

NAVIGATION FOR AUTOMATIC GUIDED VEHICLES USING OMNIDIRECTIONAL OPTICAL SENSING

presented by

BENJAMIN JOHANNES KOTZE

Thesis submitted in fulfilment of the requirements for the degree

DOCTOR TECHNOLOGIAE: ENGINEERING: ELECTRICAL

in the

Department of Electrical, Electronic and Computer Engineering

of the

Faculty of Engineering and Information Technology

at the

Central University of Technology, Free State

Promoter: Prof. G.D. Jordaan, DTech (Eng.)

Co-Promoter: Prof. H.J. Vermaak, PhD (Eng.)

Bloemfontein

December 2013

Declaration

I, BENJAMIN JOHANNES KOTZE, identity number [REDACTED], and student number 9326855, do hereby declare that this research project which has been submitted to the Central University of Technology Free State, for the degree DOCTOR TECHNOLOGIAE: ENGINEERING: ELECTRICAL, is my own independent work and complies with the Code of Academic Integrity, as well as other relevant policies, procedures, rules and regulations of the Central University of Technology, Free State, and has not been submitted before by any person in fulfilment (or partial fulfilment) of the requirements for the attainment of any qualification.

.....
SIGNATURE OF STUDENT

.....
DATE

Acknowledgements

I would like to thank the following persons and institutions for their unselfish assistance and support during the research project:

My promoter, Prof. Jorrie Jordaan, for his guidance and assistance during the research period and co-promoter, Prof. Herman Vermaak, for his contribution to the final product.

The Central University of Technology, Free State for the indirect help and support that made my studies possible.

The Research Group in Evolvable Manufacturing Systems for the equipment and assistance throughout the research period.

Piet Swanepoel for doing research at CUT on a related topic.

My colleagues, Pieter Veldtsman for help with the development of the microcontroller board, Dr Nicolaas Luwes for exchanging thoughts and ideas and Johan Niemann for the development of the DELMIA system.

A word of thanks to De Ville Weppenaar for the initial development of the Bibliography style, Yves Dhondt for the original .xsl and .xml files to use and edit, and Adriaan Nel for his guidance in developing the final product in this regard.

Finally, and especially my wife, Elseri, and my children, Dirko and Linandri that have suffered the loss of so much attention and time during the long period of study.

“The only place success comes before work is in the dictionary.”

Vince Lombardi
(1913-1970)

Abstract

Automatic Guided Vehicles (AGVs) are being used more frequently in a manufacturing environment. These AGVs are navigated in many different ways, utilising multiple types of sensors for detecting the environment like distance, obstacles, and a set route. Different algorithms or methods are then used to utilise this environmental information for navigation purposes applied onto the AGV for control purposes. Developing a platform that could be easily reconfigured in alternative route applications utilising vision was one of the aims of the research.

In this research such sensors detecting the environment was replaced and/or minimised by the use of a single, omnidirectional Webcam picture stream utilising an own developed mirror and Perspex tube setup. The area of interest in each frame was extracted saving on computational recourses and time. By utilising image processing, the vehicle was navigated on a predetermined route.

Different edge detection methods and segmentation methods were investigated on this vision signal for route and sign navigation. Prewitt edge detection was eventually implemented, Hough transfers used for border detection and Kalman filtering for minimising border detected noise for staying on the navigated route.

Reconfigurability was added to the route layout by coloured signs incorporated in the navigation process. The result was the manipulation of a number of AGV's, each on its own designated coloured signed route. This route could be reconfigured by the operator with no programming alteration or intervention. The YCbCr colour space signal was implemented in detecting specific control signs for alternative colour route navigation.

The result was used generating commands to control the AGV through serial commands sent on a laptop's Universal Serial Bus (USB) port with a PIC microcontroller interface board controlling the motors by means of pulse width modulation (PWM).

A total MATLAB[®] software development platform was utilised by implementing written M-files, Simulink[®] models, masked function blocks and .mat files for sourcing the workspace variables and generating executable files. This continuous development

system lends itself to speedy evaluation and implementation of image processing options on the AGV.

All the work done in the thesis was validated by simulations using actual data and by physical experimentation.

Abstrak

Geoutomatiseerde Geleide Voertuie (GGVs) word al hoe meer dikwels gebruik in 'n produksie-omgewing. Hierdie GGV's navigeer op baie verskillende maniere, met behulp van verskeie vorme van sensors vir die identifisering van hul omgewing soos afstand, hindernisse en 'n vasgestelde roete. Verskillende algoritmes of metodes word dan gebruik om hierdie omgewingsinligting vir navigasie toe te pas op die GGV en vir beheer doeleindes aan te wend. Ontwikkeling van 'n platform wat maklik aangepas kan word vir die gebruik op alternatiewe roete toepassings deur gebruik te maak van visie was een van die doelwitte van die navorsing.

In hierdie navorsing is hierdie omgewingsidentifiseringsensors vervang en/of verminder deur gebruik te maak van 'n enkele, omnidireksionele kameraprentjie stroom met 'n eie ontwikkelde spieël en perspexbuis opstelling. Die area van belang in elke prentjie raam is benut vir 'n besparing op rekenaarhulpbronne en prosesseringstyd. Deur gebruik te maak van beeldverwerking is die voertuig genavigeer op 'n voorafbepaalde roete.

Verskillende rand-opsporingmetodes en segmenteringsmetodes is ondersoek op hierdie visie sein vir roete- en tekennavigasie. Prewitt randopsporing is uiteindelik geïmplementeer, Hough oordragfunksies is gebruik vir die grens-opsporing en Kalman filtrering vir die vermindering van die grens opgespoor geraas om op die roete te bly navigeer.

Herprogrammeerbaarheid is bygevoeg in die roete-uitleg deur van gekleurde tekens in die navigasie-proses gebruik te maak. Die resultaat was die manipulasie van 'n aantal GGV's, elk op sy eie aangewese gekleurde-teken roete. Hierdie roete kan aangepas word deur die operateur met geen programmeringsverandering of -ingryping nie. Die YCbCr kleurkaartsein is geïmplementeer in die opsporing van spesifieke beheer kleur tekens vir 'n alternatiewe roete navigasie.

Die navigasie uitkoms is gebruik om die bevele te genereer vir die beheer van die GGV deur seriaal die beheer opdragte vanaf 'n skootrekenaar te stuur op die Universele Seriële Bus (USB) poort met 'n PIC mikrobeheerderkoppelvlakbord vir die beheer van die motors deur middel van pulswydte modulاسie (PWM).

In totaliteit is 'n MATLAB[®] sagteware-ontwikkelingsplatform gebruik deur die implementering van geskrewe M-lêers, Simulink[®] -modelle, gemaskerde funksieblokke en .matlêers vir die voorsiening van die werkplekveranderlikes en generering van uitvoerbare lêers. Hierdie voortdurende ontwikkelingstelsel leen hom tot vinnige evaluering en implementering van die beeldverwerking opsies deur die GGV.

Al die werk wat gedoen is in die proefskrif is bevestig deur simulاسies met behulp van werklike data en deur fisiese eksperimentering.

Table of Contents

Declaration.....	ii
Acknowledgements.....	iii
Abstract.....	iv
Abstrak.....	vi
List of Figures.....	xii
List of Tables	xix
Statement	xx
1 Introduction	1
1.1 Preface	1
1.2 Motivation and objective of thesis	2
1.3 Hypothesis	2
1.4 Methodology of research.....	2
1.5 Outline of thesis.....	4
2 Omnidirectional vision, AGV navigation and control	5
2.1 Introduction	5
2.2 Vision concept.....	7
2.3 Omnidirectional sensing.....	8
2.3.1 The Taylor model.....	10
2.3.2 Calibration of omnidirectional pictures	11
2.4 Edge detection for object and route recognition	12
2.4.1 Roberts operator.....	13
2.4.2 Laplace operator.....	14
2.4.3 Prewitt operator.....	14
2.4.4 Sobel operator	15
2.4.5 Robinson operator.....	15

2.4.6	Kirsch operator.....	15
2.4.7	Canny edge detection.....	16
2.4.8	Dilation and erosion.....	17
2.5	Techniques used for tracking and detecting objects utilising vision.....	17
2.5.1	Colour space conversion.....	18
2.5.2	Segmentation.....	20
2.5.3	Correlation.....	21
2.5.4	Bounding boxes.....	22
2.5.5	Optical flow.....	22
2.5.6	Hough transform.....	23
2.5.7	Kalman filter.....	26
2.5.8	Neural networks.....	27
2.5.9	Genetic algorithms.....	29
2.6	AGV platform, navigation and control.....	30
2.6.1	Dead reckoning.....	31
2.6.2	Ultrasonic triangulation.....	32
2.6.3	Control and avoidance.....	32
2.6.4	Path navigation.....	32
2.6.5	Sign navigation.....	34
2.7	Summary.....	36
3	Development of an omnivision system for navigational purposes for an AGV	37
3.1	Introduction.....	37
3.2	Mirror and camera development for omnidirectional sensing.....	38
3.2.1	Improvement on previous omnidirectional design.....	43
3.3	Development of omnidirectional sensing software.....	47
3.4	Area of interest and utilising a low resolution webcam.....	51
3.5	Transferring the omnisoftware from computer to laptop platform.....	54

3.6	Conclusion.....	58
4	Navigation development for the AGV	59
4.1	Identifying the navigational goals	59
4.2	Detection of movement whilst navigating utilising dead reckoning.....	60
4.3	Development of a new AGV platform	62
4.4	Overview of the vision guided navigation system	64
4.5	Route navigation concept	65
4.5.1	Short description of the MATLAB [®] 's Chroma-based Road Tracking demo which was the starting point for road navigation in this research	65
4.5.2	Edge detection and chroma segmentation used for AGV route tracking.....	66
4.5.3	Border detection for route identification on edge and chroma signal.....	68
4.5.4	Route Tracking and Route Merging	72
4.5.5	Display of detected line, edge, chroma and tracking information for evaluation purposes	75
4.6	PC to motor speed control interface	79
4.6.1	AGV controls generated from the visual route navigation system	81
4.6.2	Obtained speeds for the AGVs used	84
4.7	Sign recognition	86
4.7.1	Sign recognition templates.....	87
4.7.2	Detection of signs	89
4.7.3	Tracking and recognising the signs.....	90
4.7.4	Displaying the recognition results	90
4.7.5	Investigating different coloured routes	91
4.7.6	Implementing sign detection command control.....	94
4.8	Conclusion.....	98
5	Results	99
5.1	Omnivision system results.....	99

5.2	AGV platform results	106
5.3	Navigation and control system	108
5.4	Reconfigurable ability of the AGV	110
5.5	Summary	116
5.5.1	Possible improvements of the refraction and reflection of the mirror and camera setup.....	117
5.5.2	Possible improvements on the vision navigation and control system....	117
5.5.3	Conclusion with regard to the relationship between – webcam resolution, template resolution and distance to a sign	119
6	Conclusion	120
6.1	Summary	120
6.2	Original contributions.....	121
6.3	Evaluation of, and the conclusion of the vision system	121
6.4	Evaluation of, and the conclusion of a navigation interface	122
6.5	Assembly of different systems in a single platform	123
6.6	Future research	123
	Appendix A – Colour inset	125
	Appendix B.....	129
	B.1 Compiling an standalone executable file utilising the <i>mcc</i> command in MATLAB [®] from an m-file	129
	References.....	131

List of Figures

Figure 1.1: Outline of thesis including research phases	4
Figure 2.1: Flowchart of research outline as seen for the research process.....	5
Figure 2.2: Flowchart of research concepts and routes taken for reaching the research goal placing the work discussed in Chapter 2 in context.....	6
Figure 2.3: The loss of size and depth perception on a 2D image	7
Figure 2.4: Levels of image processing used in the identification of objects for processing.....	8
Figure 2.5: AGV with hyperbolic mirror setup	9
Figure 2.6: Example of a vision system satisfying the single viewpoint property of an omnidirectional camera with a hyperbolic mirror [4, p. 11]	9
Figure 2.7: (a) Coordinate system in catadioptric case; (b) Sensor plane and conversion	10
Figure 2.8: Camera setup of Aliaga with parabolic mirror and acrylic half sphere on a video camera [11]	11
Figure 2.9: Aliaga's model, which allows accurate computation between the focal- and 3D point.....	12
Figure 2.10: (a) Original image with edges due to different phenomena; (b) Detected edges by means of the Sobel operator.....	13
Figure 2.11: Results obtained by utilising different operators viewing only a section of the image used in Figure 2.10(a) – (a) Roberts operator result; (b) Prewitt operator result; (c) Canny edge result	16
Figure 2.12: (a) Original binary image; (b) Image with 3-pixel dilation; (c) Image with 3-pixel erosion; (d) Edge detection by subtracting the eroded image from the original	17
Figure 2.13: CIE chromaticity diagram 1931 [17].....	18
Figure 2.14: RGB colour space with primary and secondary colours indicating grey scale	19
Figure 2.15: HSV colour model illustrated as a cylinder [18].....	19
Figure 2.16: Different shaped parts detection from a noisy image, with different segmentation models	21

Figure 2.17: Segmentation by correlation; matched pattern with location of best match	21
Figure 2.18: Optical flow of a moving tennis ball, (a) time t_1 ; (b) time t_2 ; (c) optical flow vectors	23
Figure 2.19: (a) PCB with capacitor; (b) edges detected; (c) Hough accumulator array using edge points and orientation; (d) circle detected by thresholding and local maxima.....	24
Figure 2.20: Hough parameters for a straight line	25
Figure 2.21: Points in a Hough array plotted with different (r, θ) values.....	26
Figure 2.22: Hough space graph plotted from several (r, θ) points.....	26
Figure 2.23: Results of a system utilising the Kalman filter – solid line the predicted result, + indicates noise [28].....	27
Figure 2.24: A simple (McCulloch-Pitts) neuron	28
Figure 2.25: A three-layered neural net structure example with four inputs and three outputs [6, p. 406].....	29
Figure 2.26: Combined flowchart representing the genetic algorithm steps	30
Figure 2.27: Notation of variables in a dead reckoning setup on an AGV	31
Figure 2.28: Example of a factory floor with lines and chroma changes [39].....	33
Figure 2.29: Extractions of Sotelo et al.'s [40] work in using border and chrominance in navigation	33
Figure 2.30: Park et al's. (a) image acquisition; (b) segmentation; (c) labelling and (d) arrow extraction	35
Figure 3.1: (a) Ultrasonic sensors used to sense a distance to an obstruction; (b) Camera used in sensing distance and image of obstacles	38
Figure 3.2: Original omnidirectional sensor setup to be placed on AGV.....	39
Figure 3.3: Half sphere mirror picture before conversion using the Basler.....	39
Figure 3.4: Converted panoramic picture	39
Figure 3.5: Graphical representation of a polar transform.....	40
Figure 3.6: Test pattern generated for polar transform tests.....	41
Figure 3.7: Results generated by polar transfer – conversion starting at 0° resulting in a mirror image of the photo.....	41
Figure 3.8: Environmental picture in circular form (680 x 670 pixels), mirror image using a Webcam.....	42

Figure 3.9: Transferred image of Figure 3.8, -90° corrected and mirror image effect corrected	42
Figure 3.10: MATLAB [®] program extract – polar to cartesian	43
Figure 3.11: Transform with image facing the front not centred, but mirror image effect corrected already	44
Figure 3.12: Correct transform with front of picture in the middle	44
Figure 3.13 C# developed GUI utilising a transformation exe-file compiled from a MATLAB [®] M-file.....	45
Figure 3.14 Image deformation using a half sphere mirror	46
Figure 3.15 Hyperbolic mirror setup on a Webcam	46
Figure 3.16: Simulink [®] model for converting omni picture to panoramic picture stream	49
Figure 3.17: Embedded MATLAB [®] function block called Imconv in Figure 3.16	49
Figure 3.18: Illustration of capturing a frame, selecting an area of interest for conversion and final resolution for conversion utilizing a Webcam.....	50
Figure 3.19: Frame from omni video stream indicating the direction of movement, area of interest and converted section of image	51
Figure 3.20: Simulink [®] model for converting omnipicture to panoramic – panoramic displayed only.....	52
Figure 3.21: Simulink [®] model for converting omnipicture to area of interest including direction of movement	53
Figure 3.22: Omnipicture indicating the four directions of area of interest selected as video input.....	54
Figure 3.23: MATLAB [®] <i>bench</i> feature being displayed as a graphical result of the PC	55
Figure 3.24: MATLAB [®] <i>bench</i> feature being displayed as a result of the PC, for the process speed in seconds	55
Figure 3.25: MATLAB [®] <i>bench</i> feature being displayed as a graphical result of the laptop	56
Figure 3.26: MATLAB [®] <i>bench</i> feature being displayed as a result of the laptop, for the process speed in seconds	57
Figure 4.1: Animated layout of a simulated factory floor developed in DELMIA	60
Figure 4.2: HMI screen capture [7, p. 67].....	61
Figure 4.3: National Instruments [™] single-board sbRIO-9632 robot platforms [55]	62

Figure 4.4: PIC microcontroller board utilised to generate the pulse width modulation.....	62
Figure 4.5: Sabertooth R/C motor speed controller used on the NI robot platforms.....	63
Figure 4.6: AGV platform utilising a laptop, NI robot platform and omnivision system.....	63
Figure 4.7: Overview flow diagram of the vision based navigation system depicting the route and sign control navigation techniques used.....	64
Figure 4.8: MATLAB [®] “Chroma-based Road Tracking” demo	65
Figure 4.9: Tracking results of the “Chroma-based Road Tracking” demo [58].....	66
Figure 4.10: (a) Frame captured from a colour camera; (b) Prewitt edge detection applied on a frame	67
Figure 4.11: (a) Edge detection – result with Prewitt detection, indicating a probable dead end; and (b) Chroma result – indicating an open end.....	67
Figure 4.12: Simulink [®] model used for edge detection and chroma segmentation, source to Figure 4.14	68
Figure 4.13: (a) Single frame of edge and chroma detection; (b) Frames split vertically in the middle; (c) Right half flipped horizontally and top right part of frame omitted; (d) Hough transform applied, detecting a border; (e) Right half flipped back and <i>Rho Theta</i> values available for merging the lines onto the frame for evaluation.....	69
Figure 4.14: Simulink [®] model for obtaining the border lines by scanning only half of the frames	70
Figure 4.15: Line detection Simulink [®] model including the Hough transform incorporated in Figure 4.14	71
Figure 4.16: Illustrating the need for filtering on the number of lines detected, because of AGV movement, (a) multiple lines detected, and (b) actual filtered lines merged on frame.....	71
Figure 4.17: Route Tracking block for left and right border, input from Detection section Figure 4.14 sourcing Route merging to be viewed in Figure 4.20	72
Figure 4.18: Function block parameters for the left and right lane subsystem of the Route Tracking model	73
Figure 4.19: Left and Right lane masked system block consisting of the Kalman Filter.....	73

Figure 4.20: Route Merging subsystem masked block.....	75
Figure 4.21: Simulink [®] model used for displaying the line detection and line tracking results	76
Figure 4.22: (a) Border detection with line merged on the display; (b) Edge display (c) Chroma display; (d) Route tracking display with command prompt merged on the display.....	77
Figure 4.23: Show valid lanes and direction of movement Simulink [®] model.....	78
Figure 4.24: Sub division of AGV directions (a) directions indicated in radians with 8 ranges, (b) resultant direction code generated and Stop if no route is identified.....	79
Figure 4.25: NI USB-6009 inside and outside its enclosure.....	80
Figure 4.26: MATLAB [®] program extract – initialisation and access of ports	80
Figure 4.27: Timing diagram of R/C motor speed control utilising PWM [60, p. 3].....	81
Figure 4.28: Output achieved by switching the NI USB-6009 port on and off in sequence without time delay	81
Figure 4.29: Motor direction and control setup for AGV movement.....	82
Figure 4.30: Function block receiving the direction information to be altered and sent via USB.....	83
Figure 4.31: Enabled subsystem block for USB serial port communications from laptop to AGV for control commands.....	84
Figure 4.32: 4-wheel NI AGV platform with <i>ski</i> implemented for correcting the tilt effect of the AGV	85
Figure 4.33: Incorporating signs for defining a reconfigurable route.....	86
Figure 4.34: Traffic Warning Sign Recognition MATLAB [®] demo Simulink [®] model block.....	87
Figure 4.35: Three signs template, generated for the detection process, STOP, left and right turn	88
Figure 4.36: Three signs templates – STOP, left and right; with three orientations each, $0^\circ + 7.5^\circ$ and -7.5° ; generated for recognition process	88
Figure 4.37: Detection block of the sign recognition demo used in evaluating command sign detection in the Cr colour space.....	89
Figure 4.38: Displayed recognised left, right and stop signs.....	90
Figure 4.39: Implementing different routes for multiple AGVs by utilising different colours	91

Figure 4.40: Windows Paint edit colours tablet for HSV and RGB pixel colour signal [64].....	92
Figure 4.41: MATLAB [®] function block extract for selecting a certain green colour selected for route identification.....	92
Figure 4.42: (a) Recognised result of a green STOP sign; (b) Boolean picture generated as a result of the Simulink [®] <i>MATLAB function</i> block.....	93
Figure 4.43: MATLAB [®] implementation of the Simulink [®] model evaluated for a set green signal.....	94
Figure 4.44: Simulink [®] model implementing AGV motor control at a set distance depending on the area of pixels.....	96
Figure 4.45: Abbreviated MATLAB [®] code for the distance function block generating STOP, direction and switch control.....	97
Figure 5.1: Mirror and different Perspex [®] tubing lengths for webcam setup.....	101
Figure 5.2: Test pattern used with the measurement setup for obtaining the results in Figure 5.3.....	102
Figure 5.3: (a) Spherical shaped mirror used in Swanepoel's research with results obtained; (b) Developed mirror used in research with results obtained.....	103
Figure 5.4: Refraction and reflection influences on the image, (a) round mirror with no reflection and refraction; (b) used mirror with acrylic setup with the reflection and refraction.....	104
Figure 5.5: (a) Selected frame in front of AGV; (b) Resulted straight edge by selecting area of interest.....	105
Figure 5.6: Decrease in frames per second along the process of image processing on the laptop platform relative to that of a PC.....	106
Figure 5.7: AGV position and orientation along a destined route plotted for evaluation.....	108
Figure 5.8: Corresponding frame captures for the positions indicated in Figure 5.7 ...	109
Figure 5.9: Indication of the degree at which the signs could be detected utilising sign recognition.....	111
Figure 5.10: Distance to a sign plotted against the pixel count of the sign detected with curve fit.....	112
Figure 5.11: Indication of the offset to the straight on angle for sign recognition.....	114
Figure 5.12: Distance from the sign determined by area at different angles of approach.....	115

Figure 5.13: Explanation for the need for a higher selection control resolution 118

Figure 5.14: Directions resolution improvement suggestion, (a) higher resolution direction division in radians; (b) resultant direction code to be generated and test if no lines for movement..... 118

List of Tables

Table 3.1:	Conversion differences of MATLAB [®] and C code from double to integer.....	48
Table 3.2:	Calculated frame rate of embedded MATLAB [®] function block conversion	50
Table 3.3:	Webcam set to 30 frames/second with relevant frame size and frame rate obtained	52
Table 3.4:	Double and single view frame rates incorporating different area of interest sizes converted from a 360° picture video stream.....	53
Table 3.5:	Double and single view frame rates incorporating different area of interest sizes compared to the results obtained on a laptop.....	57
Table 4.1:	Direction control commands of the AGV and its values sent serially on USB to the PIC board for maximum speed movement.....	83
Table 4.2:	Summary of distance from AGV to signs with respect to image pixel count	95
Table 5.1:	Processor platforms specifications used in evaluations	99
Table 5.2:	Cameras used in research with applicable software specifications	100
Table 5.3:	Vertical sizes (heights) of the four different squares shown in Figure 5.2, depicted in pixels by the results obtained from semi-spherical and hyperbolic shaped mirrors respectively.....	102
Table 5.4:	Comparative speeds of the 3- and 4-wheeled AGVs used in the research without vision.....	107
Table 5.5:	Corresponding movement noted for evaluated test run of AGV with respect to the corresponding frames in Figure 5.8	109
Table 5.6:	Experimental results with predicted distance of signs detected from AGV	113

Statement

“MATLAB[®] was used as the software platform for the development, implementation and assessment of a comprehensive machine vision and navigational control system for Automatic Guided Vehicles as researched in this project. This included image acquisition and processing, navigation and control of such. The work done by the author were the development of concepts to solve the research problem utilising building blocks of MATLAB[®] such as .m files, functions and Simulink[®] blocks and is not a copy of the MathWorks[®] teams.”

Ben Kotze

Chapter 1

Introduction

This chapter gives an overview of the research problem, aim, methodology, and hypothesis with the chapter layout necessary to develop and report on a reconfigurable Automatic Guided Vehicle (AGV), capable of sensing the environment and to navigate in a small pseudo-manufacturing setup.

1.1 Preface

The operation of Automatic Guided Vehicles (AGVs) involves several aspects, including its power source, environmental detection and its drive system to name a few. One of these is object observation and/or recognition. Infrared sensing, ultrasonic and whisker sensors are but a few sensing techniques used for detecting objects in the path of an AGV as well as the distance to the object [1, p. 2]. 2-D and 3-D images are also used to obtain the distance to and information about an object in the way of a functioning AGV [2, pp. 157-160]. Cameras, with associated image processing techniques, can improve the quality of information provided to the AGV due to the unique versatility of vision. However, it presents particular challenges, as it requires acquisition techniques dependent on a changing environment and a tremendous amount of image processing. Depending on the application, thermal images and the like – which will not form part of this research project – can also be utilised in meeting specialised sensing requirements [3].

1.2 Motivation and objective of thesis

Infrared sensing, ultrasonic and whisker sensors, to name a few, are becoming increasingly inadequate in sensing the environment for navigation. Using vision, more information is available for controlling the AGV. The first objective of the research project was to investigate the possible use of a single digital camera to secure omnidirectional (360°) vision for an AGV. In this manner, images of the environment around the vehicle would be acquired dynamically to facilitate automated guidance of the AGV in a predominantly set environment.

A reconfigurable solution for manufacturers could be the reprogramming of such a vehicle for utilising alternative routes and keeping the operators programming input to a minimum, rather than implementing altering conveyor systems to transport the goods.

1.3 Hypothesis

Omnidirectional machine vision and image processing can be utilised to optimise the environmental sensing capability of an AGV in order to facilitate effective control of the vehicle. Such a vision or similar vision system should also facilitate the successful incorporation of programmed and unprogrammed reconfigurable movement by the AGV.

1.4 Methodology of research

In obtaining such a reconfigurable AGV system; vision, an AGV platform and a reconfigurable control system were identified as essential elements.

Vision

With digital image acquisition and processing, the resolution of pictures is very important. Hence, careful consideration will be given to the determination of the optimum resolution required for the system as proposed, taking into account the minimum required quality of vision required by an AGV. The associated image processing requirements, as well as commercially available industrial and non-professional cameras will also be investigated. Due consideration of these characteristics should enable determining a

suitable machine vision configuration. The maximum speed of the AGV will be a function of the processing speed of the vision system and, thus, a function of the computer hardware and software utilised.

A horizontally positioned, suitable shaped reflector, reflecting an omnidirectional image of the AGV's immediate environment onto the camera's sensor, is contemplated. The reflector size, shape and placement – relative to the position of the camera and its lens – will play an important role in the quality and nature of images acquired. These characteristics will have to be optimised mathematically and the satisfactory functioning of the comprehensive optical system verified experimentally [2, pp. 157-160][4, p. 53].

Visual identification of any physical object entails the identification of a substantial correlation between a perceived entity and its physical equivalent. Hence, the first objective of the optical configuration would be to process the acquired image such as to obtain an accurate image of the camera's immediate surroundings – enabling the substantive identification of known physical phenomena. This would necessitate image correction by means of significant signal processing algorithms. An example of this in ordinary photography is with the use of fish-eye lenses where the distorted image is mentally transformed by the human viewer into its real format. This technique will be modelled in the project using MATLAB[®] before being implemented on a practical model.

Finally, having obtained a suitable image, possible ways to minimise the amount of processing required, enabling real-time vision by the AGV, will be studied. This will include ascertaining whether it is viable to teach such a system to recognise the size, distance from and patterns of particular, predefined objects by using intelligent algorithms like Neural Networks, Genetic Algorithms or other optimisation techniques that might prove to be suitable for this application [5].

AGV platform and control

A suitable platform/s need to be used and selected. The AGVs have to be controlled in a reconfigurable way with no need for the operator to change the program or with minimal programming changes.

Reconfigurable System

A reconfigurable system as referred to in the study refers to an AGV moving from one point to another on a set route, which could be changed to another origin/destination in another sequential run with minimal or even no software changes or alterations.

1.5 Outline of thesis

The flowchart in Figure 1.1 shows the basic outline of the Thesis. Theory utilised in the research covering vision, AGVs, and control is discussed. The methodology of the research is addressed. The aim of the research was not only to develop an omnisensor, but to look at using vision in a reconfigurable control manner.

The results are noted, evaluated and discussed with a conclusion on the study and results obtained.

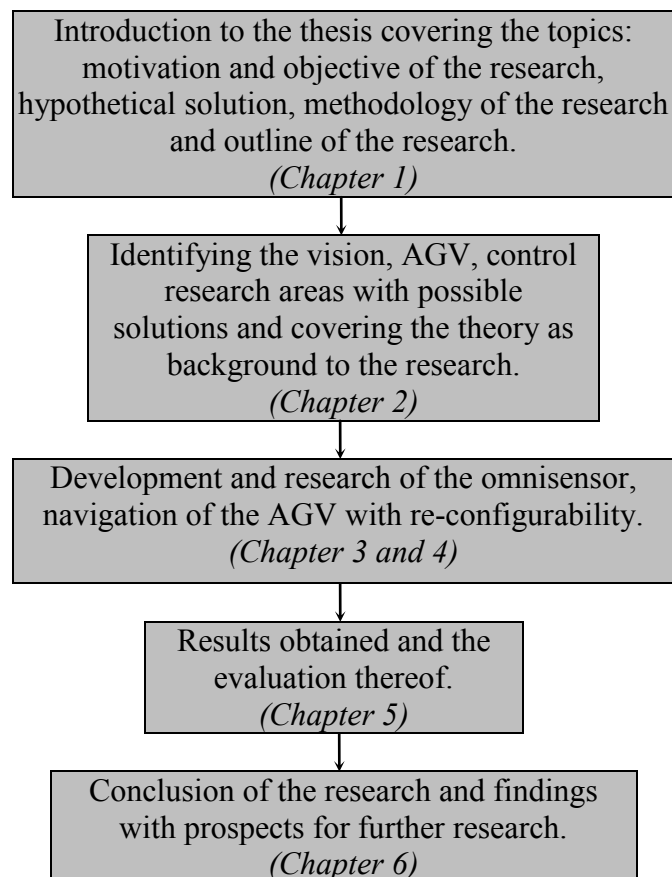


Figure 1.1: Outline of thesis including research phases

Chapter 2

Omnidirectional vision, AGV navigation and control

This chapter gives an overview of the proposed systems necessary to sense the environment of an Automatic Guided Vehicle (AGV), navigation thereof and the control of a specific platform in performing its predefined duties. Only the theories relevant and seemingly important for this study are discussed.

2.1 Introduction

In this chapter the topics for vision, AGV navigation and control are addressed as background for the research.

Figure 2.1 gives an outline of the whole process.

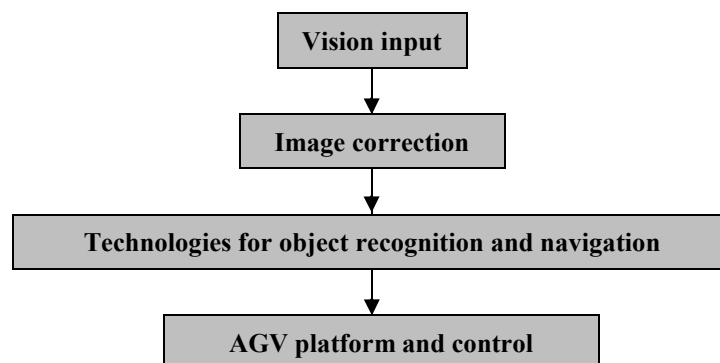


Figure 2.1: Flowchart of research outline as seen for the research process

Figure 2.2 is depicting the concepts as a visualisation of the possible solutions and directions used and investigated for the research to obtain the goal depicted by the project's goal and hypothesis. The topics covered were selected as seemingly the most suitable for the planned research converting an image into identifiable objects or routes and tracking their movement relative to the AGV for navigation and control purposes.

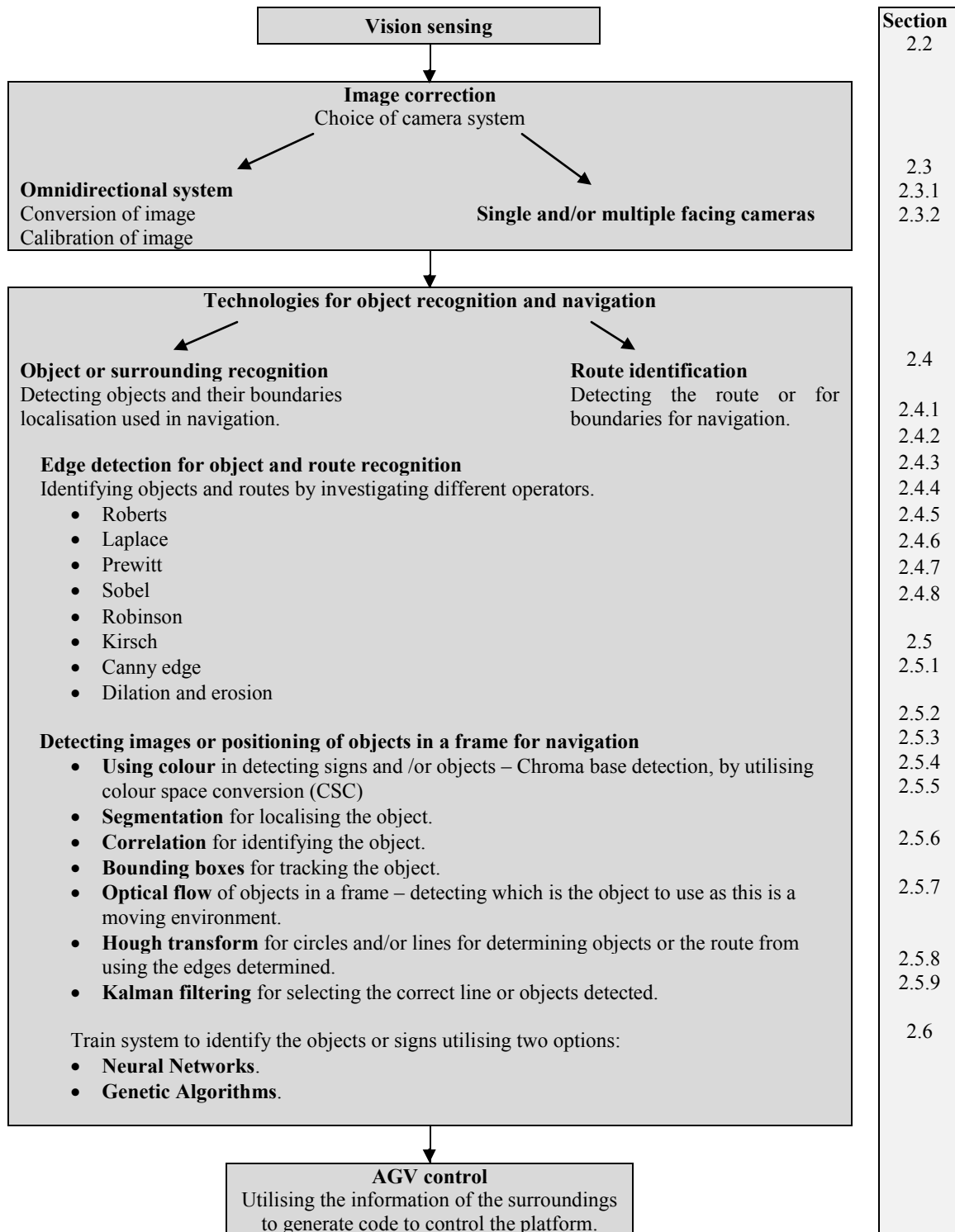


Figure 2.2: Flowchart of research concepts and routes taken for reaching the research goal placing the work discussed in Chapter 2 in context

Work done in AGV navigation and control by the Research Group in Evolvable Manufacturing Systems (RGEMS) at the CUT, Free State is discussed. Possible alternatives to be investigated in obtaining reconfigurable navigation are also looked at.

2.2 Vision concept

Machine vision (MV) is used on the AGV, replacing the more traditional distance and environmental scanning devices mentioned by Fend et al. [1, p. 2]. The vision system implemented must then give a more detailed picture of the environment for navigation and control of the AGV.

A problem in discarding scanning devices and distance sensing is the loss of depth perception in using a 2D picture as can be seen in Figure 2.3. The object may no longer appear to be its real size.

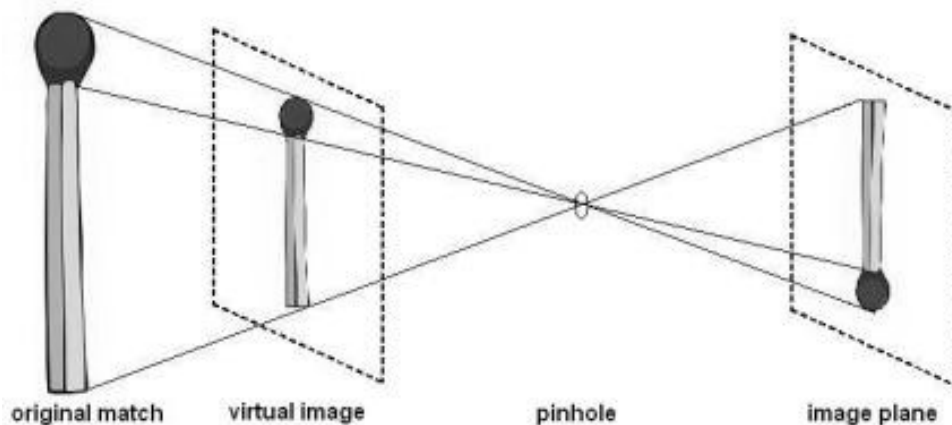


Figure 2.3: The loss of size and depth perception on a 2D image

Figure 2.4 gives a more detailed background of the aim to be achieved by vision sensing. The environment of the AGV captured in the first scene is converted into a digital image where processing needs to take place, extracting the features and/or objects necessary for navigation and control [6, p. 6].

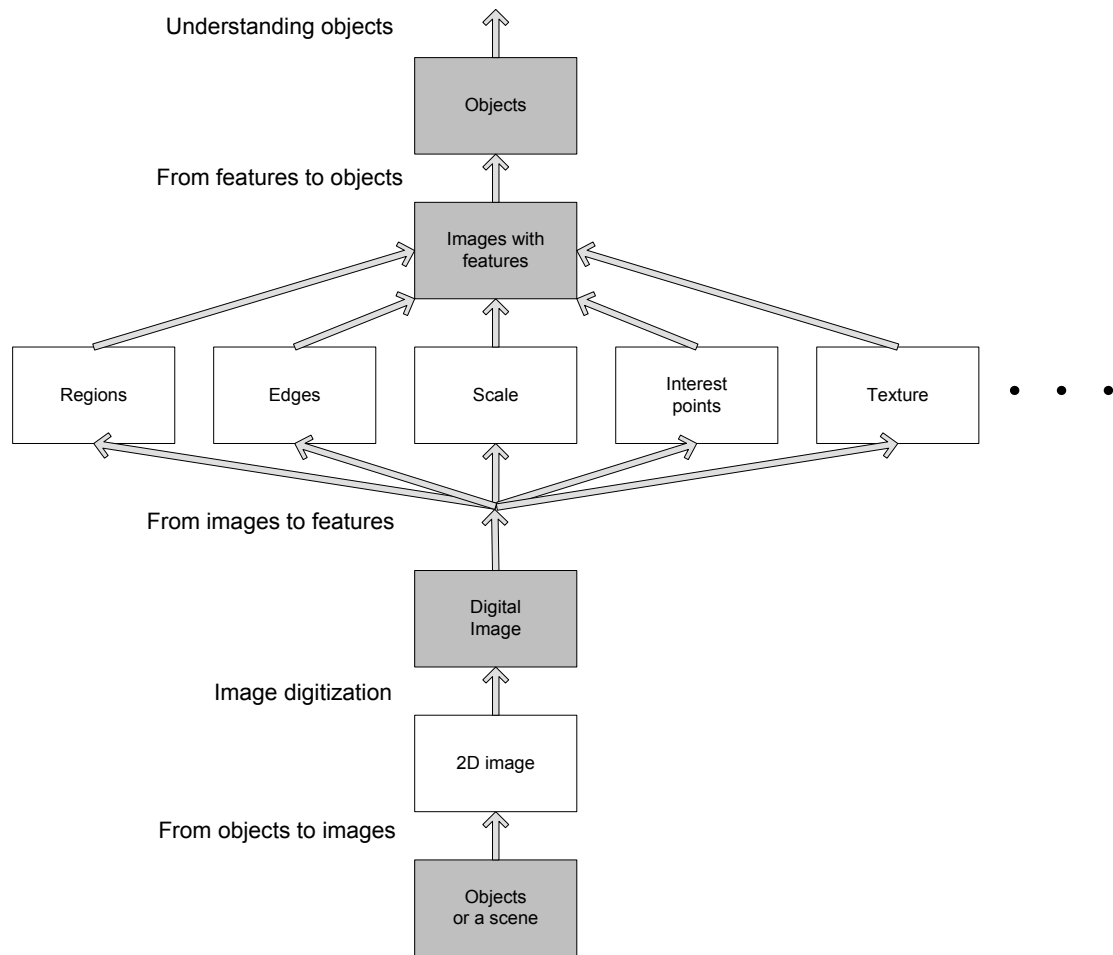


Figure 2.4: Levels of image processing used in the identification of objects for processing

2.3 Omnidirectional sensing

In omnidirectional sensing the capture of the environment, and the conversion and implementation of the image form the crucial parts of the study background. The omnidirectional background is based on a hyperbolic mirror setup. A similar configuration was also implemented in previous research done in the RGEMS group.

