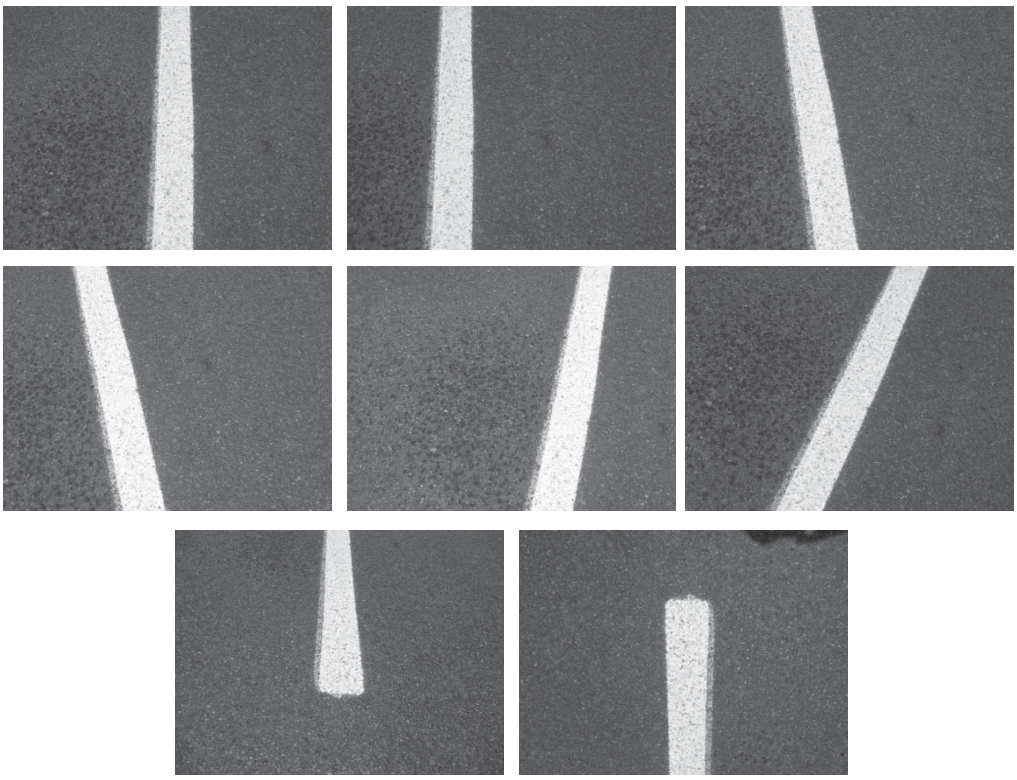

Figure 6.1 – Images of normal road marking situations.

Figure 6.2 - Difficult situations for road marking recognition

The process followed by the algorithm generates a set of quadrilaterals that can be used to approximate the line. In the normal road marking situations, such as those shown in figure 6.1, the algorithm picked up the line accurately and well. The quadrilateral models can be seen imposed on the original images in figure 6.3. Where there are cases of line being repainted not exactly over each other the algorithm is able to ignore the old line and only focus on the newer line for the next remarking.

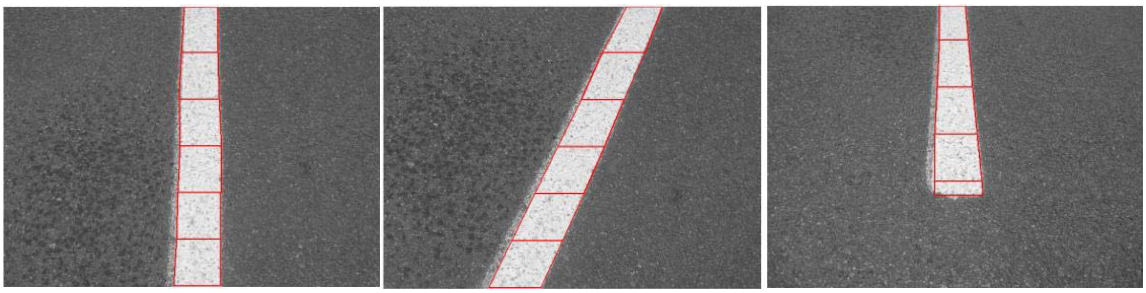

Figure 6.3 – Quadrilateral models superimposed on original images.

The fact that this algorithm works well in normal road marking situations is very good, but the point of road remarking is to repaint dirty, faded and otherwise degraded lines. In situations where the line has been completely wiped out or covered over, such as when a small part of the road has been repaired, the algorithm is able to fill in the gap and assume the line is present, figure 6.4. This depends on the size of the quadrilaterals used in the model and the size of the gap in the line. The algorithm also works well with cracked lines, ignoring the cracks and finding the edges very well, figure 6.5.

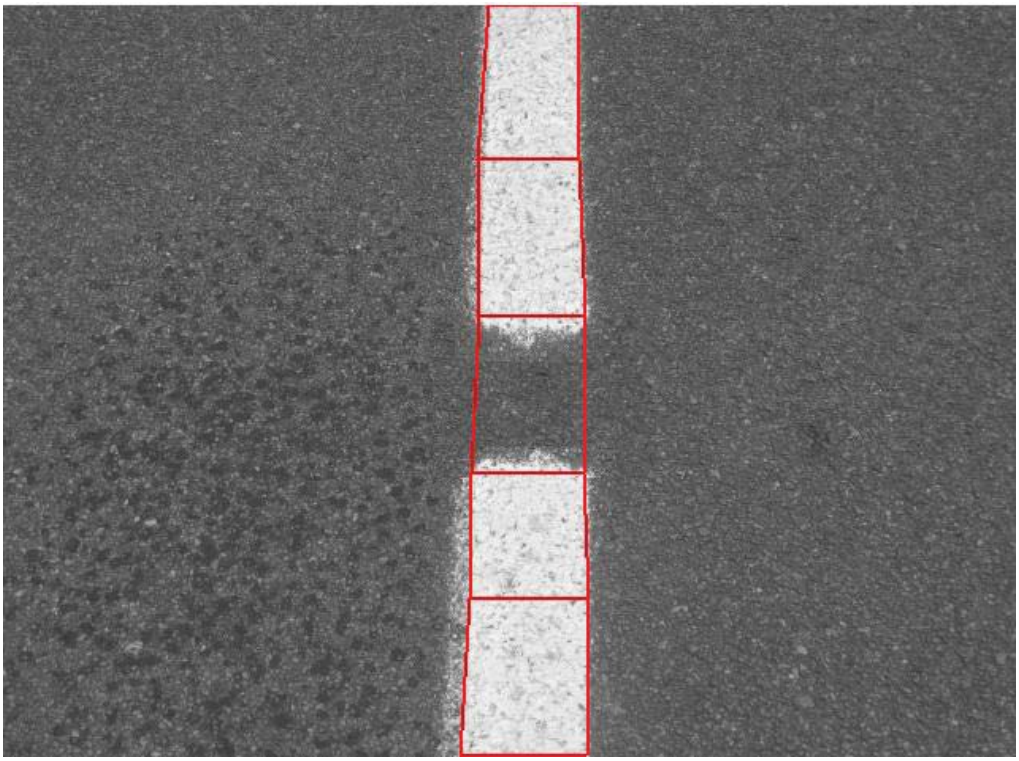

Figure 6.4 – Ignoring small holes in the line.

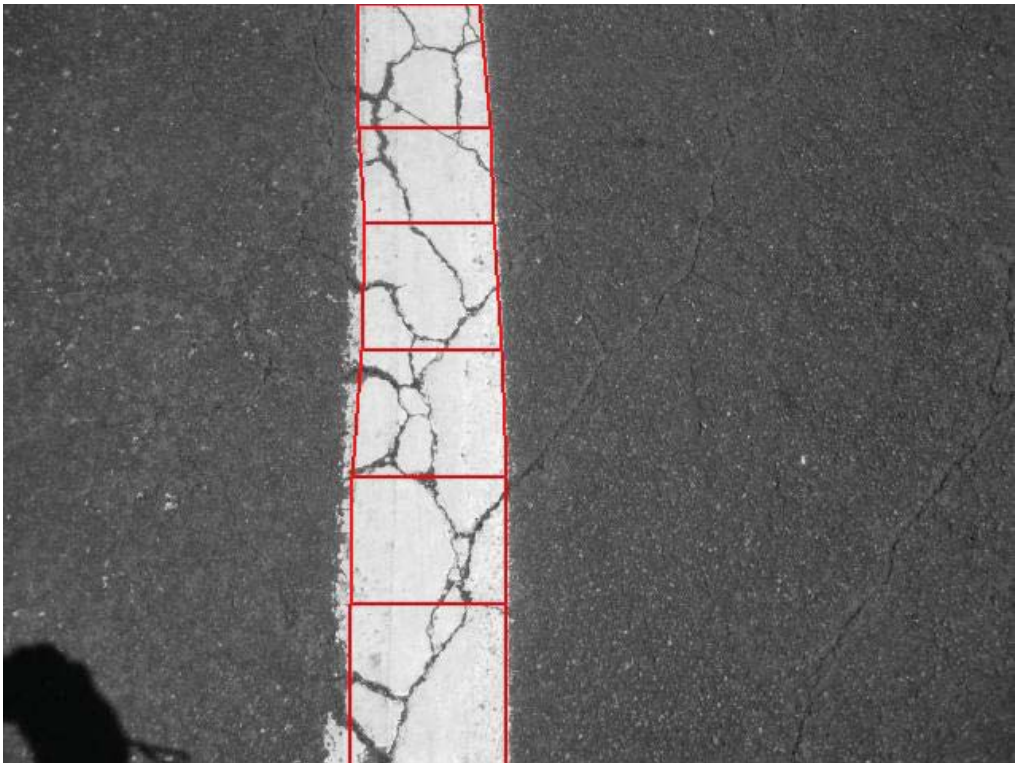

Figure 6.5 - Cracked Lines.

The biggest test of the algorithm is with very faded lines and faded lines with shadows on them. Figure 6.6 shows the result of the algorithm working on an image with varied shading all over the faded line. Despite the regions of high intensity caused by a lack of shade in some areas, the algorithm manages to detect where the line is most of the time. The errors that are visible occur when the shade less areas bound the edge of the line, causing the algorithm to extend the width of the line beyond the actual boundaries. Similarly with very faded lines the algorithm detects lines to be wider than they should be, but still manages to detect where the line is, figure 6.7. The time taken to process these images was not looked at, as factors such as the regions of interest were not able to be used for the processing. Hence the time taken for these images would be larger than for a continual video stream.

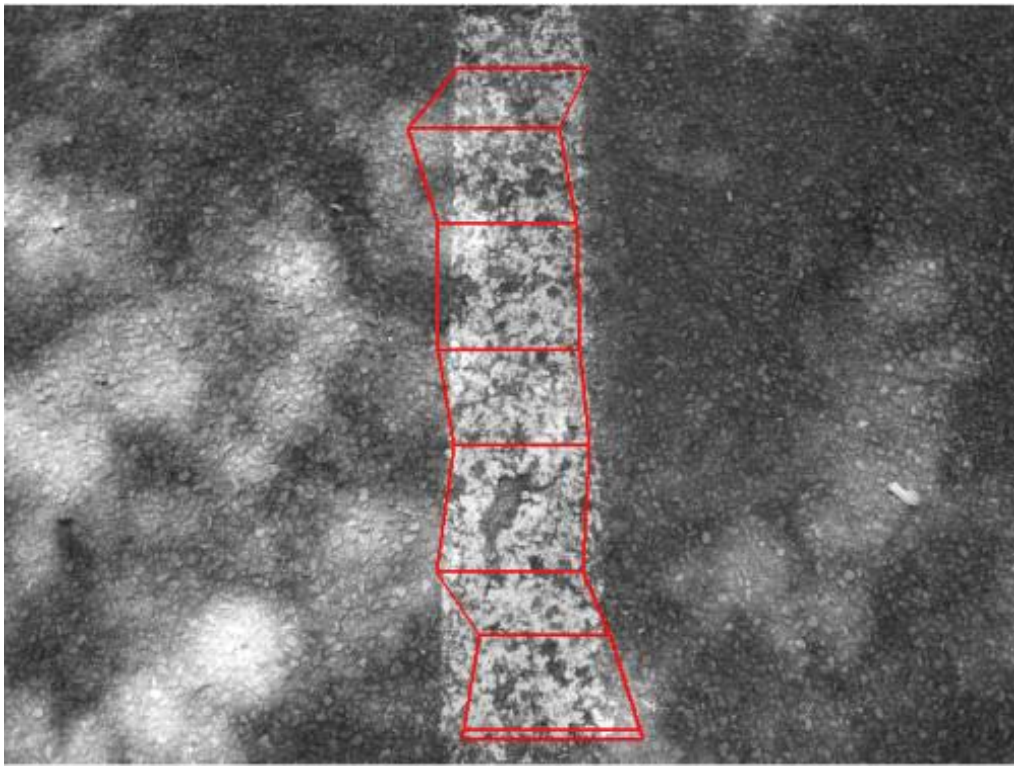

Figure 6.6 – Result on a shaded worn line.