The hyperbolic mirror setup can clearly be seen in Figure 2.5 [7, p. 58].



Figure 2.5: AGV with hyperbolic mirror setup

The Taylor model, discussed by Scaramuzza [4, p. 23], was originally evaluated to be used for the omnidirectional conversion in the research. There are, however, a variety of models to choose from for developing omnidirectional vision, including the linear model [4, p. 20] and an derivative of such a linear model by Swanepoel [7, pp. 65-66]. Figure 2.6 depicts a 2D model representation of such a setup.

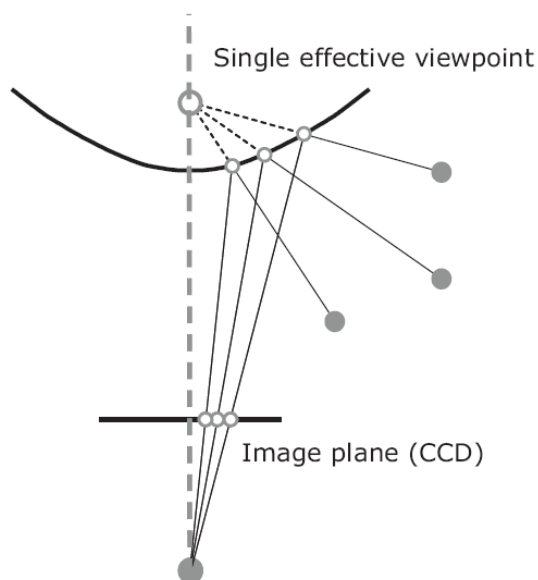


Figure 2.6: Example of a vision system satisfying the single viewpoint property of an omnidirectional camera with a hyperbolic mirror [4, p. 11]

2.3.1 The Taylor model

A little bit of background to the Taylor model is given in this section as this was the original model the omnidirectional conversion was based on. The Taylor model is a unified model for dioptric and catadioptric central omnidirectional cameras derived by Scaramuzza et al. which is suitable to different kinds of vision sensors [8][9].

Equation (2.1) shows the transfer function for the Taylor model based on the variables depicted in Figure 2.7 [4, p. 24]:

$$\lambda p'' = \lambda \left[g(\|u''\|) \right] = P \cdot X \quad (2.1)$$

where p'' is the projected image point, u'' is the mapped point, g represents the function of the lens, P is the projection matrix and X represent a scene point passing through an optical centre of a camera, utilising a scaling parameter $\lambda > 0$ [10, p. 229].

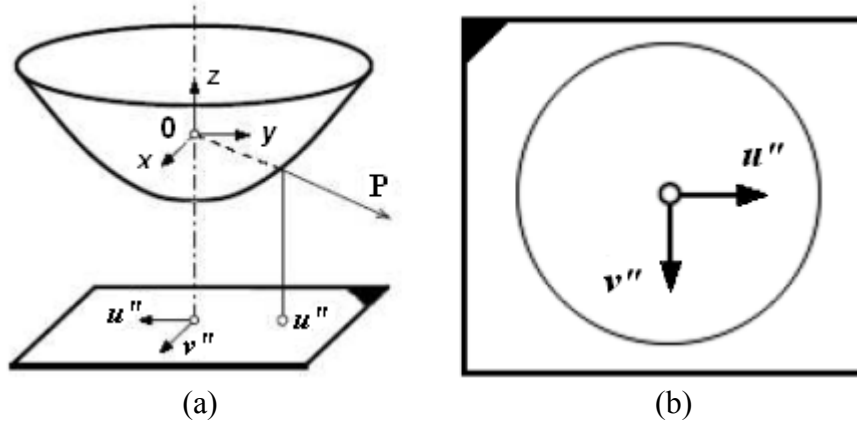


Figure 2.7: (a) Coordinate system in catadioptric case; (b) Sensor plane and conversion

In [8] the proposed polynomial for g is given as:

$$g(\|u''\|) = a_0 + a_1\|u''\| + a_2\|u''\|^2 + \dots + a_n\|u''\|^n \quad (2.2)$$

where the coefficients a_0, a_1, \dots, a_n and the polynomial degree n are calibration parameters. Calibration was investigated, because calibration seemed necessary in the conversion process of omnidirectional pictures.

2.3.2 Calibration of omnidirectional pictures

Calibration problems occur in different camera setups as Aliaga [11, pp. 127-134] explains in his catadioptric system with a parabolic mirror and an orthographic lens to produce an omnidirectional image with a single centre-of-projection. This setup can be seen in Figure 2.8.



Figure 2.8: Camera setup of Aliaga with parabolic mirror and acrylic half sphere on a video camera [11]

The model is based on the following equation with respect to Figure 2.9 indicating the variable parameters. Calibration is needed because of possible reflective errors where the focal centre point overshoots at c :

$$d = \frac{(p_z m_r)}{m_z} - \frac{m_z}{\tan \alpha} + m_r \quad (2.3)$$

where d is the actual distance, p_z is the given points height, m_r is the radius vector of point m , m_z is the height at this point and α the angle.

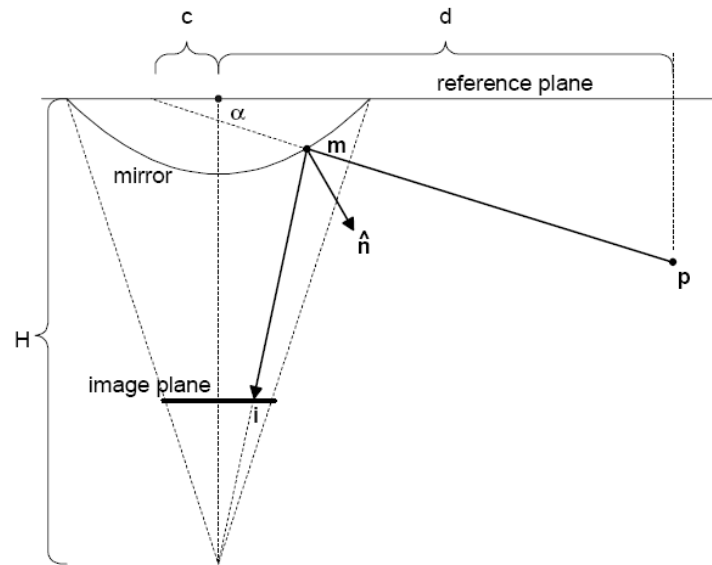


Figure 2.9: Aliaga's model, which allows accurate computation between the focal- and 3D point

This is, however, only one adaptation and there are many other calibration techniques to choose from or to modify depending on the application, as Scaramuzza proved with his checker board correction [4, p. 35].

2.4 Edge detection for object and route recognition

Figure 2.4 indicates that edge detection can be an interim stage between the digital image detection and identification of an image with features to be used in image processing. Edge detection is thus an integral step in identifying the edges of a possible route to be followed by the AGV. Edge detection is done on a binary or greyscale image represented by pixel information.

Images contain a lot of features that could be detected or changed to a perceived edge as Figure 2.10 indicates [6, p. 133]. There are many different apparent edges but the user must decide which to use and/or if it is a proper edge.

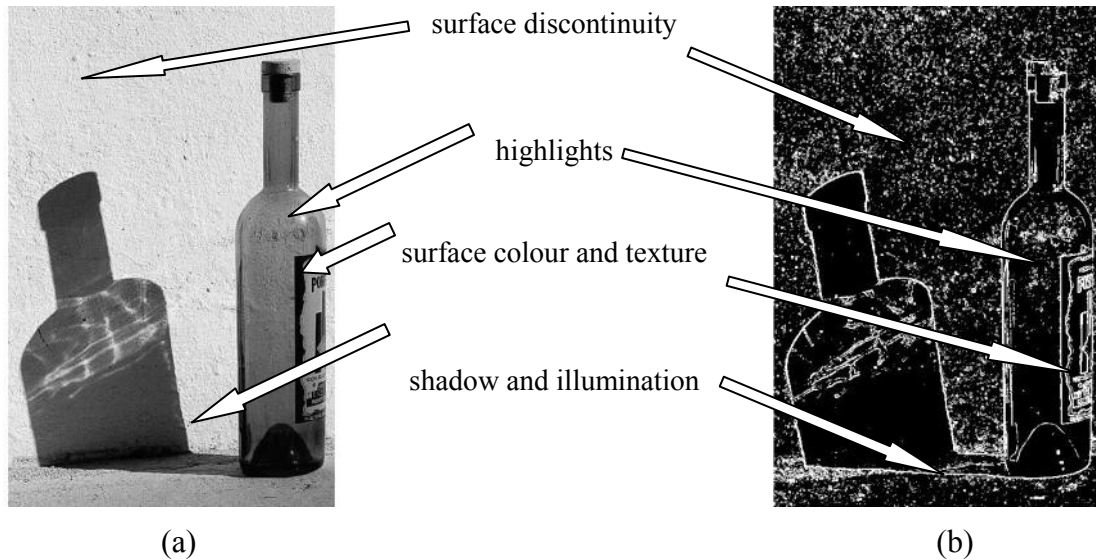


Figure 2.10: (a) Original image with edges due to different phenomena; (b) Detected edges by means of the Sobel operator

Sonka et al. discussed the gradient operator of edge detectors as belonging to one of three categories [6, p. 135]:

1. Image functions using differences in the approximation of derivatives obtained from masks (simple patterns).
2. Operators on the zero-crossing of a function derivative of an image.
3. Operators attempting to match a function to a parametric model of edges.

The following paragraphs give a short overview of the different operators used and evaluated for edge detection. These operators were investigated in the research project identifying its differences, mainly indicated by their convolution masks. The Prewitt operator was used extensively because of its direction gradient property.

2.4.1 Roberts operator

The Roberts operator is one of the oldest and first to be developed. It is based on a 2×2 neighbourhood of pixels format system [12]. The operator is to approximate the gradient of an image as could be seen in the diagonally adjacent pixels of its convolution masks h_x :

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad (2.4)$$

and the magnitude of the edge is computed as,

$$|g(i, j) - g(i + 1, j + 1)| + |g(i, j + 1) - g(i + 1, j)| \quad (2.5)$$

where g depict the specified pixel at location i and j .

The Roberts operator do not perform well with a noisy picture because of the low number of pixels used in the operator, but due to use of this small amount of pixels it's also one of the faster operators.

2.4.2 Laplace operator

The Laplace operator is an approximation of the second derivative giving the gradients magnitude. A 3×3 convolution mask is often used for 4-neighborhood pixels and 8-neighborhoods pixels and defined as:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (2.6)$$

The disadvantage of the Laplace operator is the doubt of a real edge in some instances [6, p. 136]. It also is a much larger matrix consuming more processing time.

2.4.3 Prewitt operator

The Prewitt operator detects edges using the approximation of the first derivative returning those points where the gradient value is a maximum. The gradient estimation for a 3×3 mask is done in eight possible directions. The convolution result of the greatest magnitude indicates the direction gradient. Equation (2.7), representing the operators convolution mask, illustrates this scenario described above by looking at the location of -1 , 0 and 1 in the mask [6, p. 136].

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \dots, h_8 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix} \quad (2.7)$$

The Sobel-, Robinson- and Kirsch operators are similar to the Prewitt operator, although the Prewitt operator were extensively used as it proved, by means of experimentation, to be the better operator for the research.

2.4.4 Sobel operator

As an example, the Sobel operator can be used in detection of horizontal and vertical edges, depicted by 0, utilizing only the convolution mask h_1 and h_3 from the available three directions:

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}, h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (2.8)$$

This means that if h_1 is represented by x and h_3 by y , the edge strength/magnitude is derived by [6, p. 137]:

$$\sqrt{x^2 + y^2} \quad (2.9)$$

which is also the case in many of the other operators as it is achieved by computing the sum of the squares of the differences between adjacent pixels.

2.4.5 Robinson operator

As with most of the operators, this operator is also direction specific as can be seen from the convolution masks, viewing the position of -1 and 1 with respect to -2:

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}, h_3 = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}, \dots \quad (2.10)$$

2.4.6 Kirsch operator

The Kirsch operator is direction specific but emphasis is placed on the gradient [6, p. 138]. This is similar to the Prewitt operator but different to the magnitude of the mask and distinct difference of the values of 3 and -5 with respect to 0.

$$h_1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix}, h_2 = \begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix}, h_3 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix} \quad (2.11)$$

2.4.7 Canny edge detection

Canny proposed an approach based on detection, localisation and one-response-criterion meaning that multiple detections could be taken as a single edge [13][14][15].

The Canny edge detector algorithm is based on seven steps [6, pp. 144-146]:

1. Convolve an image with a Gaussian scale.
2. Estimate local edge directions using an equation for each pixel in the image.
3. Find the location of the edges.
4. Compute the *edge strength* based upon the approximate absolute gradient magnitude at the location.
5. Threshold edges in the image with hysteresis to eliminate spurious responses.
6. Repeat steps 1 to 5 for ascending values of the standard deviation.
7. Aggregate the final information for the edges on a greater scale using the “feature synthesis” approach.

The differences were marginal but could be illustrated using Figure 2.10, with the base of the bottle as area of interest. Figure 2.11 illustrates the major difference in using the Roberts operator with a small pixel footprint, resulting in non-continuous lines, against the Prewitt with more substantial edges (mainly utilised and evaluated in the research) versus the Canny edge where multiple detections could be taken as a single edge, resulting in more unwanted edges in this scenario.



Figure 2.11: Results obtained by utilising different operators viewing only a section of the image used in Figure 2.10(a) – (a) Roberts operator result; (b) Prewitt operator result; (c) Canny edge result

2.4.8 Dilation and erosion

Assume that a binary picture is used, where the black pixels constitute the image and the white pixels are the background. Dilation could be described by an increase of black pixels, and erosion as a decrease of black pixels, best illustrated by Figure 2.12. Edge detection could also be accomplished by subtracting the eroded image from the original as can be seen in Figure 2.12 (d).

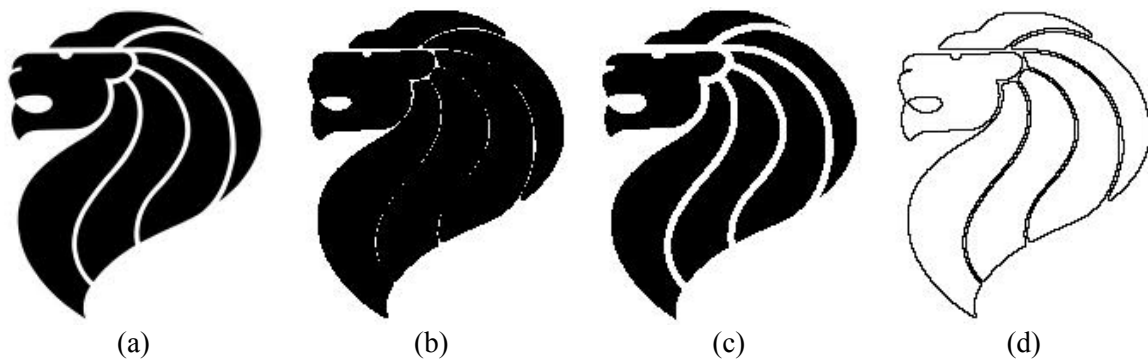


Figure 2.12: (a) Original binary image; (b) Image with 3-pixel dilation; (c) Image with 3-pixel erosion; (d) Edge detection by subtracting the eroded image from the original

Both dilation and erosion are morphological operations as being described using Minkowski's formalism by Haralick and Shapiro [16]. Although dilation and erosion seemed a possible option in edge detection it was not used in the final setup of experimentation.

2.5 Techniques used for tracking and detecting objects utilising vision

The following topics covered are a background to the possible techniques used and investigated in obtaining the vision goals for navigation and control of an AGV.

2.5.1 Colour space conversion

A lot could be derived from a binary or greyscale image, but adding colour adds another dimension. This concept was adopted in the re-configurability of the route tracks of the AGV to be followed. The representative colour values could be seen in the International Commission on Illumination (CIE) chromaticity diagram, shown in Figure 2.13. Each colour has a frequency value along the λ axis. The primary colours red, green and blue (RGB), or any other colour, could be identified by an x and y position/value on the diagram.

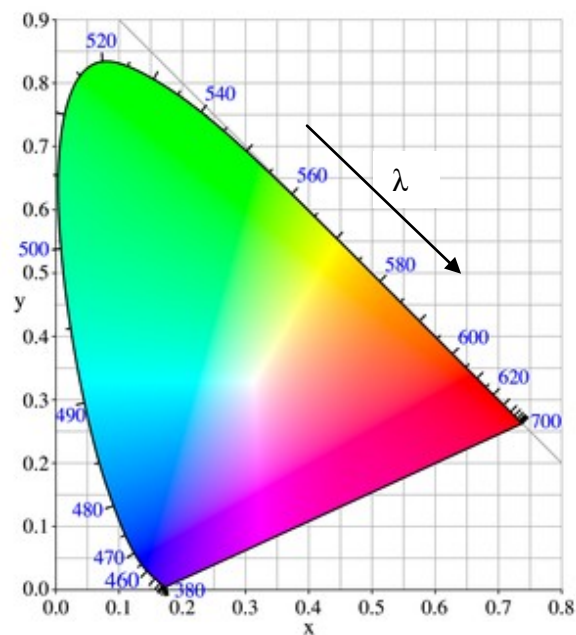


Figure 2.13: CIE chromaticity diagram 1931 [17]

The RGB colour space with primary colours and secondary colours yellow, cyan and magenta with its possible conversion to a colour value or grey scale option is best illustrated by Figure 2.14 [6, p. 37]. This is an RGB model, which was introduced and evaluated but did not produce the desired results.

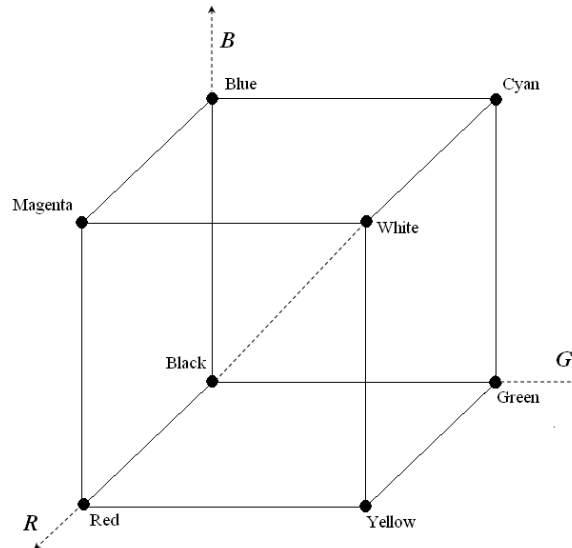


Figure 2.14: RGB colour space with primary and secondary colours indicating grey scale

Colours are also represented by hue, saturation and value (HSV) as can be seen as a cylindrical model in Figure 2.15. The hue represents a colour value, saturation the chroma or depth of the colour and value the shade of the colour.

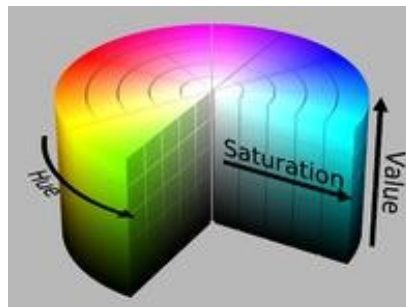


Figure 2.15: HSV colour model illustrated as a cylinder [18]

There is also the YCbCr family of colour space used in video and digital photography systems. Y represents the luminance component, Cb the blue- and Cr the red-difference chrominance components. Each of these values is being calculated with the following equations [19]:

$$Y = k_r \cdot R + k_g \cdot G + k_b \cdot B \quad (2.12)$$

$$C_B = -k_r \cdot R - k_g \cdot G + k_b \cdot B \quad (2.13)$$

$$C_R = +k_r \cdot R - k_g \cdot G - k_b \cdot B \quad (2.14)$$

where k represents the colour constant of the ratio of the individual R, G and B components resulting in the desired chrominance.

2.5.2 Segmentation

Segmentation is a vast topic but is one of the most important steps in analysing an image [6, pp. 175-327]. In this research edge- and region-based segmentation is taking priority. Thresholding plays a big role in segmentation determining borders, edges and lines. Gray-level thresholding is one of the simplest segmentation processes. The disadvantage is that the threshold must be set to a predetermined level and lighting plays a big role in altering this threshold value. Optimal- and multi-spectral thresholding are only but a few of the methods used in the segmentation process. Border tracing and region operations of an object also form part of segmentation thus leading to blob analysis, a binary representation of the object. Lu et al. proved in their research on detecting human heads and hands analysing movement gestures that, using colour in addition to thresholding for blob analysis was a successful approach [20, pp. 20c-30c]. Thus using colour in blob analysis could also be applied to other object detection applications. The main goal of segmentation would be to analyse an image by dividing the image into sections that have a strong correlation with objects depicted by these sections of the image.

The segmentation concept is best illustrated by Figure 2.16, which represents shape segmentation done by Chan and Vese [21]. Segmentation is applied to three different shapes, each representing an object. A certain segmentation model is applied to the four figures (1-4) in Figure 2.16 and outlined by a white border which represents the segmentation result. Thus four different shapes obtained by these different segmentation models representing the objects (parts), with the fourth model/step the obvious choice as the three shapes are recognisable.

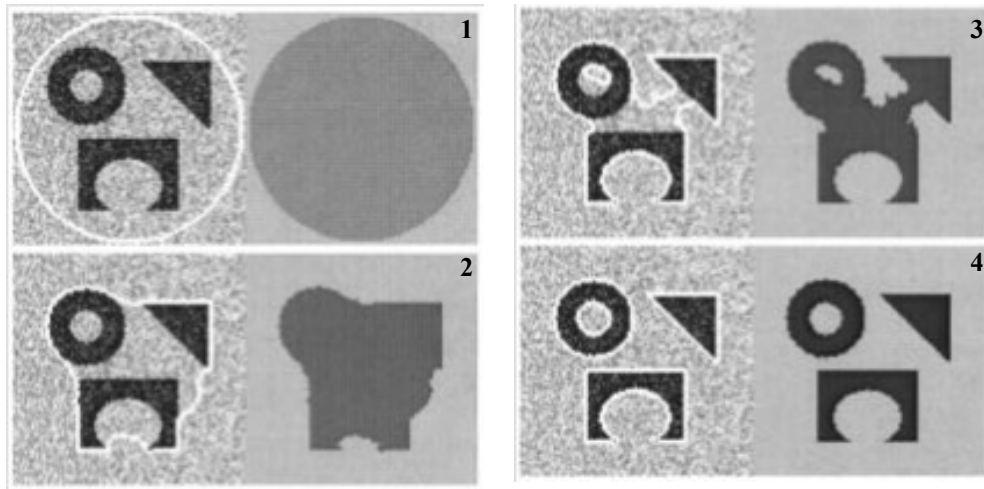


Figure 2.16: Different shaped parts detection from a noisy image, with different segmentation models

2.5.3 Correlation

Correlation could also be described as image matching and is used to locate objects in an image. An example is depicted in Figure 2.17 where a desired pattern is located in the image.

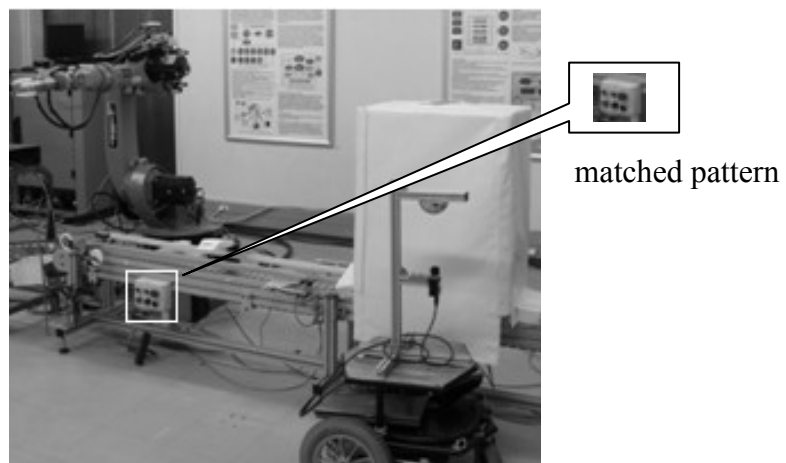


Figure 2.17: Segmentation by correlation; matched pattern with location of best match

The algorithm for correlation is based on the following criteria [6, p. 238]:

- Evaluate a match for each location and rotation of the pattern in the image,
- Locate a maximum value exceeding the preset threshold represented by the pattern location in the image.

A typical equation for correlation between a pattern and the search image data is shown in equation (2.15).

$$C_1(\mathbf{u}, \mathbf{v}) = \frac{1}{1 + \max_{(i,j) \in V} |f(t+\mathbf{u}, j+\mathbf{v}) - h(i,j)|} \quad (2.15)$$

where f is the image processed, h is the search pattern, V the set of image pixels represented by it's location (i, j) and C_1 the correlation result with (u, v) representing the location of the matched position.

There is, however, a variety of matching criteria models to choose from and this one was only used in evaluating the concept for possible utilisation in the research.

2.5.4 Bounding boxes

When an object is identified in a picture or field of view, the smallest rectangle that encloses the figure is called a bounding box [22, p. 119]. The co-ordinates of a bounding box are usually in pixels and this is used in different applications like measurements and localisation.

2.5.5 Optical flow

As part of this research navigation, object recognition and movement seems to be important as investigated in previous research, where movement and movement detection were investigated and implemented [7, pp. 69-70][23]. This was the reason for furthering the investigation on optical flow as the AGV will detect the surroundings to be moving relative to itself [24, pp. 460-463].

The optical flow concept is best explained by Figure 2.18. The ball is moving towards the viewer. There is a large movement towards the bottom of the picture. This is relevant to the speed at which it's moving. The smaller movement arrows to all directions indicates the ball is becoming a larger object, with the conclusion that the ball is moving in the direction of the viewer as a larger object is perceived to be closer to a viewer. Optical flow is based on two assumptions [6, p. 758]:

- The brightness of the image stays constant over time; and
- Nearby points in the image move in a similar manner (velocity smoothness constraint).

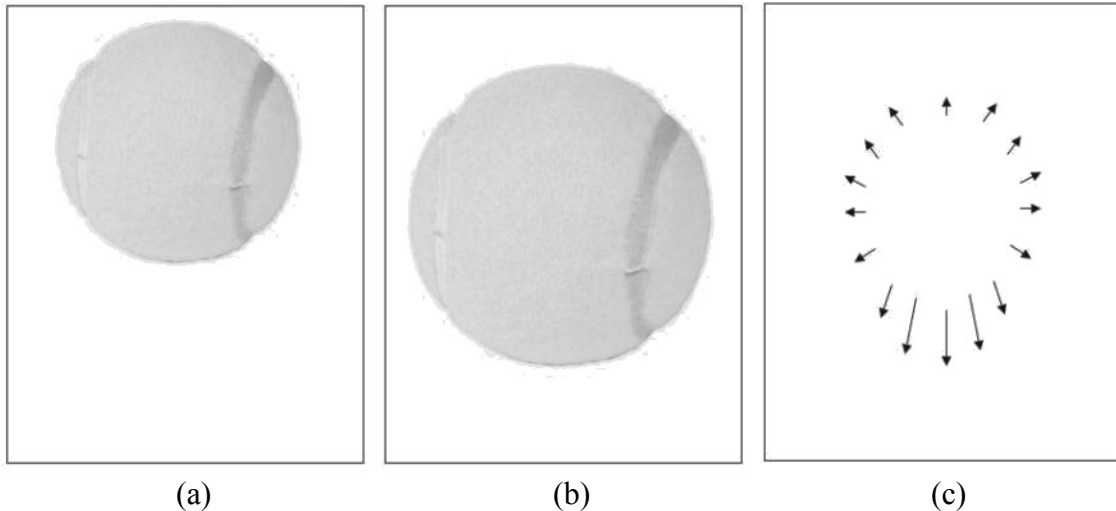


Figure 2.18: Optical flow of a moving tennis ball, (a) time t_1 ; (b) time t_2 ; (c) optical flow vectors

The aim in such an example would be to calculate the velocity (c) as indicated in equation (2.16).

$$c = \left(\frac{dx}{dt}, \frac{dy}{dt} \right) \quad (2.16)$$

where x and y is the position of the corresponding pixel coordinates.

2.5.6 Hough transform

The Hough transforms for circle and line detection also forms part of segmentation. Detecting a circle using the Hough transform as an example could be seen in Figure 2.19 [22, p. 227]. The importance of the Hough transform is its ability to generate the gradient vector for the edges detected. The Hough accumulator array is obtained by using the edge points and edge orientation. The circle is then detected by using thresholding applied to the accumulator and finding the local maxima of the edges detected in the accumulator array [25, p. 304].

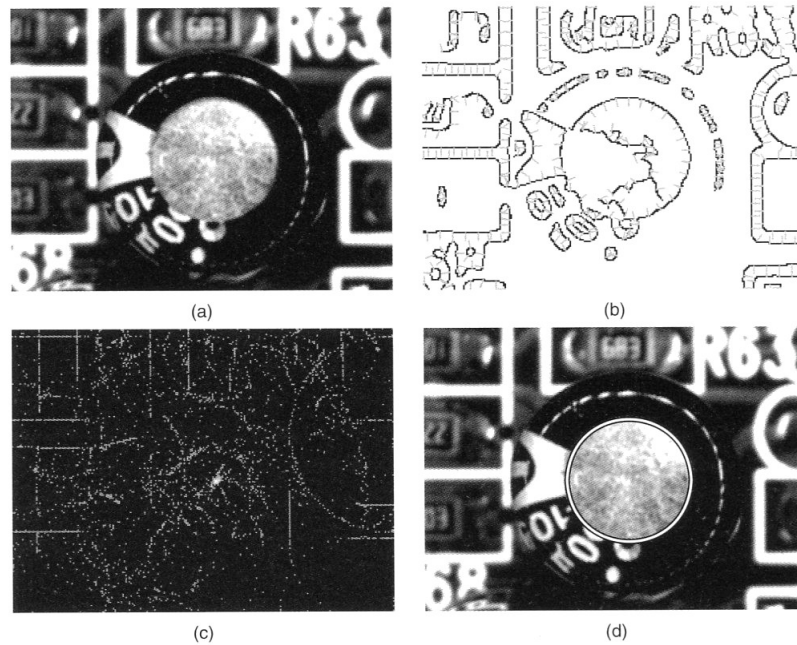


Figure 2.19: (a) PCB with capacitor; (b) edges detected; (c) Hough accumulator array using edge points and orientation; (d) circle detected by thresholding and local maxima

The Hough transfer is also used for line detection. A line is described by equation,

$$y = mx + c \quad (2.17)$$

and can be plotted by a pair of image points (x, y) . The Hough transform do not take the image points $(x_1, y_1), (x_2, y_2)$ into account, but rather use the slope parameter m and y crossing value c . There is a problem when facing a vertical line where m and c becomes unbounded values. It is therefore better to use the parameters denoted r and θ (theta) as can be seen in Figure 2.20.

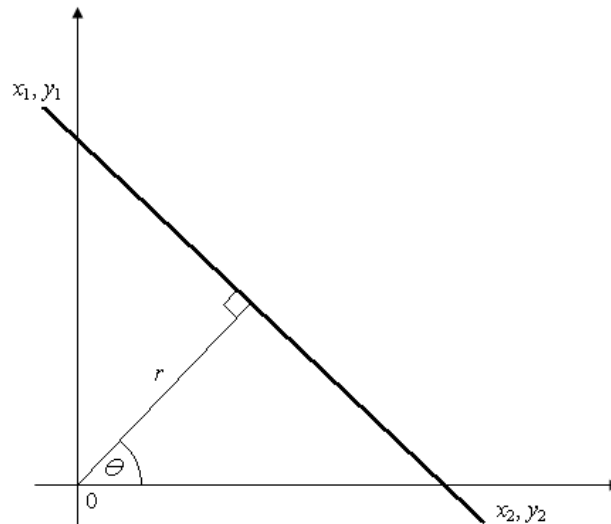


Figure 2.20: Hough parameters for a straight line

The parameter r represents the distance between the line and origin, while θ is the angle of the vector from the origin to the closest point on the line. Thus the new equation representing the line could be written as:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right)x + \left(\frac{r}{\sin\theta}\right) \quad (2.18)$$

Each line in an image is represented by a unique pair (r, θ) . Again thresholding takes place, determining a real line by placing the values in an array and finding the local maxima [25, p. 304].

These data points in the array do not always represent a particular line as the lengths are unknown. As an example, (r, θ) values for these data points could be sampled and plotted as can be seen in Figure 2.21. Thus the use of a Hough space graph seen in Figure 2.22, obtained from data points in the array, to determine which points belong to which line [26, pp. 6, 7].

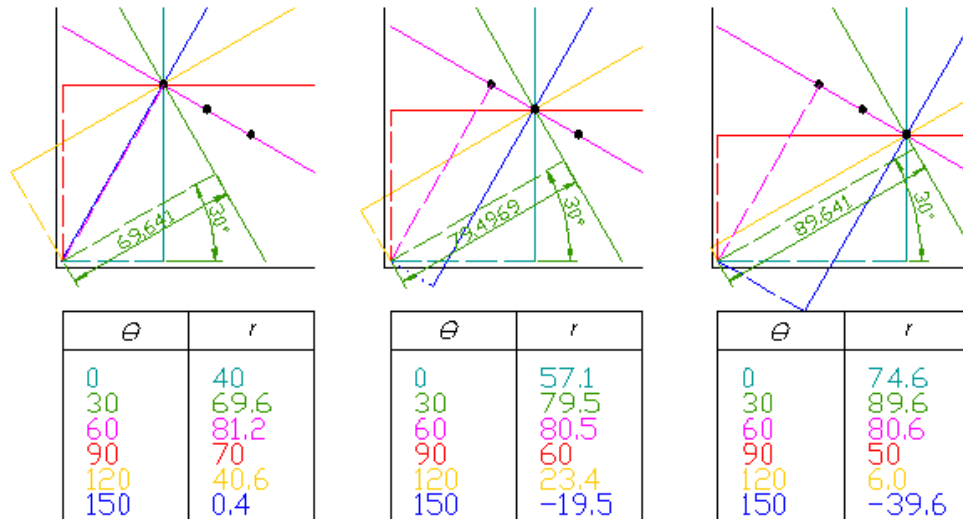


Figure 2.21: Points in a Hough array plotted with different (r, θ) values

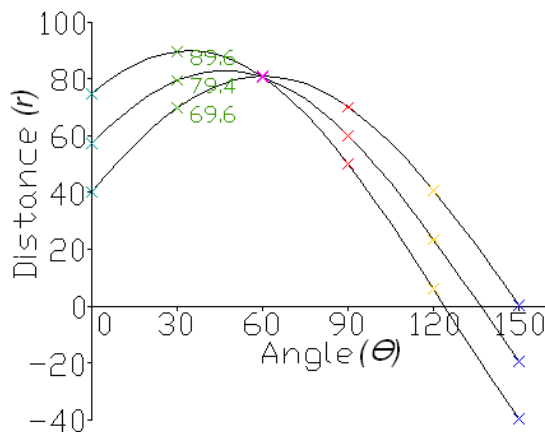


Figure 2.22: Hough space graph plotted from several (r, θ) points

The point where the lines intersect on the Hough space graph gives a distance (r) and angle (θ) of the points being tested of a definite line.

2.5.7 Kalman filter

The Kalman filter can be used for many applications. It is mainly used to predict or estimate system states of a dynamic system from a series of incomplete and/or noisy measurements [27]. In the research it could be used for filtering the amount of lines and/or bounding boxes to minimise the amount of data to be analysed.

Equation (2.19) and (2.20) represent such systems where x_k is the linear system and z_k the measured system:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \boldsymbol{\omega}_{k-1} \quad (2.19)$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v} \quad (2.20)$$

A , represents the state transition matrix and H the measured matrix.

ω_{k-1} and v represent noise and errors in the system respectively.

The result could be best explained by Figure 2.23 where a system had to predict the result using the Kalman filter with a set amount of iterations with a true value constant $x = -0.37727V$ [28].

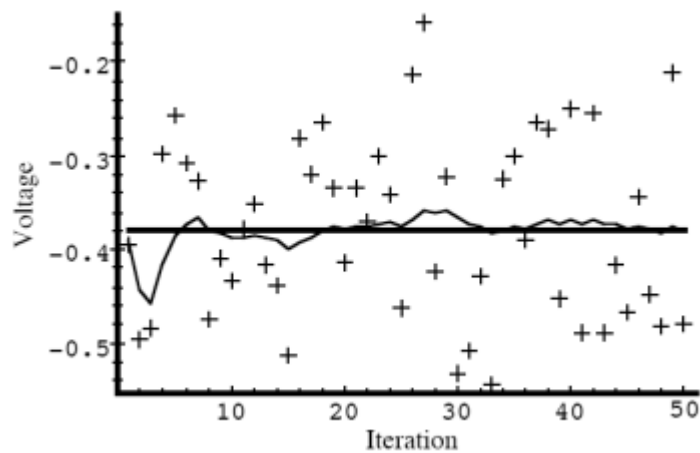


Figure 2.23: Results of a system utilising the Kalman filter – solid line the predicted result, + indicates noise [28]

The result indicates that the prediction is eventually almost the same as the expected result in the presence of noise after 50 iterations.

2.5.8 Neural networks

Object recognition plays a big role in image processing. Neural networks (NN) proved in the past to be a solution for problems such as pattern recognition [29]. NN is trained rather than designed. It was found that bridged multilayer perceptron (BMLP) is a much better architecture than popular multi layer perceptron (MLP) architecture. It is faster to train and more complex problems can be solved with fewer neurons [30, pp. 15-22].

Most neural approaches are based on combinations of elementary processors (neurons), each of which take a number of inputs and generate a single output. Each input carries a weight and the output is a weighted sum of inputs as can be seen in Figure 2.24 [31].

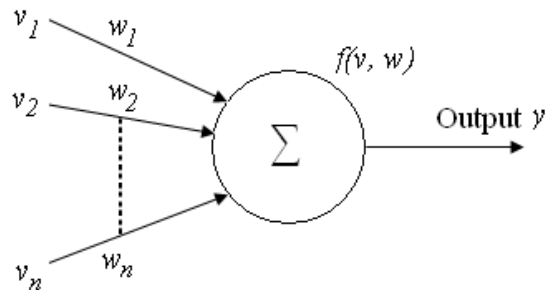


Figure 2.24: A simple (McCulloch-Pitts) neuron

The total input to the neuron is calculated as:

$$x = \sum_{i=1}^n v_i w_i \quad (2.21)$$

where v_1, v_2, \dots is seen as the inputs and w_1, w_2, \dots the weights of the individual inputs.

Also associated with a neuron is the transfer function $f(x)$ which determines the output as the following example indicates:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases} \quad (2.22)$$

The general idea is to connect such neurons in a network mimicking the human brain. The way this is done specifies the network. Such a neural net structure example can be seen in Figure 2.25.

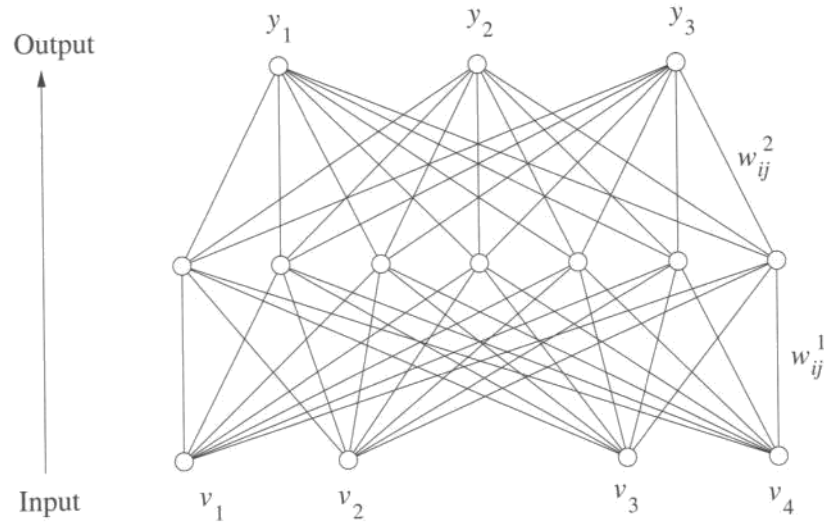


Figure 2.25: A three-layered neural net structure example with four inputs and three outputs [6, p. 406]

Structures such as these also exist with hidden layers [32, pp. 48-50]. NN could also be applied to other fields such as control and navigation.

2.5.9 Genetic algorithms

Genetic algorithms (GA) use a process similar to natural evolution to search for an optimum solution and are used in recognition and machine learning [6, pp. 425-427].

GAs distinguish themselves from other techniques by the following characteristics [33, pp. 20-21]:

- The manipulation of variables takes place in string format instead of the variable itself;
- The use of multiple points form a population, rather than a single point to prevent false peaks for the solution of the problem;
- GA entails a blind problem solving technique of which only the result is of importance; and
- GAs use a stogastic model rather than a deterministic one.

GAs are based on **reproduction** of populations, utilising **crossover** and **mutation** to render changes towards an optimum solution using a **fitness** function.

Figure 2.26 shows a combined flowchart of Sonka, Hlavac, Boyle [6, p. 427] and Kotze [33, pp. 21-22] representing the algorithm steps.

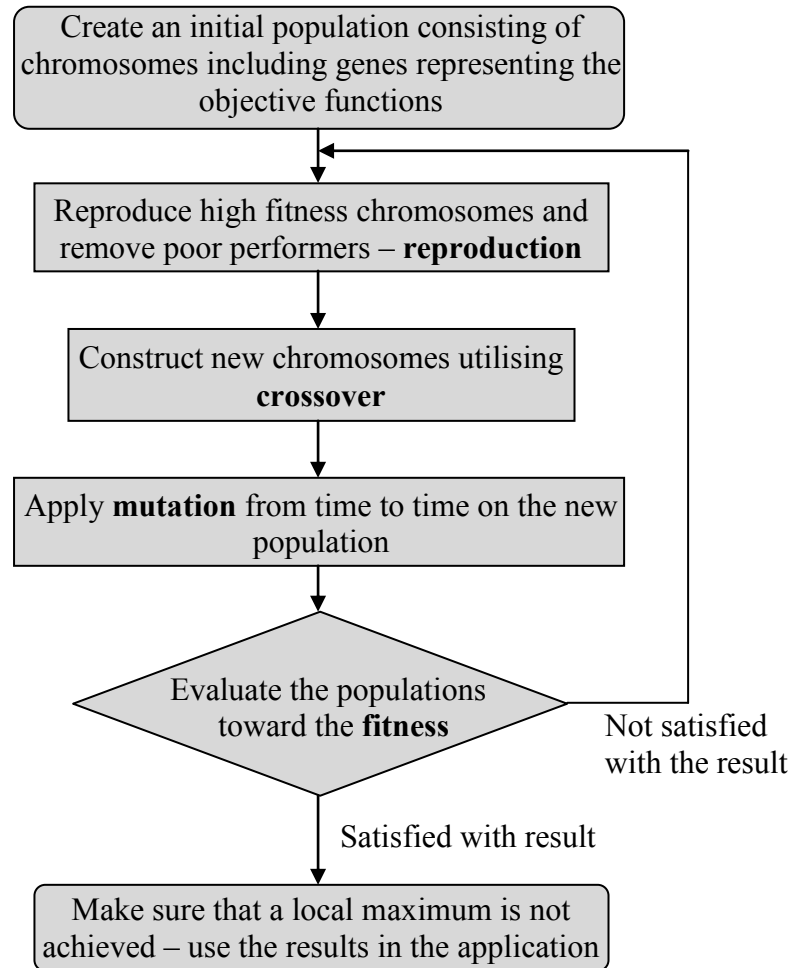


Figure 2.26: Combined flowchart representing the genetic algorithm steps

GAs lend itself to evolve to a relative optimal result but not always the global optimum.

2.6 AGV platform, navigation and control

An AGV platform is user and application specific. Navigation relies on the environment and the application, and this is facilitated by the control thereof.

Location determination plays a big role and this is where Global Positioning Systems (GPS) are used in open space environments [34, p. 1180]. Inside a building or factory other alternatives need to be investigated.

2.6.1 Dead reckoning

Dead reckoning as used by Swanepoel [7, p. 20] proved to be workable, but does not incorporate wheel slip as can be seen from equations (2.23), (2.24) and (2.25) depicting the coordinates (x and y) as well as the heading (θ), resulting in a gradual decrease in positional accuracy.

$$\Delta\theta = 2\pi \frac{R_w T_1 - T_2}{D} \quad (2.23)$$

$$\Delta x = R_w \cos(\theta) (T_1 + T_2) \frac{\pi}{T_r} \quad (2.24)$$

$$\Delta y = R_w \sin(\theta) (T_1 + T_2) \frac{\pi}{T_r} \quad (2.25)$$

where T_1 is the encoder pulses received by the left wheel, T_2 the encoder pulses from the right wheel, R_w represents the radius of the wheels, D is the distance between the wheels (taken from the centre of the wheel track to the other wheel's centre of the track) and T_r the total number of pulses recorded in a travelled distance. This is applied in a wheel placement as seen in Figure 2.27.

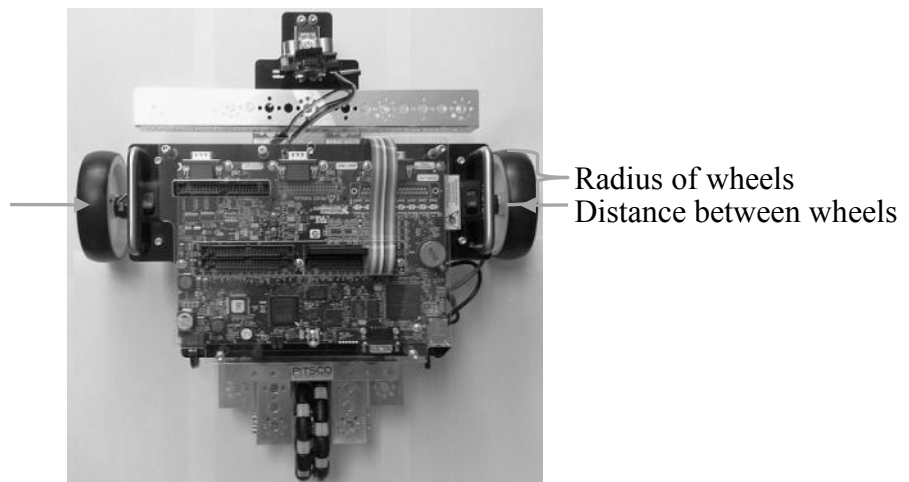


Figure 2.27: Notation of variables in a dead reckoning setup on an AGV

2.6.2 Ultrasonic triangulation

Boje used an ultrasonic triangulation system to keep track of movement and position in an enclosed environment [35, pp. 70-88]. Three transmitters were mounted at known positions on the ceiling of the test environment and the AGV detected these signals, relayed to it by means of wireless communications to a base station calculating the AGV position in the unknown space – overcoming the primary limitation of dead reckoning referred to in paragraph 2.6.1.

2.6.3 Control and avoidance

Control of the AGV to navigate and to avoid obstacles implies the use of different techniques. This is AGV specific and Swanepoel used serial commands from the controller to the motor drive, utilising ultrasonic object detection in a telemetric manner using a microcontroller interface [7, pp. 27-48]. Applying avoidance techniques is just as vast a field of study and Lubbe used GAs for making decisions on object avoidance, utilising Single-Chromosome-Evolution-Algorithms in the decision making process [36, pp. 48-56].

The possibility exists that more than one AGV will be used in a reconfigurable environment, thus the reason for looking at communication between the vehicles seen in the work of Nguyen et al. [37, pp. 35-40]. The results obtained by Lee, address the issue of collision avoidance for mobile robots [38, pp. 136-141]. This is also significant for the current research.

2.6.4 Path navigation

In a factory or manufacturing environment the walkways have lines – an example of which can be seen in Figure 2.28 - or a chroma variation.



Figure 2.28: Example of a factory floor with lines and chroma changes [39]

Having to change as little as possible in a factory, this idea was taken as a possible solution in navigation as can be seen by the work done by Sotelo et al. [40] with its application on a road seen in Figure 2.29. The Figure 2.29 shows some extractions of their work indicating the border identification of such a scenario to be used for navigation by staying on a pathway.

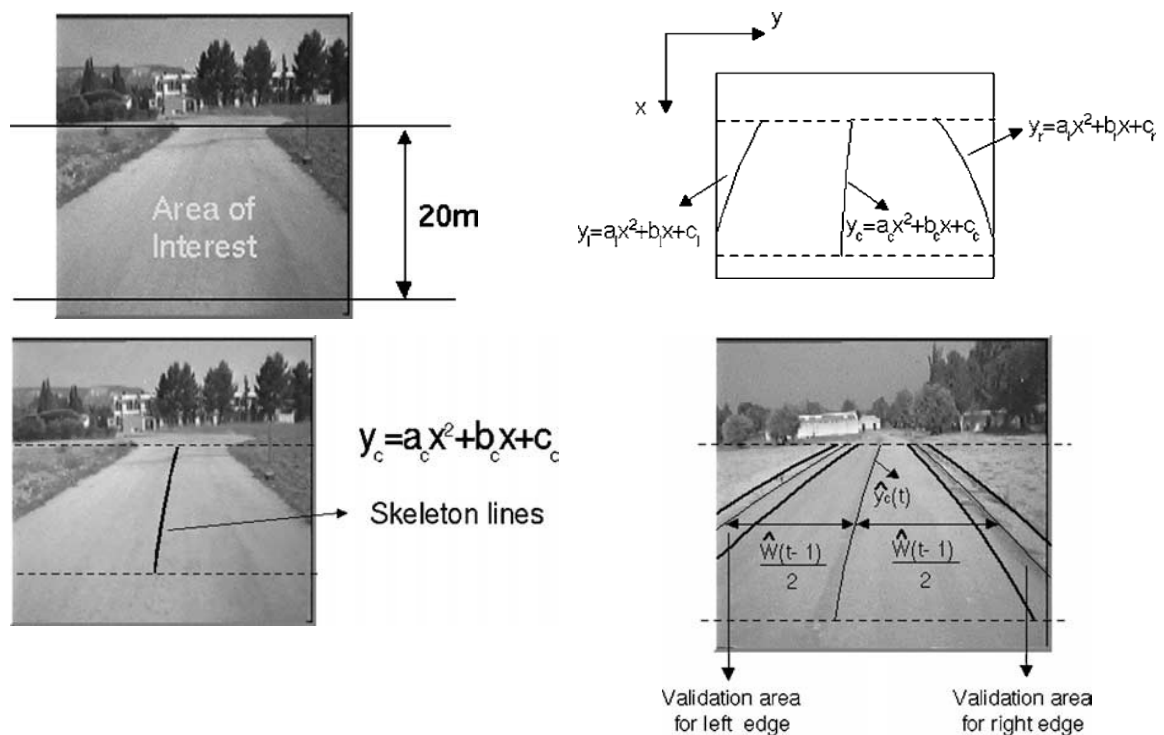


Figure 2.29: Extractions of Sotelo et al.'s [40] work in using border and chrominance in navigation

Figure 2.29 indicates an area viewed as area of interest. Skeleton lines are created as a route to follow derived from the polynomials created from the validated edges of the route travelled on.

2.6.5 Sign navigation

During the course of the research other researchers were also found, opting for the implementation of signs in the navigation process.

The research of Goedemé et al. implemented an omnidirectional camera as sensor. A topologically organised environmental map was created, using a fast feature matching algorithm between a pair of images taken from different viewpoints, focussing on man-made objects or patterns [41].

Park et al. introduced arrow signs for robot navigation utilising a wireless camera and implementing image processing algorithms [42, pp. 382-386]. Their process was based on the following steps:

- Convert the picture to a binary image,
- Remove small objects and noise from the image,
- Do region segmentation,
- Label the segmented image,
- Give a different colour to each segment and number them,
- Identify the arrow sign from the different segmented regions and separate it, and
- Find out whether the arrow is a left- or a right direction arrow.

This process is illustrated by Figure 2.30 (a) to (d).

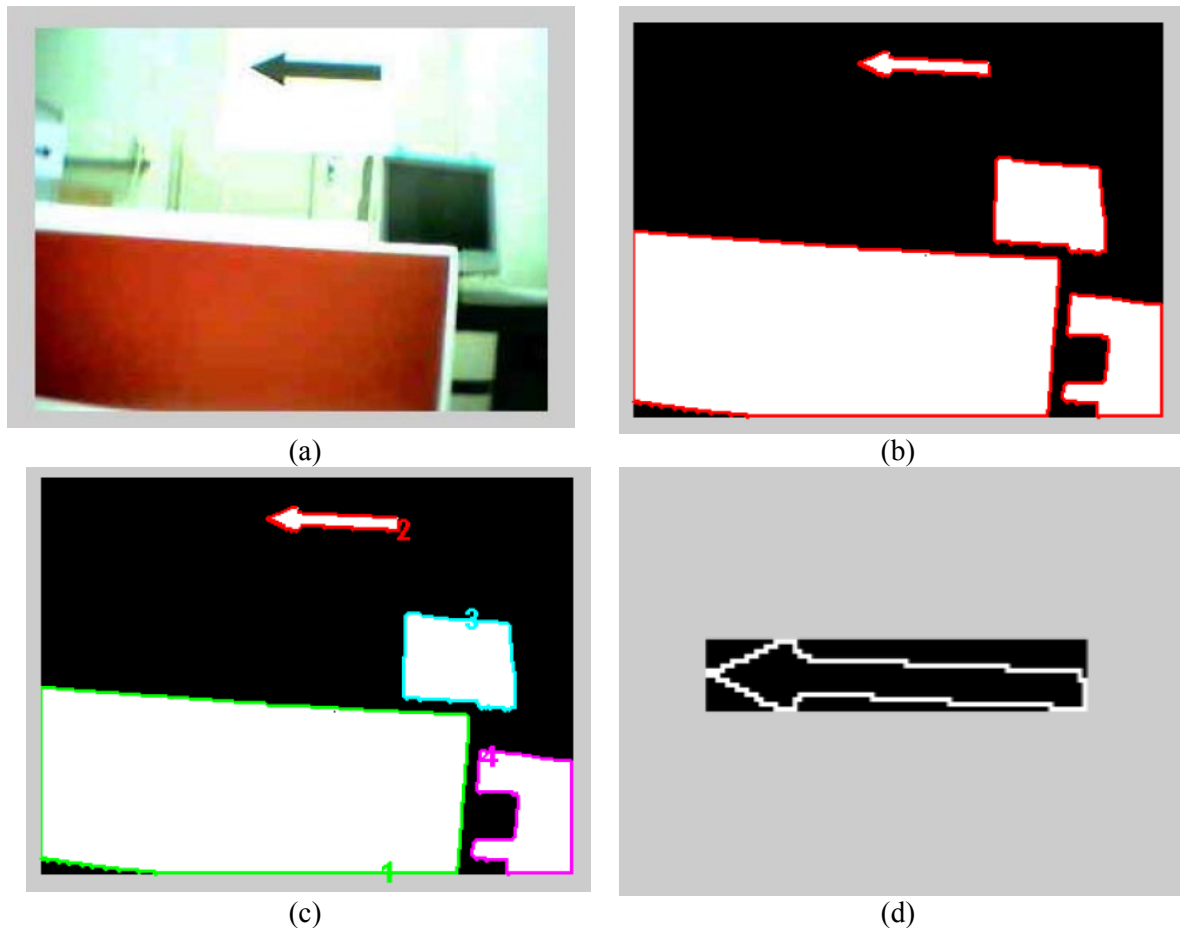


Figure 2.30: Park et al.'s. (a) image acquisition; (b) segmentation; (c) labelling and (d) arrow extraction

The researches of Zakir et al. have presented an approach towards road sign detection and recognition. Their system utilizes a method of colour segmentation by employing the HSV colour space and using empirically determined threshold values suitable for various illumination conditions. A shape classification methodology was proposed in which road sign shapes are classified by introducing the use of Contourlet Transform with a support vector machine (SVM) classifier. The recognition stage introduces the SVM classifier with the local energy based shape histogram (LESH) features. They are currently working on real time application of the algorithm within an in-car navigation system [43].

2.7 Summary

This chapter covers the vision input and possible manipulation of a picture for navigation and control purposes. No mention was made of the cameras, lighting, hardware (AGV) and the software platform for the support of these systems as most of these components were already available and specified for use in the research project.

Some location and navigation aspects are addressed with mention made of communication between AGVs and collision avoidance. The possible use of road and sign navigation is also mentioned as a solution to navigate an AGV.

Chapter 3

Development of an omnivision system for navigational purposes for an AGV

This chapter covers the development process of the omnivision system, the choice of camera and the software development platform decided on. The objective was to be able to use these separate systems as an integrated vision unit in the final product, generating usable vision outputs to be used for the control and navigation process.

3.1 Introduction

The intelligent navigation and control of AGVs involve environmental detection. Such capability can be mounted onboard or remotely. Sensors and cameras used for detecting objects in the path of an AGV, as well as the distance to the object as mentioned in section 1.1, were to be replaced by a single omnivision sensor for navigation and control purposes. This concept of utilising vision rather than ultrasonics is best illustrated by Figure 3.1 where the ultrasonic sensor only returns a distance to an obstruction for manipulating purposes and the picture from the camera could be utilised for analysing the environment for a more informed decision making process.

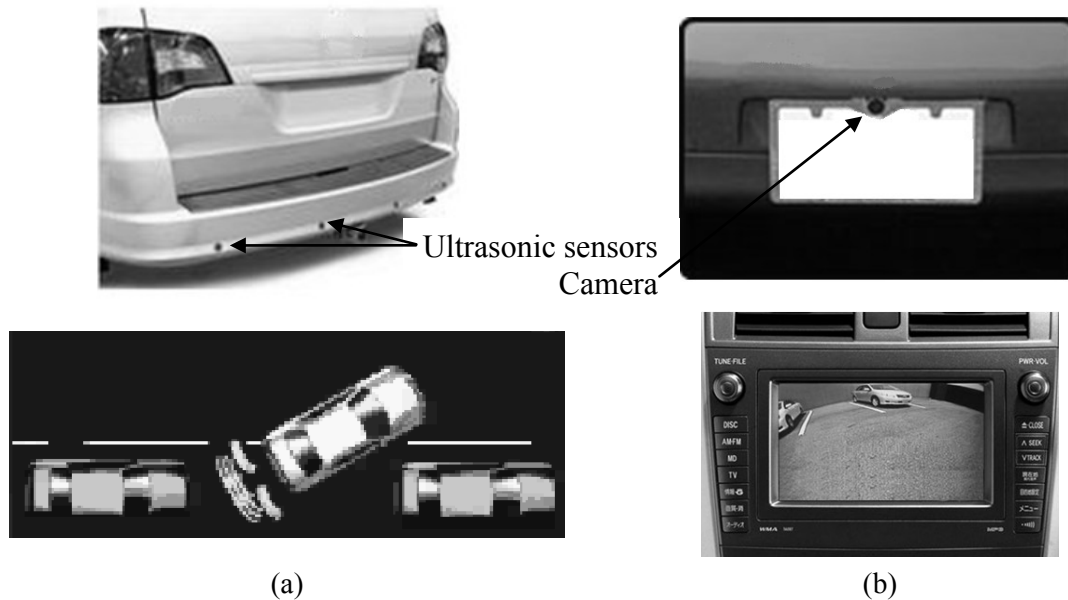


Figure 3.1: (a) Ultrasonic sensors used to sense a distance to an obstruction; (b) Camera used in sensing distance and image of obstacles

In the project the objective then was the development of an omnivision, navigation and control unit on a suitable software platform producing the necessary AGV control outputs. This concept was tested and evaluated with a program producing panoramic pictures from the omnidirectional camera setup.

3.2 Mirror and camera development for omnidirectional sensing

An omnidirectional sensor was first developed, consisting of a half sphere mirror and camera connected to a Central Processing Unit (CPU) via Universal Serial Bus (USB) and fire wire (IEEE 1394) depending on the different cameras used. The setup in Figure 3.2 was mounted on top of an AGV as the omnidirectional sensor [44].

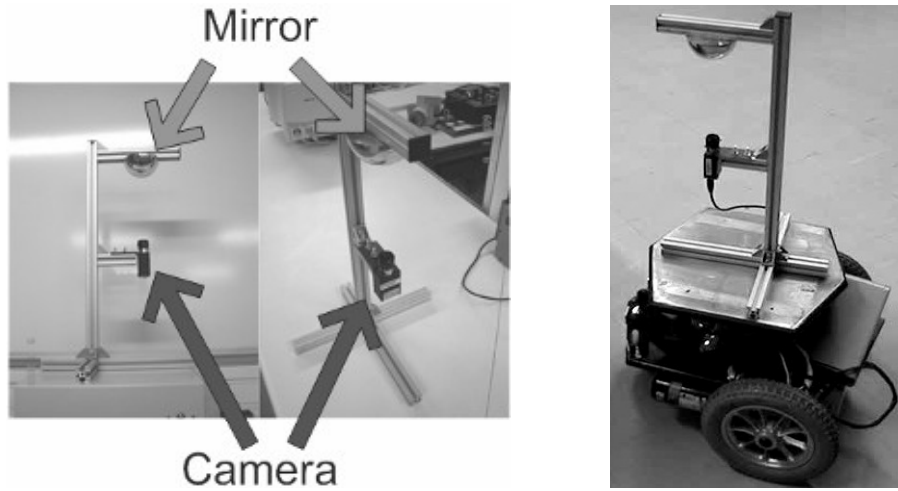


Figure 3.2: Original omnidirectional sensor setup to be placed on AGV

The resultant pictures taken, or video streamed, are in a circular shape as shown in Figure 3.3.



Figure 3.3: Half sphere mirror picture before conversion using the Basler

This signal was then converted by means of a polar transform to a panoramic picture as shown in Figure 3.4.



Figure 3.4: Converted panoramic picture

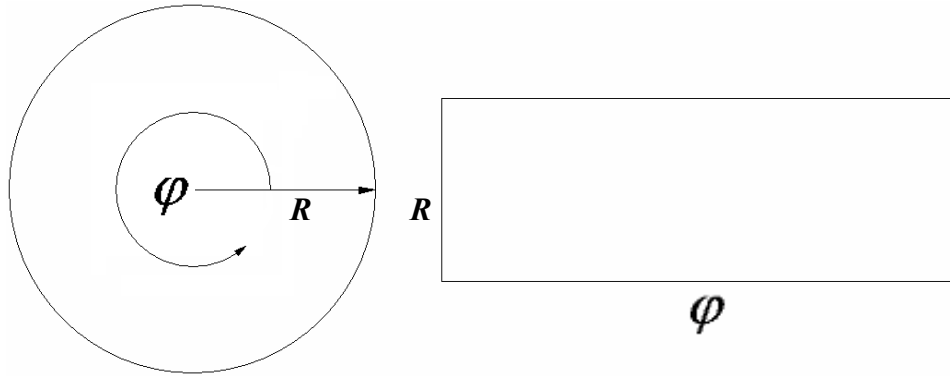


Figure 3.5: Graphical representation of a polar transform

The polar transform from Figure 3.3 to Figure 3.4 is executed using equation (3.1) shown below:

$$\begin{aligned}
 R &= \text{radius} \\
 \phi &= \left(\varphi \times \frac{\text{resolution}}{180^\circ} \times \pi \right) \text{rad} \\
 X_{\text{Pixel position}} &= R \cos(\phi) + X_{\text{Centre offset}} \\
 Y_{\text{Pixel position}} &= R \sin(\phi) + Y_{\text{Centre offset}} \\
 0 &\leq R \leq \text{Maximum radius} \\
 0^\circ &\leq \varphi \leq 360^\circ
 \end{aligned} \tag{3.1}$$

where *Maximum radius* represents the height of the frame to be converted and φ the resolution width of the panoramic view as can be seen in Figure 3.5.

The polar transfer function was developed in MATLAB[®] and then re-written in the Microsoft[®] Visual Studio[®] 2008 C# compiler. Reasons for this were to minimise transfer or calculation errors which could be overlooked or might be difficult to test for, if the code was written directly in C#. It is also recognised as good practice by academia to test the accuracy and functionality of mathematical functions on a mathematical platform.

A test pattern generated for testing the accuracy of these transfers can be seen in Figure 3.6. This test pattern and similar ones were used scientifically in determining the accuracy of measurements with the transfer of pictures from the round shape to a panoramic view.

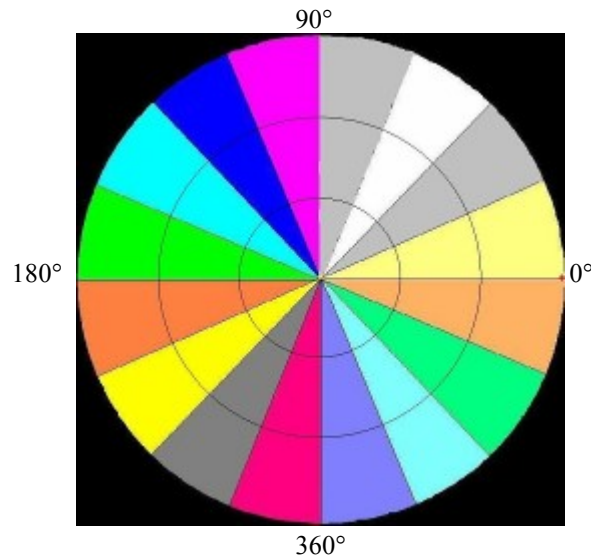


Figure 3.6: Test pattern generated for polar transform tests

The result obtained by the MATLAB[®] and C# functions of the polar transfer function with the test pattern as input can be seen in Figure 3.7 [45, pp. 1835-1839].

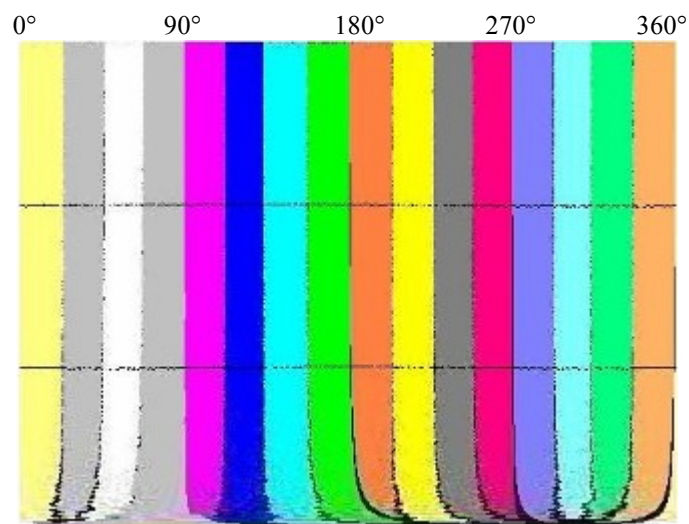


Figure 3.7: Results generated by polar transfer – conversion starting at 0° resulting in a mirror image of the photo

Figure 3.7 indicates a slight sinusoidal distortion of the transferred image. This is due to choosing the incorrect centre point on the mirror and camera setup or selecting the wrong image centre point when running the software. This distortion, if any, is not visible or is

negligible in normal environmental images - as can be seen in the image, Figure 3.9, which is a transferred image of Figure 3.8.

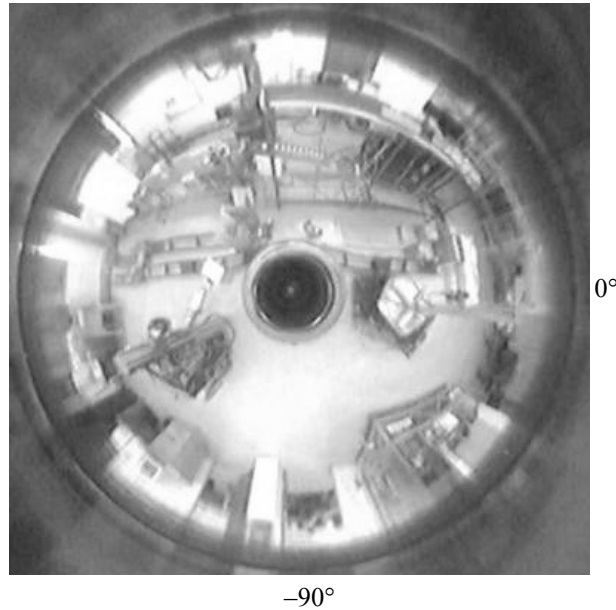


Figure 3.8: Environmental picture in circular form (680 x 670 pixels), mirror image using a Webcam



Figure 3.9: Transferred image of Figure 3.8, -90° corrected and mirror image effect corrected

Figure 3.9 was generated with the MATLAB[®] function with a radial step resolution of 1° [46] [47]. This function does the transform on pixel level and is very time consuming. It took almost 1 second for the image of 2.25 MB to be transformed with this function, on an Intel[®] Pentium[®] 3.4GHz CPU with 3.25 GB of RAM.

Figure 3.10 shows an extract of the MATLAB[®] M-file for creating the result shown in Figure 3.7 using the test pattern in Figure 3.6 at a resolution of 0.8° .

```

% select picture for processing
A = imread('C:\Testpatern.JPG');           % Figure 3.6
% select area of interest
A = A(40:726,24:702,:);
figure, imshow(A)

% centre and the radius
xc = 340;
yc = 342;
radius = 333;
% display centre and radius
hold on;
plot(round(xc),round(yc),'yx','LineWidth',2); % yellow centre
plot(round(xc+radius),round(yc),'r+','LineWidth',2); % red + at end

startradius = round(radius);
stopradius = 0;
degrees = 0.8; % resolution or width
stopang = round(360/degrees);

for thetac = 0:1:stopang
    rst = 0;
    for rsteps = startradius:-1:stopradius
        Ypix =
            round((rsteps*sin(thetac*degrees/180*pi))+(yc));
        Xpix =
            round((rsteps*cos(thetac*degrees/180*pi))+(xc));
        rst = rst + 1;
        tranf(rst,(stopang-thetac)+1,:)=A(Ypix,Xpix,:);
    end
end

figure
imshow(tranf)

```

Figure 3.10: MATLAB[®] program extract – polar to cartesian

3.2.1 Improvement on previous omnidirectional design

With previous transforms a mirror image and the direction (front of AGV to be in the centre of the conversion) of the AGV were not correctly transformed as can be seen in the transforms of Figure 3.3 to Figure 3.4 and depicted in Figure 3.11. The mirror image effect was already corrected but the forward direction of the AGV was not yet centred in the transformation.



Figure 3.11: Transform with image facing the front not centred, but mirror image effect corrected already

Figure 3.12 shows a transform where the forward direction is depicted in the middle of the transform.

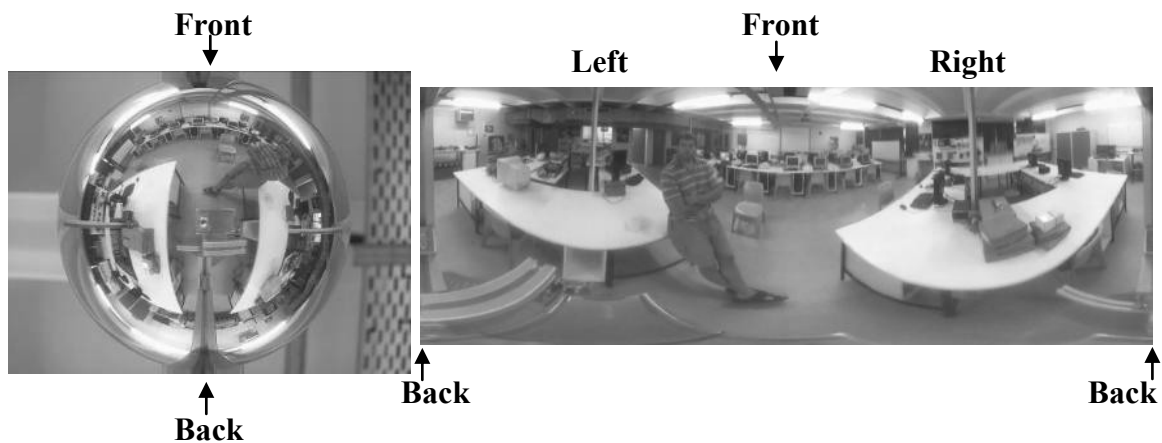


Figure 3.12: Correct transform with front of picture in the middle

The improved transform was accomplished by having the start angle at 270° and incrementing the angle for conversion in an anti-clockwise direction. The transform was generated by a MATLAB[®] M-file written as a function and compiled to an exe-file by the *mcc* command [discussed with a reference to the readme file from MATLAB[®], Appendix B.1]. The exe-file was then used in a Graphical user interface (GUI) written in C# for more graphical flexibility in auto fitting the picture for import, counting of pixel positions for input of conversion and getting the user settings [48]. Figure 3.13 is a representation of the form of the transformation GUI written in C#.



Figure 3.13 C# developed GUI utilising a transformation exe-file compiled from a MATLAB[®] M-file

In the image to be converted, depicted in Figure 3.13, it is evident that there is some image deformation. A circumference half the diameter of the mirror corresponds to around 30° in a mirror angle. The image deviation is more than double that at an angle of 60° . This is more evident in Figure 3.14 showing the relative image sizes of two identical letter A's at different angles from the centre of the mirror.

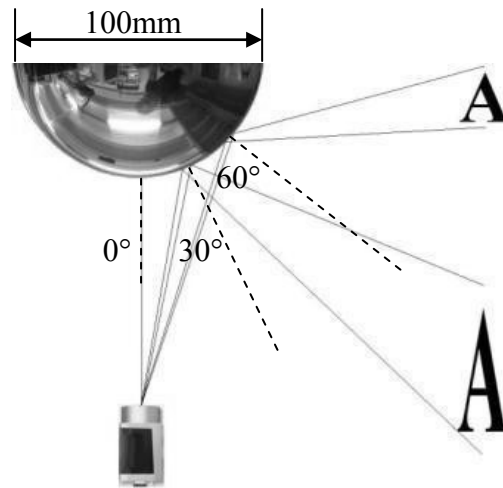


Figure 3.14 Image deformation using a half sphere mirror

A hyperbolic mirror, with a diameter of 25 mm, was then implemented instead of the two half sphere shaped mirrors, diameters of 150 mm and 100 mm respectively. The original sizes of 150 mm and 100 mm were changed because of the focal length and physical size of the setup. The results of Scaramuzza's research proved that a polar transfer function is not enough for creating a good panoramic image [4]. The implementation of a hyperbolic mirror, mounted in a round Perspex tube located on a Webcam (shown in Figure 3.15), does improve the quality of transformation. The deformation of images (letter As) at different angles reflected is compensated for by the shape of the mirror rather than software compensation utilising valuable conversion time.

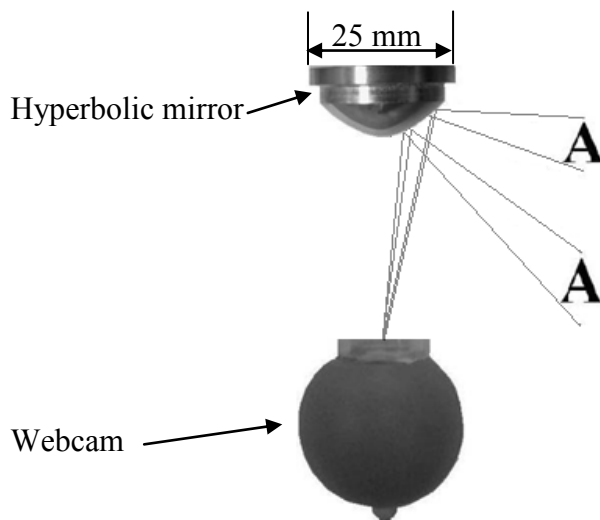


Figure 3.15 Hyperbolic mirror setup on a Webcam

The use of the hyperbolic mirror proved to be a great improvement, but insignificant deformation still exists. Calibration and interpolation may rectify these imperfections. The level of illumination is dependent on the diameter of the reflective mirror, as shown in equation (3.2).

$$P = |I| \times A_{surface} = |I| \times 4\pi r^2 \quad (3.2)$$

where P represents the power reflected from the object, I the light intensity, and r the radius of the area of a sphere reflected on. If the light intensity stays constant, a change in the radius of the reflective surface would result in a drop in the power reflected, causing a lower intensity picture [49].

Although the final omnidirectional transform used in the research was not calibrated and a low level of illumination/light intensity achieved, the transform generated could be used successfully for image processing.

3.3 Development of omnidirectional sensing software

Throughout the research process appropriate mathematical models were developed and tested in MATLAB[®]. The functions were then transferred to a C# compiler environment to create an .exe file for implementation on the hardware. The reasoning was to have an industry ready code available when finished with the research.

This process of changing MATLAB[®] code in an m-file format to C code was not without conversion problems. For example, converting a variable from the unit double to integer in MATLAB[®] meant that the value was rounded off to the closest integer value. In cases where the value overflows on its maximum bit count a value of zero was expected, but MATLAB[®] codes it to a possible maximum value – unexpected but mathematically correct, called “saturate on integer overflow” [50]. With C code the value after the decimal point is simply omitted. Table 3.1 gives a more detailed description of these conversions. The explicit System.Convert class of C# compensates for this possible error.

Table 3.1: Conversion differences of MATLAB[®] and C code from double to integer

Original value	MATLAB [®] code	MATLAB [®] result	C code	C code result
x = 23.5000	<code>y = int16(x)</code>	<code>y = 24</code>	<code>double x = 23.5;</code> <code>int y;</code> <code>y = x;</code>	<code>y = 23</code>
X = 23.4000	<code>Y = int16(X)</code>	<code>Y = 23</code>	<code>double X = 23.4;</code> <code>int Y;</code> <code>Y = X;</code>	<code>Y = 23</code>

The initial vision and control system utilised a controller driven by C code on the AGV for vision and control purposes and to act as communication hub for implementation of these algorithms.

MATLAB[®] proved to be capable of enabling this whole process on the same software development platform without converting it to C# [50]. Consequently the system was adapted, utilising a laptop personal computer as a vision and control system running on MATLAB[®] code.

This initial development was done on single pictures taken in the omnidirectional setup that needed to be changed to a video streamed system. Simulink[®] was incorporated for this purpose. Figure 3.16 shows the initial conversion model development in accessing the camera by utilising the *From Video Device* data block. The *Embedded MATLAB Function* was written (depicted in Figure 3.17), incorporating the conversion model seen in Figure 3.10. The *X Centre*, *Y Centre* and *Insert Text* blocks were used for setting up the camera for the omnidirectional hardware and small calibration changes for image centring purposes.