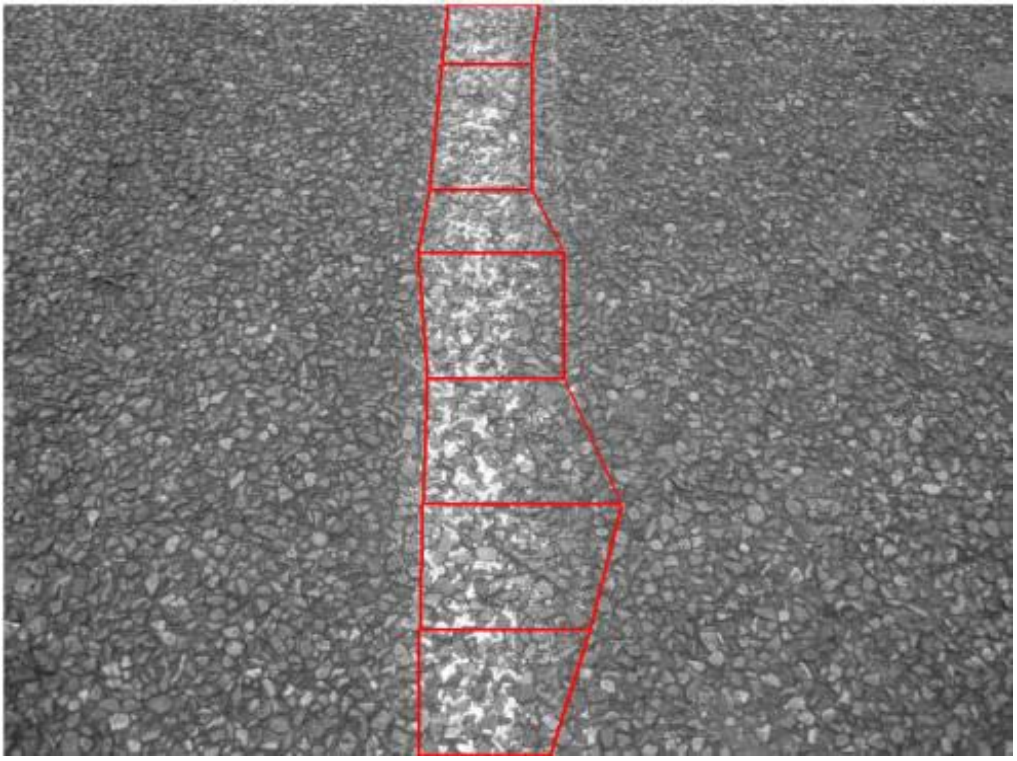

Figure 6.7 – Result on a highly worn line.

Along with the images being used for analysis it was important to also look at how the algorithm responds to video or sequences of images following one after the other. Looking at images in isolation gives a good idea as to how the algorithm works in various situations and under difficult conditions. Each of the images took between 0.3 and 0.5 seconds to process and for remarking that is far too slow. Ideally it would be able to work at 20 to 30 times per second or between 0.03 and 0.05 seconds per image. Fortunately the similarity between frames in the actual application of road remarking mimics those from a video of a line and so these can be used to determine the speed of the algorithm. Video was taken at a resolution of 640 x 480 and at 30 frames per second in colour. The algorithm was then asked to work through these sequences of images and an average of the time taken was calculated, the first value was ignored each time as there was no Region of Interest information for this frame. The sequences of frames were all around 100 frames in length.

The results were obtained from within Matlab, without compiling the program and with only a small amount of optimisation. Running the full algorithm gave results around 0.11 seconds per frame. As the information required from the algorithm is obtainable from the bottom of the image, it is not necessary to perform the full algorithm and model the entire line, if all that is required is in the bottom section. How far up the image the algorithm will need to go will depend on the speed of marking, by decreasing the amount the algorithm will need to model, the algorithm will be sped up. Limiting the algorithm to the lower half of the image gave results around 0.067 seconds per frame or 15 frames per second and limiting it to the lower third gave results of 0.051 seconds per frame or 19.5 frames per second. The output from one of these frames can be seen in figure 6.8.

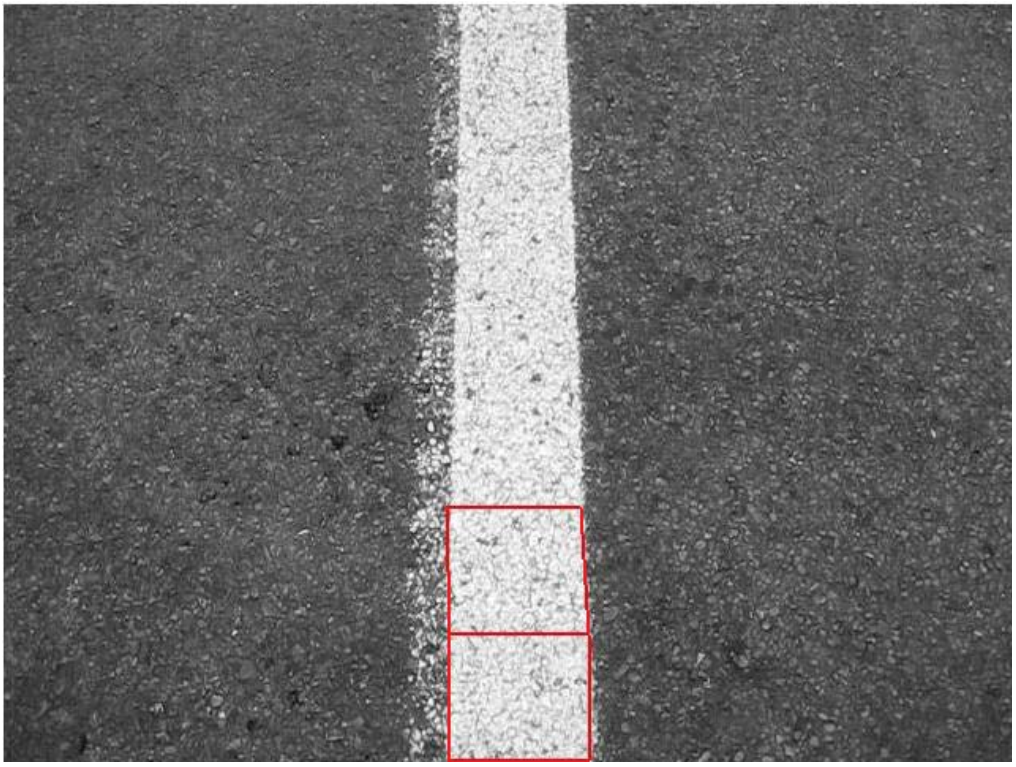

Figure 6.8 – Result of video running at 19.5 frames per second.

6.1.2 Analysis

The results from the algorithm give a good idea as to the overall performance of the algorithm. It manages to cope with a variety of different orientations and positions of the line with no problem. It struggles a little more with difficult situations, but still manages to find most of the line.

The major problem in the detection of the line was correctly detecting the width of the line. Often in difficult situations the algorithm would detect the line to be wider than it actually was. One simple solution to this is to instead of just having the minimum width check, having a range of values that the width needs to be between. If it is below the minimum then discard it and if it is above the maximum then perform a radon transform of the surrounding area to take into account the vertical correlation of the lines intensity values. This will eliminate a large amount of the error that occurred in highly shaded images and also in the very faded lines.

The speed of the detection when using video to simulate the actual input can be fast enough, if only part of the image is scanned. Taking into account the fact that the algorithm was tested on a PC running Windows and other assorted programs and also that the algorithm was not compiled or optimised to any real extent, it outperformed expectations. The speed of 0.051 seconds per frame was above the anticipated and expected outcome.

This thesis has not looked at situations where there are multiple lines in a single image. When marking the centre lines, occasionally there will be situations where there are two lines, either solid or dashed. The algorithm would not require much changing to accommodate this, but would require some tracking to keep track of which line is which. This problem would also occur when a single line diverges into two when there is a side road or a median strip begins. Setting the algorithm to follow a specific line could be easily accomplished, again with small modifications to the algorithm.

There is no intelligence in this algorithm between frames other than the regions of interest. For the purposes of safety and robustness it would be advantageous to have some sort of checking on the output of the algorithm. Keeping track of the outputs and measuring them against previous outputs and the expected range of outputs. This would protect against erroneous readings and errors caused by the algorithm and would make sure that if the line is not found or is found in the incorrect place that reading would not be translated into actuator movement.

6.2 Control and Actuation

6.2.1 Results

There are many different components that can be looked at when determining how well this system operates. Seeing how well it functions in keeping the line smooth, determining what the distribution of the errors are, looking at its step response, and seeing how it responds under varying loads are all good indicators of the systems suitability for the purpose it was designed for.

The control and actuation is used to position the nozzle over the line and so a very good indicator of this is to see how well it achieves that purpose. The full stepper motor model is used and has a constant load of 0.3N applied to it. Its operation seems to follow the line in a manner that is pleasing to the eye. It is important that the control changes direction and speed smoothly and not too suddenly for the motors safety, but also for the painted line. It is unsightly for the line to have sharp changes in direction and these will be noticed much more than the occasional smoothly swerving line. This is achieved and can be seen in figure 6.9.

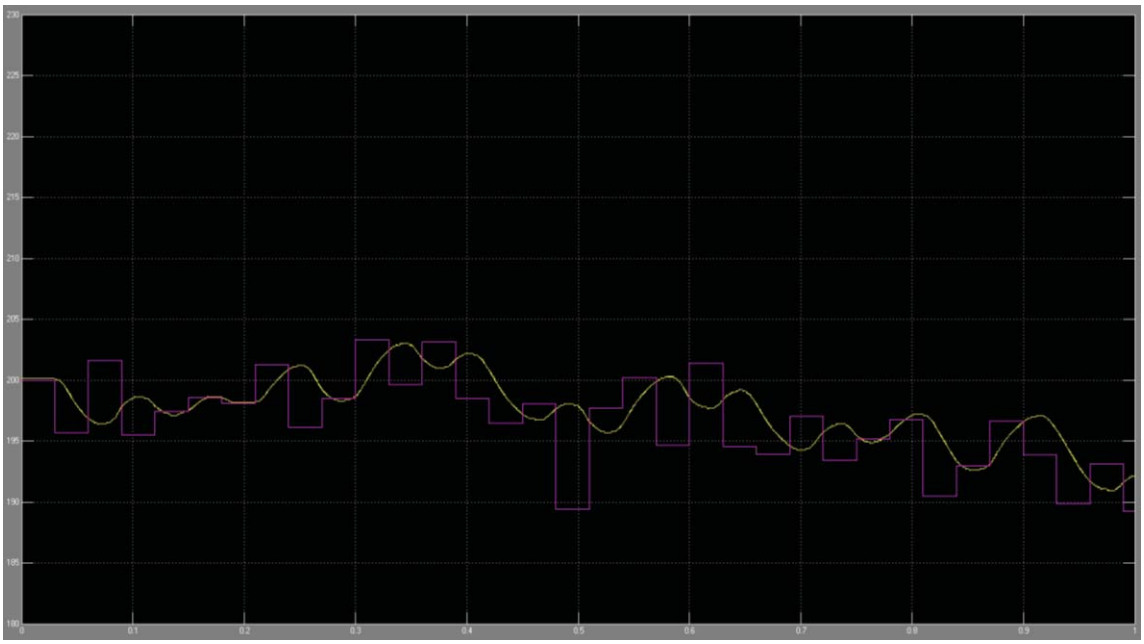

Figure 6.9 – Smoothly swerving output position of the actuator along with its set point.

The results for this project are all subjective and it is important to get some quantitative results as well. The error of the system is one of these results, it is important and needs to be looked at. The position of the actuator needs to mirror the line as closely as possible, although a small error will not be noticed by motorists due to the nature of paint and the surface being marked. This is very easy to measure, a simulation was set to go for ten seconds and the information about the set point error was recorded 500 times per second. The results were recorded in histogram format, figure 6.10, showing the distribution of the errors and the cumulative percentage. 50% of the time the actuator was within two millimetres of the centre of the line, and 85% of the time it was within five millimetres.

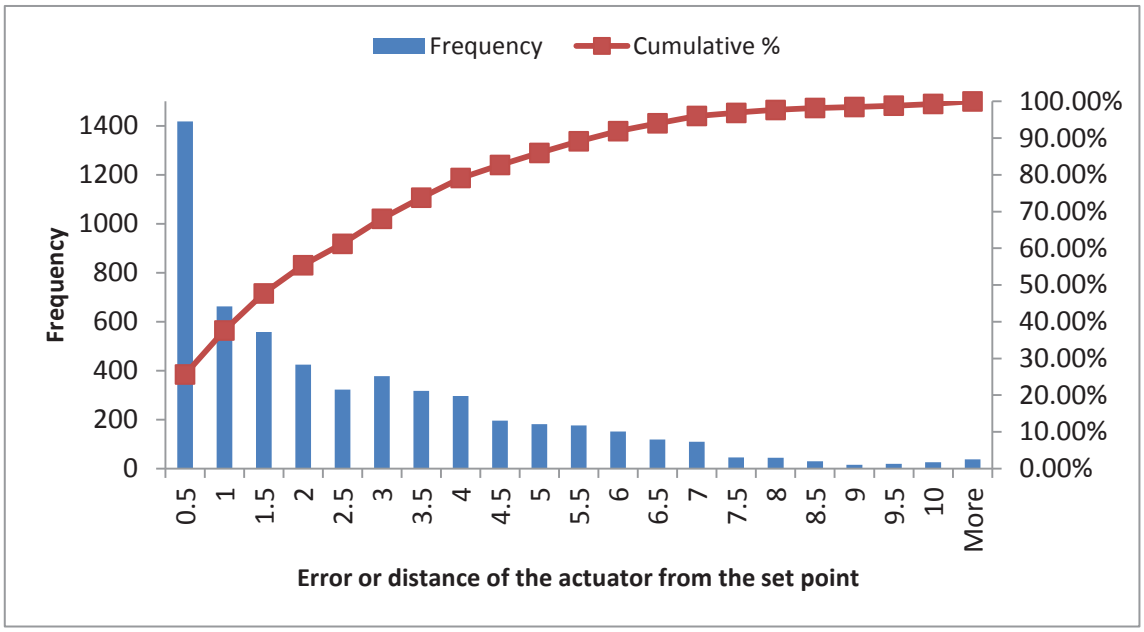

Figure 6.10 – Histogram of the distance from the set point to the actuator.

The speed and step response of the system also need to be looked at. A step of size 30 mm was used to test the response. Figure 6.11 shows the set point and resulting output. It took approximately 0.23 seconds to settle from the change in set point and this equals 130 millimetres per second. This is slower than the desired 150 millimetres per second but can be accounted for. There is a period of speed up and slow down at each end of this step causing the slower speed to be recorded. Looking at the slope of the actuator position gives an accurate measure of the top speed, it is very close to the desired 150 millimetres per second. There is also a minimal overlap of one step or 0.118 millimetres.

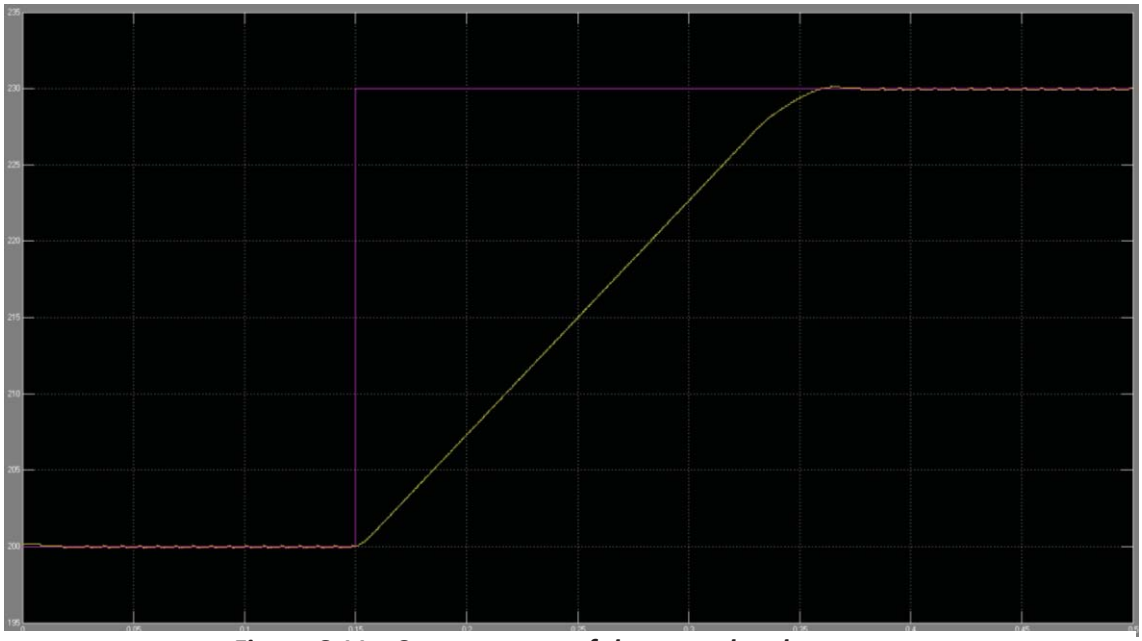

Figure 6.11 – Step response of the control and actuator.

The actuator will not be subjected to a constant load, from the jolting and jarring of driving to the changes in weight of the apparatus, the load will vary. It is important to test the control and actuation under these conditions. A changing load every 0.05 seconds was applied to the motor and the results were observed, figure 6.12. Minimal if any change was observed from the point of view of smoothness and accuracy. It does take longer to respond to changes in set point when there is a particularly high load torque applied or a sudden change occurs. This may cause a slightly higher error but should have minimal overall impact on the system. All other previous tests were carried out with a constant load of 0.3 newtons.

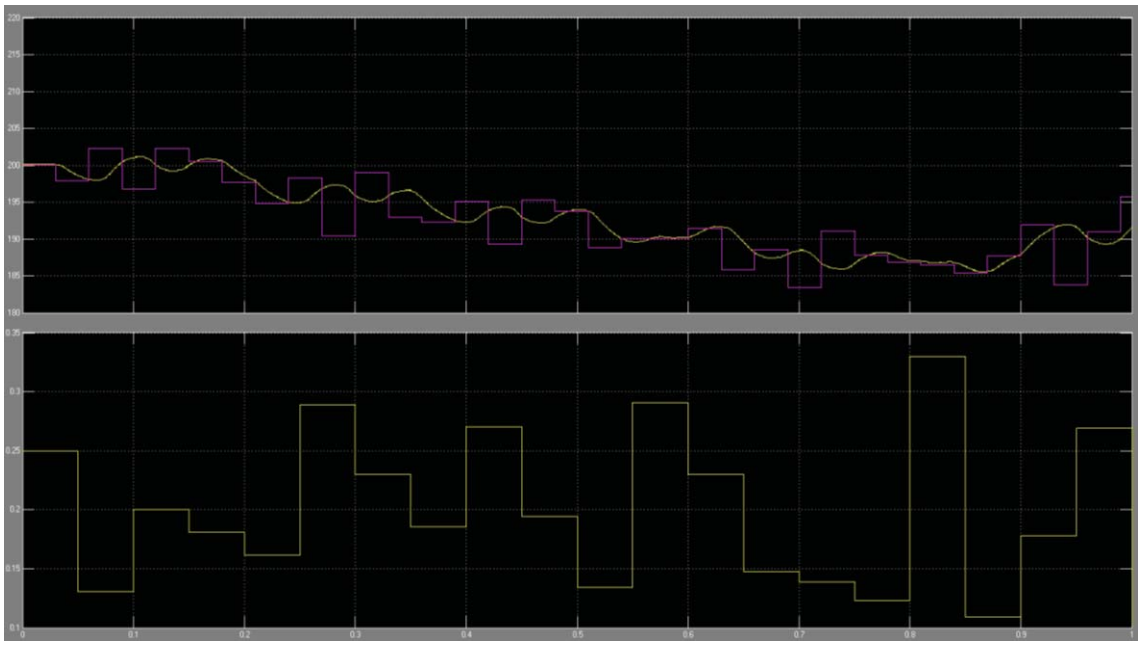

Figure 6.12 – Actuator set point and position (above) and load torque applied to the motor (below).

6.2.2 Analysis

All these results show promising control and actuation that would be suitable for remarking. The positioning of the actuator shows smooth transitions from one direction to another without any sharp changes. This may not be critical as the speeds of the change in directions and the frequency of these changes may be high enough that they are not noticeable. Along with the error that comes when spraying paint from a height onto a road that can be very uneven, driving at speeds mean the driver will not notice very small variations in the change. If when actual testing takes place the error is deemed to large it is possible to sharpen up the transitions giving a slightly faster response.

It is also very easy to change the membership functions of the fuzzy logic controller to give a higher top speed for the motor if it is deemed that the speed is insufficient. The low current top speed combined with the gearbox will give enough torque without needing to change the motor if this is decided upon.

The results for this section are not as accurate as they could be due to the use of a random number generator as the error. By adding or subtracting a random value to or from the current position, no account was made for the look ahead of the previous algorithm. The position it is moving to is the expected position of the line when the next measurement is received. The random values represent the driver error or change in direction of the vehicle based on the driver. These will not follow a completely random distribution and the drivers can be trained to minimise these errors. There will be minimal situations where the error goes from a very negative error suddenly to a very positive error. This would mean the driver is swerving back and forth and would not happen.

6.3 Overall System

Through the design of the algorithm it became apparent that the position of the camera should be directly over the spraying nozzle. It is a fine balance between look ahead, to anticipate where the line will be once the processing has occurred, and taking into account that the truck is also moving and so the line will move in unpredictable ways. By positioning the camera looking at an angle so the bottom of the image is as close to above the nozzle as possible, it will allow the software to determine the amount of look ahead required depending on the speed of the truck. The camera will be mounted on the actuator with the nozzle so ideally, the line will always be in the centre of the image. There are three reasons for this: first, is that the camera would need to be mounted reasonably far back to get the entire width of the actuator in without using a wide angle lens that would distort the image. This gives a smaller line to find and hence reduced

accuracy. Second, if the camera is statically mounted, when the lines are at either extreme there would be perspective issues around positioning the nozzle that would require computational time to solve. By mounting the camera with the actuator, actual feedback of the position of the nozzle in relation to the line can be easily acquired. Third, if the camera is mounted on the actuator it is closer to the line and the line will fill up more of the image. This will make it easier to detect the line particularly when it is very worn.

This makes it not plausible to simulate the entire system together. Without the actuation of the camera it is impossible to combine the two sections together into a single unit. A modular approach has been utilised for design as much as possible so as to minimise the issues that may arise when the system is connected together. Figure 6.13 shows the position of each of the components on the right hand side of the truck.

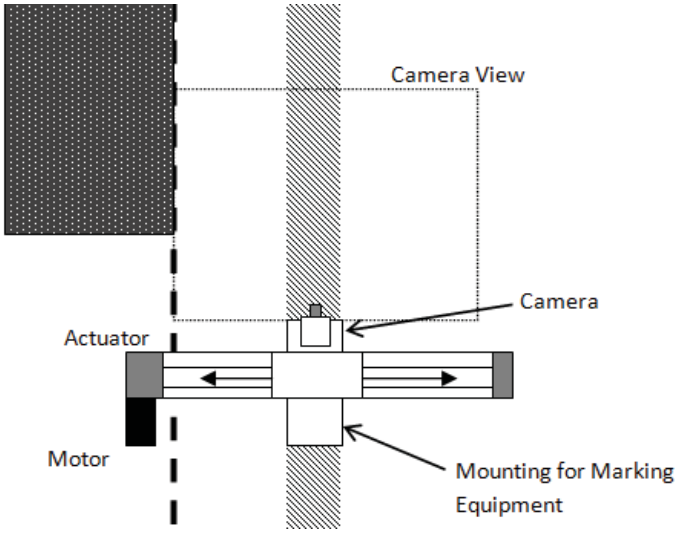

Figure 6.13 – Road marking equipment layout.