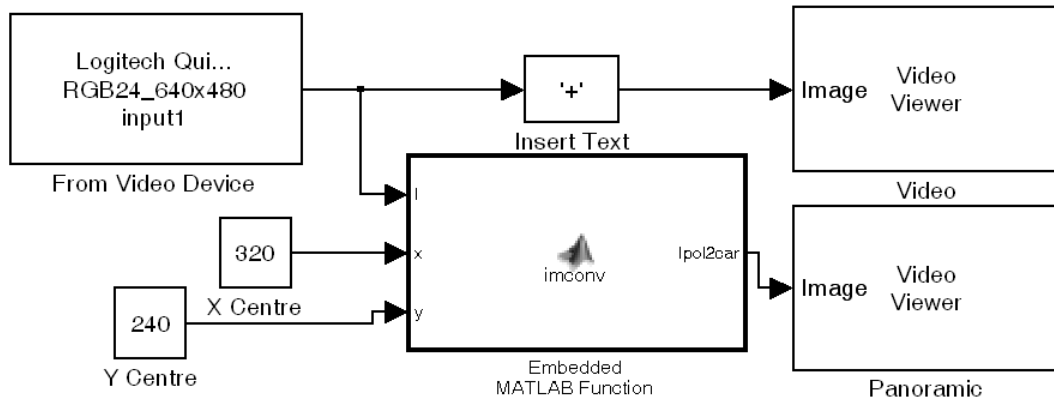


Figure 3.16: Simulink[®] model for converting omni picture to panoramic picture stream

```
function Ipol2car = imconv(I, x, y)
%Function to unwrap poly to cartesian
%tic
xc = x;
yc = y;
startradius = 200;
stopradius = 0;
degrees = 1;
stopang = round(360/degrees + 90); %450 start 90% offset
res = 360;
Ini = I;
Ino = zeros(startradius+1, res, 3);
for thetac = 90:1:stopang % 90deg - 450deg
    rst = 1;
    for rsteps = startradius:-1:stopradius
        Ypix =
            round((rsteps*sin(thetac*degrees/180*pi))+(yc));
        Xpix =
            round((rsteps*cos(thetac*degrees/180*pi))+(xc));
        if((rst ~= 0)&&((stopang-thetac)~=0))
            Ino(rst, (stopang-thetac), :) = Ini(Ypix, Xpix, :);
            %upright mirror
        end
        rst = rst + 1;
    end
end
Ipol2car = Ino;
%toc
```

Figure 3.17: Embedded MATLAB[®] function block called Imconv in Figure 3.16

The embedded MATLAB[®] function block depicted in Figure 3.17 was evaluated to determine the expected frame rate for the polar to cartesian conversion in conjunction with the video acquisition and display thereof. The *tic* and *toc* m-functions were used to start and stop the timer in determining the elapsed conversion time (see Figure 3.17, function extract).

A selected area of interest from the initial frame of the Webcam was selected, having the input frame size of 492×738 pixels (Figure 3.18). Various output frame sizes were selected to correlate with the necessary area of interest converted and eventually the frame size of an .mp4 recorder (96×128) was used for generating video clips in testing the developed software in MATLAB[®]. This is best illustrated by Figure 3.18. The obtained results are shown in Table 3.2.

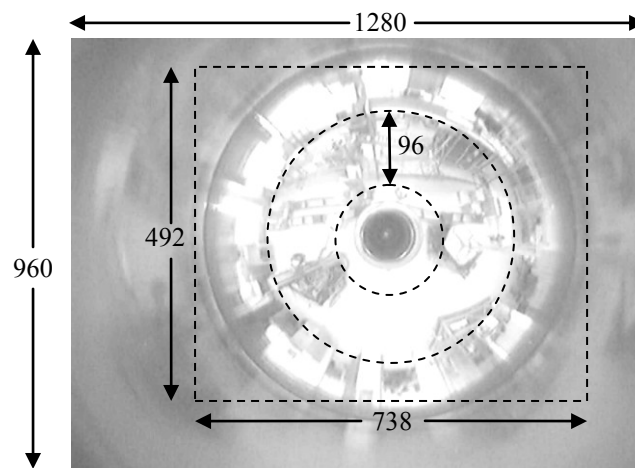


Figure 3.18: Illustration of capturing a frame, selecting an area of interest for conversion and final resolution for conversion utilizing a Webcam

Table 3.2: Calculated frame rate of embedded MATLAB[®] function block conversion

Input Frame size	Output frame size	Time elapsed	Calculated frame rate
492 X 738	201 X 360	0.249487 seconds	4 frames per second
492 X 738	96 X 180	0.059298 seconds	≈17 frames per second
492 X 738	96 X 128	0.043832 seconds	≈23 frames per second

With these results it is evident that the frame size is very important and a compromise had to be reached between frame size and sufficient information in the picture frame, in reaching the goal of having a vision instrument for mobile omnidirectional sensing and control.

3.4 Area of interest and utilising a low resolution webcam

With the results obtained in Table 3.2, it seemed imperative to reduce the time of computation for acquisition, conversion and display. The time required by MATLAB[®] for acquisition and display was optimised within the limitation of the software. More viable options were utilising a lower resolution camera, selecting a limited area of interest and changing the omnidirectional transform program thus reducing the processing time. A Webcam was implemented, replacing the BASLER A600f camera, resulting in a lower resolution.

The reasoning behind the concept of a limited area of interest was that the AGV would need only the information of a limited area in the direction of movement. This concept is illustrated by Figure 3.19.

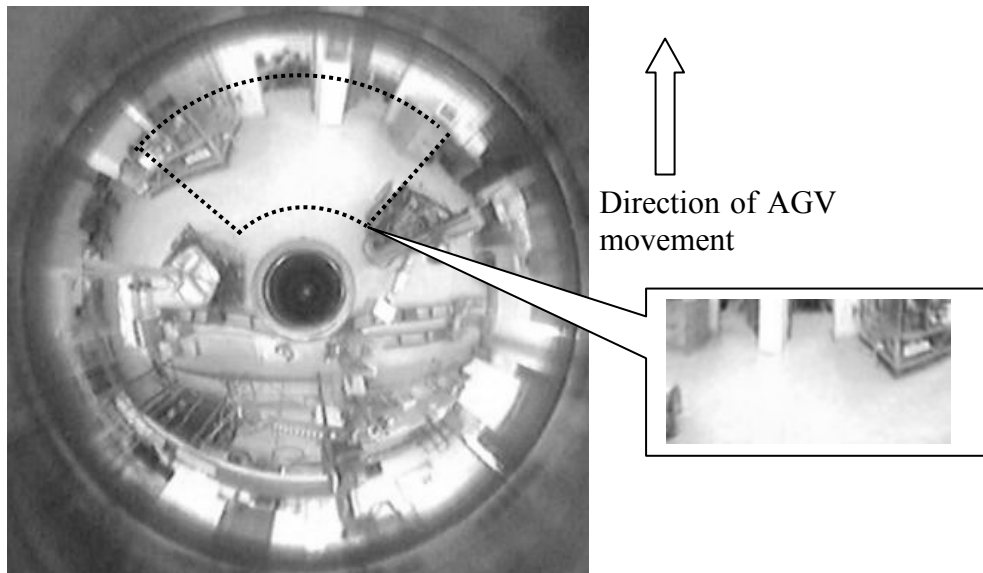


Figure 3.19: Frame from omni video stream indicating the direction of movement, area of interest and converted section of image

The evaluation of the conversion time saved was initiated by obtaining the maximum frame rate the Webcam can produce by displaying the obtained video stream directly, without conversion. Evaluating the acquisition and display time only, the acquired frame size was altered, as could be seen in Table 3.3. The Webcam settings were set to obtain a frame rate of thirty frames per second (30 frames/second).

Table 3.3: Webcam set to 30 frames/second with relevant frame size and frame rate obtained

Acquisition Frame size	Displayed frame size	Frame rate obtained
320 X 240	320 X 240	15 frames per second
640 X 480	640 X 480	15 frames per second

Thus the frame size made no real difference to the frame rate in comparison to the acquisition and display of the video stream, which produced a loss of 15 frames per second. The *From Video Device* and *Video Viewer* incorporated the equivalent processing time of 15 frames per second. This was further tested by removing the double *Video Viewer* configuration from the setup shown in Figure 3.16 to the diagram shown in Figure 3.20, thus losing the preview of the image setup.

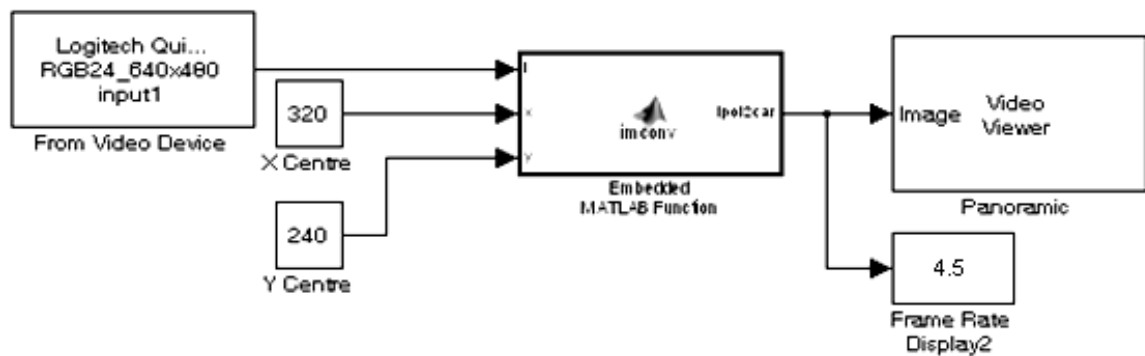
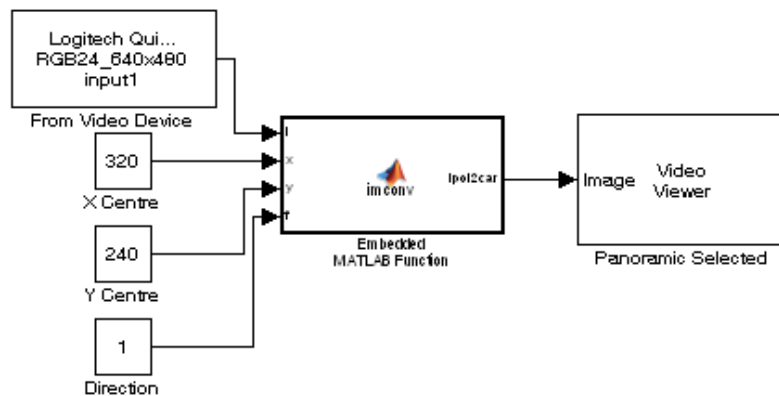
**Figure 3.20: Simulink® model for converting omnipicture to panoramic – panoramic displayed only**

Table 3.4 gives a more detailed layout of the results obtained by having a dual *Video Viewer* for the video stream received by the camera and converted panoramic view compared to a single *Video Viewer* (only the converted image) and incorporating a smaller area of interest.

Table 3.4: Double and single view frame rates incorporating different area of interest sizes converted from a 360° picture video stream

	Acquisition Frame size	Output frame size as Figure 3.19 indicate	Frame rate obtained
Double view	640 X 480	720 X 186	3.5 frames per second
Single view	640 X 480	720 X 186	4.5 frames per second
Single view	640 X 480	360 X 186	7.5 frames per second
Single view	640 X 480	180 X 96	14 frames per second

Opting for the single view only improved the frame rate by one frame per second (28.6% improvement). The largest change was by using a smaller frame size to convert. This prompted the design of having an input to select the viewing direction as shown in Figure 3.21.

**Figure 3.21: Simulink® model for converting omnipicture to area of interest including direction of movement**

This feature resulted in an initial option of only having four possible viewing directions, to be controlled by the AGV movement control software, determining in which direction to look, as can be seen in Figure 3.22.

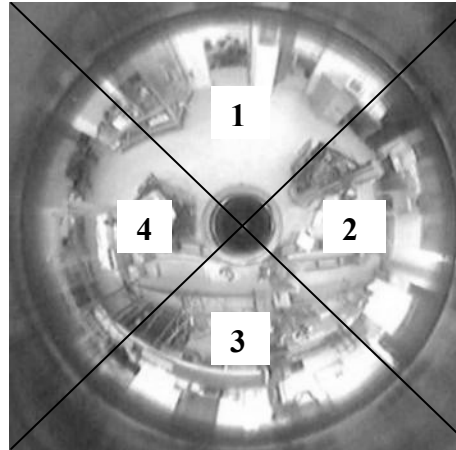


Figure 3.22: Omnipicture indicating the four directions of area of interest selected as video input

The option substantially increased the number of frames per second available for image processing as the area of interest is in the direction of possible future movement, hence smaller in frame size. However this concept was never implemented.

3.5 Transferring the omnisoftware from computer to laptop platform

All of the development work was done on a Microsoft Windows XP Professional Version 2002 with Service Pack 3 and an Intel[®] Core™ Duo CPU E8400 @ 3.00GHz with 2.98 GHz, 1.99 GB of RAM personal computer (PC).

MATLAB[®] has a feature, called *bench*, to evaluate the processing strength of the machine in different calculating areas. The result for the specific computer is shown in Figure 3.23 and Figure 3.24.

This machine compared well to the other computer platforms compared to in the group, in terms of computation performance.

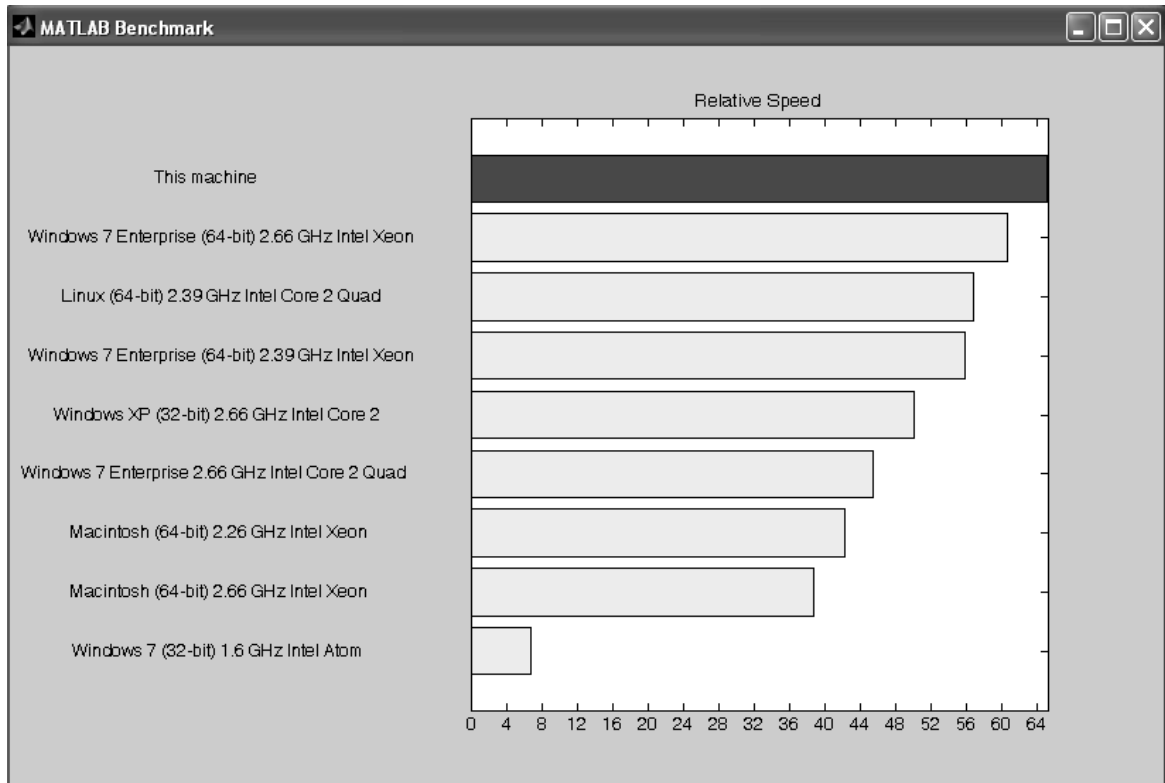


Figure 3.23: MATLAB[®] *bench* feature being displayed as a graphical result of the PC

The figure shows a window titled 'MATLAB Benchmark (times in seconds)'. It contains a table with columns for Computer Type, LU, FFT, ODE, Sparse, 2-D, and 3-D. The rows list various computer configurations and their corresponding times in seconds for each operation.

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
This machine	0.1844	0.3456	0.1140	0.2003	0.2933	0.1903
Windows 7 Enterprise (64-bit) 2.66 GHz Intel Xeon	0.0938	0.1735	0.1235	0.1551	0.3180	0.7287
Linux (64-bit) 2.39 GHz Intel Core 2 Quad	0.1288	0.3144	0.1981	0.2535	0.2713	0.2232
Windows 7 Enterprise (64-bit) 2.39 GHz Intel Xeon	0.1051	0.2110	0.1349	0.1913	0.3423	0.6994
Windows XP (32-bit) 2.66 GHz Intel Core 2	0.2319	0.3741	0.1257	0.2325	0.3262	0.5788
Windows 7 Enterprise 2.66 GHz Intel Core 2 Quad	0.1994	0.3168	0.1465	0.2504	0.4267	0.7367
Macintosh (64-bit) 2.26 GHz Intel Xeon	0.0711	0.2452	0.2363	0.2359	0.4108	0.9290
Macintosh (64-bit) 2.66 GHz Intel Xeon	0.1293	0.3442	0.2280	0.3114	0.5021	0.7656
Windows 7 (32-bit) 1.6 GHz Intel Atom	7.3691	2.0275	0.7489	1.5453	2.3965	1.9017

Place the cursor near a computer name for system and version details. Before using this data to compare different versions of MATLAB, or to download an updated timing data file, see the help for the bench function by typing 'help bench' at the MATLAB prompt.

Figure 3.24: MATLAB[®] *bench* feature being displayed as a result of the PC, for the process speed in seconds

Figure 3.23 and Figure 3.24 clearly indicates that the particular PC used outperformed the other computer platforms to which it was compared. The comparison data for other computer platforms is stored in a text file, “bench.dat”. Updated versions of this file are available from MATLAB[®] Central [51].

The MATLAB[®] code was transferred to a laptop to be used on the AGV with Microsoft Windows XP Professional Version 2002 with Service Pack 3 and an Intel[®] Core™ Duo CPU T7500 @ 2.20GHz with 789 MHz, 1.99 GB of RAM. The MATLAB[®] version used on the PC and laptop was MATLAB[®] 7.12 (R2011a).

The *bench* feature was used again on the laptop and the results are shown in Figure 3.25 and Figure 3.26. This laptop did not fare as well as expected compared to the others in the group in terms of computational performance.

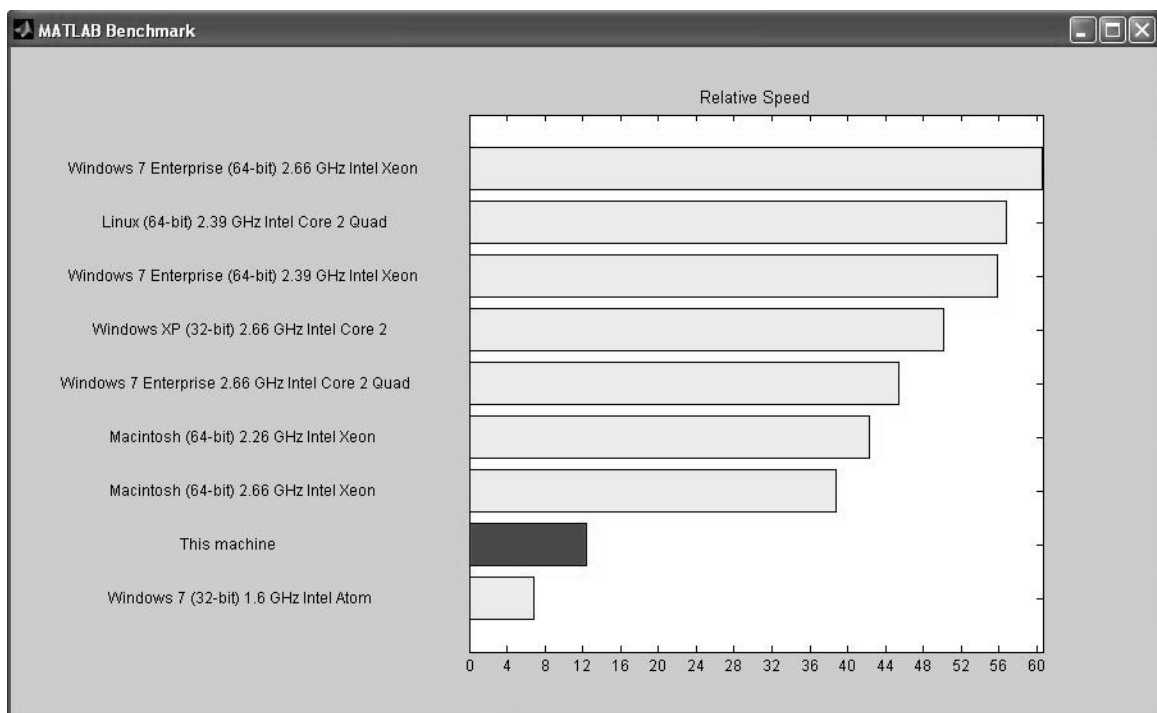


Figure 3.25: MATLAB[®] *bench* feature being displayed as a graphical result of the laptop

Computer Type	LU	FFT	ODE	Sparse	2-D	3-D
Windows 7 Enterprise (64-bit) 2.66 GHz Intel Xeon	0.0938	0.1735	0.1235	0.1551	0.3180	0.7287
Linux (64-bit) 2.39 GHz Intel Core 2 Quad	0.1288	0.3144	0.1981	0.2535	0.2713	0.2232
Windows 7 Enterprise (64-bit) 2.39 GHz Intel Xeon	0.1051	0.2110	0.1349	0.1913	0.3423	0.6994
Windows XP (32-bit) 2.66 GHz Intel Core 2	0.2319	0.3741	0.1257	0.2325	0.3262	0.5788
Windows 7 Enterprise 2.66 GHz Intel Core 2 Quad	0.1994	0.3168	0.1465	0.2504	0.4267	0.7367
Macintosh (64-bit) 2.26 GHz Intel Xeon	0.0711	0.2452	0.2363	0.2359	0.4108	0.9290
Macintosh (64-bit) 2.66 GHz Intel Xeon	0.1293	0.3442	0.2280	0.3114	0.5021	0.7656
This machine	1.0519	1.4081	0.6333	1.0605	1.8049	1.1064
Windows 7 (32-bit) 1.6 GHz Intel Atom	7.3691	2.0275	0.7489	1.5453	2.3965	1.9017

Place the cursor near a computer name for system and version details. Before using this data to compare different versions of MATLAB, or to download an updated timing data file, see the help for the bench function by typing 'help bench' at the MATLAB prompt.

Figure 3.26: MATLAB[®] bench feature being displayed as a result of the laptop, for the process speed in seconds

Keeping these results in mind, the work done in obtaining the results in Table 3.4 was repeated on the laptop, providing the results shown in Table 3.5.

Table 3.5: Double and single view frame rates incorporating different area of interest sizes compared to the results obtained on a laptop

	Input frame size	Output frame size	Frame rate obtained PC	Frame rate obtained laptop
Double view	640 X 480	720 X 186	3.5 frames per second	0.5 frames per second
Single view	640 X 480	720 X 186	4.5 frames per second	0.7 frames per second
Single view	640 X 480	360 X 186	7.5 frames per second	1.3 frames per second
Single view	640 X 480	180 X 96	14 frames per second	2.4 frames per second

This made it evident that the frames available per second for the control and interpreting of the environment with the laptop, is very low and possibly insufficient for navigation and control purposes.

3.6 Conclusion

The research covered in this chapter proved the viability of the development of a usable omnidirectional conversion algorithm written in MATLAB[®] tested as executable in a user friendly C# GUI.

The alteration from the original half sphere mirror to a hyperbolic mirror shape saved omnicalibration- and interpolation time.

Selecting a Webcam and making use of an area of interest, enabled the saving of valuable computational time in converting an image. The vision sensor development provided a cost effective alternative to a range of sensors traditionally used in detecting the environment of an AGV, for navigation and control purposes.

MATLAB[®] was chosen as the complete software platform, generating results, evaluating the camera setup and mirror configuration on a PC and finally a laptop platform without converting the code to C# and compiling it to an executable application.

The results obtained proved that the laptop processing time was too slow for omnivision purposes for the mobile system. Implementing the concept of an area of interest in the direction of movement, provided a possible solution in using a single camera facing in the direction of movement saving computational time in developing the navigation and control software.

Chapter 4

Navigation development for the AGV

This chapter covers the navigation goals, development of a navigational system and the implementation and control of the AGV platform. This development was a move away from dead reckoning used as navigation technique on previous locally-developed AGVs, towards a vision-based navigation system.

4.1 Identifying the navigational goals

In a reconfigurable environment it should preferably be possible to alter the route that the AGV needs to travel, depending upon ordering information (origin or pickup point) and delivery of parts (destination). It is also possible that some manufactured components need to be returned for rework or final rejection. This creates a scenario where there should be flexibility in the order and route of parts to be conveyed, as can be seen in Figure 4.1 generated by DELMIA V5 Release 21. This figure indicates a few roaming AGVs each of which needs to follow a predetermined route in fetching and delivering parts or components. The use of AGVs allows flexible routes, to be changed by the operator rather than using a fixed route like a conveyor system [52].

This concept was taken as a starting point for the navigation and control of an AGV in this project. The assumption that the AGV is going to travel on a factory floor with lines and chroma changes as depicted by Figure 2.28, an example of a factory floor, is also evident in this figure.

The omnivision system was to be used for navigation and assessing the AGV's environment.

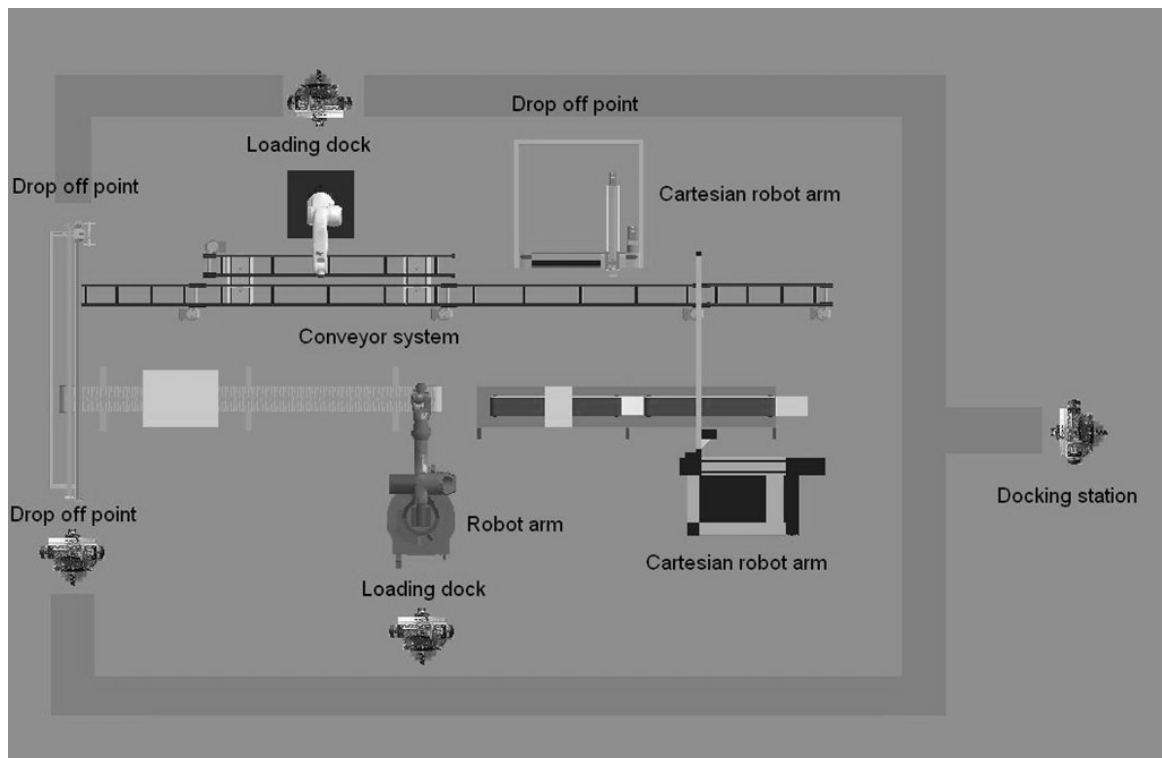


Figure 4.1: Animated layout of a simulated factory floor developed in DELMIA

4.2 Detection of movement whilst navigating utilising dead reckoning

The original platform used in the research utilised ultrasonic sensors to provide a proximity picture of the environment around the AGV [7, p. 18]. An omnidirectional video stream was implemented in detecting movement as well as to assist the operator in accessing a 360° view of the AGV surroundings.

A system of dead reckoning was utilised in determining the AGV's position in the environment and to assist in navigating the planned route.

The driven wheels were controlled through a PIC microcontroller circuit board receiving and sending commands in series to and from a Human Machine Interface (HMI) [53]. Only three such AGVs were built that were based on an electrical wheelchair platform utilising its motors and motor drive.

The serial data was transmitted and received telemetrically through a WLAN connection. The WLAN was used for its radio frequency (RF) bandwidth and range [54].

The HMI is depicted in Figure 4.2 where the distances received from the ultrasonic sensors detecting obstacles in the vicinity of the AGV were being displayed. The orientation of the AGV is reflected in the HMI by a straight line obtained by wheel distance travelled. The controller commands could be selected and the video feed could be monitored on the same HMI.

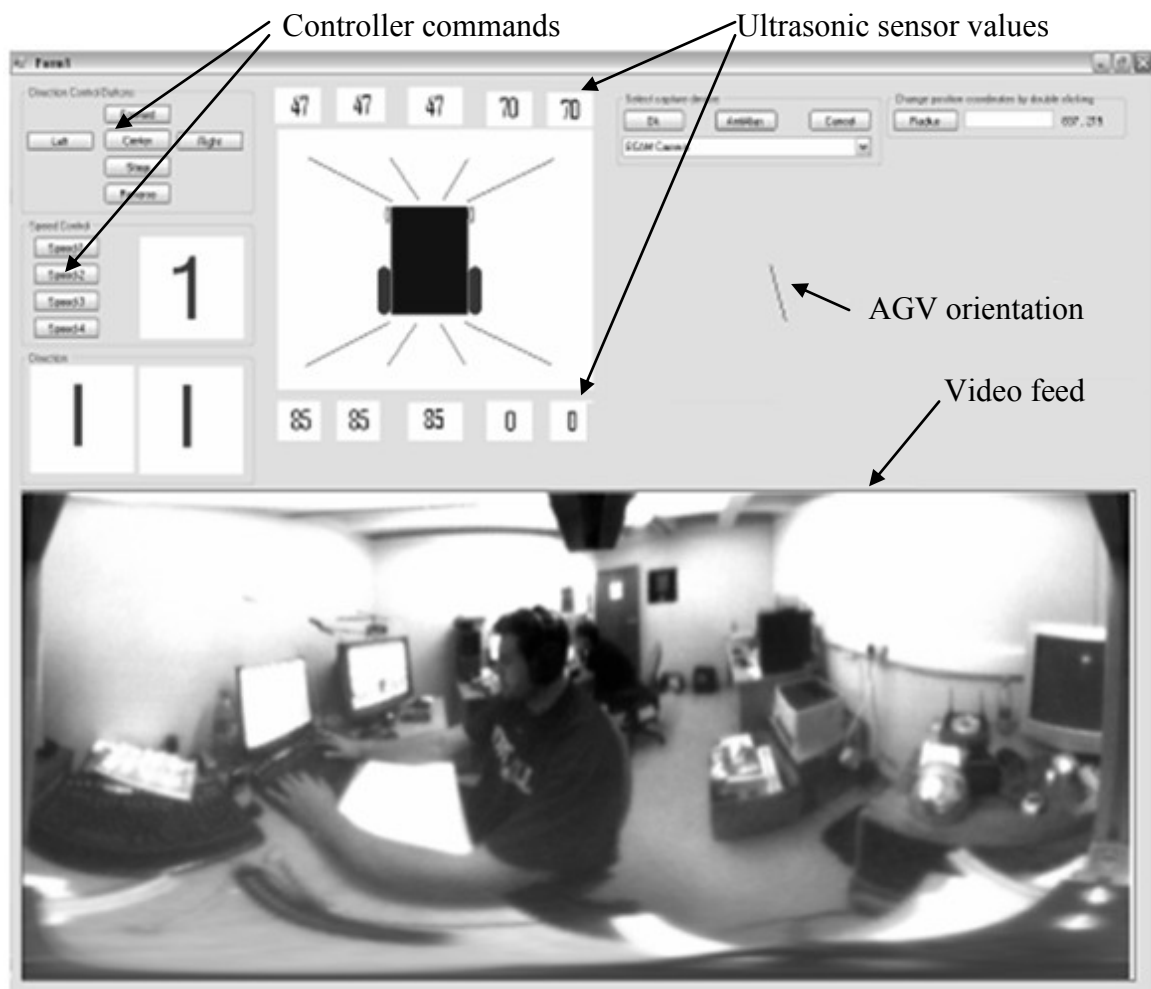


Figure 4.2: HMI screen capture [7, p. 67]

4.3 Development of a new AGV platform

The platform discussed in section 4.2 was replaced by the National Instruments™ (NI) single-board sbRIO-9632 robot platform, seen in Figure 4.3.

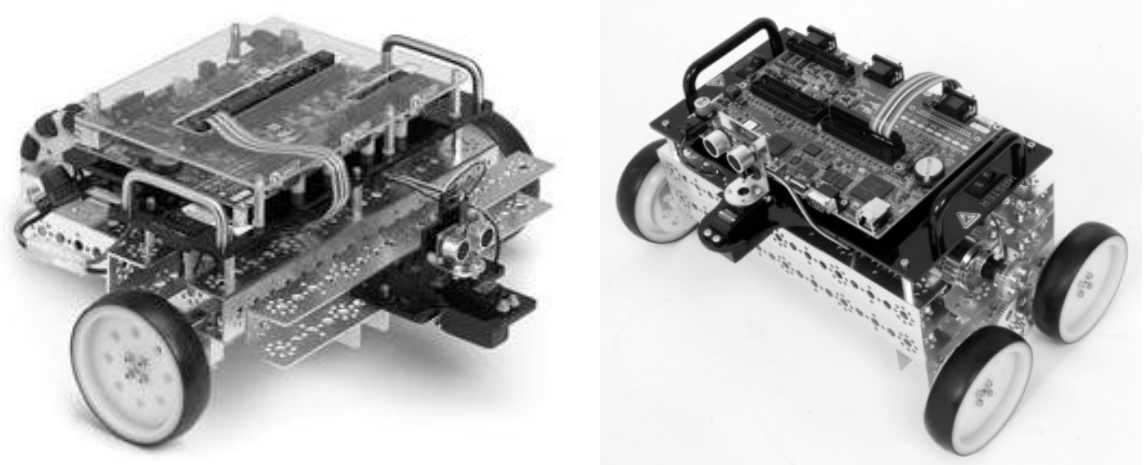


Figure 4.3: National Instruments™ single-board sbRIO-9632 robot platforms [55]

A laptop operating on MATLAB® code was utilised as processor, directly communicating through the Universal Serial Bus (USB) port to a PIC microcontroller board, developed in the RGEMS group. This board was utilised to generate the two-channel pulse width modulation (PWM) for the radio control (R/C) motor speed controller.

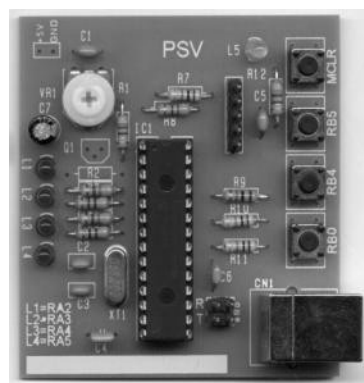


Figure 4.4: PIC microcontroller board utilised to generate the pulse width modulation

The Sabertooth R/C motor speed controller (Figure 4.5) of the NI robot platform is used for controlling two TETRIX[®] geared motors [56].

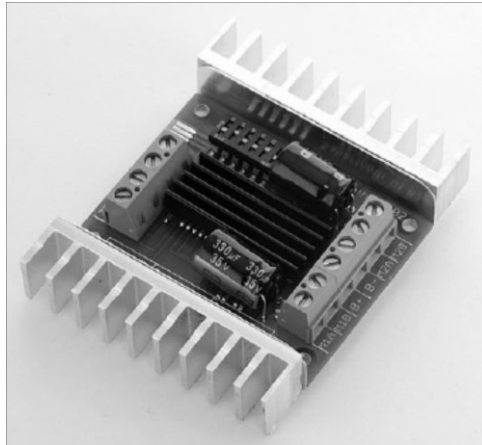


Figure 4.5: Sabertooth R/C motor speed controller used on the NI robot platforms

The final setup of the platform is shown in Figure 4.6.



Figure 4.6: AGV platform utilising a laptop, NI robot platform and omnivision system

This setup, with the option for selecting between a single or omnivision camera, was used for developing the navigation and control capabilities of the AGV [57].

4.4 Overview of the vision guided navigation system

The vision guided navigation system consists in essence of the vision capturing section split into a parallel system, using these images for route navigation and detection of colour signs for controlled navigation. Each of these two systems creates outputs which are used jointly to control the AGV platform. The complete developed system is summarised by the flow diagram in Figure 4.7 with each section referring to the work described in this chapter.

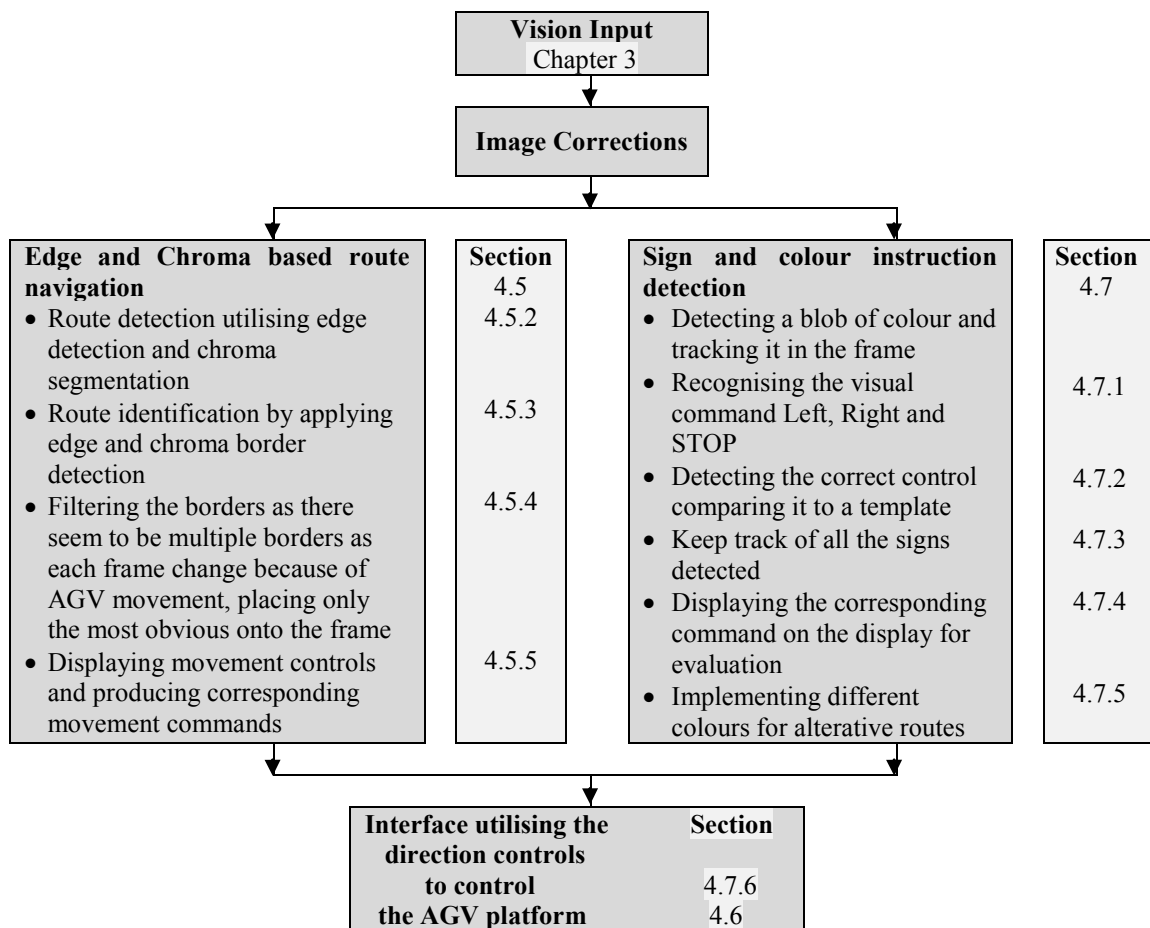


Figure 4.7: Overview flow diagram of the vision based navigation system depicting the route and sign control navigation techniques used

4.5 Route navigation concept

The use of lines on the side of a route, walkway or a chroma route is similar to a normal road surface. This is the primary reason for the evaluation of Sotelo et al.'s work, referenced in Chapter 2 [40]. MATLAB[®]'s "Chroma-based Road Tracking" demo was a starting point for the route navigation of the AGV in this research.

4.5.1 Short description of the MATLAB[®]'s Chroma-based Road Tracking demo which was the starting point for road navigation in this research

Figure 4.8 illustrates the Simulink[®] model of the demo [58]. When running the demo a pre-recorded video stream is used as source to be processed for evaluating the road tracking concepts used. The model then uses the chroma information of the frames to detect and track the road edges. The "Chroma-based Road Tracking" demo model illustrates the use of the *Colour Space Conversion* block, the application of *Hough Transform* block, and the advantage of the *Kalman Filter* block to detect and track information utilising hue and saturation values of the frames from the video.