The decision to mount the actuator behind the wheel has advantages and disadvantages. It allows much more control over the lines being painted; sharper turns can be remarked without the possibility of driving over the painted line and wider lines can be painted. The downside is the need for increased control due to this position moving more when the truck is turned. The increased sensitivity to movement may be more difficult for operators to get used to but as the algorithm should be operating at about 30 times per second this will not be an issue.

Each of the required parts of the road marking process will be mounted under the actuator, figure 6.14. They all need to be in a line and mounting them under the actuator allows them to be closer together and frees up the space above for the support structure. The entire apparatus will need to be able to swing back when not in use to make sure it is not damaged when travelling or the other side is in operation.

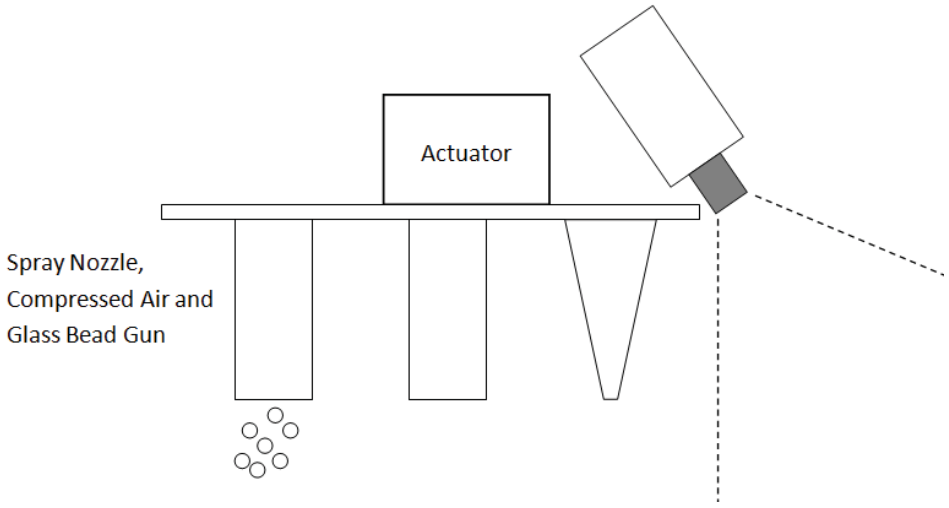

Figure 6.14 - Position of components on the actuator.

The positioning of the camera in the actuator also raises an issue with the algorithm that was unanticipated. There is a high probability that during the actuation, the camera will pick up the side of the vehicle or the wheel and this may have an adverse effect on the line detection process. The easiest way to remedy this would be to keep track of the actuators position and, as the vehicle will not be moving relative to the actuator, ignore the sections with information from the vehicle. It would be possible to map out where on the images the vehicle will be visible for any given actuator position.

Another consideration when remarking is the camber of the road and actuator. If the vehicle is on a lean then it would cause the position of the nozzle to not be directly over the line. The solution to this would be to include a gyroscope/accelerometer to detect the angle of the actuator and then it would be possible to compensate for that by offsetting the actuator from the line by a set amount.

6.4 Conclusion

The image processing algorithm worked very well in most situations and still found the line in the most difficult of images. The speed was very good but could be improved by running it on a dedicated system. A small number of tweaks could also be implemented to make it even more robust, but the underlying algorithm and principals function very well. The control and actuation for this section meet the theorised requirements for actuation of the nozzle. There is room within this design to easily modify the control without having to change any components if the results are not as desired.

7 Conclusion

This thesis looked at the development of an assisted road remaking system. Methods and components were chosen based on their individual traits and use in other applications and put together to form a system. An algorithm was developed to detected road markings in a variety of conditions: dirty, faded, and shady. The output of this algorithm is able to be fed into a control system which was developed to position a chosen actuator over where the line should be. It was modelled and tested under a variety of conditions and found to be suitable for the application it was designed for.

Initially this project was split up into three sections: sensing, control, and actuation. As the project progressed and the modelling became important, it was necessary and logical for the control and actuation sections became one.

The process followed for the sensing section involved first looking at various research articles for line following and detection projects. It was found that the majority used CCD cameras for their line detection as it has various advantages over CMOS cameras and any other sensors that were used. The algorithms used by these projects were also looked at and their advantages and disadvantages were weighed up. It was found that the radon transform method would be best suited for finding faded and worn lines, but that often a variety of methods were employed simultaneously to make the process faster and more robust. Regions of interest were also found to speed up the process of finding lines.

Once the research and decisions on the method had been made, the algorithm was developed. Various approaches were trialled and an algorithm gradually grew. Different methods and improvements were trialled and the ones that functioned well were used and others were discarded. The algorithm modelled the line with a set of quadrilaterals. Once the algorithm had reached an appropriate level it was tested on a variety of images to test its ability to find the line in different situations. Some of these included dirty, cracked or very faded lines. The process was also tested on a series of sequences of images to test its speed. It performed very well finding the lines and detected them almost perfectly most of the time. It struggled somewhat with very faded lines and lines with lots of partial shading. It was still able to find them but the line was too wide. The speed of the processing was found to be about 0.051 seconds.

For the control and actuation section a similar approach was carried out. Different methods of motor control were looked into and the various articles looked at compared Fuzzy Logic and PID control. In most of the situations it was found that fuzzy logic proved superior when dealing with

a varying load. Using the information obtained from the research articles and other sources of information, the components of the actuation and control part were chosen based on their advantages and the way they would work together. A belt driven actuator was chosen as the most suitable for the purposes of this project and it would be driven by a stepper motor. The stepper motor has the advantage of not needing any external feedback allowing the position to be known at all times.

Once the specifications of the actuator and motor were known it was possible to build a Simulink model of the system to simulate the control. Using membership functions and lookup tables similar to other applications as a starting point, it was possible to develop a controller that would control the motor correctly. This was tested in a variety of ways including a having constant load and varying load. The speed of the motor and the error were also looked at. It was found that the speed was almost exactly what was anticipated. The error showed that 85% of the time the actuator was within five millimetres of the set point. A varying load also showed minimal impact on the control. Most importantly the line appeared smooth to the eye, having gradual transitions from one direction to the other.

Deciding that the camera should be mounted above and in front of the spray nozzle meant that it was impossible to combine the two components together without actually building the system. This position was deemed best as it gave the algorithm a large line to find, removed most of the perspective issues and allows a simple camera without the need for a wide angle lens.

The overall costs of the components and the system as a whole is not high. All of this could be purchased and setup for a fraction of the savings it would be responsible for. The ongoing maintenance would be a small cost but compared to wages and new truck setups it is minimal.

The objectives were to develop a system that can detect and follow a line to a high degree of accuracy and at a high speed. This needed to be coupled with smooth control so that the line would appear visually smooth. Overall the objectives were met: the line was detected with high accuracy, 85% of the time it was within 5 mm, and the actuator position was smoothly controlled and would give a visually smooth line.

The overall system that was created and designed is as follows. A CCD camera mounted above the spraying nozzle taking images of the current road markings. These images are processed by an algorithm that uses the radon transform and local thresholding to find the line. Once the line is found a value corresponding to the error of the camera is sent to the control. The control uses the error along with the current speed to determine the correct values to send to the stepper motor

controller. This in turn controls the stepper motor and through a gearbox controls the actuator and camera. Connected to the actuator is the road marking apparatus which will also follow the line.

The final system is able to:

- Detect a line in a variety of conditions and at various qualities.
- Position an actuator over a line accurately and fast.
- Improve the quality and speed of road remarking
- Allow faster training and improved safety.
- Reduce the cost of the remarking process.

7.1 Further Work

There is always scope for improvement and further work. The system that has been designed and modelled has not been built. The logical next stage would be the building of this system to further test its capabilities. Often modelling cannot accurately predict every situation or occurrence and the building of the system would allow more testing to be carried out. It would also allow the two main sections of line detection and control and actuation to be connected together and trialled. Although a modular approach was taken there are always issues that arise when systems are connected together. By compiling the algorithm and running it on a dedicated system, the speed of detection would also increase giving an even higher frame rate.

The algorithm works very well but as was touched upon in the analysis, there are several things that can be changed. Implementing a maximum width for the line would eliminate many of the problems in detecting the faded or heavily shaded lines. Also before it was able to be used in a real world application the algorithm would need to be given the ability to track multiple lines at once and also lines that diverge and come back together. It would need to recognise these situations and know which line to follow. Some form of error checking should also be implemented to increase robustness and stop any erroneous readings making it through the control and into incorrect horizontal movement.

Further investigation is also warranted into how fast the actuator should transition from one direction to another and the top speed of the actuator. If it is not visible to the human eye, then it makes sense to speed up transitions and speeds giving a lower overall error. This would need to be carried out once the system had been built.

The inclusion of an accelerometer would also be advantageous to check and counter any error caused by the camber of the vehicle and the road.

8 References

[1] (n.d.). Retrieved May 2010, from Kadcam Enterprises Ltd: http://www.kadcamenterprises.com/home.htm

[2] Croft, D. (2002, January). *Guidelines for rural road marking and delineation RTS 5.* Retrieved Febuary 2010, from New Zealand Tranport Agency: http://www.nzta.govt.nz/resources/road-traffic-standards/docs/rts-05.pdf

[3] Charbonnier, P., Diebolt, F., Guillard, Y., & Peyret, F. (1998). Road Markings Recognition Using Image Processing. *IEEE Conference on Intelligent Transportation System* (pp. 912-917). Boston: IEEE.

[4] McCall, J. C., & Trivedi, M. M. (2006). Video-Based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation. *IEEE Transactions on Intelligent Transportation Systems , 7* (1), 20-37.

[5] LeBlanc, D. J., Johnson, G. E., Venhovens, P., Gerber, G., DeSonia, R., Ervin, R. D., et al. (1996, December). CAPC: A Road-Departure Prevention System. *Control Systems Magazine, IEEE , 16* (6), pp. 61-71.

[6] Assidiq, A., Khalifa, O., Islam, R., & Khan, S. (2008). Real Time Lane Detection for Autonomous Vehicles. *Proceedings of the International Conference on Computer and Communication Engineering 2008* (pp. 82-88). Kuala Lumpur: IEEE.

[7] Lipski, C., Scholz, B., Berger, K., Linz, C., & Stich, T. (2008). A Fast and Robust Approach to Lane Marking Detection and Lane Tracking. *IEEE Southwest Symposium on Image Analysis and Interpretation* (pp. 57-60). Santa Fe: IEEE.

[8] Barickman, F. S., & Stoltzfus, D. L. (1999). A Simple CCD Based Lane Tracking System. *SAE Technical Paper Series, International Congress & Exposition.* Detroit.

[9] Goldbeck, J., & Huertgen, B. (1999). Lane Detection and Tracking by Video Sensors. *Proceedings of International Conference on Intelligent Transportation Systems* (pp. 74-79). Tokyo: IEEE.

[10] Macek, K., Williams, B., Kolski, S., & Siegwart, R. (2004). A Lane Detection Vision Module for Driver Assistance. *Mechatronics and Robotics .* Aachen: IEEE.

[11] Kheyrollahi, A., & Breckon, T. P. (2012). Automatic real-time road marking recognition using a feature driven approach. *Machine Vision and Applications , 23* (1), 123-133.

[12] Bertozzi, M., & Broggi, A. (1998). GOLD: A Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection. *IEEE Transactions on Image Processing , 7* (1), 62-81.

[13] Bertozzi, M., & Broggi, A. (1997, July). Vision Based Vehicle Guidance. *Computer , 30* (7), pp. 49-55.

[14] Bellino, M., Lopez de Meneses, Y., Ryser, P., & Jacot, J. (2005). Lane Detection Algorithm for an Onboard Camera. (T. P. Pearsall, Ed.) *Photonics in the Automobile , 5663*, 102-111.

[15] Kastrinaki, V., Zervakis, M., & Kalaitzakis, K. (2003). A Survey of Video Processing Techniques for Traffic Applications. *Image and Vision Computing* , 359-381.

[16] Vacek, S., Schimmel, C., & Dillmann, R. (n.d.). *Road-Marking Analysis for Autonomous Vehicle Guidance.* Retrieved April 24, 2009, from Freiburg University: http://ecmr07.informatik.uni-freiburg.de/proceedings/ECMR07_0034.pdf

[17] Viet, T., Tarel, J.-P., Nicolle, P., & Charbonnier, P. (2008). Evaluation of Roadmarking Feature Extraction. *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems* (pp. 174-181). Beijing: IEEE.

[18] Asif, M., Arshad, M. R., & Wilson, P. A. (2005). AGV Guidance System: An Application of Simple Active Contour for Visual Tracking. *Proceedings of World Academy of Science, Engineering and Technology*, (pp. 74-77). Madrid.

[19] McCall, J. C., & Trivedi, M. M. (2004). An Integrated, Robust Approach to Lane Marking Detection and Lane Tracking. *Proceedings of the IEEE Intelligent Vehicles Symposium* (pp. 533-537). IEEE.

[20] Zhang, Q., & Couloigner, I. (2007). Accurate Centerline Detection and Line Width Estimation of Thick Lines Using the Radon Transform. *IEEE Transactions on Image Procssing , 16* (2), 310-316.

[21] Dickmanns, E., Behringer, R., Dickmanns, D., Hildebrandt, T., Maurer, M., Thomanek, F., et al. (1994). The Seeing Passenger Car 'VaMoRs-P'. *Proceedings of theIntelligent Vehicles '94 Symposium* (pp. 68-73). IEEE.

[22] Nagel, H., Enkelmann, W., & Struck, G. (1995). FhG-Co-Driver - From Map-Guided Automatic Driving by Machine Vision to a Cooperative Driver Support. *Mathematical and Computer Modelling , 22* (4-7), 185-212.

[23] Tang, K., Fung Man, K., Chen, G., & Kwong, S. (2001). An Optimal Fuzzy PID Controller. *IEEE Transactions on Industrial Electronics* (pp. 757-765). IEEE.

[24] Santos, M., de la Cruz, J. M., Dormido, S., & de Madrid, A. P. (1996). Between Fuzzy-PID and PID-ConventionalControllers: a Cood Choice. *1996 Biennial Conference of the North American Fuzzy Information Processing Society.* , (pp. 123-127). Berkeley.

[25] Joe Qin, S. (1994). Auto-Tuned Fuzzy Logic Control. *American Control Conference, 1994*, (pp. 2465-2469).

[26] Muoz-Cesar, J., Merchan-Cruz, E., Hernandez-Gomez, L., Guerrero-Guadarrama, E., Jimenez-Ledesma, A., & Jaidar-Monter, I. (2008). Speed Control of a DC Brush Motor with Conventional PID and Fuzzy PI Controllers. *Electronics, Robotics and Automotive Mechanics Conference* (pp. 344-349). Morelos: IEEE.

[27] Khongkoom, N., Kanchanathep, A., Nopnakeepong, S., Tanuthong, S., Tunyasrirut, S., & Kagawa, R. (2000). Control of the Position DC Servo Motor by Fuzzy Logic. *TENCON 2000, Proceedings* (pp. 354-357). Kuala Lumpur : IEEE.

[28] Le-Huy, H., & Hamdi, M. (1993). Control of a Direct-Drive DC Motor by Fuzzy Logic. *Conference Record of the 1993 IEEE Industry Applications Society Annual Meeting* (pp. 732-780). Toronto: IEEE.

[29] Alexei, Z., & Sandor, H. (1997). Robust Speed Fuzzy Logic Controller for DC Drive. *IEEE International Conference on Intelligent Engineering Systems* (pp. 385-389). Budapest: IEEE.

[30] Ying, H., Siler, W., & Buckley, J. J. (1990). Fuzzy Control Theory: A Nonlinear Case. *Automatica , 26* (3), 513-520.

[31] *FireWire vs. USB 2.0.* (2005, December). Retrieved August 31, 2010, from Q Imaging: http://www.qimaging.com/support/pdfs/firewire_usb_technote.pdf

[32] Litwiller, D. (2007). *CCD vs. CMOS.* Retrieved September 1, 2010, from DALSA: http://www.dalsa.com/corp/markets/CCD_vs_CMOS.aspx

[33] Gonzalez, R. C., & Woods, R. E. (2008). *Digital Image Processing Third Edition.* New Jersery: Pearson Prentice Hall.

[34] *Radon Transform, Image Processing Toolbox User's Guide, Matlab Help Document*. Natick, MA: MathWorks, Inc., 2005.

[35] (n.d.). Retrieved May, 2011, from Omega Engineering Inc: http://www.omega.com/prodinfo/Stepper_Torque.html

[36] Cameron, M., & Al-Bahadly, I. (2012 in press). A line Detection Algorithm for Road Remarking. *Sensors and Transducers Journal.* ISSN 1726-5749.

Appendix 1 – Main Algorithm

```
function  [Marking NewROI Time] = LineFinding(BWImage, ROI,
Threshold, RectangleHeight, AveragingDistance, RoiDistance,
RadonRatio, TopWidth, BottomWidth)
% Creating Rectangles
% Rectangle(x).BL       = Bottom Left coordinates
% Rectangle(x).BC       = Bottom Centre coordinates
% Rectangle(x).BR       = Bottom Right coordinates
% Rectangle(x).TL       = Top Left coordinates
% Rectangle(x).TC       = Top Centre coordinates
% Rectangle(x).TR       = Top Right coordinates
% Rectangle(x).Angle    = Angle of the rectangle

tic
[X Y] = size(BWImage);
CurrentRow = X+20 - 1; %Minus 1 as camera has a gray line at
bottom of image
Finished = false;
Found = false;
First = true;
RectangleCounter = 1;
Left = NaN;
End = false;

while (~Finished)
    if ~Found, ForwardSearch(); end
    if End
        return
    end

    Rectangle(RectangleCounter).BL = [Left CurrentRow];
    Rectangle(RectangleCounter).BR = [Right CurrentRow];
    Rectangle(RectangleCounter).BC = [(Left + Right)/2
CurrentRow];

    PreviousRow = CurrentRow;
    CurrentRow = CurrentRow - RectangleHeight - 20;

    BackSearch();

    if isnan(Left) && ~Finished  %If the previous search doesn't
find anything
        CurrentRow = PreviousRow - RectangleHeight;
        Found = false;
    elseif isnan(Left) && Finished
        Rectangle(RectangleCounter) = [];
    else
        Rectangle(RectangleCounter).TL = [Left CurrentRow];
        Rectangle(RectangleCounter).TR = [Right CurrentRow];
        Rectangle(RectangleCounter).TC = [(Left + Right)/2
CurrentRow];
        Rectangle(RectangleCounter).Angle =
atan((Rectangle(RectangleCounter).TC(1)
```

```
                            - Rectangle(RectangleCounter).BC(1)) /
(CurrentRow - Rectangle(RectangleCounter).BL(2))) * 180/pi;
        RectangleCounter = RectangleCounter + 1;
        Found = true;
    end
end

if Found
    Size = max(size(Rectangle));
    if Rectangle(1).BL(2) <= size(BWImage, 1) - 10 %If the marking
doesn't start at
                                                                the
bottom of the image
        [LeftCord RightCord] = RadonDown(BWImage,Rectangle);
        for i = Size:-1:1
            Rectangle(i+1) = Rectangle(i);
        end
        Rectangle(1).TL = Rectangle(2).BL;
        Rectangle(1).TC = Rectangle(2).BC;
        Rectangle(1).TR = Rectangle(2).BR;
        Rectangle(1).BL = LeftCord;
        Rectangle(1).BR = RightCord;
        Rectangle(1).BC = [(LeftCord(1) + RightCord(1))/2
LeftCord(2)];
        end

%-----------------------------------------------------------------
    %Creating ROI using an area equal to a multiple of the width
of the current
    %line. Achieved by assuming that the line is approximately
straight and
    %therefore calculating the gradient and zero crossing  to form
an equation
    %to easily calculate the edge for any row (value of y) using y
= m*x + c
    %or in this case reversed to be x = (y - c) / m.
%-----------------------------------------------------------------

    Width = 0;
    for j = 1:RectangleCounter-1
        Width = Width + Rectangle(j).BR(1) - Rectangle(j).BL(1);
    end
    Width = round(Width / (RectangleCounter-1));

    if ~First
        Lx1 = Rectangle(RectangleCounter-1).TL(1) -
Width*RoiDistance;
        Lx2 = Rectangle(1).BL(1) - Width*RoiDistance;
        Ly1 = Rectangle(RectangleCounter-1).TL(2);
        Ly2 = Rectangle(1).BL(2);

        Rx1 = Rectangle(RectangleCounter-1).TR(1) +
Width*RoiDistance;
        Rx2 = Rectangle(1).BR(1) + Width*RoiDistance;
        Ry1 = Rectangle(RectangleCounter-1).TR(2);
        Ry2 = Rectangle(1).BR(2);
```

```
        [Lm Lc] = GetMandC(Lx1,Ly1,Lx2,Ly2);
        [Rm Rc] = GetMandC(Rx1,Ry1,Rx2,Ry2);
    else
        Lm = NaN;
        Lc = NaN;
        Rm = NaN;
        Rc = NaN;
        Rectangle(1).BL(1) = NaN;
    end
else
    Lm = NaN;
    Lc = NaN;
    Rm = NaN;
    Rc = NaN;
    Rectangle(1).BL(1) = NaN;
end

NewROI = [Lm Lc;Rm Rc];
Time = toc;
Marking = Rectangle;
end
```

Appendix 2 – Subroutines

Forward Search

```
    function ForwardSearch ()
        while isnan(Left) && ~Finished
            CurrentRow = CurrentRow - 20;
            if CurrentRow < 1 && First     %No line found in image
at all
                Marking(1).BL(1) = NaN;
                NewROI = [NaN NaN; NaN NaN];
                Time = 0;
                End = true;
                break
            elseif CurrentRow < 1
                CurrentRow = 1;
                Finished = true;
            end
            if ~isnan(ROI(1,1))            %There is a region of
interest
                [LeftROI RightROI] = RoiValues (CurrentRow, ROI);
                if LeftROI < 1, LeftROI = 1; end
                if RightROI > Y, RightROI = Y; end
                [Left Right] =
GetLargest(BWImage(CurrentRow,:),LeftROI,RightROI);
            else
                [Left Right] =
GetLargest(BWImage(CurrentRow,:),1,Y);
            end
        end
        if ~isnan(Left)
            First = false;
        end
    end
```

Backwards Search

```
    function BackSearch ()
        LineFound = false;
        while ~LineFound
            %Searches for the next row values
            CurrentRow = CurrentRow + 20;
            if CurrentRow >= PreviousRow
                break
            elseif CurrentRow < 1
                CurrentRow = 1;
                Finished = true;
            end

            if ~isnan(ROI(1,1))             %There is a region of
interest
                [LeftROI RightROI] = RoiValues (CurrentRow, ROI);
                if LeftROI < 1, LeftROI = 1; end
```

```
                if RightROI > Y, RightROI = Y; end
                [Left Right] =
GetLargest(BWImage(CurrentRow,:),LeftROI,RightROI);
            else
                [Left Right] =
GetLargest(BWImage(CurrentRow,:),1,Y);
            end
            if ~isnan(Left)
                Line.X = [Left Right;
Rectangle(RectangleCounter).BL(1)

Rectangle(RectangleCounter).BR(1)];
                Line.Y = [CurrentRow;
Rectangle(RectangleCounter).BL(2)];
                LineThere = RadonIsThere(BWImage, Line,
RadonRatio);
                if LineThere
                    First = false;
                    LineFound = true;
                end
            end
        end
    end
```

Largest Line in Row

```
    function [Left Right] = GetLargest(BWImageRows,RoiLeft,
RoiRight)
        Width = TopWidth + 2*CurrentRow / (2 * X / (BottomWidth -

TopWidth));
        [LeftArray RightArray] = LineFinder(BWImageRows,
Threshold,
                        AveragingDistance, Widths, RoiLeft,
RoiRight);

        biggest = 10;
        Place = NaN;
        for i = 1:max(size(LeftArray))
            width = RightArray(i) - LeftArray(i);
            if width > biggest
                biggest = width;
                Place = i;
            end
        end

        if isnan(Place)
            Left = NaN;
            Right = NaN;
        else
            Left = LeftArray(Place);
            Right = RightArray(Place);
        end
```

```
    end
```