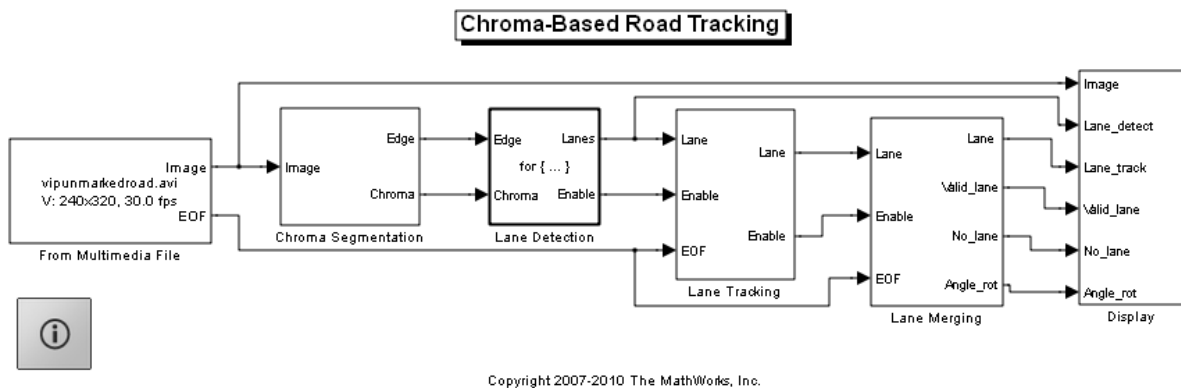


Figure 4.8: MATLAB[®] "Chroma-based Road Tracking" demo

The demo model performs a search operation to define the left and right edges of a road by analysing video frames for a change in colour behaviour. The model then selects a line either because of an edge detected, or a line created by a change of chroma pixels,

whichever have the greater precedence. The search is initiated from the bottom-centre of each frame and moves to both the upper-left and upper-right corners of each frame.

When both road sides are visible, the demo shows an arrow in the centre of the road in the direction calculated by averaging the directions of the left and right sides as could be seen in Figure 4.9.



Figure 4.9: Tracking results of the “Chroma-based Road Tracking” demo [58]

4.5.2 Edge detection and chroma segmentation used for AGV route tracking

The route used for simulation in the research project was either a route defined by blocks of carpets or a corridor with almost the same colour and shaded carpets. A single camera with low resolution (160 X 120) was initially used to detect the borders of the carpet route, utilising edge detection (as shown in Figure 4.10). The area of interest was then sized to (128 X 96) by the *Pad* function of MATLAB[®]. This frame size was selected to be able to correlate the results to those in Table 3.4 and Table 3.5 where the area of interest was selected from the omnivision system. The use of a single camera rather than the omnivision conversion in the simulation was because of the slow frame rate after conversion.



Figure 4.10: (a) Frame captured from a colour camera; (b) Prewitt edge detection applied on a frame

Prewitt edge detection gave a promising result as can be seen in Figure 4.10 (b). This was to be expected from the background theory on edge detection covered in section 2.4. In Figure 4.10 (b) it is also evident that the furthest point of the track could result in a false edge, as the lighting from a cross corridor was perceived as if the route is a dead end, which is not the case. Chroma segmentation was implemented with a threshold as the carpet beyond the apparent dead end was assumed to be of the same colour. This was achieved by utilising the *Colour space Conversion* and selecting the saturation value of the signal to be used in the thresholding process. Figure 4.11, the output of the edge and chroma detection section, indicates the generated result with the prospect of obtaining a more realistic result for the combined scene as a possible route and not a dead end.

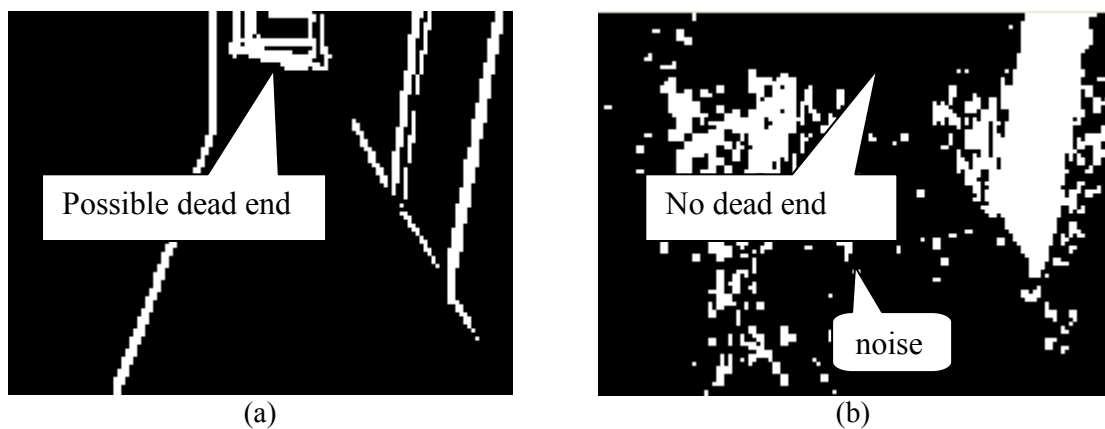


Figure 4.11: (a) Edge detection – result with Prewitt detection, indicating a probable dead end; and (b) Chroma result – indicating an open end

Figure 4.12 indicates the developed Simulink[®] model for generating the result shown in Figure 4.11 (a) and (b). The threshold value plays a big role in the number of pixels available after colour space conversion for the segmentation process comparing the so-called noise in the frame passed to those necessary for determining an edge.

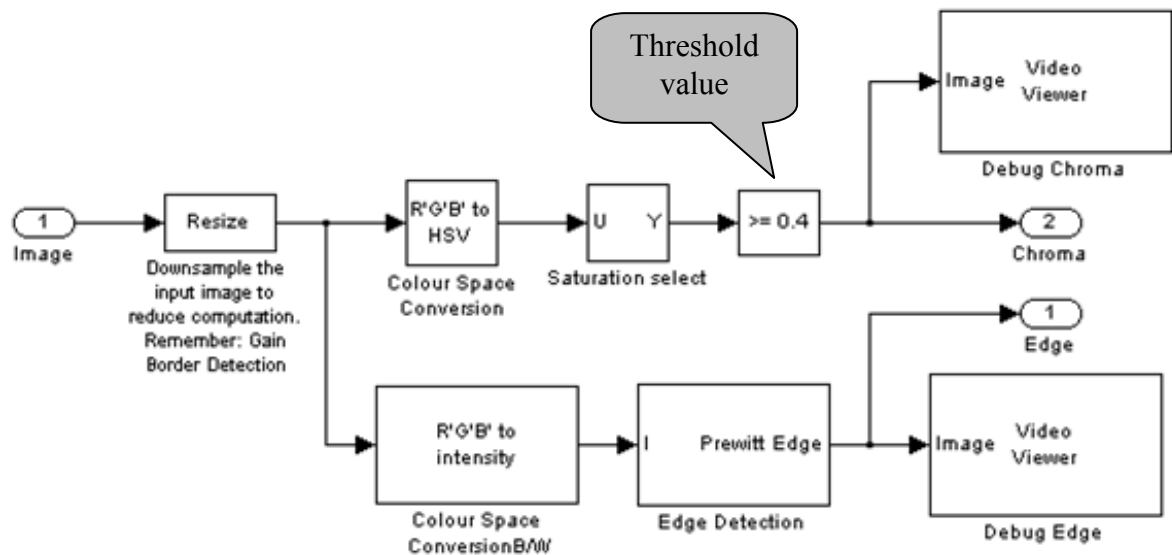


Figure 4.12: Simulink[®] model used for edge detection and chroma segmentation, source to Figure 4.14

The result gave a positive result, showing that utilising both Prewitt and chroma based detection combined gave better results than either technique individually.

4.5.3 Border detection for route identification on edge and chroma signal

As explained in section 2.5.6 the Hough transform was a good choice for detecting the route's border from the edge and chroma edge signal. As shown in Figure 2.20 a search had to be conducted through the frame detecting possible lines, utilising the *Rho* and *Theta* of those lines. This line search is process intensive and the strategy used in the MATLAB[®] demo model was adopted in splitting the frames vertically in two, utilising only half of this split frame at a time for the search process [58].

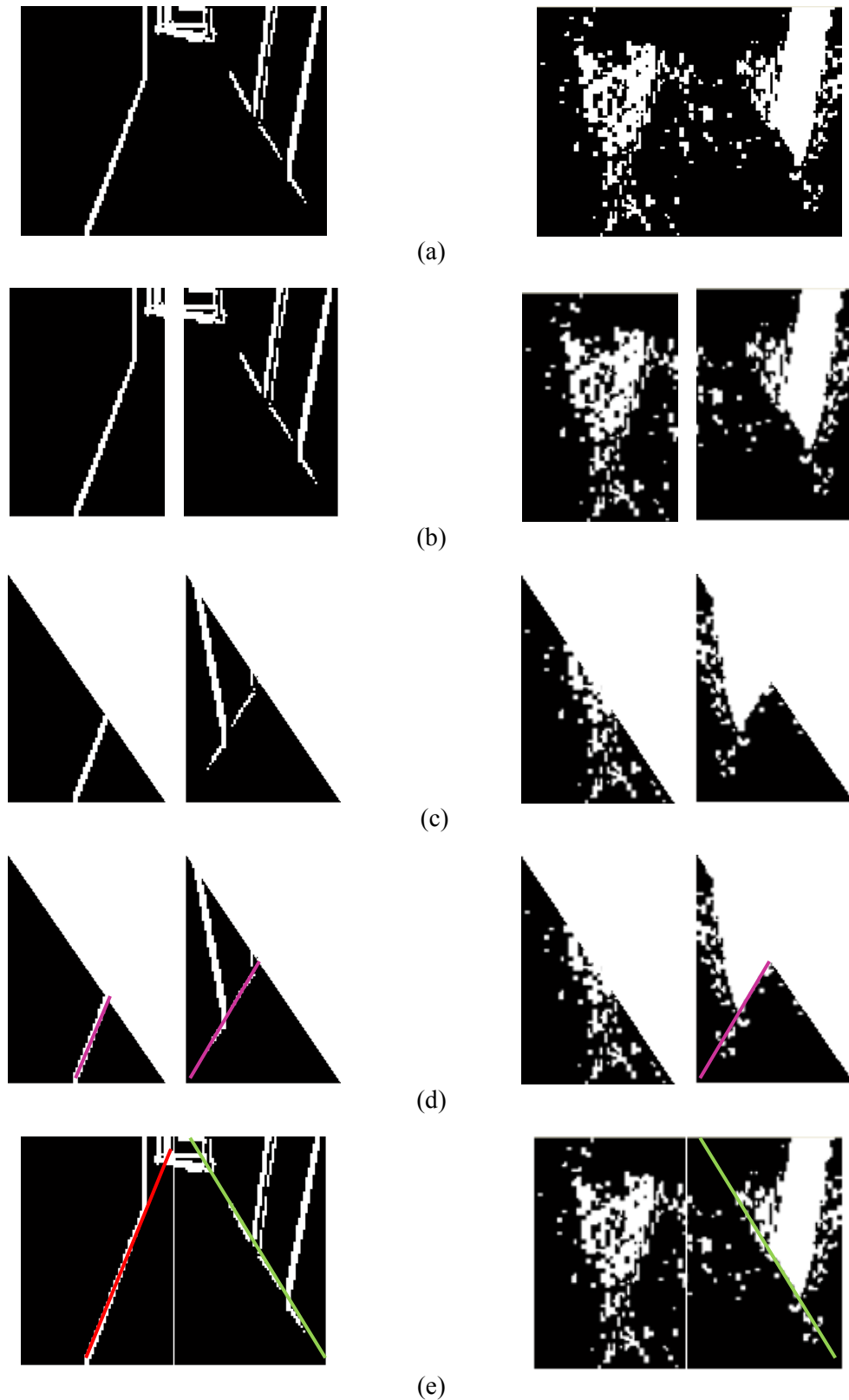


Figure 4.13: (a) Single frame of edge and chroma detection; (b) Frames split vertically in the middle; (c) Right half flipped horizontally and top right part of frame omitted; (d) Hough transform applied, detecting a border; (e) Right half flipped back and Rho $Theta$ values available for merging the lines onto the frame for evaluation

Splitting the frame as in Figure 4.13(b), already localise the search to a much smaller area. Flipping the right part of the frame has the advantage of the *Theta* staying in the same quadrant. Omitting the top part of the same search area which does not contribute to any real influence on the result minimises the search area even further (see Figure 4.13(c)). This application is possible, as a route usually diminishes towards the farthest end of the frame, as is evident in Figure 4.13 (c and d).

The results obtained in Figure 4.13 were accomplished by the Simulink[®] model, as developed by the researcher and illustrated in Figure 4.14.

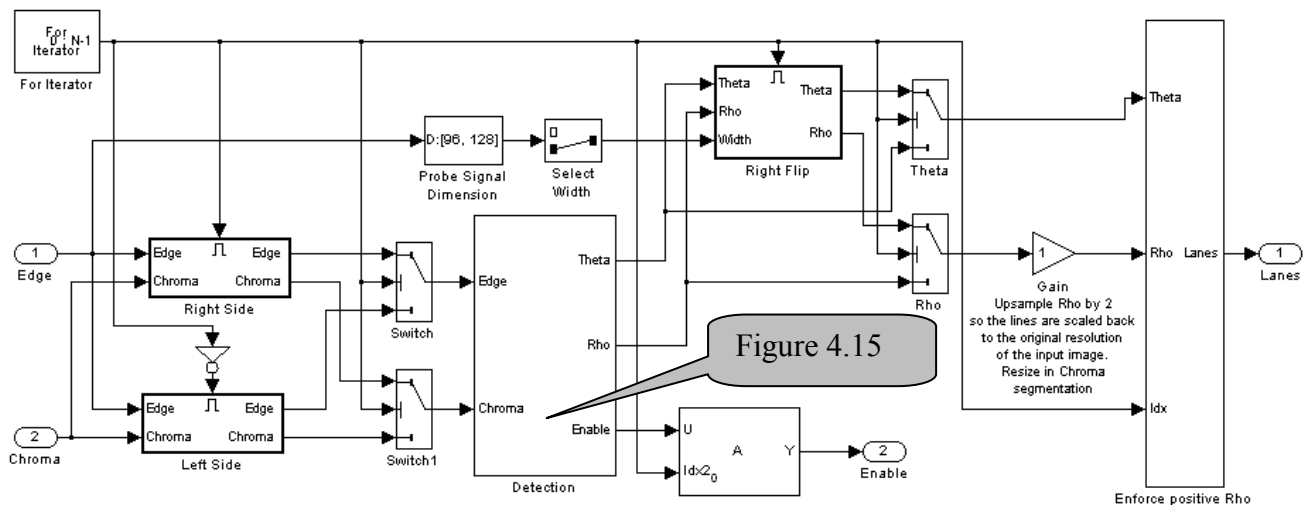


Figure 4.14: Simulink[®] model for obtaining the border lines by scanning only half of the frames

The *Detection* block in the Simulink[®] model contains the Hough transform and the algorithms for finding the most apparent line in the frame. The threshold selections of the chroma pixel number (specifying the chroma level per pixel) as well as the edge pixel number (number of pixels in close proximity to form a possible line) are also available as options as displayed in Figure 4.15.

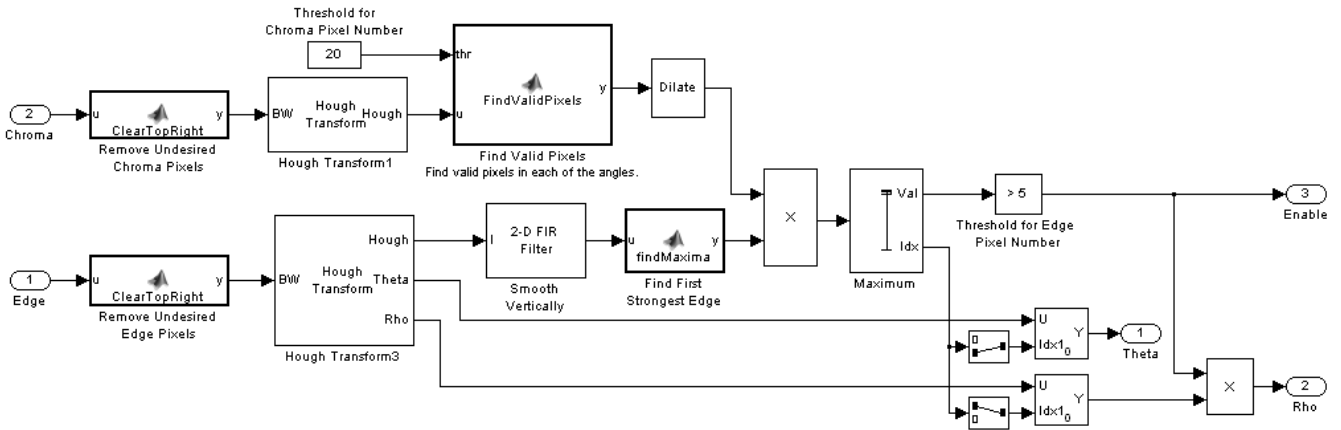


Figure 4.15: Line detection Simulink[®] model including the Hough transform incorporated in Figure 4.14

With this Simulink[®] model multiple lines are detected from a single border, because of the movement of the AGV. These needs to be filtered out and the best applicable line must be merged with the final display indicating the border of the route. This is best illustrated by Figure 4.16.

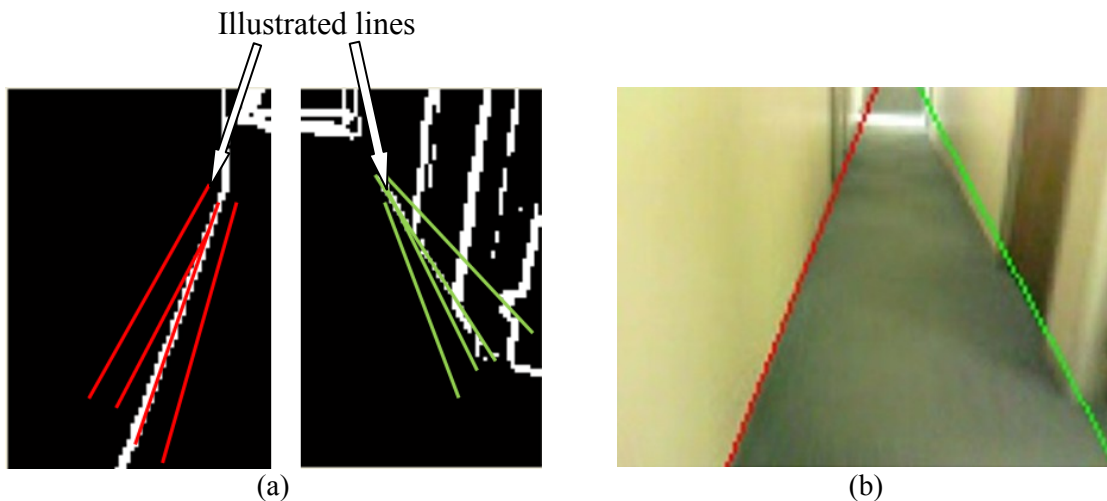


Figure 4.16: Illustrating the need for filtering on the number of lines detected, because of AGV movement, (a) multiple lines detected, and (b) actual filtered lines merged on frame

In Figure 4.16(a) it is clear that several lines for a route border seem to be detected. By utilising filtering a single route is identified. This route needs to be tracked and the result displayed (merged on the frame) for evaluation. This gives rise to the development of

Route Tracking and *Route Merging* doing just this, the result of which can be seen in Figure 4.16(b).

4.5.4 Route Tracking and Route Merging

This is largely achieved by the *Route Tracking* (Lane Tracking) and *Route Merging* (Lane Merging) blocks adopted and altered for AGV navigation, from the Simulink® demo model, seen in Figure 4.8. The *Route Tracking* block utilised is displayed in Figure 4.17.

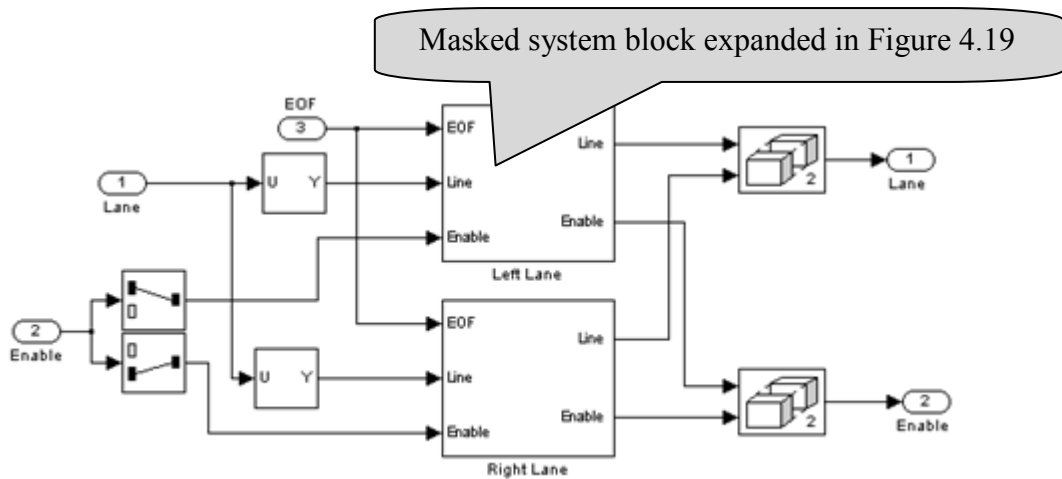


Figure 4.17: Route Tracking block for left and right border, input from Detection section Figure 4.14 sourcing Route merging to be viewed in Figure 4.20

The *Route Tracking* consists of two similar masked subsystem blocks for certain parameter settings to be changed for optimal performance. These parameters are evident and can be viewed in Figure 4.18.

Parameters

Maximum number of lanes to find in the current frame:

Maximum number of lanes to store in the tracking repository:

Maximum allowable change of lane distance metric between two frames:

Weight of the angular difference for calculating the distance metric:

Minimum number of frames a lane must be detected to become a valid lane:

Maximum number of frames a lane can be missed without marking it invalid

Figure 4.18: Function block parameters for the left and right lane subsystem of the Route Tracking model

The two blocks itself (Left and Right lane in Figure 4.17), however, consist of a *Matching* section, the *Kalman Filter* and an *Update* section, seen in Figure 4.19.

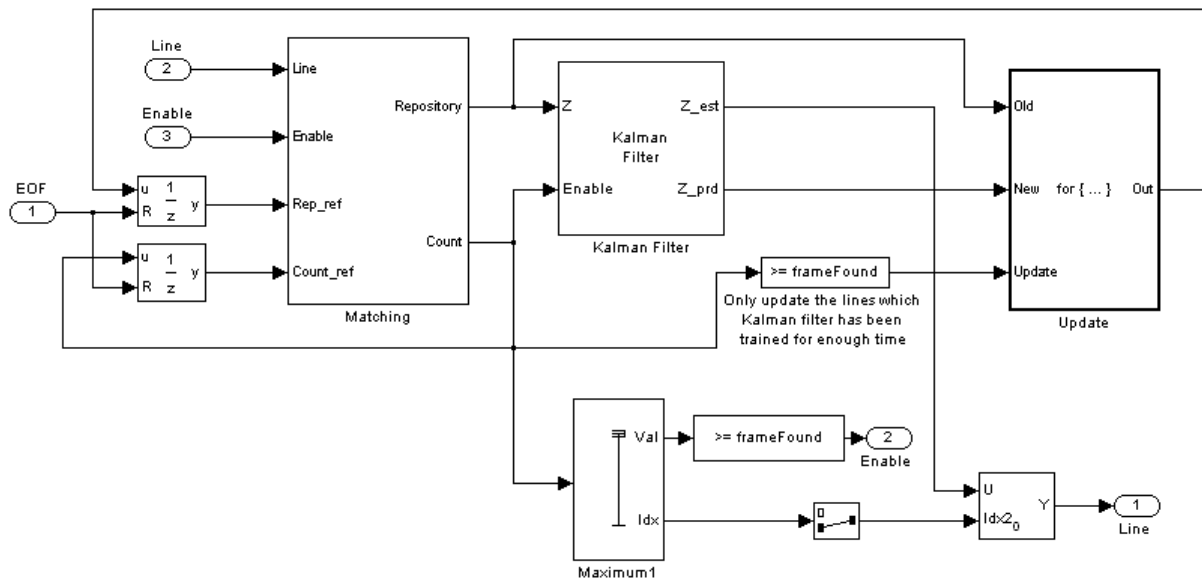


Figure 4.19: Left and Right lane masked system block consisting of the Kalman Filter

The *Matching* section calculates the *distance between the lines* found in the frame (illustrated in Figure 4.16(a)) to those in the repository. The calculation is done by equation (4.1).

$$\mathbf{Distance\ between\ lines} = |\mathbf{Rho1} - \mathbf{Rho2}| + |\mathbf{Theta1} - \mathbf{Theta2}| \times m \quad (4.1)$$

where *Rho* and *Theta* represent the separate lines and *m* the weight of the angular difference for calculating the distance between the lines.

The *Matching* section then finds the best matches between the newly detected lines and those in the repository (the multiple lines detected because of AGV movement). Selecting the most appropriate line between all these lines detected the *Kalman Filter* is implemented based upon the history of a line position and predicting the line most suitable. Simultaneously the *Matching* section updates the list through the *Update* block which the *Kalman Filter* has passed to be most suitable after being “trained enough” (enough suitable lines passed through the filter).

The *Route Merging* subsystem seen in Figure 4.20 consist of a masked block *Finding lanes* which is used to set the minimum number of frames in which a line must be detected consecutively to become a valid line. It then uses this information generating the results for the line attributes such as if the line exists, if the frame have one or two lines or if the line where merged onto a frame. This is then forwarded by the *Merge lanes* block as a line position on a frame and at which angle of rotation the line is to be displayed (illustrated in Figure 4.16(b)).

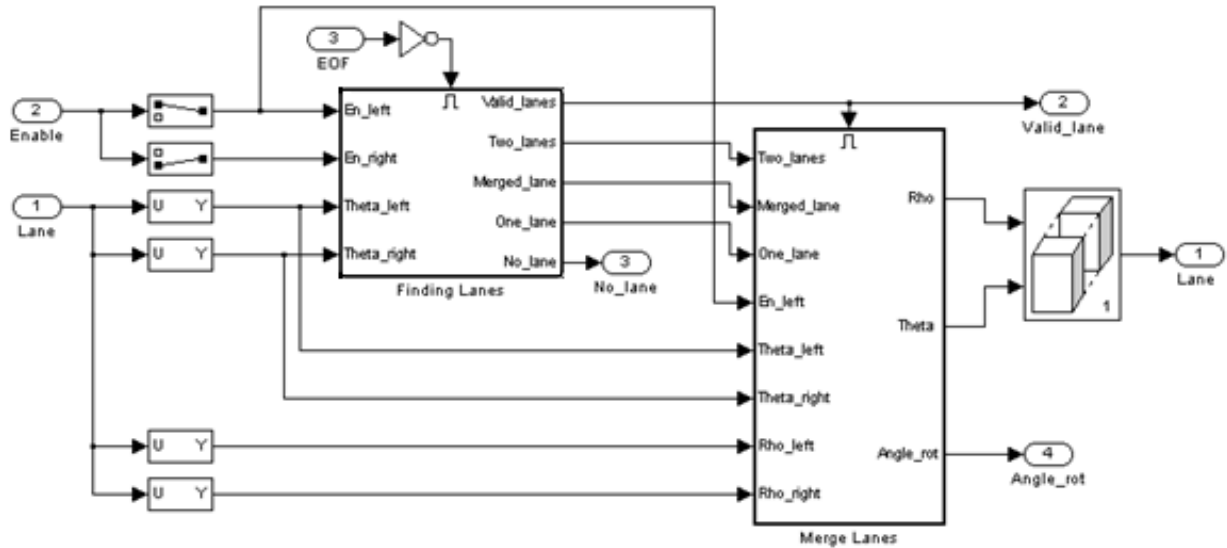


Figure 4.20: Route Merging subsystem masked block

4.5.5 Display of detected line, edge, chroma and tracking information for evaluation purposes

Although the route navigation's main aim is to generate information to control the AGV, the researcher needed some feedback on the performance of the developed systems and algorithms. This is the reason for displaying the detection of border lines (lanes), the edge and chroma signal, as well as the resultant route tracking frame with the possible command displayed as a merged signal on it. A recording feature was implemented to evaluate the results obtained by viewing the video recorded. The edge and chroma is displayed by the Simulink[®] model depicted in Figure 4.12. Figure 4.21 includes the model for displaying the border detection and route tracking results.

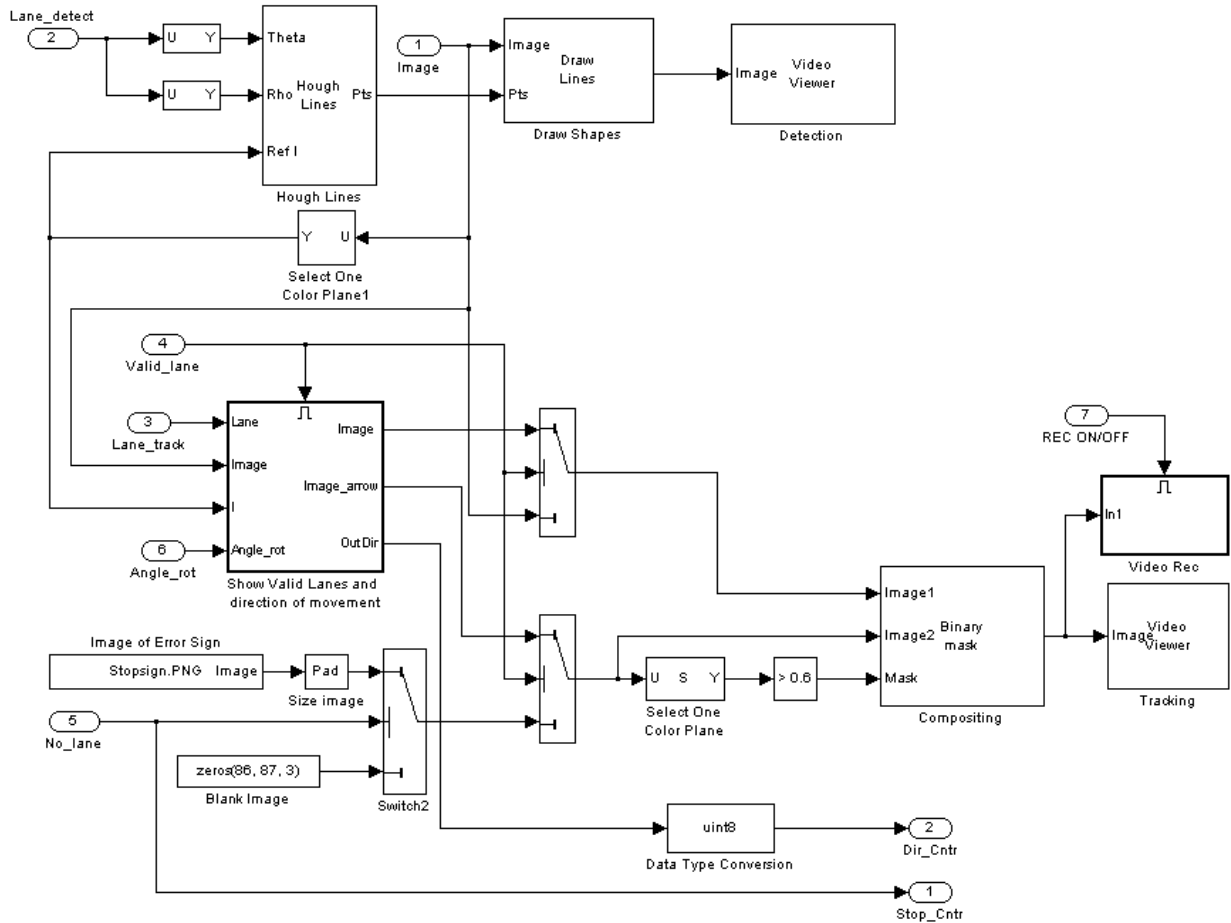


Figure 4.21: Simulink[®] model used for displaying the line detection and line tracking results

The information from the *Show Valid Lanes and direction of movement* block, in Figure 4.21, is producing the frame to evaluate with the applicable direction control, its direction and if there is a no-Route-to-follow signal. This *Show Valid Lanes and direction of movement* block is expanded in Figure 4.23. The *Theta* and *Rho* line information is used by the *Hough lines* block (Figure 4.23) to display the merged lines on top of the video frame, as shown in Figure 4.22(a). The *Binary mask*, displays the control commands as a merged signal on the Route tracking display (Figure 4.22(d)).

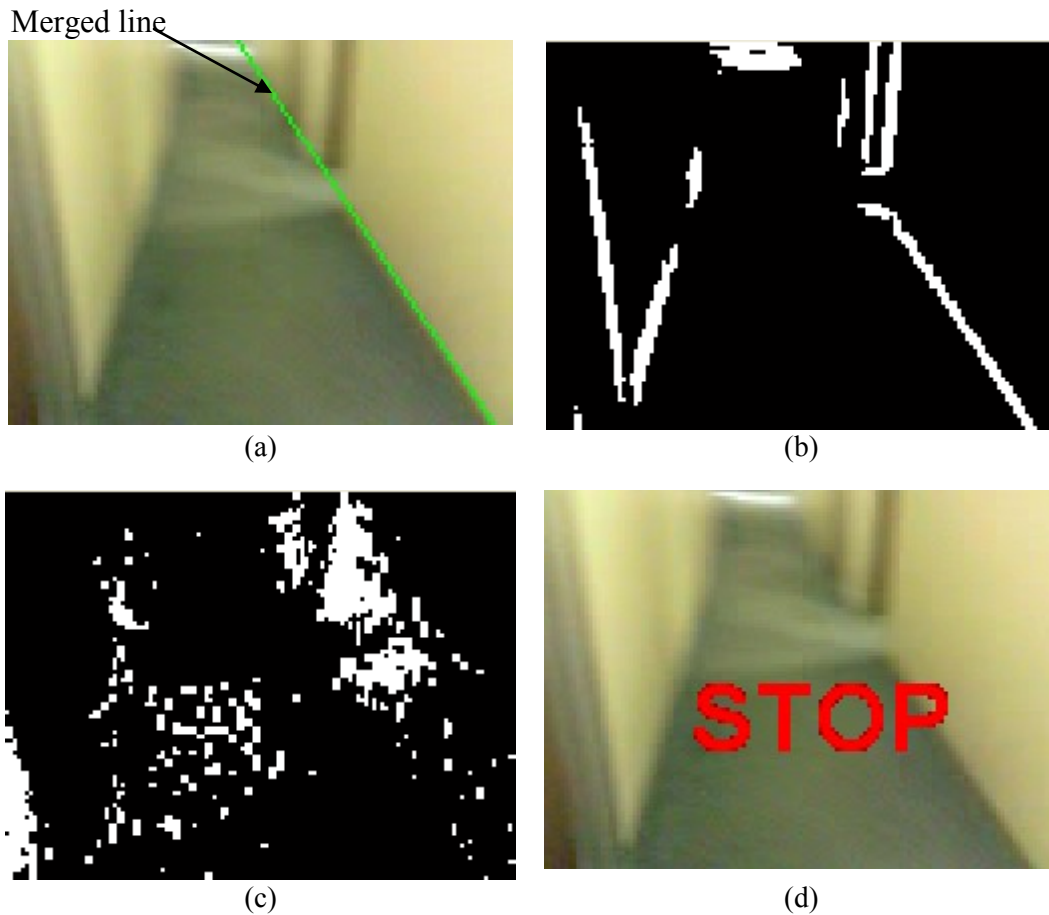


Figure 4.22: (a) Border detection with line merged on the display; (b) Edge display (c) Chroma display; (d) Route tracking display with command prompt merged on the display

A “no Route to follow signal” (STOP) was generated in the test run depicted by Figure 4.22, because Figure 4.22(b) indicates a possible dead end and Figure 4.22(c) have an indication of high noise on a possible route. Thus, the STOP was generated as combined result.

The *Show Valid Lanes and direction of movement* block in Figure 4.21 consist of three main sections:

- the *Hough lines* block responsible for the correct position for the lines on a frame to be merged on the video frame for evaluation (visually viewed if it corresponds to the edge detected, Figure 4.22(a));
- the arrow picture to be merged onto the video frame indicating the direction of movement for the AGV; and

- the *Direction Options* function block generating the values for the direction controller (see Figure 4.23).

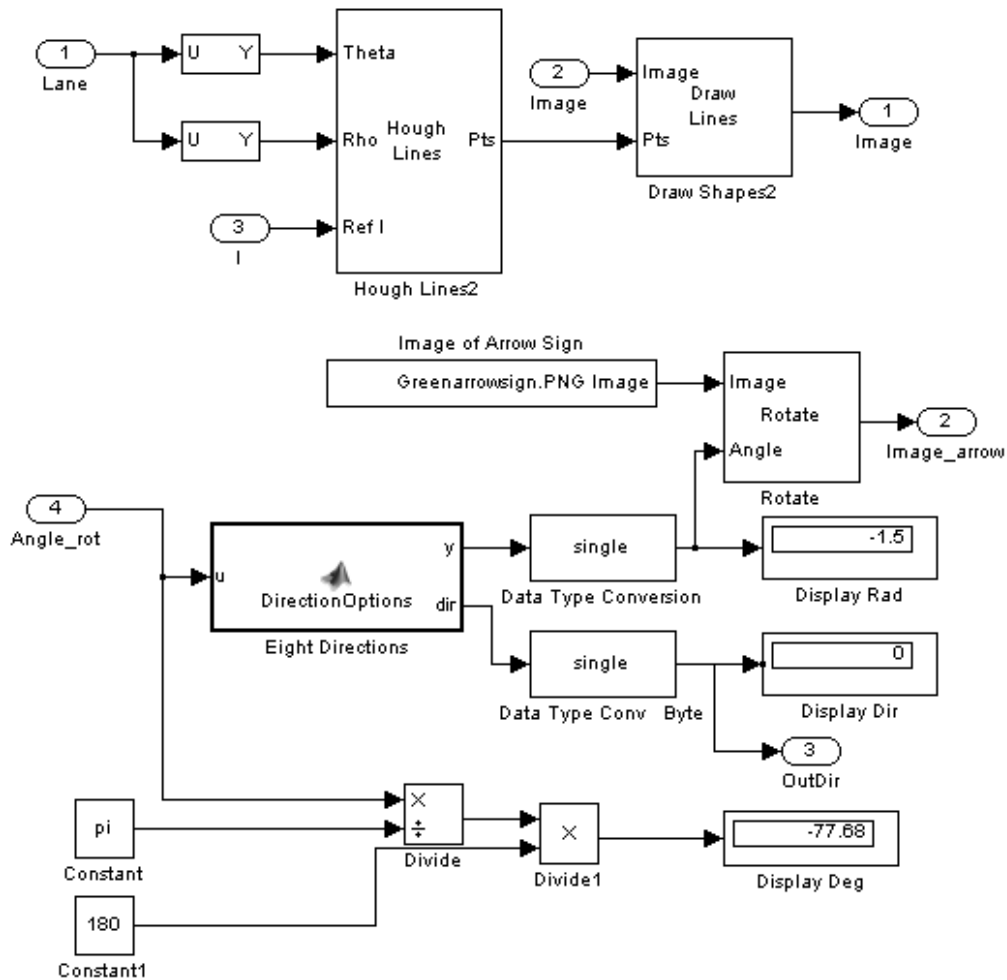


Figure 4.23: Show valid lanes and direction of movement Simulink® model

The direction the AGV must be steered in, determined from the route information detected, is represented by the *Angle_rot* signal generated in radians, signal 4 in Figure 4.23. Thus any direction (360°) of movement for the AGV was divided into eight initial directions consisting of a range of radian values. The reasoning behind the eight directions was to minimise the control commands, limiting it to forward, reverse, left, right and the directions in-between, as could be seen in Figure 4.24(b). The *Direction Options* function block detects in which of the eight directions of possible movement the *Angle_rot* falls, indicated by Figure 4.24(a). *Angle_rot* is also displayed in degrees in

Figure 4.23 to assist in the evaluation. Coded in the *Direction Options* function block STOP has precedence, but reverse was also coded for possible inclusion in a maize application with a dead-end possibility.

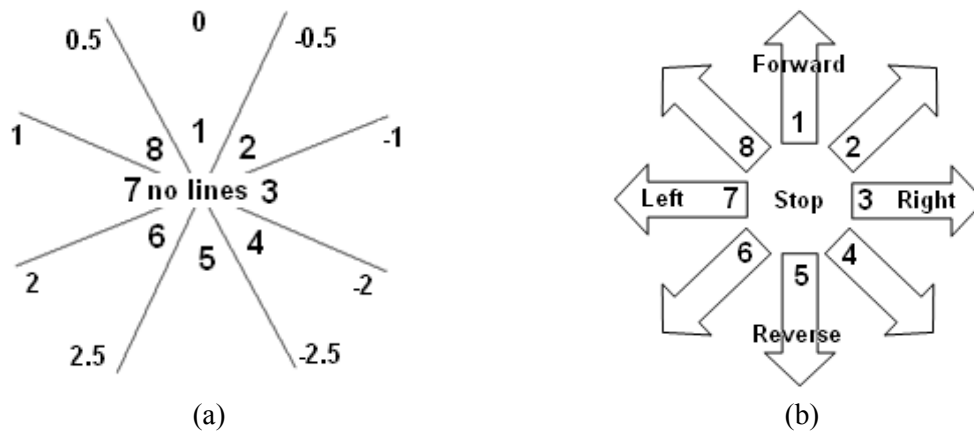


Figure 4.24: Sub division of AGV directions (a) directions indicated in radians with 8 ranges, (b) resultant direction code generated and Stop if no route is identified

4.6 PC to motor speed control interface

There was a need to communicate the associated direction commands from the images of the camera system to the AGV platform. As a possible solution, it was initially planned to use a National Instruments USB-6009 multifunction I/O, already a manufacturing standard, seen in Figure 4.25 [59]. It consists of analogue to digital converter input ports, with fourteen bit resolution. Digital to analogue converter output ports and twelve digital input/output ports with a 32-bit event counter, to name the most prominent specifications.



Figure 4.25: NI USB-6009 inside and outside its enclosure

The device was initialised in MATLAB[®] as Dev1. Port 0 and 1 was used as digital outputs and inputs respectively. Figure 4.26 shows some of the code extracted from MATLAB[®] to initialise and access the ports for evaluation purposes.

```
% initialize port0 as output and port1 as input
dio = digitalio('nidaq', 'Dev1');
addline(dio, 0:3, 0, 'Out');
addline(dio,0,1,'In');
%write logic ones to the port0 output
putvalue(dio.Line([1 2 3 4]),[1 1 1 1]);
%read the logic level on port1
value =getvalue(dio.Line(5))
```

Figure 4.26: MATLAB[®] program extract – initialisation and access of ports

This proved that MATLAB[®] could be used to write out commands via the NI USB-6009 to the Sabertooth motor speed controller. The speed control, however, is PWM controlled with the specifications seen in Figure 4.27.

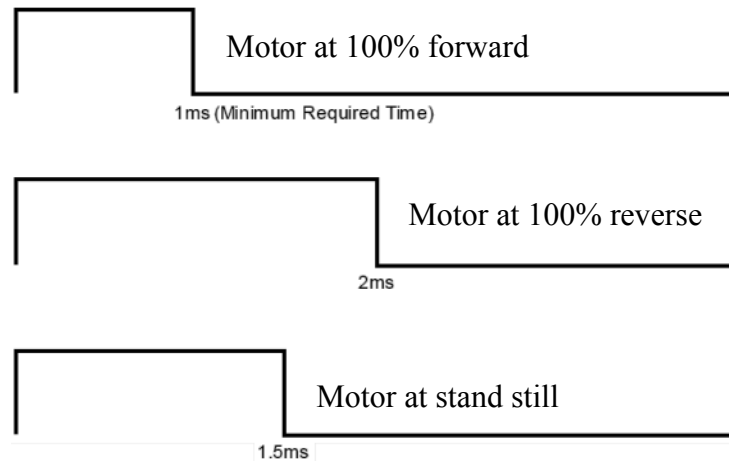


Figure 4.27: Timing diagram of R/C motor speed control utilising PWM [60, p. 3]

The fastest on/off switching to simulate the PWM, in succession, achieved by the NI USB-6009 was two milliseconds (2 ms) as could be seen in Figure 4.28. This made it impossible to use the NI USB-6009 in such an application, opting for the development of the PIC microcontroller board shown in Figure 4.4.

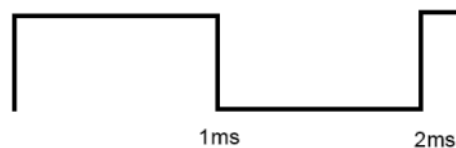


Figure 4.28: Output achieved by switching the NI USB-6009 port on and off in sequence without time delay

MATLAB[®] proved capable of communicating to NI equipment if the need do arise in such an application.

4.6.1 AGV controls generated from the visual route navigation system

The PIC microcontroller board and software was developed in such a way that the direction control signal of the AGV was sent serially via the USB port of the PC to the PIC board. Because the AGV has two drive motors, the speed and direction of the respective motors determines the AGV's speed and direction. The control command consists of two single hexadecimal values (one for each motor). These hexadecimal

values were combined as a byte value and had to be declared as double to be sent from the PC to the PIC board serially. The two command values have a hexadecimal range between 0x0 and 0xF. The range makes it possible to have a resolution of sixteen steps for the speed of the motors, from full reverse (0xF) to full forward (0x0) of each individual motor. The stop position of the motors need to be in the middle between the values 0x0 to 0xF and was selected to be 0x7.

From Figure 4.29 it is evident that, because of the construction of the AGV, the movement of the motor drives is in opposing directions.

For example; forward is the byte value 0xF0. The left motor, least significant hexadecimal, value is 0x0 (full forward). The right motor has the hexadecimal value of 0xF (full reverse) (see Figure 4.29).

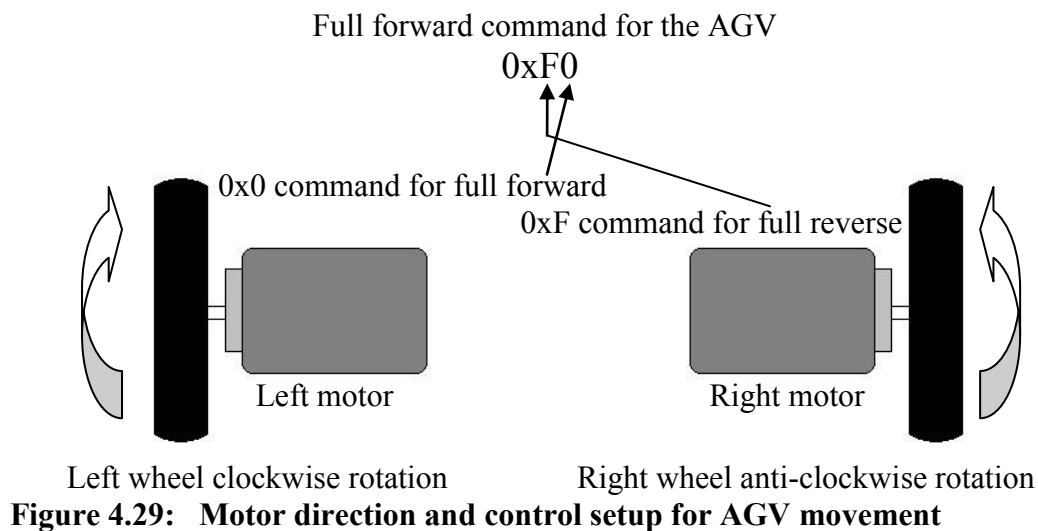


Table 4.1 reflects only the four most important directional control words of the AGV and STOP control for the motors as examples of the code to be sent to the PIC board. These control words represent the full speed equivalents in the different directions.

Table 4.1: Direction control commands of the AGV and its values sent serially on USB to the PIC board for maximum speed movement

Direction	Hexadecimal control value	Decimal control value
Forward	0xF0	240
Reverse	0x0F	15
Left	0xFF	255
Right	0x00	0
Stop	0x77	119

The *Display* Simulink® model was developed to generate the *Direction-* and *Stop-Control* to correlate with the arrow and STOP sign displayed on the *Route Tracking* display (Figure 4.9 and Figure 4.22(d)). This direction-control signal was then altered to the correct control byte by the *DirectionCntrl* function block to be sent by the USB port to the PIC motor speed controller. This Simulink® model is displayed in Figure 4.30.

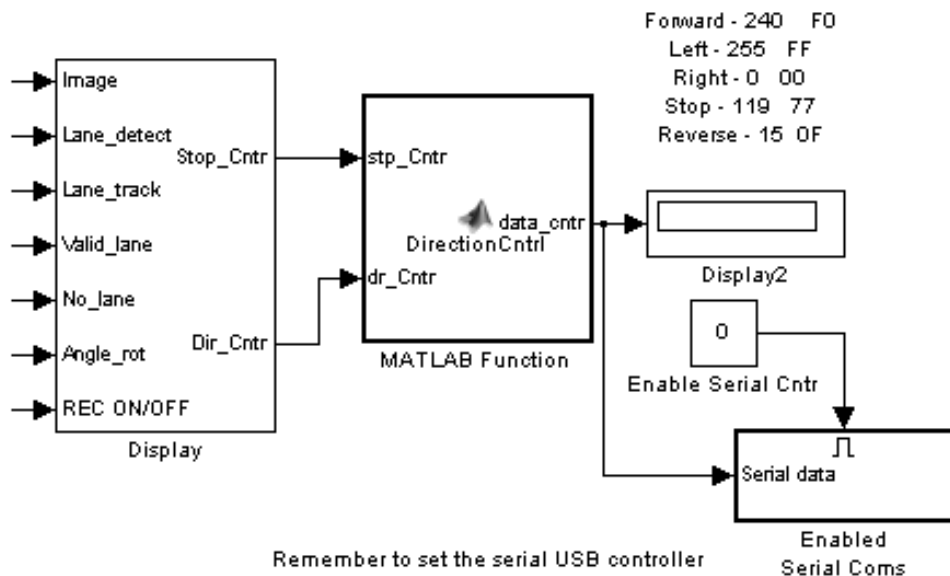


Figure 4.30: Function block receiving the direction information to be altered and sent via USB

The USB serial communications is controlled by an Enabled Subsystem block depicted in Figure 4.31.

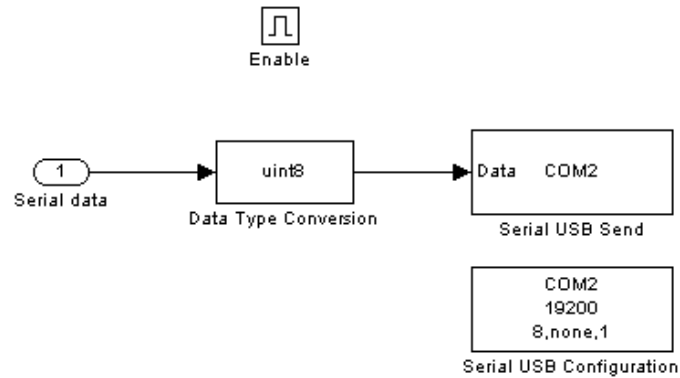


Figure 4.31: Enabled subsystem block for USB serial port communications from laptop to AGV for control commands

4.6.2 Obtained speeds for the AGVs used

By generating a full forward or reverse command the maximum speed which the 3- and 4-wheel AGV platforms can reach is calculated from the data sheet and actual specifications shown in equation (4.2). Both these platforms depicted in Figure 4.3 were used in the research and evaluation.

$$\text{Speed of AGV} = \text{motor speed (rpm)} \times \text{gear ratio} \times \text{wheel circumference} \quad (4.2)$$

Substituting the following parameters of the four- (4) wheels NI AGV platform:

motor speed (rpm) no load = 154 revolutions per minute (rpm) [61],

motor speed (rpm) actual = 140 revolutions per minute (rpm),

gear ratio 2:1 and

wheel circumference = 314.16 mm.

gives a maximum speed of the four- (4) wheeled NI AGV = 21.9 m/min which relates to 1.3 km/h.

Substituting the same parameters of the three- (3) wheeled NI AGV platform:

motor speed (rpm) no load = 154 revolutions per minute (rpm) [61],

motor speed (rpm) actual = 144 revolutions per minute (rpm),

no gear ratio and

wheel circumference = 314.16 mm.

gives a maximum speed of the three- (3) wheeled NI AGV = 45.24 m/min which relates to 2.7 km/h. This is double the speed of the 4-wheeled NI AGV platform because there is no gear ratio to the wheels.

Testing the route navigation at 1.3 km/h which is maximum speed for the 4-wheel AGV platform resulted in the vehicle tilting forward and backward. The tilting effect was because of the maximum forward and backward control thrust from the navigation system, the short wheelbase between front and rear wheels of only 127 mm and the high centre of gravity. A speed of 1.3 km/h was still very slow for the AGV and rather than implementing gradual acceleration a *ski* was implemented as seen in Figure 4.32.

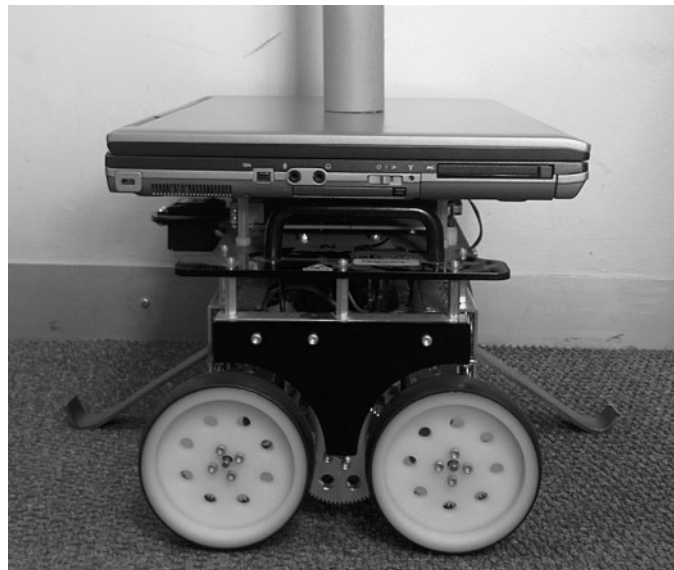


Figure 4.32: 4-wheel NI AGV platform with *ski* implemented for correcting the tilt effect of the AGV

The 3-wheel NI platform was faster and did not need such alterations, because of a lower centre of gravity and a distance of 192 mm between the front and rear wheels. Still, a gradual acceleration and deceleration would be better for reliability on the gear system.

This then concluded the complete system for route navigation from detecting a path and navigating the AGV from one point to another, utilising vision by selecting an area of interest. The research thus far did not include reconfigurability other than placing the AGV on different routes and the system could not determine which route to take when encountering a split route, like a T-junction. The selection of route direction would be random.

This resulted in the development of a system that could indicate a direction to take and a STOP command at the destination, keeping reconfigurability in mind.

4.7 Sign recognition

The concept of sign detection in conjunction with route tracking is to provide the AGV controller with an indication to which route is to be taken when encountering more than one option. This is accomplished by incorporating left and right turn signs with a stop sign at its destination. This gives the AGV a reconfigurable route, determined by the operator, without programming intervention or changes, by placing the signs along a changeable route. This is best illustrated by Figure 4.33.

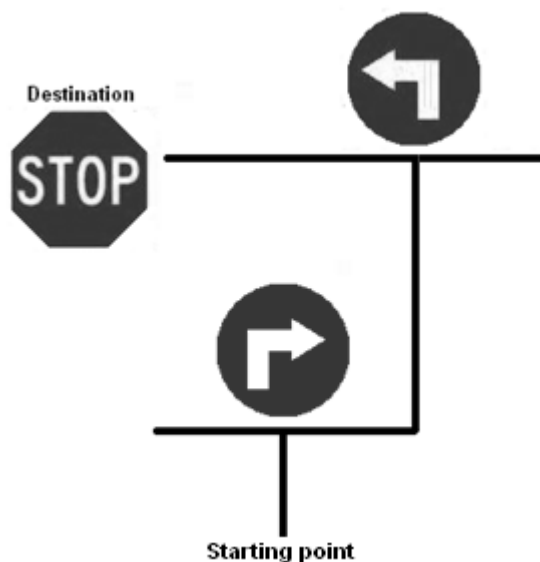


Figure 4.33: Incorporating signs for defining a reconfigurable route

The AGV starts at the starting point, encountering a T-junction where after it turns right because of the sign. It then follows the route irrespective of the turns and at the second junction needs to turn left, because of the sign command, before reaching its destination where it is stopped.

The “Traffic Warning Sign Recognition” MATLAB[®] demo served as starting point of this development, depicted in Figure 4.34 [62]. This model was altered to suit the research objective.

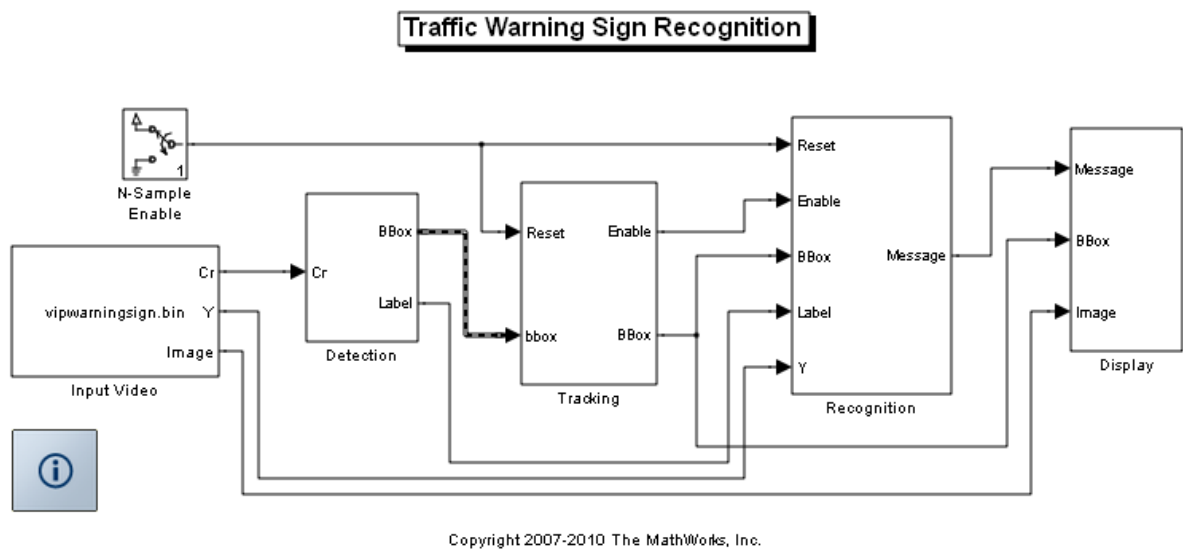


Figure 4.34: Traffic Warning Sign Recognition MATLAB[®] demo Simulink[®] model block

4.7.1 Sign recognition templates

As discussed in section 2.5.3 an option for correlation is to use a template. Two sets of templates were generated, one for detection and the other for recognition of the signs. The detection templates were generated in a low resolution (12 × 12 per sign) with the advantage of saving on computational resources (see Figure 4.35). Only one template was generated per sign to be utilised in the detection process. As a STOP sign is mostly red this red colour pixel was mainly used in the detection process.



Figure 4.35: Three signs template, generated for the detection process, STOP, left and right turn

The recognition process required a higher resolution (18 x 18 per sign) for the signs, with the implementation of alternative orientations for each. Orientations of the signs of plus and minus 7.5 degrees were selected for this purpose originally. The sign information, like the STOP, left and right arrow was generated as white pixels for the recognition process, seen in Figure 4.36.



Figure 4.36: Three signs templates – STOP, left and right; with three orientations each, 0° + 7.5° and -7.5° ; generated for recognition process

These templates were generated by the altered “vipwarningsigns_templates.m” file to read the sign input pictures in Portable Network Graphics (.png) format [63]. The resultant templates from the function were stored in the “vipwarningsigns_templates.mat” file. This .mat file in turn was loaded, as *Workspace* variables for the simulation, by the Simulink[®] model block by defining the file in the *Model initialisation function* under the *Model Properties* below the *Model callbacks* heading.

Using one size of templates in such a setup makes the recognition result distance depended between the AGV and sign. This is discussed in more detail in section 4.7.6 and the results section 5.4.

4.7.2 Detection of signs

The *Detection* block previously indicated in Figure 4.34 as part the whole system analysis each video frame in the YCbCr colour space, as can be seen in seen in Figure 4.37 where the red signal (Cr) was isolated and used.

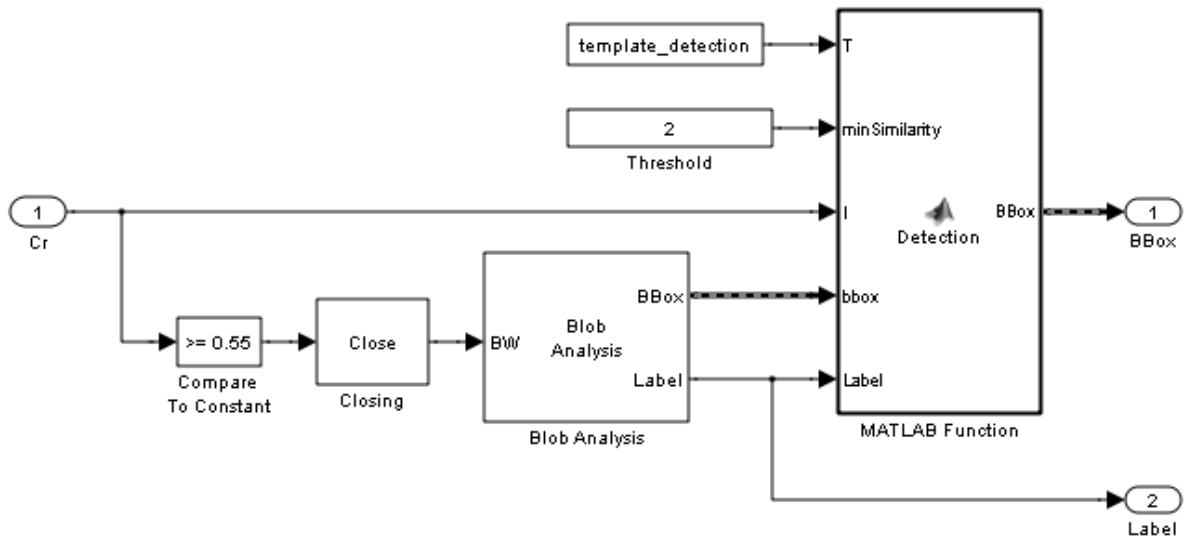


Figure 4.37: Detection block of the sign recognition demo used in evaluating command sign detection in the Cr colour space

Thresholding the Cr-channel (input1, named Cr) and performing morphological operations on the same signal, imply that the model concentrates on the predominantly red pixels of a picture (the sign colour). The morphological operation *Closing* performs a dilation operation followed by an erosion operation using a predefined neighbourhood of pixels. Utilising the *Blob Analysis* block, the model finds the red pixels and bounding box for each blob, as this block returns the centroid of a large section in a binary image with a count of these red pixel blob occurrences. The model then compares the blob with each sign detection template. If a blob is similar to any of the sign detection templates, it is a potential command sign.

4.7.3 Tracking and recognising the signs

The model compares the bounding boxes around the red blobs of the potential command signs detected in the current video frame with those in the previous frame. The model compares this blob with the sign recognition templates only if a potential sign is detected in a set number (default is 4) of consecutive video frames. If the potential command sign is similar enough to a command recognition template in a set number of consecutive frames (default is 3), the model considers the potential command sign to be an authentic sign.

After successful sign recognition the models continue to track the sign in the frame without returning to the recognition sequence to save on computation resources.

4.7.4 Displaying the recognition results

After a potential sign has been detected in four (a set value) or more consecutive video frames, the model draw a yellow rectangle around the particular sign, utilising the *Draw Shape* block. On recognising a sign, the *Insert Text* block is used to write the name of the sign on the video frame for evaluation purposes. The term *Tag* is used to indicate the order in which the signs are detected with a maximum set at nine before restarting the count.



Figure 4.38: Displayed recognised left, right and stop signs

At this stage the AGV could detect these three signs along the set route for the specific command execution. A reconfigurable route could be set out by the operator without

programming changes to the AGV. This did not solve the possibility of more AGVs using the same route with different destinations each.

4.7.5 Investigating different coloured routes

Altering the colour which the AGV respond to, gave rise to alternative routes for different AGVs to follow as Figure 4.39 depicts.

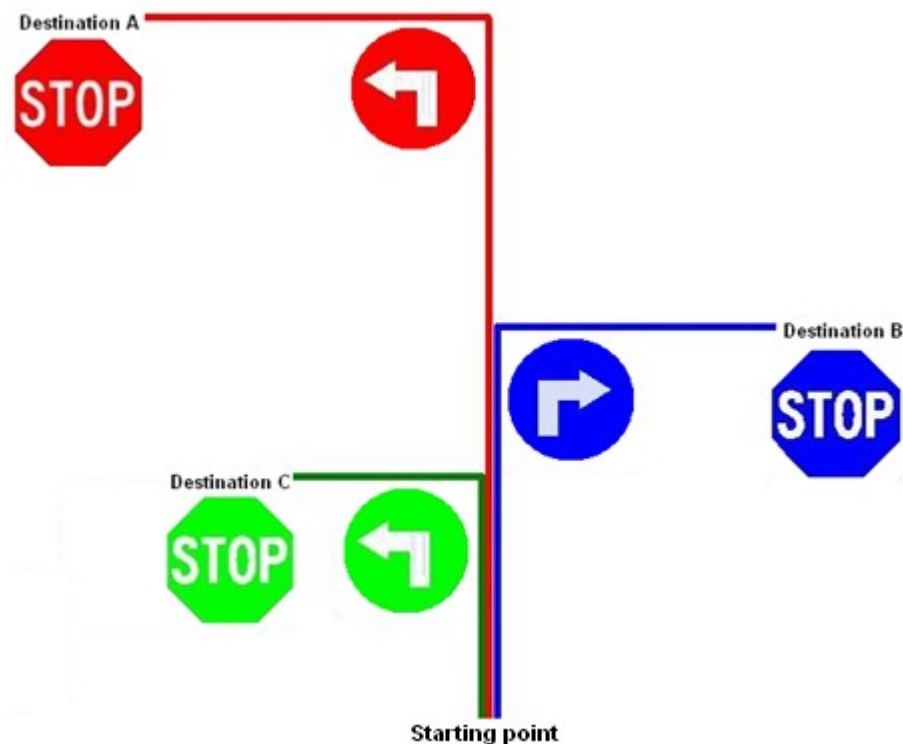


Figure 4.39: Implementing different routes for multiple AGVs by utilising different colours

Detecting a blob of red pixels (using the Cr signal) or blue pixels (using the Cb signal) did not pose a problem as it is available in the YCbCr signal. A green signal, or any other colour, had to be sourced and selected differently. For this reason, different methods were investigated.

Method 1: The RGB video signal was used in selecting the specific colour depicting a certain AGV's route. Choosing the separate R, G and B values and implementing a tolerance for each colour signal representing this selected route colour.

Using a colour tablet shown in Figure 4.40, a specific green colour was selected for a specific route as indicated on the chart. The RGB values obtained are; Red (79), Green (183) and Blue (53). These specific RGB values were then used for the Simulink[®] *MATLAB Function* block developed, of which part of the code is shown in Figure 4.41. The resultant outcome for the selected colour and generating a Boolean picture is shown in Figure 4.42.

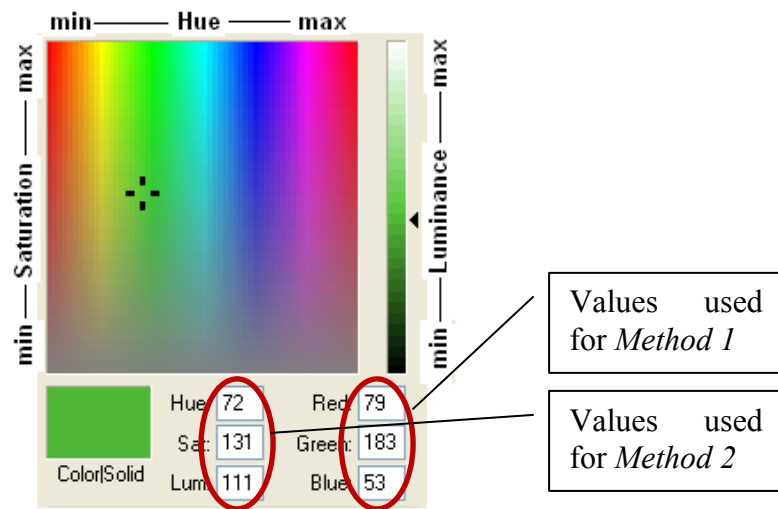


Figure 4.40: Windows Paint edit colours tablet for HSV and RGB pixel colour signal [64]

```
function Io = fcn(Ii)
%RGB selection
%Get size of video frame
y = single(size(Ii, 1));
x = single(size(Ii, 2));

%Create output frame
Io = Ii(1:y,1:x,1);

for col = 1:1:y           %height of frame
    for row = 1:1:x       %width of frame
        if (Ii(col,row,1) >= 70/255) && (Ii(col,row,1) <= 85/255)&&...
            (Ii(col,row,2) >= 175/255) && (Ii(col,row,2) <= 190/255)&&...
            (Ii(col,row,3) >= 50/255) && (Ii(col,row,3) <= 60/255)
            %values of RGB is fraction ex 0.45
            Io(col,row,1) = 1;           %binary value
        else
            Io(col,row,1) = 0;
        end
    end
end
end
```

Figure 4.41: MATLAB[®] function block extract for selecting a certain green colour selected for route identification

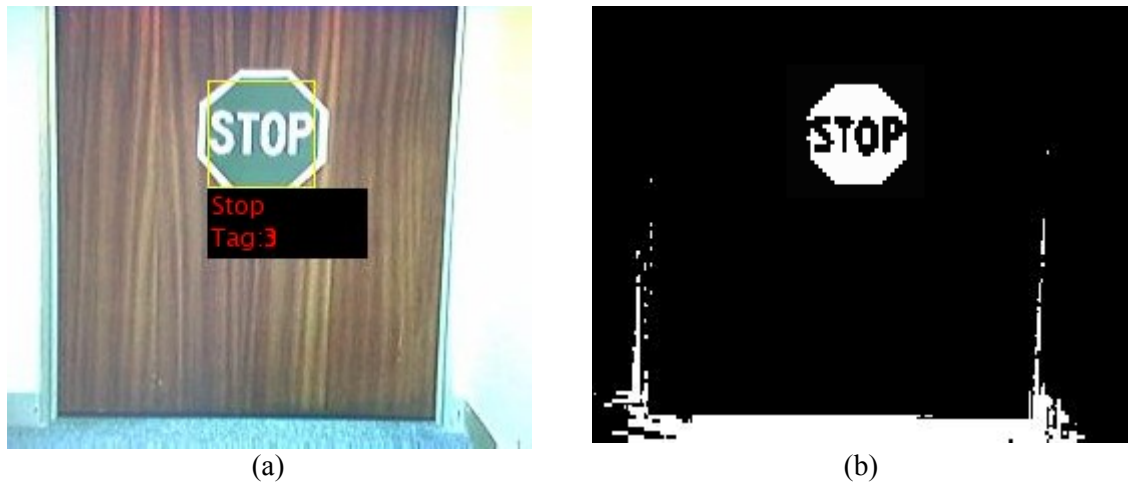


Figure 4.42: (a) Recognised result of a green STOP sign; (b) Boolean picture generated as a result of the Simulink[®] MATLAB function block

Method 2: Method 1 was evaluated utilising the HSV (72, 131 and 111) signal rather than the RGB signal. This gave similar results to that of Method 1 and was still time consuming.

Method 3: This method was implemented for trying to save time on computing using only the single hue (H) value representing a certain colour range (72) excluding the saturation and intensity signal.

All three methods used did function, but not satisfactorily. Lighting played a big role as the luminance changed significantly in the colour selected. The luminance value altered the shade of the colour to a very large extent, representing at maximum the colour white and minimum the colour black.

The unsatisfactory results obtained for selecting a specific colour from the RGB and HSV signal, resulted in a re-evaluation of the YCbCr signal as this produced better results on using colour signals red (Cr) and blue (Cb). Utilising equations (2.12), (2.13) and (2.14) and substituting typical constants for the Y signal, equation (4.3) was derived. This equation was implemented with the Simulink[®] model shown in Figure 4.43.

$$C_g = -1.5R + 2G - 0.54B + 0.5 \quad (4.3)$$

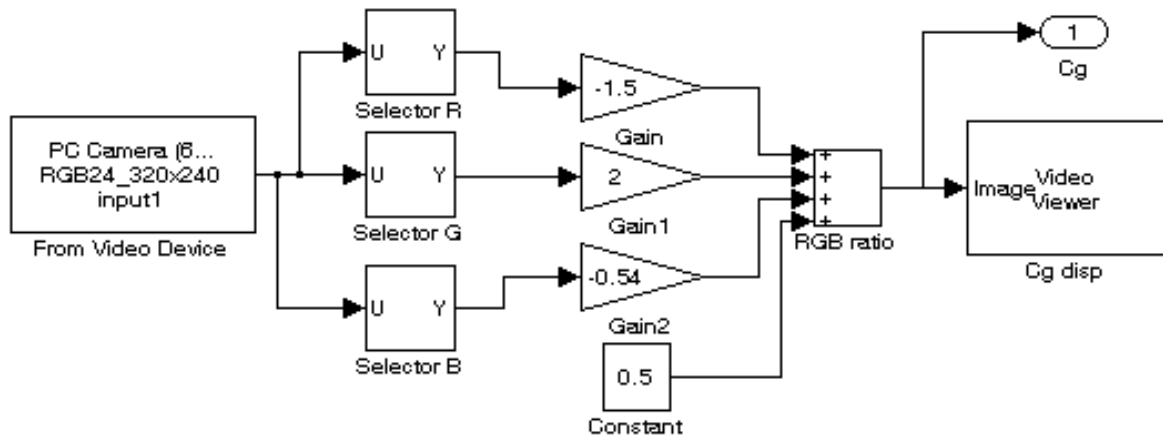


Figure 4.43: MATLAB[®] implementation of the Simulink[®] model evaluated for a set green signal

This method proved experimentally the most successful as the colour selected made the output signal less sensitive to variations in lighting levels. In a similar way any other colour could be specified as a weighted RGB signal rather than the red, green and blue value at that specific colour selection.

4.7.6 Implementing sign detection command control

Detecting the command signs successfully posed a problem with respect to the reaction time to execute the relevant command. This made a difference in the distance from the sign to the specific position of the AGV. For example; when the AGV detects a STOP sign, how long does it take till the STOP command is applied?

The size of all the signs were standardised to be approximately 18 cm by 18 cm. Knowing the sign size the distance from the AGV to the sign could be calculated utilising the number of pixels representing the image size recognised [65, pp. 324-329].

Table 4.2 gives a summary of the distance relevant to pixel count, obtained experimentally utilising the webcam to be used.

Table 4.2: Summary of distance from AGV to signs with respect to image pixel count

Distance to a sign	Approximate pixel count
40 cm	174 X 174
50 cm	144 X 144
60 cm	120 X 120
70 cm	106 X 106
80 cm	96 X 96
90 cm	84 X 84
100 cm	76 X 76
110 cm	66 X 66

A safe distance from the AGV to a sign or obstruction was selected to be between 70 cm and 90 cm. The reason was that this represented the width of most of the routes used in the simulation and evaluation process, making it possible for the AGV to turn within this distance. This resulted in the choice of image size, representing the stochastic distance to a sign selected and evaluated of approximately 84 x 84 pixels (total of 7 056 pixels), viewed from any direction. The distance selection to the sign was developed to be a variable input in the Simulink[®] model.

Determining this distance to the sign was achieved by utilising the area of the bounding box placed around the sign detected and then comparing this pixel count with the required size (total pixel count). When true, the relevant sign command detected was executed. Provision was made for a multiple count of signs detected in a single frame during consecutive frames. This value for the number of signs detected was set to a single digit count of nine during evaluation but could be altered by the variable “maxNumSigns” as part of the *Blob Analysis* function. It was accomplished by placing the particular code in a *MATLAB Function* block within the *Recognition* block, located in the complete system block diagram previously featured in Figure 4.34.

The route tracking controller (*Direction Cntrl*) has two inputs controlling the motor speed interface, i.e. STOP control and direction control (speed in a certain direction), already shown in Figure 4.30. These two signals are also generated as outputs by the developed sign controller function block. They are switched into operation by a switch control acting

as multiplexer with a binary input selector on a valid sign input at the correct chosen distance. Figure 4.44 shows the implemented Simulink[®] model block where the area of the detected sign (*Prod*) and the variable distance (*Dist*) are needed as input with the STOP, direction and switch control generated as output. This Simulink[®] model block also makes provision for the STOP and direction control signals as input coming from the *Display* block as well as the outputs going to the direction control block in Figure 4.30.

In this example, depicted by Figure 4.44, the STOP sign was placed the closest and then the left turn followed by the right turn. The correct signs were detected and only the STOP command was executed as could be seen in the figure. The order of detection could be judged by the tag numbers with the STOP sign first, then the left, followed by the right turn.

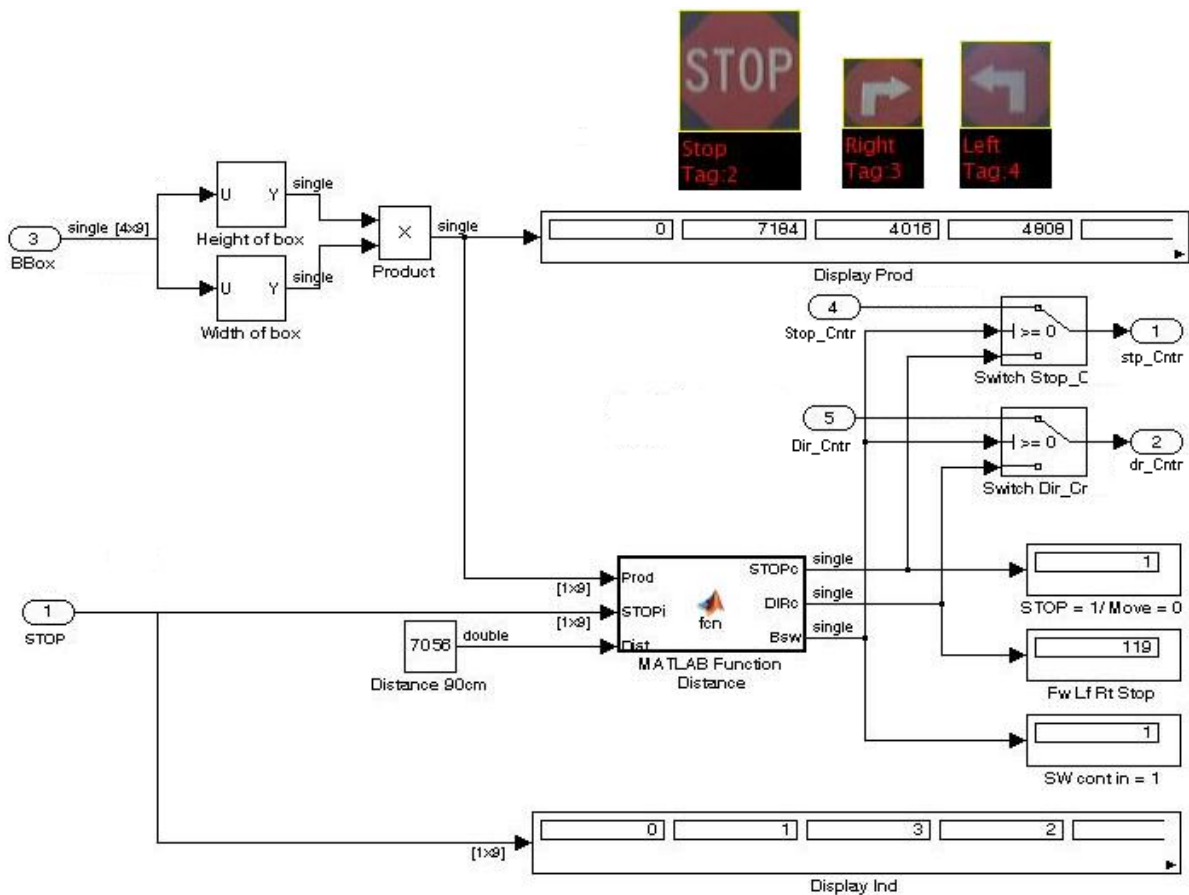


Figure 4.44: Simulink[®] model implementing AGV motor control at a set distance depending on the area of pixels

Figure 4.45 indicates an abbreviated version of the MATLAB[®] function block code generating the control and switching signals. Only the forward, left, right and STOP signals are shown for illustration purposes.

```
function [STOPc, DIRc, Bsw] = fcn(Prod, STOPi, Dist)
%STOPc - STOP(1) no lane control output
%DIRc - Forward, Left and Right Direction Control
%Bsw - Boolean switch
%Prod - Area of specific BBox, to be compared to distance value
%STOPi - STOP(1), Left(2) and Right(3) Direction control input
%Dist - distance setting
STOPc = single(0);           %set movement control to default
DIRc = single(240);         %default forward
Bsw = single(0);
Tag = single(length(Prod)); % = amount of tags;
if (Tag > 0)                 % depending on variable maxNumSigns
    for ind = 1 : Tag
        if (Prod(ind)>Dist)
            if (STOPi(ind) == 1)           %STOP
                STOPc = single(1);         %STOP control
                DIRc = single(119);        %no direction - STOP
                Bsw = single(1);           %switch to control output
            end
            if (STOPi(ind) == 2)           %Left
                STOPc = single(0);         %STOP control - moving
                DIRc = single(255);        %direction control LEFT
                Bsw = single(1);           %switch to control output
            end
            if (STOPi(ind) == 3)           %Right
                STOPc = single(0);         %STOP control - moving
                DIRc = single(0);          %direction control RIGHT
                Bsw = single(1);           %switch to control output
            end
        end
    end
end
end
```

Figure 4.45: Abbreviated MATLAB[®] code for the distance function block generating STOP, direction and switch control

4.8 Conclusion

The navigational goals, utilising vision, as described in this chapter were successfully met by the developed AGV platform and the route navigation with the sign recognition and control implemented.