Local Averaging Line Finder

```
function [Left Right] = LineFinder(Array, Threshold,
AveragingDistance,
                                                MinWidth, LeftROI,
RightROI)

Going = false;
Count = 0;
NoUnder = 999;
length = size(Array,2);

for i = LeftROI:RightROI
    LeftValue = i - AveragingDistance;
    RightValue = i + AveragingDistance;
    if LeftValue < 1
        LeftValue = 1;
    end
    if RightValue > length
        RightValue = length;
    end
    LeftAve(i) = mean(Array(LeftValue:i-1));
    RightAve(i)= mean(Array(i+1:RightValue));









    if (Array(i) > (LeftAve(i) + Threshold)) && (Array(i) >
(RightAve(i)
                                                                +
Threshold))
        if ~Going
            if NoUnder > 3 % Number under allowed before it won't
link up
                Start = i;

            else
                Count = Count - 1;
            end
            Going = true;
        end
    else
        if Going
            End = i - 1;
            NoUnder = 1;
            Going = false;
            Count = Count + 1;
```

```
            LeftValues(Count) = Start;
            RightValues(Count) = End;
        else
            NoUnder = NoUnder + 1;
        end
    end
end

for i = 1:Count-1
    Finished = false;
    if i >= (Count)
        break
    end
    while ~Finished
        Gap = LeftValues(i+1) - RightValues(i);
        Size = RightValues(i) - LeftValues(i);
        if (Gap < 20) && (Gap > 0) && (Size > 1)
            RightValues(i) = RightValues(i+1);
            LeftValues(i+1) = 0;
            RightValues(i+1) = 0;
            for j = i+1:Count-1
                LeftValues(j) = LeftValues(j+1);
                RightValues(j) = RightValues(j+1);
            end
            Count = Count - 1;
            None = false;
        else
            Finished = true;
        end
    end
end

Left = NaN;
Right = NaN;

j = 0;
for i = 1:Count
    if (RightValues(i) - LeftValues(i)) > MinWidth
        j = j + 1;
        Left(j) = LeftValues(i);
        Right(j) = RightValues(i);
    end
end
```

ROI Value Finder

```
function [LeftValue RightValue] = RoiValues (Row, ROI)
%Gets the left and right ROI values for the specified row from the
image ROI

if ROI(1,1) == 0
    LeftValue = ROI(1,2);
else
    LeftValue = round((Row - ROI(1,2))/ROI(1,1));
end
```

```
if ROI(2,1) == 0
    RightValue = ROI(2,2);
else
    RightValue = round((Row - ROI(2,2))/ROI(2,1));
end
```

Equation Calculator

```
function [M C] = GetMandC(x1,y1,x2,y2)
%Returns the gradient M and zero intercept C for the line that
passes %through the points (x1,y1) and (x2,y2). If it is a
horizontal line then %M is 0 and C is the Y intercept.
if (x1 ~= x2)
    M = double((y2 - y1) / (x2 - x1));
    C = double((y2*x1 - y1*x2) / (x1 - x2));
else
    M = 0;
    C = double(x1);
end
```

Radon Line Check

```
function [There] = RadonIsThere(BWImage, Line, RadonContrast)
%   Line.X = [UpperLeft UpperRight]
%            [LowerLeft LowerRight]
%   Line.Y = [UpperRow]
%            [LowerRow]

ExtraArea = 1.0; % x the width each side.

%-----------------------------------------------------------------
-------
%Cuts the radon square from the image and fills in the outside
with zeros
%-----------------------------------------------------------------
-------
[Y X] = size(BWImage);
Width = (Line.X(1,2) + Line.X(2,2) - Line.X(1,1) - Line.X(2,1)) /
4; % Width of the painted line

%Finds the left and right most points so that the smaller tile can
be cut
%from the main image
if Line.X(1,1) < Line.X(2,1)
    LeftMost = int16(Line.X(1,1)-Width*ExtraArea);
else
    LeftMost = int16(Line.X(2,1)-Width*ExtraArea);
end
if Line.X(1,2) > Line.X(2,2)
    RightMost = int16(Line.X(1,2)+Width*ExtraArea);
else
```

```
    RightMost = int16(Line.X(2,2)+Width*ExtraArea);
end
if LeftMost < 1, LeftMost = 1; end
if RightMost > X, RightMost = X; end

%Cuts out the tile required
ImageArea = BWImage(int16(Line.Y(1)):int16(Line.Y(2)),
LeftMost:RightMost);

%Calculates coefficients for the lines that determine what the
area is
%that is to be used for the radon transforms
[Lm Lc] = GetMandC(Line.X(1,1)-Width*ExtraArea, Line.Y(1),
Line.X(2,1)-Width*ExtraArea, Line.Y(2));
[Rm Rc] = GetMandC(Line.X(1,2)+Width*ExtraArea, Line.Y(1),
Line.X(2,2)+Width*ExtraArea, Line.Y(2));
Interest = [Lm Lc;Rm Rc];
[Y X] = size(ImageArea);

%Makes zero all the pixels within the tile that fall outside the
area
%required
for i = Line.Y(1):Line.Y(2)
    [Left Right] = RoiValues (i, Interest);
    Left = floor(Left) - LeftMost + 1;
    Right = ceil(Right) - LeftMost + 1;
    for j = 1:X
        if (j <= Left) || (j >= Right)
            ImageArea(i-Line.Y(1)+1,j) = 0;
        end
    end
end

%-----------------------------------------------------------------
-------
% Finds where the different points of the radon transform start
and stop
%-----------------------------------------------------------------
-------
CP = floor((size(ImageArea)+1)/2); %CP = Centre of image tile
%Corrections to make 0,0 the centre of the image tile
ZeroCorX = -CP(2)-LeftMost;
ZeroCorY = -CP(1)-Line.Y(1);

%Calcluates line coefficients for the ROI lines using centre of
the image
%as 0,0
[Lm Lc] = GetMandC(Line.X(1,1)-Width*ExtraArea + ZeroCorX,
Line.Y(1)+ ZeroCorY, Line.X(2,1)-Width*ExtraArea + ZeroCorX,
Line.Y(2)+ ZeroCorY);
[Rm Rc] = GetMandC(Line.X(1,2)+Width*ExtraArea + ZeroCorX,
Line.Y(1)+ ZeroCorY, Line.X(2,2)+Width*ExtraArea + ZeroCorX,
Line.Y(2)+ ZeroCorY);
```

```
Angle = atan(((Line.X(1,1)+Line.X(1,2)/2) -
          (Line.X(2,1)+Line.X(2,2)/2))/(Line.Y(1)-Line.Y(2))) *
180/pi;
RadonLinePoint = [30*cos(Angle*pi/180) -30*sin(Angle*pi/180)];
Radm = RadonLinePoint(2)/RadonLinePoint(1); %Gradient of
projection axis

%Using simultaneous equations to find the points along the
projection
%axis where the line crosses them

if Lm == 0 %Vertical
    LeftIntX = double(Lc);
    LeftIntY = 0;
else
    LeftIntX = Lc / (Radm - Lm);
    LeftIntY = Lm*LeftIntX + Lc;
end
if Rm == 0 %Vertical
    RightIntX = double(Rc);
    RightIntY = 0;
else
    RightIntX = Rc / (Radm - Rm);
    RightIntY = Rm*RightIntX + Rc;
end

LeftOutsideRadius = round(-1*sqrt(LeftIntX^2 + LeftIntY^2)+1);
RightOutsideRadius = round(sqrt(RightIntX^2 + RightIntY^2));

[Lm Lc] = GetMandC(Line.X(1,1) + ZeroCorX, Line.Y(1)+ ZeroCorY,
Line.X(2,1)+ ZeroCorX, Line.Y(2)+ ZeroCorY);
[Rm Rc] = GetMandC(Line.X(1,2) + ZeroCorX, Line.Y(1)+ ZeroCorY,
Line.X(2,2)+ ZeroCorX, Line.Y(2)+ ZeroCorY);

if Lm == 0
    LeftIntX = double(Lc);
    LeftIntY = 0;
else
    LeftIntX = Lc / (Radm - Lm);
    LeftIntY = Lm*LeftIntX + Lc;
end
if Rm == 0
    RightIntX = double(Rc);
    RightIntY = 0;
else
    RightIntX = Rc / (Radm - Rm);
    RightIntY = Rm*RightIntX + Rc;
end

LeftLineRadius = round(-1*sqrt(LeftIntX^2 + LeftIntY^2)+1);
RightLineRadius = round(sqrt(RightIntX^2 + RightIntY^2));

[R,xp] = radon(ImageArea,Angle);

LeftOutsideRadiusValue = find(xp==LeftOutsideRadius);
RightOutsideRadiusValue = find(xp==RightOutsideRadius);
```

```
LeftLineRadiusValue = find(xp==LeftLineRadius);
RightLineRadiusValue = find(xp==RightLineRadius);

RadLeft = mean(R(LeftOutsideRadiusValue:LeftLineRadiusValue));
RadCentre = mean(R(LeftLineRadiusValue:RightLineRadiusValue));
RadRight = mean(R(RightLineRadiusValue:RightOutsideRadiusValue));

ProjectionContrast = RadCentre/max(RadLeft,RadRight);

if ProjectionContrast > RadonContrast
    There = true;
else
    There = false;
end
```

Bottom of Line Finder

```
function [LeftCord RightCord] = RadonDown(BWImage, Rectangles)
%Uses the lowest rectangle and extends the sides of it all the way
to the
%bottom of the image. The image is then radon transformed at an
angle
%perpendicular to the direction of the line. Based on the results
of this
%the actual end of the line can be more accurately determined.
[Y X] = size(BWImage);

TL = Rectangles(1).TL;
TR = Rectangles(1).TR;

[Lm Lc] =
GetMandC(Rectangles(1).BL(1),Rectangles(1).BL(2),Rectangles(1).TL(
1),Rectangles(1).TL(2));
BL = [round((480 - Lc) / Lm) 480];
[Rm Rc] =
GetMandC(Rectangles(1).BR(1),Rectangles(1).BR(2),Rectangles(1).TR(
1),Rectangles(1).TR(2));
BR = [round((480 - Rc) / Rm) 480];

if TL(1) < BL(1),  LeftMost = TL(1);
else LeftMost = BL(1); end

if TR(1) > BR(1),  RightMost = TR(1);
else RightMost = BR(1); end

if LeftMost < 1, LeftMost = 1; end
if RightMost > X, RightMost = X; end

ImageArea = BWImage(TR(2):BR(2), LeftMost:RightMost);
UsedArea =[Lm Lc;Rm Rc];
[Y X] = size(ImageArea);

for i = TL(2):BL(2)
    [Left Right] = RoiValues (i, UsedArea);
```

```
    Left = floor(Left) - LeftMost + 1;
    Right = ceil(Right) - LeftMost + 1;
    for j = 1:X
        if (j <= Left) || (j >= Right)
            ImageArea(i-TL(2)+1,j) = 0;
        end
    end
end

CP = floor((size(ImageArea)+1)/2); %CP = Centre of image tile
ZeroCorY = -CP(1)-Rectangles(1).TL(2);
Angle = atan((Rectangles(1).TC(1) -
         Rectangles(1).BC(1))/(Rectangles(1).TC(2)-
Rectangles(1).BC(1)));
[R,xp] = radon(ImageArea,-90);

First = 1;
Second = 1;
[Num Bin] = hist(R);
Num(1) = 0;




for i = 2:max(size(Num))
    if Num(i) > Num(First)
        Second = First;
        First = i;
    elseif Num(i) > Num(Second)
        Second = i;
    end
end

Cutoff = (Bin(First)+Bin(Second)) / 2;

[C,I] = max(R);
Difference = 9999;
Place = 0;

for i = I:max(size(R))
    CurrentDist = abs(R(i)-Cutoff);
    if CurrentDist < Difference
        Difference = CurrentDist;
        Place = i;
    end
end

YCord = xp(Place)*cos(Angle) - ZeroCorY;
[Left Right] = RoiValues (YCord, UsedArea);

LeftCord = [Left YCord];
RightCord = [Right YCord];
```