A reconfigurable layout could be achieved with relative success utilising an AGV recognising only a set colour for its specific route.

This method of navigation and control improved the flexibility over dead reckoning navigation with an added saving in programming alteration time and complexity.

Results stated in this chapter only point out the alterations and direction took in the research. Detailed results are discussed in the next chapter.

Chapter 5

Results

In this chapter the results of the study for the omnivision system, AGV platform, navigation- and control system and the reconfigurability of the AGV is described.

5.1 Omnivision system results

In this section, the performance of the omnivision system is evaluated through several experimental results on different processor platforms and cameras using both a three- and four wheels NI AGV platform.

The platforms being used are shown in Table 5.1: the PC for the original MATLAB[®] development and the Laptop for deployment on the NI platforms.

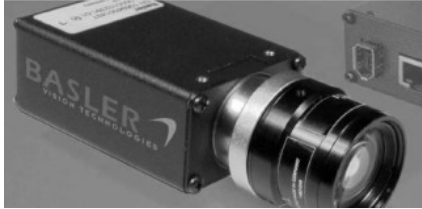


Table 5.1: Processor platforms specifications used in evaluations

Personal computer (PC)	Laptop
Microsoft Windows XP	Microsoft Windows XP
Professional Version 2002 with Service Pack 3	Professional Version 2002 with Service Pack 3
Intel [®] Core [™] Duo CPU E8400 @ 3.00GHz	Intel [®] Core [™] Duo CPU T7500 @ 2.20GHz
2.98 GHz, 1.99 GB of RAM	789 MHz, 1.99 GB of RAM

Looking at the specifications of the PC and laptop used; the speed of the processor and that of the RAM were the major factors that caused the difference in processing power. The architecture of the two machines also had an influence on the final results, as could be seen in the previous mentioned benchmark test for the PC Figure 3.24 and that of the laptop Figure 3.26.

The cameras used are shown in Table 5.2: the Basler was later omitted because of its physical size and the optical focal length. Hence the Webcams were used on the 3- and 4-wheeled NI platforms.

Table 5.2: Cameras used in research with applicable software specifications

	BASLER A600f	Logitech® QuickCam® Pro 4000	Connix Webcam 6009CIF
			
Software	BCAM Viewer BCAM Version 1.9.0020 ©Basler AG	Logitech® QuickCam® Software Version 10.5.1.2029 ©1996-2007	PRO-Q PC-Camera
Sensor type	Micron MT9V403 - 1/2 inch, CMOS, Global Shutter	CCD	CMOS
Number of pixels	656(H) x 490(V)	640(H) x 480(V)	640(H) x 480(V)
I/O interface	IEEE 1394	USB	USB

The research already showed that the processing speed on a PC is more acceptable than on a laptop (section 3.5), thus the reason for incorporating processing time saving measures. One of them was the use of a self designed and manufactured mirror rather than the round shaped mirror used in Swanepoel's research [7, pp. 57-58]. This caused a saving on necessary image corrections as stated in section 3.2.1. The conversion function (seen in Figure 3.10 – the actual function, not only an extract) could be shortened leaving out the section of the code addressing the deformation, saving on executing program line code.

The focus distance of the BASLER with lens setup was much longer than the webcams in a similar setup to that shown in Figure 3.2 and the BASLER setup also used larger diameter mirrors. The use of the BASLER camera was, however, discontinued as the move to a webcam with lower resolution and the implementation of the area of interest made it redundant and too large a setup to be used on the AGV platform. Another reason not opting for the BASLER setup was the high centre of gravity with respect to the smaller AGV.

Figure 5.1 shows the mirror and the Perspex® tubing used for the different webcams as the cameras' focal distances differed. It was constructed and connected on top of the camera as displayed in Figure 3.15 and on the AGV in Figure 5.2.



Figure 5.1: Mirror and different Perspex® tubing lengths for webcam setup

Figure 5.2 shows the omnidirectional webcam setup used in obtaining the deformation results of 10 cm x 10 cm squares reflected in Figure 5.3.

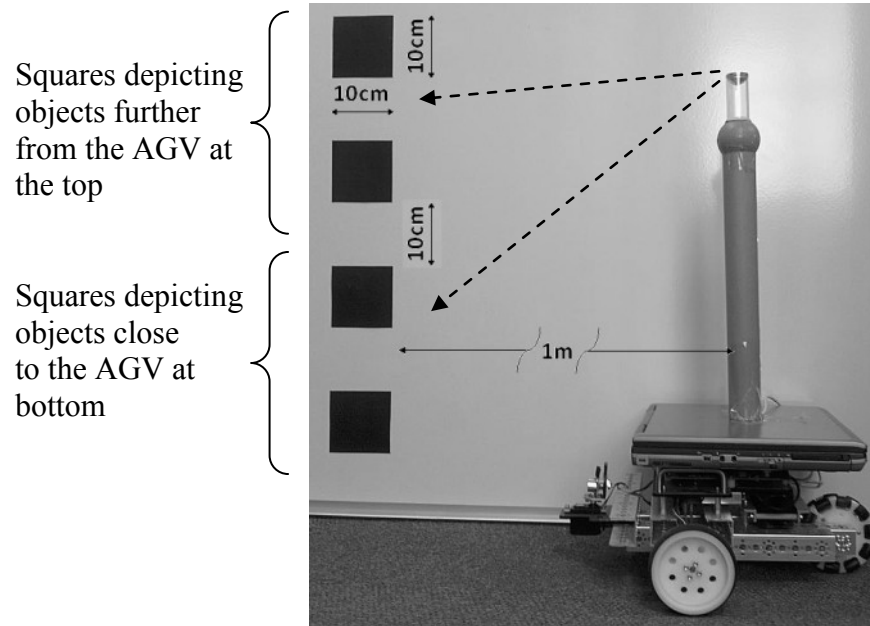


Figure 5.2: Test pattern used with the measurement setup for obtaining the results in Figure 5.3

Figure 5.3 and Table 5.3 shows the deformation results obtained (discussed in section 3.2.1, shown by Figure 3.14 and Figure 3.15 – difference in deformation for different shaped mirrors) as an indication of the difference between the round shaped mirror (Figure 5.3(a)) and the mirror used in the research (Figure 5.3(b)). The pictures were taken by the same camera and resolution with similar lighting conditions.

Table 5.3: Vertical sizes (heights) of the four different squares shown in Figure 5.2, depicted in pixels by the results obtained from semi-spherical and hyperbolic shaped mirrors respectively

	Spherical mirror	Hyperbolic mirror
Top Square	30	38
Square second from the top	36	39
Square third from the top	42	42
Bottom square	42	42

Obviously the sizes rendered by the hyperbolic mirror are more accurate and less dependent on its vertical position relative to the AGV than the same for the semi-spherical mirror.

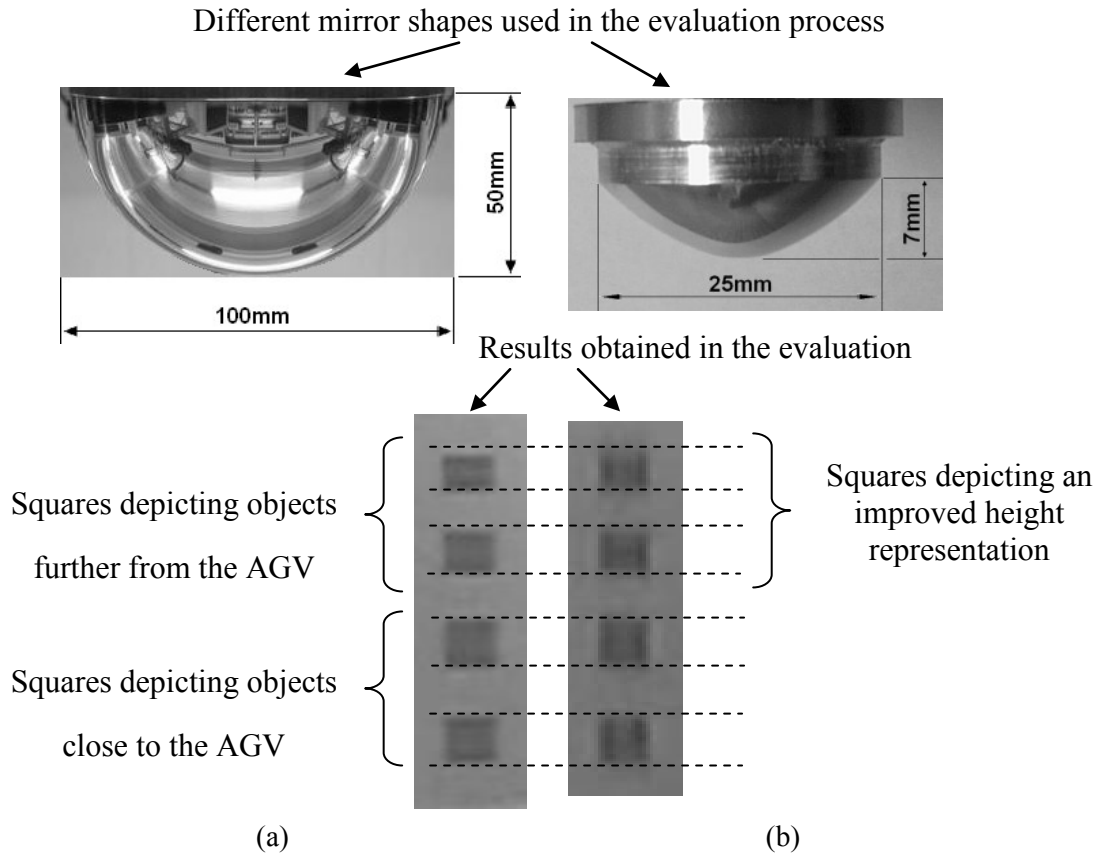


Figure 5.3: (a) Spherical shaped mirror used in Swanepoel’s research with results obtained; (b) Developed mirror used in research with results obtained

Figure 5.3(b) shows an improvement over the results obtained in Figure 5.3(a) with respect to the ratio in shape and size of the squares representing objects close to the vision system (bottom part of figure) to objects further from the vision system (top part of figure) of the AGV. The deformation of the shape at the top of Figure 5.3(a) is more (with a decrease in height) than that of Figure 5.3(b). These ratios reflect a square object without image corrections or calibration.

The smaller shape of the mirror used in the research had a darker appearance of the picture (as seen in Figure 5.3(b)) as expected because of the smaller diameter of the mirror. This issue was discussed, that the level of illumination drops as the diameter decreases, in section 3.2.1.

A ghost reflection is also evident because of the acrylic properties as can be seen in the differences between Figure 5.3(a) and (b). Figure 5.4 shows this pattern of refraction (Snell’s law [66]) and reflection (Schön [67]), which have an influence on the apparent

focus of the image. This ghost reflection however did not make a difference as the edge could still be detected with a specific threshold level for the edge detection.

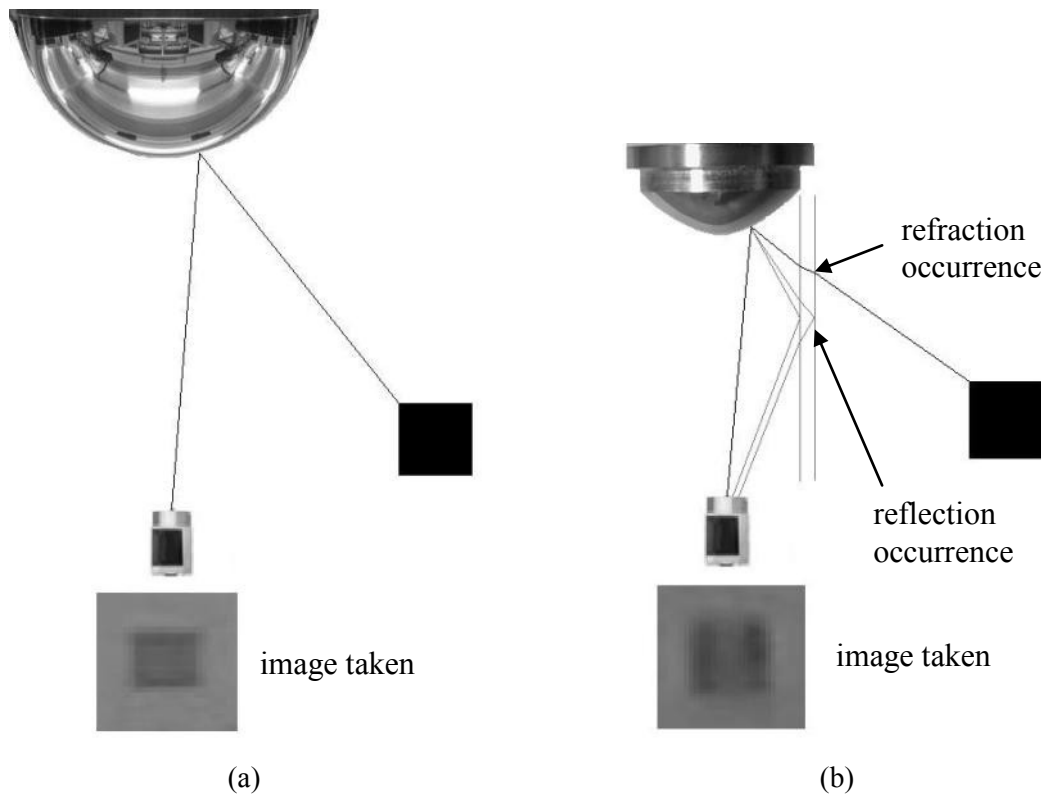


Figure 5.4: Refraction and reflection influences on the image, (a) round mirror with no reflection and refraction; (b) used mirror with acrylic setup with the reflection and refraction

Utilising Table 3.2, selecting a frame size of 201×360 , relates to only 4 frames per second available for image processing (i.e. navigation and control) after omnidirectional conversion compared to 23 frames per second selecting a frame size of 96×128 . A pixel count of approximately 6 times less resulted in an improvement of approximately 6 times more frames per second available for image processing, without the image acquisition time taken into consideration. It is evident that, changing from a high resolution picture frame (high resolution camera) to a lower resolution picture (low resolution webcam) saved considerable processing time as the function for converting a single frame is almost directly related to the image size.

Selecting the area of interest also assisted with the edge detecting process, selecting the lines to be sent through the Kalman Filter for route navigation. The area of interest included the straightest lines section of the converted frame indicated in Figure 5.5(a), making image corrections and implementing calibration unnecessary, saving valuable processing time.

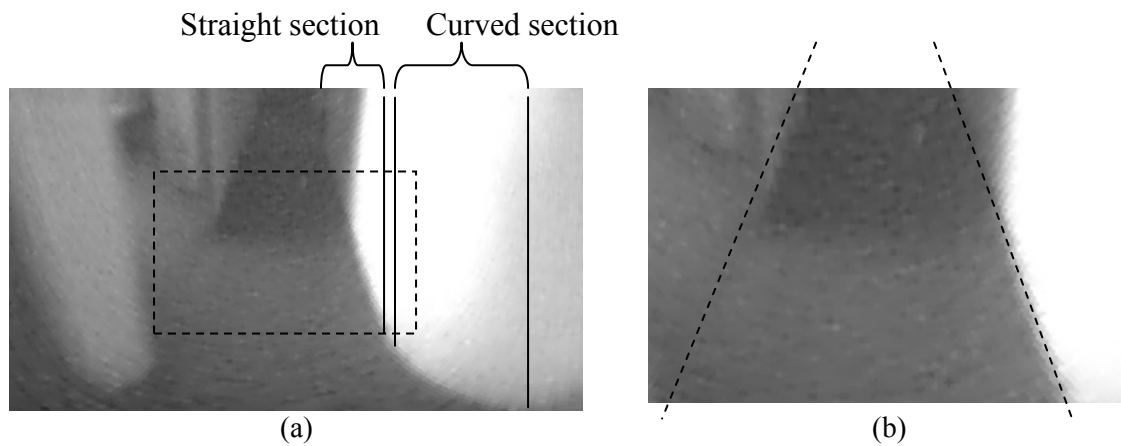


Figure 5.5: (a) Selected frame in front of AGV; (b) Resulted straight edge by selecting area of interest

In Figure 5.5(b) it is evident that both left and right route edges are more linear than those in Figure 5.5(a) making image corrections unnecessary. The area selected and shown in Figure 5.5(b) is still covering enough area in front of the AGV making navigation possible and not missing obstructions close to the AGV platform.

As the software target platform used in the research was a specific laptop, the frames per second achieved by the system were evaluated on the laptop. Figure 5.6 shows the drastic decrease in frames per second available to work with in image processing after each stage on the system, including that obtained by the PC for comparison.

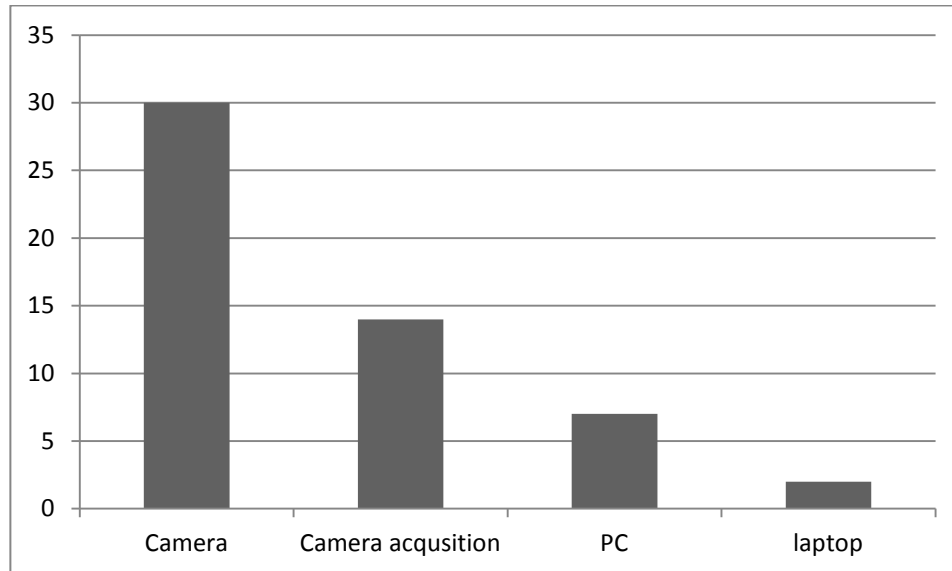


Figure 5.6: Decrease in frames per second along the process of image processing on the laptop platform relative to that of a PC

The frame rate of 30 frames per second available from the camera is decreased to almost 14 frames available after acquisition with a selected frame size of 96×128 . This frame rate is further decreased to 2 frames per second after omni conversion process available on the laptop and 7 frames per second on the PC for image processing. The 2 frames per second was too slow to evaluate the navigation properly thus the reason for selecting a camera facing the front (simulating the area of interest) for the evaluation of the navigational system.

5.2 AGV platform results

This section gives an overview of the AGV platforms' evaluated performances in terms of their speed and physical attributes through several experimental results on the two different NI platforms used, indicated in Figure 4.3 (3 and 4 wheel NI AGVs).

In section 4.6.2 it was noted that the maximum speeds the individual AGVs achieved were 2.7 and 1.3 kilometres per hour without utilising any vision, depicted in Table 5.4. The maximum frame rates achieved by using the omnidirectional vision in the study were 7 frames per second for the PC and 2 frames per second for the laptop, seen in Figure 5.6. Using this information relates to a distance travelled of 36.5 cm per second with the

slowest AGV of the two. This distance travelled by the AGV between sequential images is calculated as 18 cm using the laptop control - which performs at 2 frames per second.

Table 5.4: Comparative speeds of the 3- and 4-wheeled AGVs used in the research without vision

	3-Wheel AGV	4-Wheel AGV
Maximum speed obtained in forward/reverse without vision	2.7 km/h	1.3 km/h
Individual speeds denoted in meter per minute	45.24 m/min	21.9 m/min
Individual speeds denoted in centimetre per second	75.4 cm/sec	36.5 cm/sec

This speed of 18cm per frame was clearly too fast to allow image processing using the laptop. The AGV's speed needed to be reduced because, at least 6 to 8 frames per second was necessary for proper vision control (section 4.5.4). This meant that the AGV travelled more than a meter at 6 frames per second (18 cm x 6 frames = 108 cm). This is more than a typical turning circle distance (90 cm) allowed in section 4.7.6, before a control decision could be made. Altering the AGV's speed to suit the processing time of the laptop related to a speed of 12 cm per second (derived from equation (5.1)), which was not suited for the final industry application.

$$\text{Speed of AGV utilising omnivision} = \frac{\text{maximum speed of 4 wheel AGV}}{\text{frames needed for image processing}} \quad (5.1)$$

$$\begin{aligned} \therefore \text{Speed of AGV utilising omni vision on laptop (2 frames/sec)} \\ &= \frac{36 \text{ cm/sec for the 4 wheel AGV}}{6 \text{ frames/sec frames needed for image processing}} \\ &= 6 \text{ cm/frame at 2 frames/sec} \\ &= 12 \text{ cm/sec} \end{aligned}$$

This gave reason for not using the omnidirectional vision in conjunction with the vision control and rather opting for a single camera and a low resolution area of interest for the rest of the evaluations.

5.3 Navigation and control system

In this section the performance of the route navigational system of the AGV was evaluated, in terms of following the route as expected and using vision with a single low resolution webcam [57].

As there was no provision made for localisation of the AGV by means of dead reckoning as in Swanepoel's [7, pp. 41-44] research or utilising laser scanners and visual odometry as in Scaramuzza et al.'s [68] work, the movement of the AGV needed to be monitored and noted by observation during assessment.

The vision recording feature of the surroundings of the AGV mentioned in section 4.5.5 was used in recording the live streaming of the AGV as it progressed on its route during evaluation. These results were compared and noted with respect to the orientation of the AGV and its position on the route. What was evident was that the AGV could follow the set route with ease and that the commands generated from the navigation system did give the desired output to the AGV drive controls.

Figure 5.7 shows some of the results plotted and the position and orientation of the AGV noted for a specific evaluation performed.

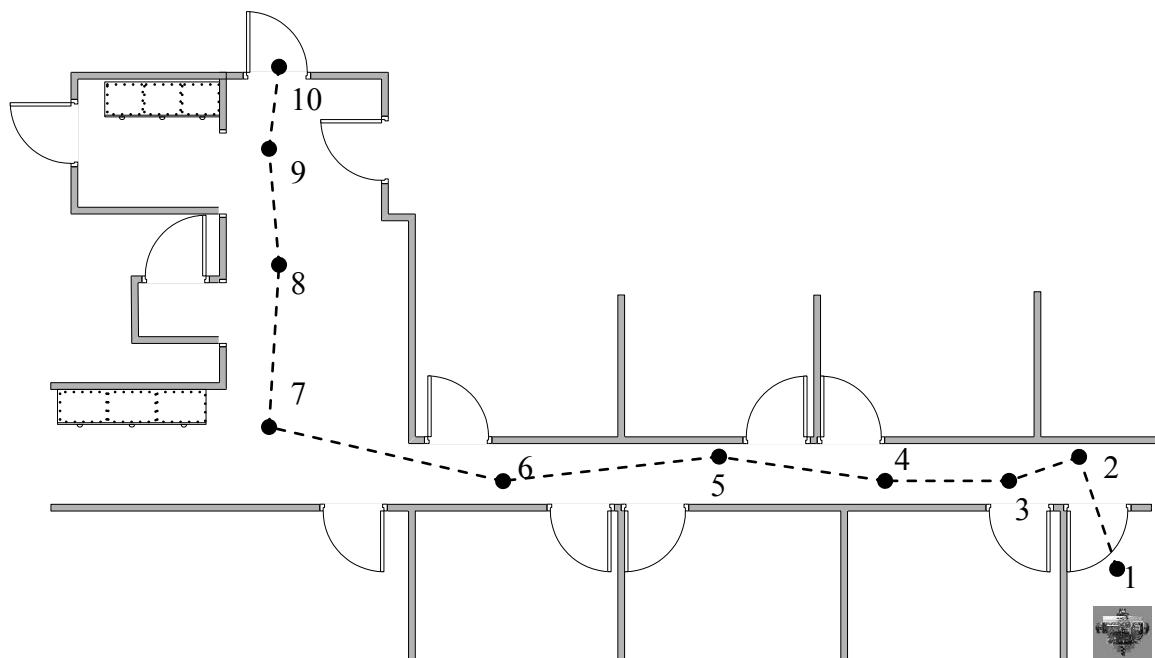


Figure 5.7: AGV position and orientation along a destined route plotted for evaluation

Figure 5.8 indicates the corresponding arrow direction control indication for monitoring purposes and AGV movement control. Point 7 in both Figure 5.7 and Figure 5.8 indicates an unexpected movement toward (expected to continue on a straight route) the right hand side of the corridor because of the chrominance edge picked up. The unexpected movement of the AGV was also part of the reason for implementing the reconfigurable control signs discussed in section 5.4.

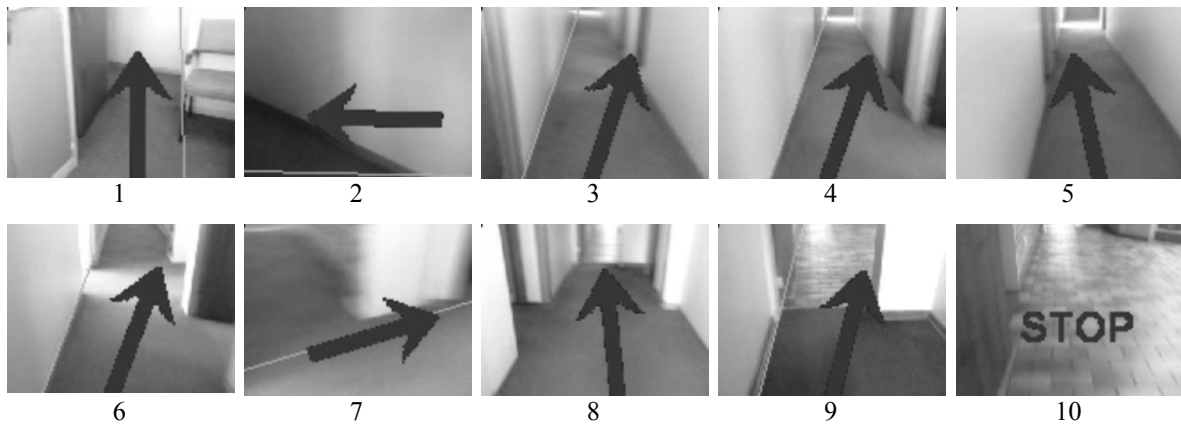


Figure 5.8: Corresponding frame captures for the positions indicated in Figure 5.7

Table 5.5 indicated the actual movement of the AGV with respect to the frame captures in Figure 5.8. The generation of the control commands and how it is created is discussed in section 4.5.5 with reference to Figure 4.24 indicating in which direction limits the route detection falls.

Table 5.5: Corresponding movement noted for evaluated test run of AGV with respect to the corresponding frames in Figure 5.8

Frame No.	Description of AGV movement
1	AGV moving straight from its starting position.
2	AGV turning 90° left because of corridor wall in front.
3	AGV turning slightly right too close to wall.
4	Another movement slightly right.
5	AGV turning slightly left.
6	A slightly right movement.
7	Unexpected sharp right movement of AGV because of chrominance border.
8	Slightly left movement.
9	Slightly right movement.
10	AGV STOP because of irregular line and chrominance in front.

Illumination of the environment played a big role not just by detecting the edges but also because chrominance processing was implemented. Utilising the webcam, the auto white balance played a big role in the RGB and HSV colour values. Applying the white balance setting in the manual option improved the results as the threshold settings could be changed to satisfaction with each different environment trial run.

5.4 Reconfigurable ability of the AGV

In this section, the performance of the sign recognition system of the AGV was evaluated. The sign recognition system provided the route reconfigurability to be applied by the operator by placing the applicable signs along the route for a specific AGV.

The sign recognition system made provision for signs to be detected at a rotated angle of $\pm 7.5^\circ$ (section 4.7.1). The signs could however be detected to a maximum rotated angle of 45° for the left and right sign and 30° rotated angle for the Stop sign. This vertical rotation detection was better than expected although the signs were never placed at such angles. Detection situations where the sign was so skew were also never encountered. Figure 5.9 indicates the results achieved in simulations testing the system as it was never placed at this angle in the actual evaluation runs but, could occur if a sign stand was knocked over or placed incorrectly.

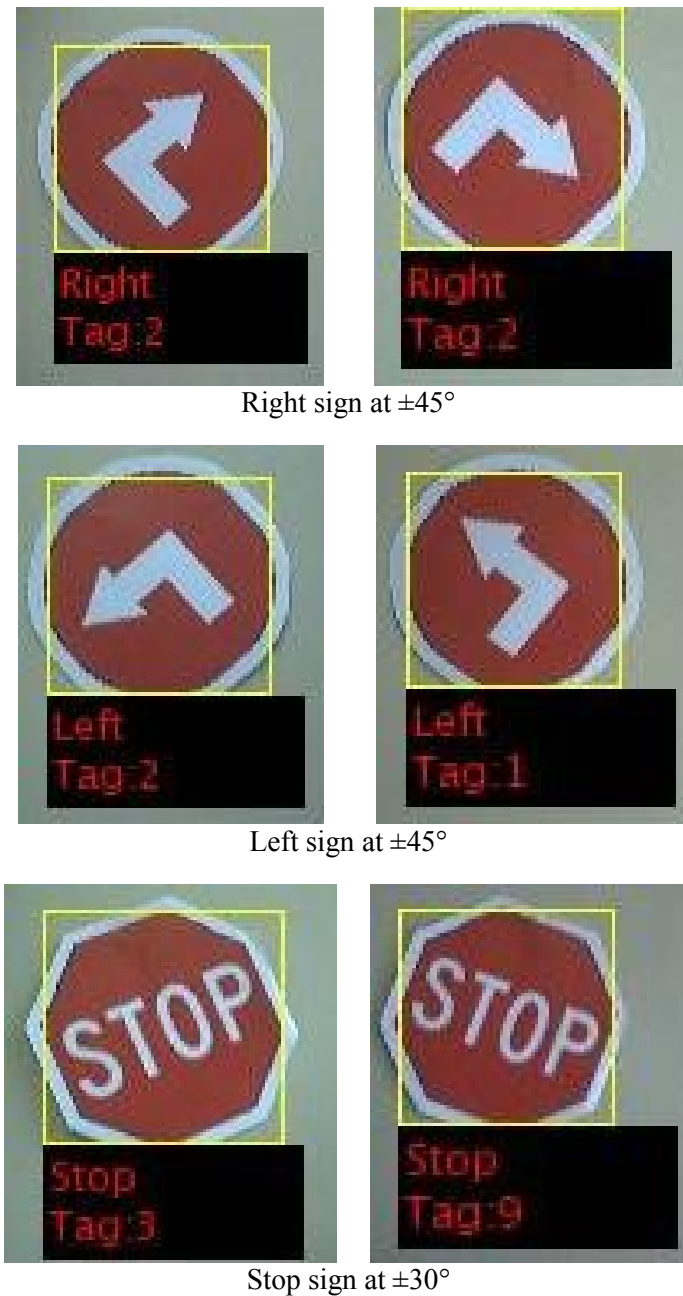


Figure 5.9: Indication of the degree at which the signs could be detected utilising sign recognition

With the left and right sign rotated beyond the 45° mark the sign was detected as indicating the opposite direction. Thus the arrow shape played very little role in the recognition phase because of the low resolution implemented in the templates. The angular shape played the predominant role as expected.

All of the signs could be detected safely at a maximum distance of 15 metres from the AGV, with the resolution of the webcam set to 640×480 pixels per frame. This resolution

was however not used as the area of interest was implemented for conserving processing time. With the frame resolution setting of 180 x 96, the signs could be detected at a maximum range of 2.4 metres, determined experimentally. The detection distance decreased dramatically with the decrease in the resolution used on the webcam settings with the signs size and templates resolution kept constant. This result correlates with the predicted distance derived from using; the data in Table 4.2, graph drawn shown in Figure 5.10 and using the prediction equation (5.2) obtained through a curve fit – producing Table 5.6.

The prediction equation calculating:

$$distance = 10e^{\frac{pixels-179.36}{54.15}} + 30 \quad (5.2)$$

where *distance* represent the distance in centimetres between the camera setup and sign to be detected and *pixels* represent the pixel count of the average height by length of the sign to be detected (represented by the bounding box and Boolean picture size). All the answers and calculations relate to a sign of approximately 18 cm x 18 cm in size.

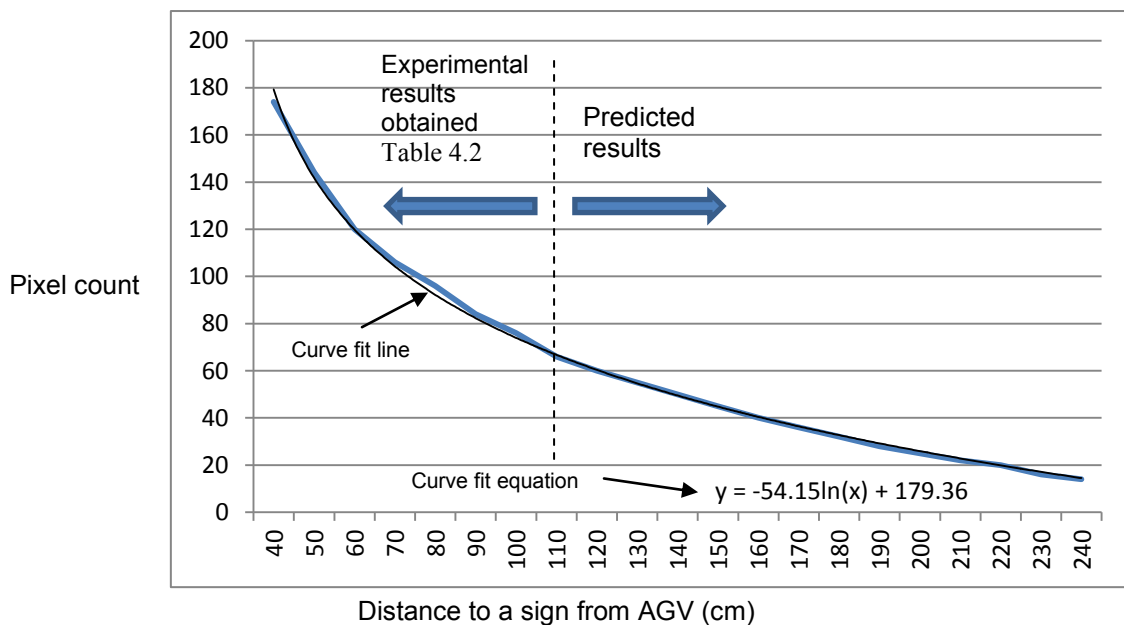


Figure 5.10: Distance to a sign plotted against the pixel count of the sign detected with curve fit

Figure 5.10 indicates the distance from the signs to the AGV plotted against the experimental results (pixel count) including the predicted results (pixel count) with the equation obtained applying a curve fit.

Table 5.6: Experimental results with predicted distance of signs detected from AGV

	Distance to a sign	Approximate pixel count	Predicted pixel count
Experimental results obtained Table 4.2	40 cm	174 X 174	179 X 179
	50 cm	144 X 144	142 X 142
	60 cm	120 X 120	120 X 120
	70 cm	106 X 106	104 X 104
	80 cm	96 X 96	92 X 92
	90 cm	84 X 84	82 X 82
	100 cm	76 X 76	74 X 74
	110 cm	66 X 66	67 X 67
	120 cm	60 X 60	60 X 60
	130 cm	55 X 55	55 X 55
	140 cm	50 X 50	50 X 50
	150 cm	45 X 45	45 X 45
	160 cm	40 X 40	40 X 40
	170 cm	36 X 36	36 X 36
	180 cm	32 X 32	33 X 33
	190 cm	28 X 28	29 X 29
	200 cm	25 X 25	26 X 26
	210 cm	22 X 22	23 X 23
	220 cm	20 X 20	20 X 20
	230 cm	16 X 16	17 X 17
	240 cm	14 X 14	14 X 14

Values obtained experimentally to evaluate the predicted values

The conclusion would be that there must be enough pixels to be used in the correlation process utilising a template size of 12 x 12 pixels for detection. The size of the detection template results in a predicted distance of 2.5 metres for sign detection. Utilising a template size of 18 x 18 pixels for recognition resulted in a predicted distance of 2.27 metres. The sign detected range at this set webcam resolution resulted in an expected distance between these two indicators at a distance of 2.4 m proven and obtained experimentally.

Each time a sign is recognised a *Tag* number is allocated to the sign while it is tracked. A registry of 9 is kept but this number could be changed by a variable setting in the

program. An indicator is also allocated to the recognised sign, 1 for *Stop*, 2 for *Left* and 3 for *Right* as could be seen in a previous Figure 4.44 mentioned to keep track of the particular sign.

Encountering the signs at an offset horizontal front angle also did not provide a problem as the deviation from the straight on position could vary to as much as 50° without failure to recognise the sign as could be seen in Figure 5.11.

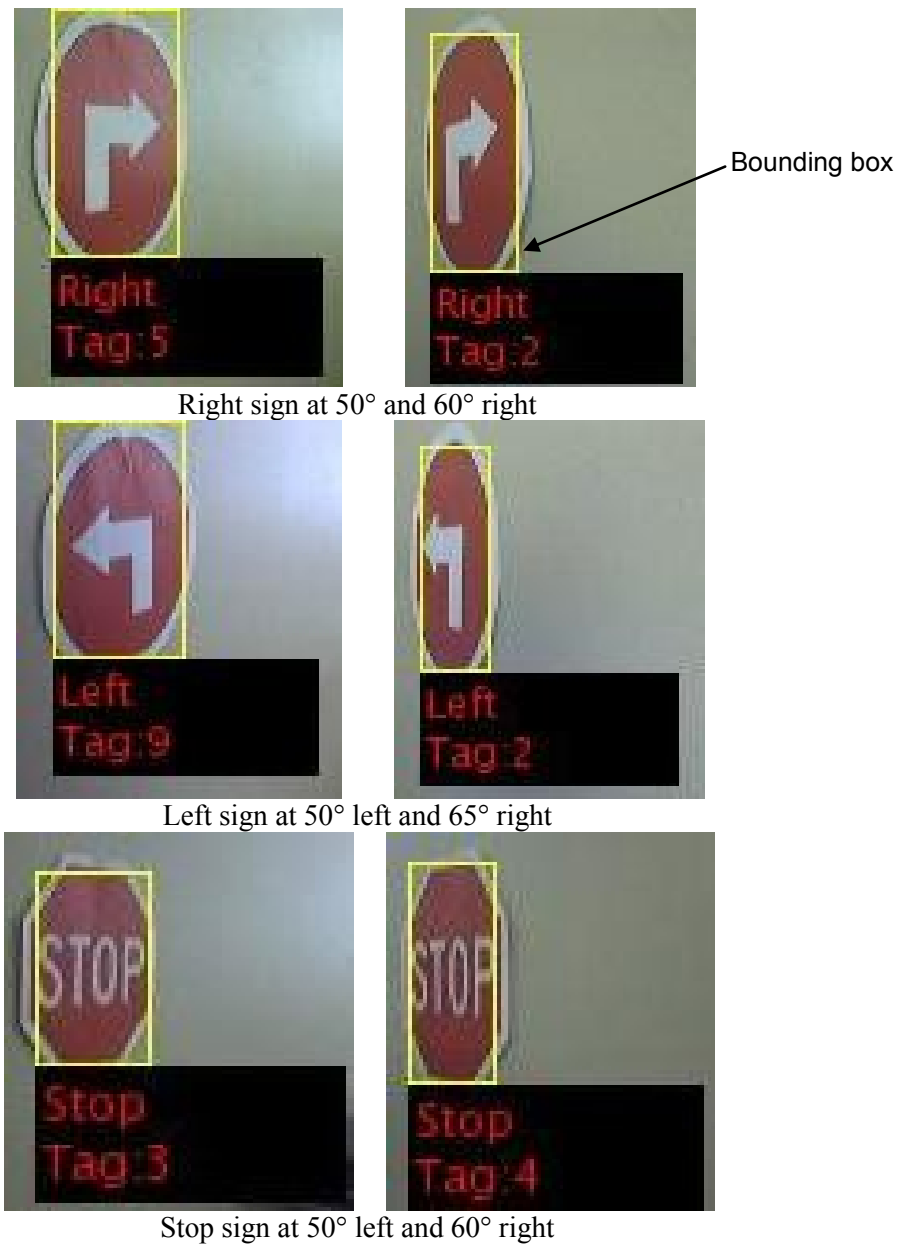


Figure 5.11: Indication of the offset to the straight on angle for sign recognition

The AGV movement control acted on the signs control function at a predefined distance, set at 70 cm for evaluation purposes, determined by the set area of the bounding box. The distance between the sign and AGV was determined by the average area of the bounding box seen in Figure 5.11 around the sign (explained in section 4.7.6). As the sign size was kept constant the biggest change experienced in the dimensions of the bounding box was the width. The reason was the angle at which the AGV approached the sign. The height of the bounding box relative to the distance between the AGV and sign also seemed to stay almost constant. The result was that the AGV acted on the sign command much closer at a large angle deviation from head on to the sign. This resulted in a distance of reaction of between 40 cm and 64 cm which did not pose any problems as the size of the platform was relatively small. This is perhaps better illustrated in Figure 5.12.