Top of Line Finder

```
function [LeftCord RightCord] = RadonUp(BWImage, Rectangle)
%Uses the topmost rectangle and extends the sides of it all the
way to
% the top of the image. The image is then radon transformed at an
angle
%perpendicular to the direction of the line. based on the results
of this
%the actual end of the line can be more accurately determined.

[Y X] = size(BWImage);
RecNum = max(size(Rectangle));
BL = Rectangle(RecNum).BL;
BR = Rectangle(RecNum).BR;

[Lm Lc] =
GetMandC(Rectangle(RecNum).BL(1),Rectangle(RecNum).BL(2),Rectangle
(RecNum).TL(1),Rectangle(RecNum).TL(2));
TL = [round((1 - Lc) / Lm) 1];
[Rm Rc] =
GetMandC(Rectangle(RecNum).BR(1),Rectangle(RecNum).BR(2),Rectangle
(RecNum).TR(1),Rectangle(RecNum).TR(2));
TR = [round((1 - Rc) / Rm) 1];

if TL(1) < BL(1),  LeftMost = TL(1);
else LeftMost = BL(1); end

if TR(1) > BR(1),  RightMost = TR(1);
else RightMost = BR(1); end


if LeftMost < 1, LeftMost = 1; end
if RightMost > X, RightMost = X; end

ImageArea = BWImage(TR(2):BR(2), LeftMost:RightMost);
UsedArea =[Lm Lc;Rm Rc];
[Y X] = size(ImageArea);

for i = TL(2):BL(2)
    [Left Right] = RoiValues (i, UsedArea);
    Left = floor(Left) - LeftMost + 1;
    Right = ceil(Right) - LeftMost + 1;
    for j = 1:X
        if (j <= Left) || (j >= Right)
            ImageArea(i-TL(2)+1,j) = 0;
        end
    end
end

CP = floor((size(ImageArea)+1)/2); %CP = Centre of image tile

Angle = atan((Rectangle(RecNum).TC(1) -
Rectangle(RecNum).BC(1))/(Rectangle(RecNum).TC(2)-
Rectangle(RecNum).BC(1)));
```

```
[R,xp] = radon(ImageArea,-90);

First = 1;
Second = 1;

[Num Bin] = hist(R);
Num(1) = 0;
for i = 2:max(size(Num))
    if Num(i) > Num(First)
        Second = First;
        First = i;
    elseif Num(i) > Num(Second)
        Second = i;
    end
end

Cutoff = (Bin(First)+Bin(Second)) / 2;

[C,I] = max(R);
Difference = 9999;
Place = 0;

for i = I:-1:1
    CurrentDist = abs(R(i)-Cutoff);
    if CurrentDist < Difference
        Difference = CurrentDist;
        Place = i;
    end
end

YCord = xp(Place)*cos(Angle) + CP(1);

[Left Right] = RoiValues (YCord, UsedArea);

LeftCord = [Left YCord];
RightCord = [Right YCord];
```

Location Finder

```
function Deviation = LocationFinder(Marking, Row)
% Finds the deviation of the line at the desired row in mm based on the
% width of the line in the image

LineThickness = 100; %in mm

if isnan(Marking(1).BL(1))
    Deviation = NaN;
else
    if Marking(1).BL(2) == 480 || Marking(1).BL(2) == 479  % If the botom
row has a line present
        CurrentCentre = Marking(1).BC(1);
        CurrentLeft = Marking(1).BL(1);
        CurrentRight = Marking(1).BR(1);
    else
```

```
        [M C] =
GetMandC(Marking(1).BC(1),Marking(1).BC(2),Marking(1).TC(1),Marking(1).TC
(2));
        if M == 0
            CurrentCentre = C;
        else
            CurrentCentre = (480 - C) / M;
        end
        [M C] =
GetMandC(Marking(1).BL(1),Marking(1).BL(2),Marking(1).TL(1),Marking(1).TL
(2));
        if M == 0
            CurrentLeft = C;
        else
            CurrentLeft = (480 - C) / M;
        end
        [M C] =
GetMandC(Marking(1).BR(1),Marking(1).BR(2),Marking(1).TR(1),Marking(1).TR
(2));
        if M == 0
            CurrentRight = C;
        else
            CurrentRight = (480 - C) / M;
        end
    end
    Ratio = LineThickness / (CurrentLeft - CurrentRight);
    Deviation = (320-CurrentCentre)*Ratio;
end
```

Appendix 3 – Testing Program

```
function ImageAndVideoTester
%   Program used to debug and test the use of the image processing
%   algorithms. It tests itself on images, videos and also live streaming
%   from an attached web cam if one is installed and the correct dlls
%   are installed
global handles

Window = findobj('tag','MainWindow');
if (isempty(Window)),

    Window = figure(...
            'numbertitle'                   ,'off',...
            'name'                          ,'Image and Video Tester',...
            'MenuBar'                       ,'none',...
            'doublebuffer'                  ,'on',...
            'CloseRequestFcn'               ,@close,...
            'tag'                           ,'MainWindow',...
            'defaultuicontrolfontsize'      , 12, ...
            'defaultuicontrolunits'                 , 'pixels', ...
            'defaultuicontrolbackgroundcolor'       , [.8 .8 .8], ...
            'defaultuicontrolHorizontalAlignment'   ,'Left',...
            'KeyPressFcn'                           ,@KeyPressed,...
            'Position'                       ,[200 150 950 600]);

    MainMenu = uimenu('Label','File','Tag','TheMainMenu');
        uimenu(MainMenu,'Label','Exit','Callback',@Close);

    CameraProcessingMenu = uimenu('Label','Video
Camera','Tag','TheVCMenu');
```

```
        uimenu(CameraProcessingMenu,'Label','Test Camera
Aquisition','Callback',@TestCameraCon,'Tag','TEST','enable','off','separa
tor','on');

    axes(...
            'Units'                         ,'pixels', ...
            'box'                           ,'on', ...
            'xlim'                          ,[0 640], ...
            'xtick'                         ,[], ...
            'ylim'                          ,[0 480], ...
            'ytick'                         ,[], ...
            'Parent'                        ,Window, ...
            'HandleVisibility'              ,'callback', ...
            'Position'                      ,[290 100 640 480], ...
            'Tag'                           ,'ImageDisplayPanel');

##########################################################################
%           Buttons
##########################################################################
    uicontrol(...
            'style'                         ,'pushbutton', ...
            'string'                        ,'Start', ...
            'callback'                      ,@Start, ...
            'enable'                        ,'off', ...
            'parent'                        , Window, ...
            'Position'                      ,[120 555 150 25], ...
            'tag'                           ,'StartBtn');

    uicontrol(...
            'style'                         ,'pushbutton', ...
            'string'                        ,'Stop', ...
            'callback'                      ,@Stop, ...
            'enable'                        ,'off', ...
            'parent'                        , Window, ...
            'Position'                      ,[120 525 150 25], ...
            'tag'                           ,'StopBtn');

    uicontrol(...
            'style'                         ,'pushbutton', ...
            'string'                        ,'Load', ...
            'callback'                      ,@Load, ...
            'enable'                        ,'on', ...
            'parent'                        , Window, ...
            'Position'                      ,[120 495 150 25], ...
            'tag'                           ,'LoadBtn');

##########################################################################
%           Radio Buttons
##########################################################################
    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[20 555 100 20],...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'radiobutton', ...
            'callback'                      ,@ImageRadioPressed, ...
            'String'                        ,'Images',...
            'Value'                         ,1,...
            'Tag'                           ,'ImageRadioBtn');

    uicontrol(...
```

```
            'Parent'                        , Window, ...
            'Position'                      ,[20 530 100 20],...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'radiobutton', ...
            'callback'                      ,@VideoRadioPressed, ...
            'String'                        ,'Video',...
            'Tag'                           ,'VideoRadioBtn');

    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[20 505 100 20],...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'radiobutton', ...
            'callback'                      ,@CameraRadioPressed, ...
            'String'                        ,'Camera',...
            'Tag'                           ,'CameraRadioBtn');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Radio Buttons 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[20 255 100 20],...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'radiobutton', ...
            'callback'                      ,@RectanglesRadioPressed, ...
            'String'                        ,'Rectangles',...
            'Value'                         ,1,...
            'Tag'                           ,'RectanglesRadioBtn');

    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[20 230 120 20],...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'radiobutton', ...
            'callback'                      ,@SingleRadonRadioPressed, ..
            'String'                        ,'Single Radon',...
            'Tag'                           ,'SingleRadonRadioBtn');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Text Boxes and Strings for Input
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[20 440 100 20],...
            'String'                        ,'Threshold:',...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'text', ...
            'Tag'                           ,'txtThreshold');
    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[170 440 100 20],...
            'HandleVisibility'              ,'callback', ...
            'String'                        ,'50',...
            'Style'                         ,'edit', ...
            'backgroundcolor'               ,'white', ...
            'Tag'                           ,'edtThreshold');
```

```
 uicontrol(...
        'Parent'                    , Window, ...
        'Position'                  ,[20 415 150 20],...
        'String'                    ,'Averaging Distance:',...
        'HandleVisibility'          ,'callback', ...
        'Style'                     ,'text', ...
        'Tag'                       ,'txtAveragingDist');
uicontrol(...
        'Parent'                    , Window, ...
        'Position'                  ,[170 415 100 20],...
        'HandleVisibility'          ,'callback', ...
        'String'                    ,'150',...
        'Style'                     ,'edit', ...
        'backgroundcolor'           ,'white', ...
        'Tag'                       ,'edtAveragingDist');

uicontrol(...
        'Parent'                    , Window, ...
        'Position'                  ,[20 390 150 20],...
        'String'                    ,'Rectangle Height:',...
        'HandleVisibility'          ,'callback', ...
        'Style'                     ,'text', ...
        'Tag'                       ,'txtRectangleHeight');
uicontrol(...
        'Parent'                    , Window, ...
        'Position'                  ,[170 390 100 20],...
        'HandleVisibility'          ,'callback', ...
        'String'                    ,'80',...
        'Style'                     ,'edit', ...
        'backgroundcolor'           ,'white', ...
        'Tag'                       ,'edtRectangleHeight');

uicontrol(...
        'Parent'                    , Window, ...
        'Position'                  ,[20 365 100 20],...
        'String'                    ,'ROI Distance:',...
        'HandleVisibility'          ,'callback', ...
        'Style'                     ,'text', ...
        'Tag'                       ,'txtROIDist');
uicontrol(...
        'Parent'                    , Window, ...
        'Position'                  ,[170 365 100 20],...
        'HandleVisibility'          ,'callback', ...
        'String'                    ,'1',...
        'Style'                     ,'edit', ...
        'backgroundcolor'           ,'white', ...
        'Tag'                       ,'edtROIDist');

uicontrol(...
        'Parent'                    , Window, ...
        'Position'                  ,[20 340 100 20],...
        'String'                    ,'Radon Ratio:',...
        'HandleVisibility'          ,'callback', ...
        'Style'                     ,'text', ...
        'Tag'                       ,'txtRadonRatio');
uicontrol(...
        'Parent'                    , Window, ...
        'Position'                  ,[170 340 100 20],...
        'HandleVisibility'          ,'callback', ...
        'String'                    ,'1.4',...
        'Style'                     ,'edit', ...
```

```
            'backgroundcolor'               ,'white', ...
            'Tag'                           ,'edtRadonRatio');

    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[20 315 200 20],...
            'String'                        ,'LineWidth  Top:',...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'text', ...
            'Tag'                           ,'txtLineWidthTop');
    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[130 315 40 20],...
            'HandleVisibility'              ,'callback', ...
            'String'                        ,'50',...
            'Style'                         ,'edit', ...
            'backgroundcolor'               ,'white', ...
            'Tag'                           ,'edtLineWidthTop');

    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[175 315 100 20],...
            'String'                        ,'Bottom:',...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'text', ...
            'Tag'                           ,'txtLineWidthBot');

    uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[230 315 40 20],...
            'HandleVisibility'              ,'callback', ...
            'String'                        ,'80',...
            'Style'                         ,'edit', ...
            'backgroundcolor'               ,'white', ...
            'Tag'                           ,'edtLineWidthBot');

##########################################################################
%           Strings For Results
##########################################################################

     uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[290 70 300 20],...
            'String'                        ,'',...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'text', ...
            'Tag'                           ,'txtTime');

     uicontrol(...
            'Parent'                        , Window, ...
            'Position'                      ,[290 40 300 20],...
            'String'                        ,'',...
            'HandleVisibility'              ,'callback', ...
            'Style'                         ,'text', ...
            'Tag'                           ,'txtLocation');

    handles = guihandles(Window);
    handles.Source = 'Image';
    handles.Image = [];
    handles.Video = [];
```

```
    handles.Stop = true;
    handles.Single = false;
    handles.LoadLocation = pwd;

else
    figure(Window);
    set(findobj('Tag','TheMainMenu'),'enable','on');
end

function ImageRadioPressed(hco,eventStruct)
global handles
set(findobj('Tag','VideoRadioBtn'),'Value',0);
set(findobj('Tag','CameraRadioBtn'),'Value',0);
set(findobj('Tag','StopBtn'),'Enable', 'off');
set(findobj('Tag','LoadBtn'),'String','Load');
handles.Source = 'Image';
handles.Stop = true;
clear vcapg2;

function VideoRadioPressed(hco,eventStruct)
global handles
set(findobj('Tag','ImageRadioBtn'),'Value',0)
set(findobj('Tag','CameraRadioBtn'),'Value',0)
set(findobj('Tag','StopBtn'),'Enable', 'on');
set(findobj('Tag','LoadBtn'),'String','Load');
handles.Source = 'Video';
handles.Stop = true;
clear vcapg2;

function CameraRadioPressed(hco,eventStruct)
global handles
set(findobj('Tag','VideoRadioBtn'),'Value',0)
set(findobj('Tag','ImageRadioBtn'),'Value',0)
set(findobj('Tag','StopBtn'),'Enable', 'on');
set(findobj('Tag','LoadBtn'),'String','Connect');
handles.Source = 'Camera';

function RectanglesRadioPressed(hco,eventStruct)
global handles
set(findobj('Tag','SingleRadonRadioBtn'),'Value',0);
handles.Single = false;

function SingleRadonRadioPressed(hco,eventStruct)
global handles
set(findobj('Tag','RectanglesRadioBtn'),'Value',0);
handles.Single = true;

function clearImageDisplayPanel
% Removes Axis markings on image display
global handles
set(handles.ImageDisplayPanel,'xtick',[]);
set(handles.ImageDisplayPanel,'ytick',[]);

function Start(hco,eventStruct)
global handles

try
    handles.Threshold =
str2double(get(findobj('Tag','edtThreshold'),'string'));
```

```
    handles.AveragingDist =
str2double(get(findobj('Tag','edtAveragingDist'),'string'));
    handles.RectangleHeight =
str2double(get(findobj('Tag','edtRectangleHeight'),'string'));
    handles.RoiDist =
str2double(get(findobj('Tag','edtROIDist'),'string'));
    handles.RadonRatio =
str2double(get(findobj('Tag','edtRadonRatio'),'string'));
    handles.TopWidth =
str2double(get(findobj('Tag','edtLineWidthTop'),'string'));
    handles.BottomWidth =
str2double(get(findobj('Tag','edtLineWidthBot'),'string'));
catch
    warndlg('Incorrect Varaible');
    return
end




% ROI is the region of interest and is used when a sequence of images are
% being looked at as the line will not have moved significantly from
% frame to frame. It is arranged as so:
%
%                 [Lm(Left Line Gradient)  Lc(Left Line Zero Crossing)  ]
%       ROI   =   [Rm(Right Line Gradient) Rc(Right Line Zero Crossing) ]
%                 [Angle of overall line   NaN                          ]
%
ROI = [NaN NaN; NaN NaN; NaN NaN];

switch handles.Source
    case 'Image'

        [Marking ROI Time] = ProcessDisplayFrame(handles.Image, 1,
                                                          handles, ROI);
        tic
        MoveDist = LocationFinder(Marking);
        Time = Time + toc;

        set(findobj('Tag','txtTime'),'string',['Time taken:  '
                                                      num2str(Time)]);
        set(findobj('Tag','txtLocation'),'string',['Move:  '
                                              num2str(MoveDist) ' mm']);

    case 'Video'

        Length = max(size(handles.Video));
        Total = zeros(Length);
        for i=1:Length
            [Marking ROI Time] = ProcessDisplayFrame(handles.Video(i), 1,
                                                          handles, ROI);
            tic
            MoveDist = LocationFinder(Marking);
            Time = Time + toc;
            set(findobj('Tag','txtLocation'),'string',['Move:  '
                                              num2str(MoveDist) ' mm']);
            if i > 1
                Total(i) = Time;
            end
        end
```

```
        Ave = sum(Total) / max(size(handles.Video));

        set(findobj('Tag','txtTime'),'string',['Average Time taken:  '
                                                    num2str(Ave)]);

    case 'Camera'
        handles.Stop = false;
        Total = 0;
        Frames = 0;

        while (~handles.Stop),
            a = vcapg2;
            colormap(gray);
            b = rgb2bw(a);

            imshow(b, 'Parent', handles.ImageDisplayPanel);
            axis off;

            [Marking ROI Time] = ProcessDisplayFrame(b, 1, handles, ROI);
            tic
            MoveDist = LocationFinder(Marking);
            Time = Time + toc;
            Total = Total + Time;
            Frames = Frames + 1;

            drawnow;
            set(findobj('Tag','txtTime'),'string',['Time taken:  '
                                                num2str(Time)]);
            set(findobj('Tag','txtLocation'),'string',['Move:  '
                                            num2str(MoveDist) ' mm']);
        end

        Ave = Total/Frames;
        set(findobj('Tag','txtTime'),'string',['Average Time taken:  '
                                                    num2str(Ave)]);

    otherwise
                disp('No Method Selected');
end

function Stop(hco,eventStruct)
global handles
handles.Stop = true;

function Load(hco,eventStruct)
global handles

switch handles.Source
    case 'Image'
        try
            [FileName,PathName,FilterIndex] =
                        uigetfile('.jpg','mytitle',handles.LoadLocation);
            Image = imread(strcat(PathName,FileName));
            handles.LoadLocation = PathName;
        catch
            warndlg('Failed to load image');
            return
        end
        Dim = size(Image,3);
```

```
        if Dim == 3
            Image = rgb2bw(Image);
        end
        imshow(Image, 'Parent', handles.ImageDisplayPanel);
        clearImageDisplayPanel;
        handles.Image = rgb2bw(Image);
        set(findobj('Tag','StartBtn'),'enable', 'on');

    case 'Video'
        try
            [FileName,PathName,FilterIndex] =
                        uigetfile('.avi','mytitle',handles.LoadLocation);
            Video = LoadMovie(strcat(PathName,FileName));
            handles.LoadLocation = PathName;
        catch
            warndlg('Failed to load video');
            return
        end
        handles.Video = Video;
        set(findobj('Tag','StartBtn'),'enable', 'on');
    case 'Camera'
        if (exist('vcapg2.dll'))
            % initialise the camera
            vcapg2();
            try
                a = vcapg2; % capture an image
            catch
                warndlg('Camera is not connected');
                return
            end
            % check size
            [n_fil, n_col, n_dim] = size(a);

            if (n_fil ~= 480)
                uiwait(warndlg({'The camera is ok but...','change the
                resolution to 620x480 !'},'Resolution Error','modal'));
            end

            % everything is correct
            set(findobj('Tag','TEST'),'enable','on');
            set(findobj('Tag','StartBtn'),'enable', 'on');
        else
            uiwait(warndlg('vcapg2.dll is not on the
                                        path...','Error','modal'));
        end
    otherwise
        disp('No Method Selected');
end

function Close(hco,eventStruct)
global handles

handles.Stop = true;
clear vcapg2;
closereq;

function KeyPressed(src,evnt)
global handles
k = evnt.Key;
if strcmp(k,'escape')
```

```
    handles.Stop = true;
end

function TestCameraCon(hco,eventStruct)
global handles

% watch what is being captured
handles.Stop = false;

while (~handles.Stop),
    a = vcapg2;
    b = rgb2bw(a);
    image(b, 'Parent', handles.ImageDisplayPanel);
    axis off;
    drawnow;
    refresh;
end



function TestCamera(hco,eventStruct)

Window = findobj('tag','MainWindow');

set(findobj('Tag','STOP'),'enable','on');

figure(Window);
subplot(1,1,1);
colormap(gray);
images = 0;
ROI = [NaN NaN; NaN NaN];

halt = findobj('Tag','STOP');
t0 = clock;
while (strcmp(get(halt,'enable'),'on')),
    % capture image
    a = vcapg2;
    b = rgb2gray(a);
    image(a);
    axis off;

    [Marking ROI Time] = FunctROI(b, ROI);
    MarkingSize = max(size(Marking));
    hold on
    if ~isnan(Marking(1).BL(1))
        for j = 1: MarkingSize
            line([Marking(j).BL(1) Marking(j).TL(1)],[Marking(j).BL(2)
Marking(j).TL(2)],'color','r'); %Left Line
             line([Marking(j).BR(1) Marking(j).TR(1)],[Marking(j).BR(2)
Marking(j).TR(2)],'color','r'); %Right Line
             line([Marking(j).BC(1) Marking(j).TC(1)],[Marking(j).BC(2)
Marking(j).TC(2)],'color','r', 'linestyle','--'); %Centre Line
             line([Marking(j).BL(1) Marking(j).BR(1)],[Marking(j).BL(2)
Marking(j).BR(2)],'color','r'); %Bottom Line
             line([Marking(j).TL(1) Marking(j).TR(1)],[Marking(j).TL(2)
Marking(j).TR(2)],'color','r'); %Top Line
        end
        if ~isnan(ROI(1,1))
             [LeftBotROI RightBotROI] = RoiValues (Marking(1).BL(2), ROI);
             [LeftTopROI RightTopROI] = RoiValues
(Marking(MarkingSize).TL(2), ROI);
```

```
            line([LeftBotROI LeftTopROI],[Marking(1).BL(2)
Marking(MarkingSize).TL(2)],'color','r', 'linestyle','--');
            line([RightBotROI RightTopROI],[Marking(1).BL(2)
Marking(MarkingSize).TL(2)],'color','r', 'linestyle','--');
        end
        if Marking(1).BL(2) == 480
            CurrentPosition = Marking(1).BC(1);
        else
            [M C] =
GetMandC(Marking(1).BC(1),Marking(1).BC(2),Marking(1).TC(1),Marking(1).TC
(2));
            CurrentPosition = (480 - C) / M;
        end
        k = 400;
        l = 0;
        while k > 380
            l = l + 1;
            k = Marking(l).TL(2);
        end
        [M C] =
GetMandC(Marking(l).BC(1),Marking(l).BC(2),Marking(l).TC(1),Marking(l).TC
(2));
        FuturePosition = (480 - C) / M;
        title(['Current Position: ' num2str(320 - CurrentPosition) '
Future Position: ' num2str(320 - FuturePosition)]);

    end
    hold off

    images = images +1;
    drawnow; %flushes out event queue
    refresh;

end
delta_t = etime(clock,t0);
title([num2str(images/delta_t) ' frames/s']);
set(findobj('Tag','STOP'),'enable','off');
```

Frame Processing

```
function  [Marking ROI Time] = ProcessDisplayFrame(Image, FigureNo,
handles, ROI)

figure(FigureNo);

if isstruct(Image)
    Image = [Image(1,:).BWFrame];
end
imshow(Image);
axis off;
tic
newimage = imadjust(Image);
% newimage = Image;
Time1 = toc;

if handles.Single
    [Marking ROI Time2] = SingleRadon(newimage, ROI, handles.Threshold,
handles.RectangleHeight,...
```

```
                    handles.AveragingDist, handles.RoiDist,
handles.RadonRatio, handles.TopWidth, handles.BottomWidth);
else
    [Marking ROI Time2] = LineBeginning(newimage, ROI, handles.Threshold,
handles.RectangleHeight,...
                    handles.AveragingDist, handles.RoiDist,
handles.RadonRatio, handles.TopWidth, handles.BottomWidth);
end

Time = Time1 +Time2;

if ~isnan(Marking(1).BL(1))
    figure(FigureNo)
%     figure(2);
%     set(figure(2), 'color', 'w')
%     imshow(Image)
    hold on
    for j = 1:max(size(Marking))
        line([Marking(j).BL(1) Marking(j).TL(1)],[Marking(j).BL(2)
Marking(j).TL(2)],'color','r','LineWidth',2); %Left Line
        line([Marking(j).BR(1) Marking(j).TR(1)],[Marking(j).BR(2)
Marking(j).TR(2)],'color','r','LineWidth',2); %Right Line
%           line([Marking(j).BC(1) Marking(j).TC(1)],[Marking(j).BC(2)
Marking(j).TC(2)],'color','r', 'linestyle','--'); %Centre Line
        line([Marking(j).BL(1) Marking(j).BR(1)],[Marking(j).BL(2)
Marking(j).BR(2)],'color','r','LineWidth',2); %Bottom Line
        line([Marking(j).TL(1) Marking(j).TR(1)],[Marking(j).TL(2)
Marking(j).TR(2)],'color','r','LineWidth',2); %Top Line
    end

%     if ~isnan(ROI(1,1))         % Shows the ROI's
%         [LeftBotROI RightBotROI] = RoiValues (Marking(1).BL(2), ROI);
%         [LeftTopROI RightTopROI] = RoiValues
(Marking(max(size(Marking))).TL(2), ROI);
%         line([LeftBotROI LeftTopROI],[Marking(1).BL(2)
Marking(max(size(Marking))).TL(2)],'color','r', 'linestyle','--',
'LineWidth', 1.5);
%         line([RightBotROI RightTopROI],[Marking(1).BL(2)
Marking(max(size(Marking))).TL(2)],'color','r', 'linestyle','--',
'LineWidth', 1.5);
%     end

end
hold off
```