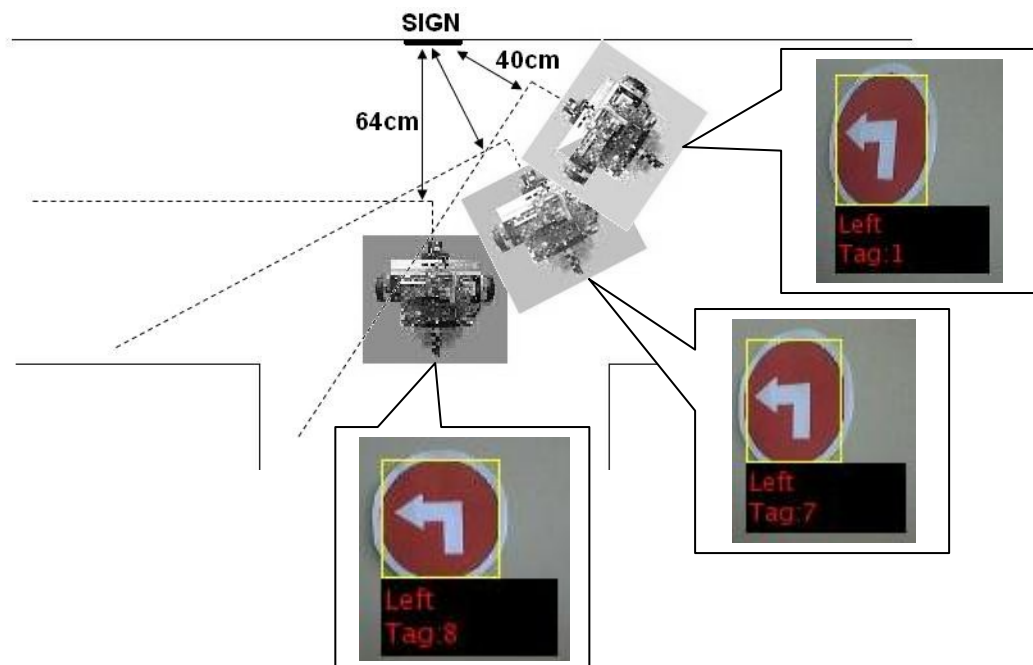


Figure 5.12: Distance from the sign determined by area at different angles of approach

This resulted perhaps in a more stochastic approach where the sign height compensated a bit for the loss in width by the increase of pixel count the closer the AGV got to the sign. This prompted an approach to just use the height of the sign rather than the bounding box. Utilising the bounding box were still a better stochastic approach leaving the possibility

of placing the signs at different heights along the route which is more flexible in an industry approach.

Different colours were introduced to provide different AGVs with the ability to follow their own routes. To select a specific colour the RGB (method 1 discussed in section 4.7.5) and hue (H) values (method 3 section 4.7.5) were successful but very vulnerable to lighting conditions. The method selecting the correct YCbCr values proved to be the most robust method with low light conditions and white balance changes utilising the webcams.

5.5 Summary

The omnidirectional vision system was evaluated by means of simulations on a PC platform before it was implemented for navigation and control purposes on a laptop placed on an AGV. This chapter includes both these results. The omnidirectional vision system performed well on the PC platform but, the laptop was lacking processing power.

The AGV platforms performed well. The 3-wheel AGV travelled at double the speed of that of the 4-wheel AGV with the same drive commands because of mechanical differences. As the omnidirectional setup had a high centre of gravity the 3-wheel AGV was more stable because of the larger base area.

Implementing route navigation with sign control utilising different colours in conjunction with the omnidirectional vision system proved difficult because of the lack in the laptops processing power. Applying the concept of area of interest, not including image corrections and calibration and utilising an own developed mirror configuration proved to be viable solutions enabling use of the laptop. Testing the navigational and control system with the omnidirectional vision configuration as acquisition system in real time resulted in a maximum operational speed for the AGV of approximately 6 cm/sec. This is not viable for an industry application. Hence, testing of the AGV control system was limited to a single camera facing in the direction of movement, representing an “area of interest” system, saving time on the omnidirectional image acquisition time. This enabled testing at a speed of 12 cm/sec – which is still slow for an industrial application, but proved the viability of a vision-based control system.

Testing different AGVs on individual routes proved the navigation and sign control concept to be a workable vision system utilising a low resolution area of interest. All of the tests performed on the AGVs were done in real time and the reconfigurability of the system proved to be successful.

5.5.1 Possible improvements of the refraction and reflection of the mirror and camera setup

The construction of the mirror and camera could be improved because of refraction and more importantly reflection. Changing the design and/or materials could improve the image quality. Glass has a typical reflection percentage of 8% [69, p. 2], quartz has a better reflective percentage of typically 4 to 5% [70] and acrylic (Perspex®) is 4% [71, p. 20]. The application of an anti-reflective (AR) coating may also help [72].

5.5.2 Possible improvements on the vision navigation and control system

Although it did not happen during a specific trial run it was evident from other evaluations that the resolution of the eight direction control and movement of the AGV would need a higher number of intervals. As the AGV got closer to the route border detected at a specific angle and no steering control was detected at that angle to steer the AGV away or in the right direction, the AGV ran over the route edge before it altered the control direction or never even corrected the direction of the AGV. This is explained by Figure 5.13 where the AGV drifted toward the left edge of the route border at an angle that would not be detected to steer it away.

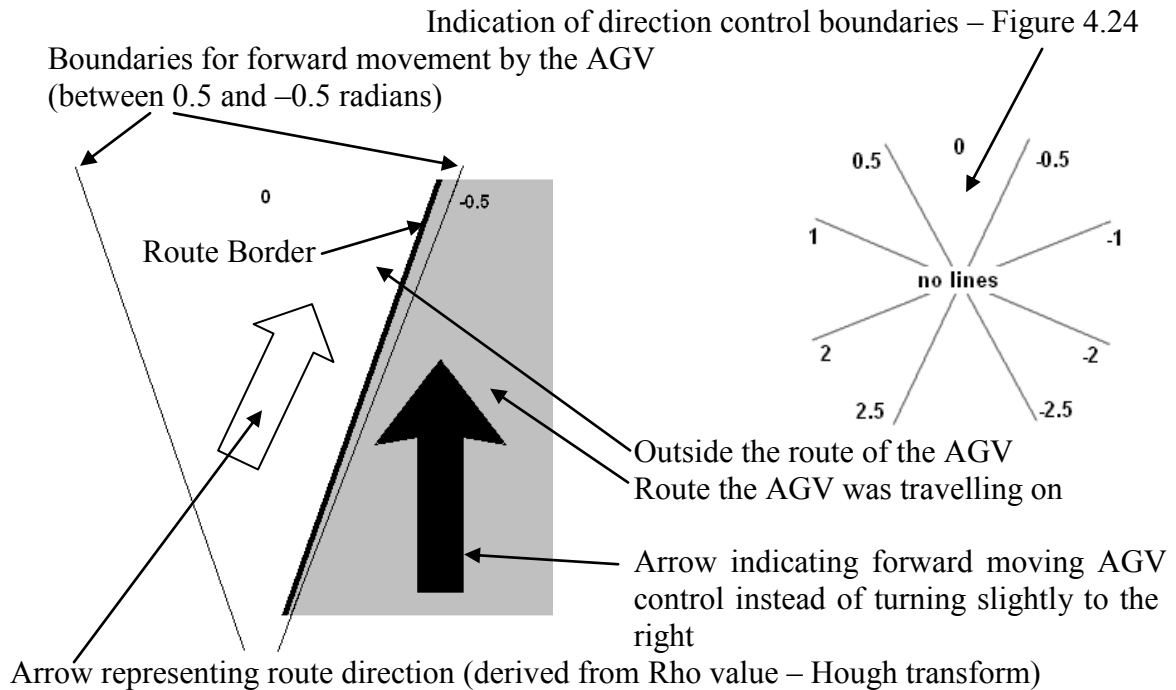


Figure 5.13: Explanation for the need for a higher selection control resolution

In Figure 5.13 the border of the route is still within the 0.5 and -0.5 radians range. Thus the AGV control generates a forward movement. The AGV is actually too close to the border and will run into the wall or over the route boundary. In this case a slight movement to right would have been preferred.

Thus the steering control between full Left and Right needs a higher resolution of steering angles to accurately control the AGV as illustrated by Figure 5.14. There was also no need for the reverse sections as this was not needed or implemented as the AGV comes to a stand still if no route is detected or if it is a dead end.

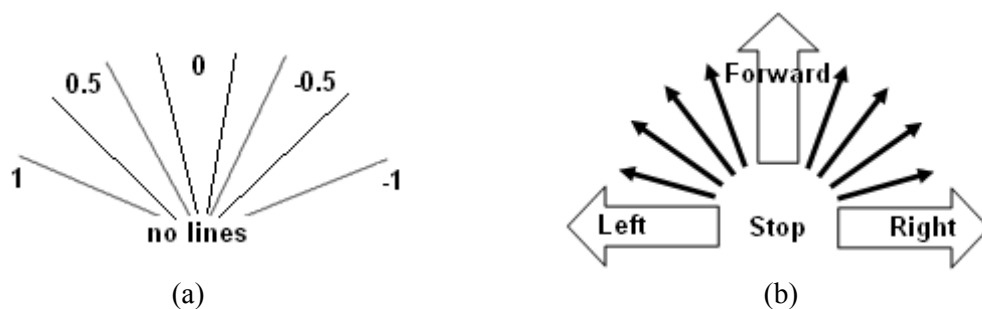


Figure 5.14: Directions resolution improvement suggestion, (a) higher resolution direction division in radians; (b) resultant direction code to be generated and test if no lines for movement

This higher resolution control was, however, not implemented during execution of the project.

5.5.3 Conclusion with regard to the relationship between – webcam resolution, template resolution and distance to a sign

In the sign navigation and control of the AGV the following relationships between:

- the distance from a sign to an AGV,
- the set resolution of the webcam or area of interest used in the vision acquisition,
- the resolution of the templates to be used, and
- sign size.

were that the further the sign needs to be recognised from the AGV the higher the resolution of the webcam or area of interest needs to be, the higher the template resolution forces the sign to be detected closer to the AGV and increasing the sign size would increase the distance for detection.

This is represented by equations (5.3) to (5.5).

$$\mathbf{distance\ to\ a\ sign} \approx \mathbf{webcam\ resolution} \quad (5.3)$$

$$\mathbf{distance\ to\ a\ sign} \approx \frac{\mathbf{1}}{\mathbf{template\ resolution}} \quad (5.4)$$

$$\mathbf{distance\ to\ a\ sign} \approx \mathbf{sign\ resolution} \quad (5.5)$$

Chapter 6

Conclusion

This chapter summarises the contributions of this thesis and possible directions for future research.

6.1 Summary

The main aim of this research was to develop an omnidirectional vision sensor observing AGV surroundings. This sensor is of no use if there are no applications for such a device. Navigation with vision was applied. In accomplishing this task the research was divided into the following facets:

- research to possible solutions and applications (Chapter 2),
- development of the omnivision sensor (Chapter 3),
- development of navigating an AGV by means of vision (Chapter 4),
- evaluating the systems developed (Chapter 5), and
- a conclusion (Chapter 6).

6.2 Original contributions

The research carried out in this project led to the development of a MATLAB[®] function used in the conversion of an omnidirectional picture to a panoramic picture displaying a 360° view of the camera position surroundings. This function was used in testing the conversion speed of the transform as well as certain possible calibrations to be used.

In testing the possibility of using either MATLAB[®] or C# as software platform, a C# program was developed in converting omnidirectional pictures to panoramic pictures selecting the area to be converted. This program made use of the executable function compiled in MATLAB[®] and C# was used to develop the graphical user interface (GUI).

As processing speed was a factor throughout the research selecting a specific area of interest in the Omnidirectional picture led to considerable savings in processing speed. This paved the way to use low-cost and resolution cameras.

Vision implemented on the AGV as sensory component, guided the research in implementing edge and chrominance route navigation. This also led to the implementation of reconfigurable sign positioning and detection for navigation applications. Specific control actions for navigation based on distance from the sign utilising vision were implemented. This concept in navigation led to a reconfigurable navigated AGV with minimal operator intervention.

The research produced publications in an accredited journal, several proceedings and a poster presented as can be verified in the references.

6.3 Evaluation of, and the conclusion of the vision system

Even though the omnidirectional vision system worked well in the development phase of the research on single frame conversions, it required high computational power with large programming execution resources for conversions of a video stream. This was too time-consuming for the laptop platform used which resulted in a slow moving AGV. Proving the possibility of using omnidirection vision for the applicable navigational application was still possible. Utilising omnidirectional vision still proved the elimination of sensors to be used in identifying the AGV's surroundings.

Utilising an in-house developed mirror setup system improved the aspect ratios of the visual environmental input of the AGV saving on omnidirectional calibration and conversion time. The research for utilising NN and GA in identifying the AGV's surroundings was also put on hold by these results.

Implementing the area of interest concept saved valuable computational time on implementation but there are other options still to be investigated for improving time management listed as options for future research.

6.4 Evaluation of, and the conclusion of a navigation interface

The navigation results proved to be working well utilising route and sign navigation. The Prewitt edge detection proved to be the best choice for edge detection in this project. The Hough line detection proved to be functional in route navigation, where both edge of an object and chrominance were utilising in route detection.

The concept of using an area of interest not only helped in saving conversion time of the omni picture but, also assisted in the detection of the route by selecting the relevant area of interest. This saved on time utilised for correction strategies of the picture and area to be converted.

Coloured sign navigation was implemented by using three different signs, applying tracking and detection. The signs colour was then used for different AGVs, defining each AGV's route. The distance to the sign was detected by using the area of the sign counting the pixels within a bounding box surrounding the sign. A pre-set distance to a sign triggered a specific navigation command.

Ambient lighting proved to be a factor in detection the route and signs as was to be expected. The lighting could be improved in each scenario or evaluation, but provision was also made in terms of variable parameters for fine tuning in the program navigation and control programming blocks. These settings and light improvements were however route and area specific.

Using a single camera without omnivision assisted in the speed of the AGV platform as less processing time was consumed to perform the conversions on the laptop platform.

Lighting quality also improved as the light path from an object did not need to travel through the Perspex and onto the small mirror area.

6.5 Assembly of different systems in a single platform

The choice of using MATLAB[®] solely as software platform for the vision, navigation and control of an AGV proved to be the best solution for this specific research and possible future development as the different concepts could be tested and combined in one platform. There is an unproven possibility that MATLAB[®] contributed to a slower processing time than the industrial implementation of, for example C# or LabVIEW[™]. This evaluation between platforms, however, did not form part of the investigation.

Combining the vision system, navigational system, sign recognition system and AGV control proved to be possible even on a laptop and did function to prove the concept, but at too slow speed for industry applications.

Even if the software simulations and AGV experimental testing provided promising results, there are still some aspects like the hardware platform, software algorithms and lighting configuration that can be improved upon to provide better performance.

6.6 Future research

In improving the processing speed and the machine vision system the following could be investigated:

- The processing power for the AGV navigation could be located on an alternative processor not situated on the AGV or machine vision system. The speed of the bidirectional communication system between the machine vision system and the operational control system of the AGV will then be a determining functional characteristic of the complete system.
- Alternatively it might be possible to “teach” the AGV to interpret sensed optical signals as representing specific physical phenomena, distorted in the same manner. In other words, rather than convert the captured omnidirectional images into its panoramic equivalent – as humans do – it might be possible to programmatically

convert the physical environment into its distorted equivalents. The relative merits of these two techniques could be investigated and the most appropriate system implemented in the final configuration. NN, GA and fuzzy logic could also be possible solutions. These were initially investigated as possible solutions to the omnidirectional navigation control system.

- 2-D versus 3-D optimisation could also be investigated by either using two photographic systems in a stereovision configuration, or via possible focus or reflector movement. The aim with this will be to accurately determine the physical size of an object and its distance from the vision system.
- Very-high-speed integrated circuits (VHSIC) hardware description language (VHDL) and Field-Programmable Gate Array (FPGA) could be incorporated in the transform from omnidirectional conversion to panoramic [73][74]. Implementing the proven concept using an area of interest (section 3.4) and perhaps a compiler generating the mapping could be investigated.
- Incorporating parallel processing by identifying cores to certain functions [50].
- Development of the vision control software on a more relevant industrial platform for example incorporating it on the sRio platform [75].
- Include a localisation technique similar to those used in Swanepoel's [7, pp. 41-44], Boje's [35, pp. 70-88] and Scaramuzza et al.'s [68] to be used for correlation in localisation and mapping.

Appendix A – Colour inset

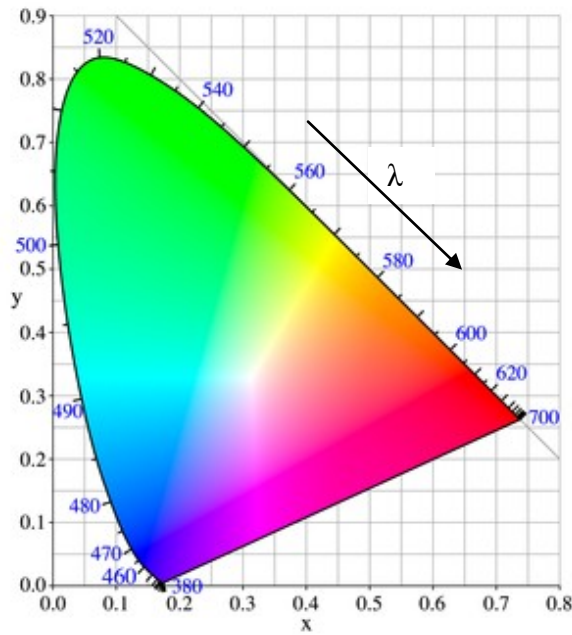


Figure 2.13, page 18.

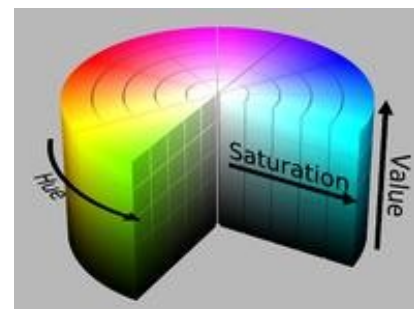


Figure 2.15, page 19.

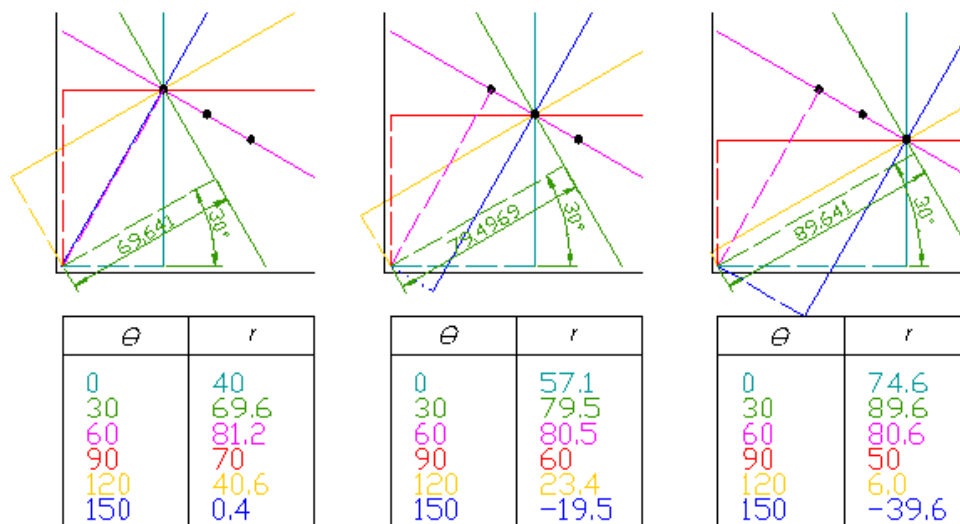


Figure 2.21, page 26.

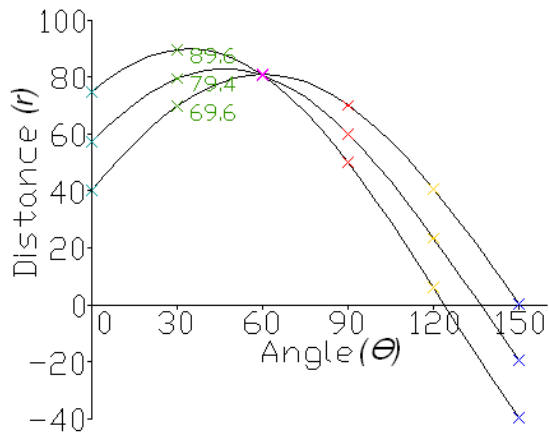


Figure 2.22, page 26.

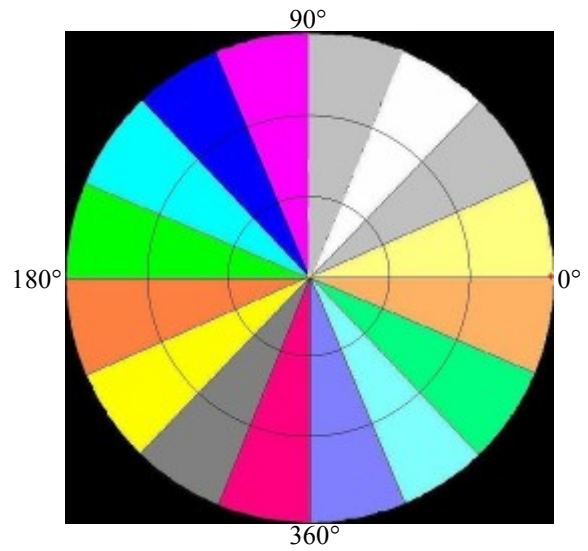


Figure 3.6, page 41.

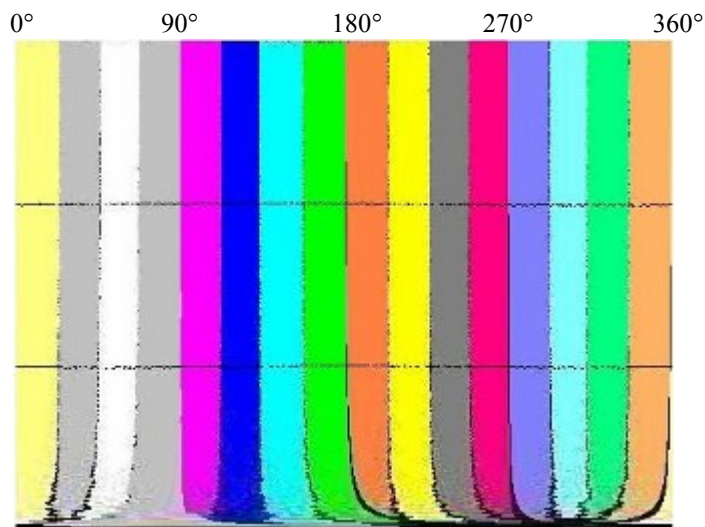
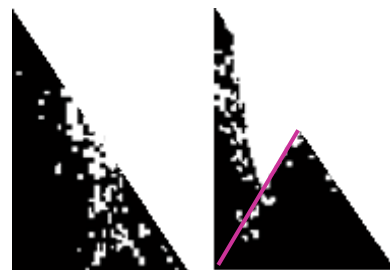
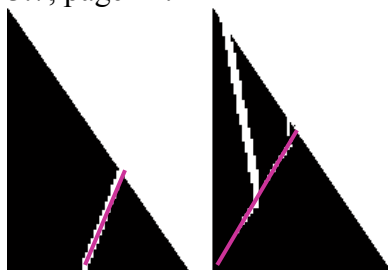
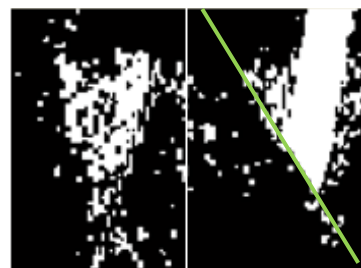
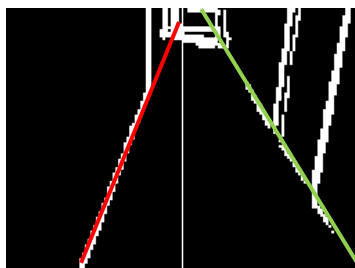


Figure 3.7, page 41.



(d)



(e)

Figure 4.13, page 69.

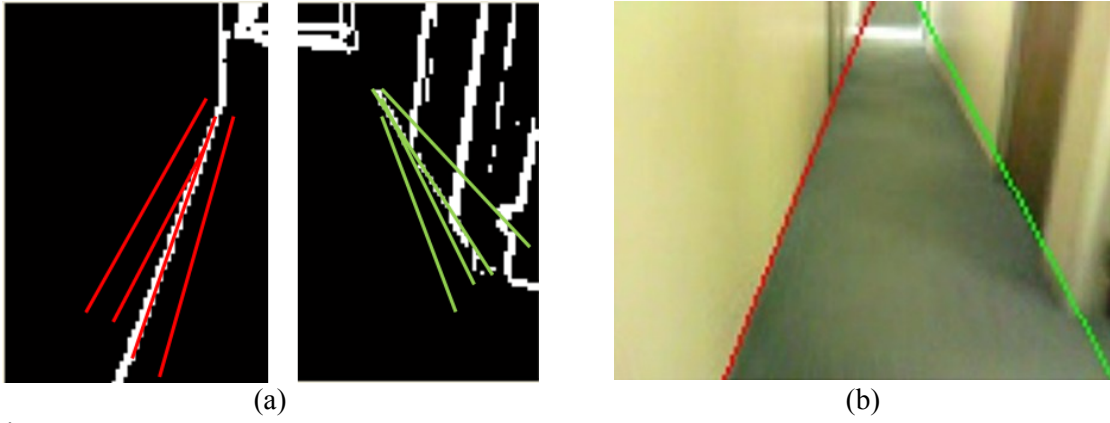


Figure 4.16, page 71.

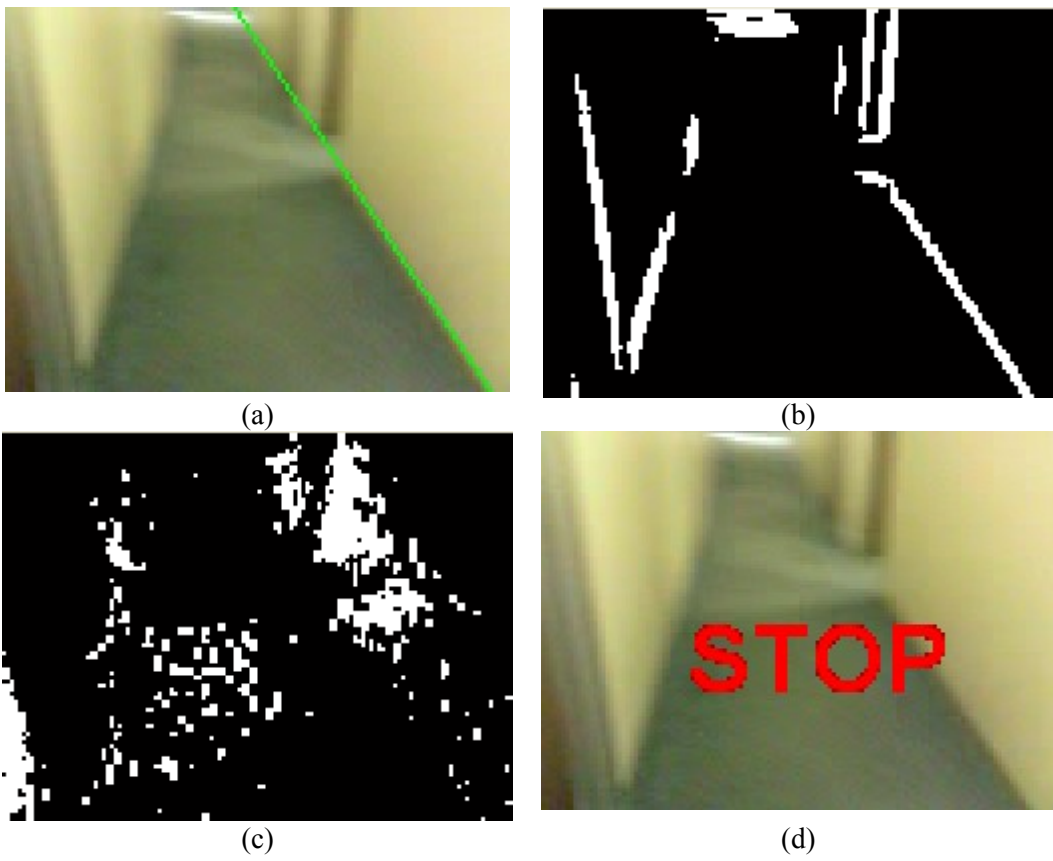


Figure 4.22, page 77.



Figure 4.38, page 90.

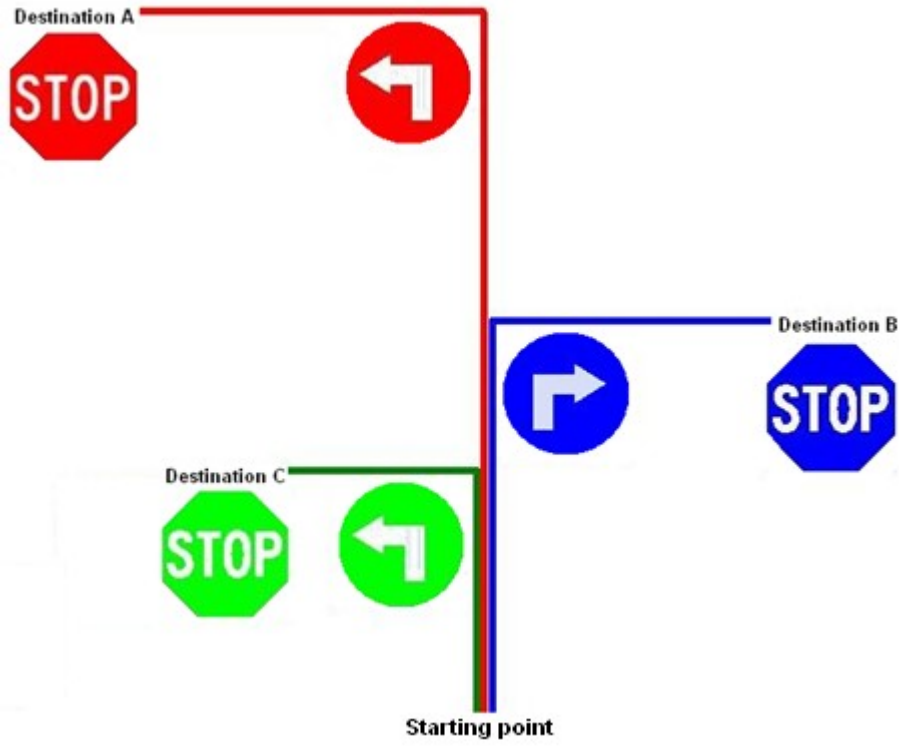


Figure 4.39, page 91.

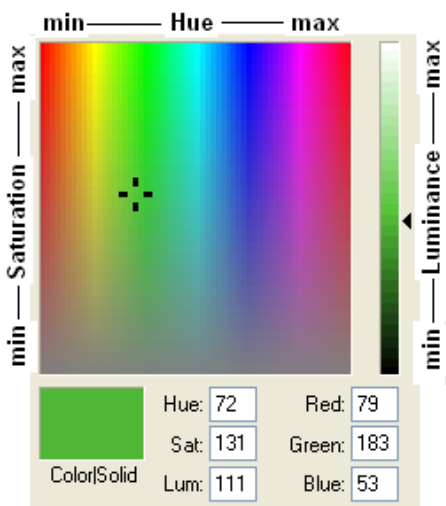


Figure 4.40, page 92

Appendix B

B.1 Compiling an standalone executable file utilising the *mcc* command in MATLAB[®] from an m-file

MATLAB Compiler

1. Prerequisites for Deployment

* Verify the MATLAB Compiler Runtime (MCR) is installed and ensure you have installed version 7.10. (*The same version that you used to compile the .m files with*)

* If the MCR is not installed, run MCRInstaller, located in:

C:\Program files\MATLAB\R2009a\toolbox\compiler\deploy\win32\
MCRInstaller.exe

For more information on the MCR Installer, see the MATLAB Compiler documentation.

NOTE: You will need administrator right to run MCRInstaller.

2. Files to Deploy and Package

Files to package for Standalone

-imnwrapbenm.exe, *the executable file generated with the mcc command.*

-MCRInstaller.exe

-include when building component by selecting “include MCR” option in deploytool.

-This readme file

3. Definitions

MCR – MATLAB Compiler uses the MATLAB Compiler Runtime (MCR), which is a standalone set of shared libraries that enable the execution of M-files. The MCR provides complete support for all features of MATLAB without the MATLAB GUI. When you package and distribute an application to users, you include supporting files generated by the builder as well as the MATLAB Compiler Runtime (MCR). If necessary, run MCRInstaller to install the correct version of the MCR. For more information about the MCR, see the MATLAB Compiler documentation.

For a complete list of product terminology, go to <http://www.mathworks.com/help> and select MATLAB Compiler.

* NOTE: <matlabroot> is the directory where MATLAB is installed on the target machine.

4. Appendix

A. On the target machine, add the MCR directory to the system path specified by the target system's environment variable.

i. Locate the name of the environment variable to set, using the table below:

<u>Operating System</u>	<u>Environment Variable</u>
Windows	PATH

ii. Set the path by doing one of the following:

NOTE: <mcr_root> is the directory where MCR is installed on the target machine.

On Windows systems:

* Add the MCR directory to the environment variable by opening a command prompt and issuing the DOS command:

```
set PATH=<mcr_root>\v710\runtime\win32;%PATH%
```

Alternately, for Windows, add the following pathname:

```
<mcr_root>\v710\runtime\win32
```

to the PATH environment variable, by doing the following:

1. Select the My Computer icon on your desktop.
2. Right-click the icon and select Properties from the menu.
3. Select the Advanced tab.
4. Click Environment Variables.

NOTE: On Windows, the environment variable syntax utilises backslashes (\), delimited by semi-colons (;).

References

- [1] M. Fend, S. Bovet and V. V. Hafner, “The Artificial Mouse - A Robot with Whiskers and Vision,” [Online]. Available: <http://www.amousse.de/>. [Accessed 2007].
- [2] J. Fernandes and J. Neves, “Using Conical and Spherical Mirrors with Conventional Cameras for 360° Panorama Views in a Single Image,” in *IEEE 3rd International Conference on Mechatronics*, Budapest, 2006.
- [3] A. Treptow, G. Cielniak and T. Duckett, “Comparing Measurement Models for Tracking People in Thermal Images on a Mobile Robot,” [Online]. Available: <http://www.aass.oru.se/~gck/papers/treptow05comparing.pdf>. [Accessed 2007].
- [4] D. Scaramuzza, *Omnidirectional vision: From calibration to robot motion estimation*, ETH Zurich: Dissertation Doctor of Science, 2008.
- [5] J. Bittencourt and F. Osório, “Adaptive Filters for Image Processing based on Artificial Neural Networks,” [Online]. Available: <http://csdl2.computer.org/comp/proceedings/sibgrapi/2000/0878/00/08780336.pdf>. [Accessed 2007].
- [6] M. Sonka, V. Hlavac and R. Boyle, *Image Processing, Analysis, and Machine Vision*, International student 3rd ed., Toronto: Thomson, 2008.
- [7] P. J. Swanepoel, *Omnidirectional Image Sensing for Automated Guided Vehicle*, Bloemfontein, Free State: Dissertation MTech, School of Electrical and Computer Systems Engineering, CUT, Free State, April 2009.
- [8] D. Scaramuzza, A. Martinelli and R. Siegwart, “A flexible technique for accurate omnidirectional camera calibration and structure from motion,” *IEEE International Conference on Computer Vision Systems (ICVS 2006)*, January 2006.

- [9] D. Scaramuzza, A. Martinelli and R. Siegwart, "A toolbox for easy calibrating omnidirectional cameras," *IEEE International Conference on Intelligent Robots and Systems (IROS 2006)*, October 2006.
- [10] C. Geyer and K. Daniilidis, "Catadioptric Projective Geometry," *International Journal of Computer Vision*, vol. 45, no. 3, pp. 223-243, 2001.
- [11] D. G. Aliaga, "Accurate Catadioptric Calibration for Real-time Pose Estimation in Room-size Enviroments," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2001.
- [12] L. G. Roberts, "Machine perception of three-dimensional solids," in *Optical and Electro-Optical Information Processing*, J. Tippett, Ed., Cambridge, MIT Press, 1965, pp. 159-197.
- [13] J. F. Canny, *Finding edges and lines in images*, Cambridge, MA: Technikal Report AI-TR-720, MIT, Artificial Intelligence Laboratory, 1983.
- [14] J. F. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, November 1986.
- [15] M. Brady, "Representing shape," in *Robotics and Artificial Intelligence*, M. Brady, L. A. Gerhardt and H. F. Davidson, Eds., Berlin, Springer and NATO, 1984, pp. 279-200.
- [16] R. M. Haralick and L. G. Shapiro, "Computer and Robot vision," vol. 1, MA, Addison-Westley, Reading, 1992, pp. 28-48.
- [17] T. Smith and J. Guild, "The C.I.E. colorimetric standards and their use," *Transactions of the Optical Society*, vol. 33, no. 3, pp. 73-134, 1931.
- [18] D. P. Greenberg and A. Allison, "Computer generated images for medical applications," *Cornell University Program of Computer Graphics*, vol. 12, pp. 196-202, August 1978.

- [19] C. Poynton, *Digital Video and HDTV*, San Francisco: Morgan Kaufman, 2003, pp. 291-292.
- [20] S. Lu, G. T. Tsechpenakis, D. N. Metaxas, M. L. Jensen and J. Kruse, "Blob Analysis of Head and Hands: A Method for Deception Detection," in *System Sciences, HICSS'05. Proceedings of the 38th Annual Hawaii International Conference, IEEE*, January 2005.
- [21] T. F. Chan and L. A. Vese, "Active contour without edges," *IEEE Trans. Image Processing*, vol. 10, no. 2, pp. 266-277, 2001.
- [22] C. Steger, M. Ulrich and C. Wiedemann, *Machine Vision Algorithms and Applications*, Berlin: Wiley-VCH, 2008.
- [23] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," in *A.I. Memo No. 572*, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, April 1980, pp. 1-27.
- [24] C.-C. Lin and M. Wolf, "Detecting Moving Objects Using a Camera on a Moving Platform," in *ICPR 2010, 2010 International Conference on Pattern Recognition*, Istanbul, 2010.
- [25] L. Shapiro and G. Stockman, *Computer Vision*, Prentice-Hall, 2001.
- [26] Wikipedia, "Hough transform," [Online]. Available: http://en.wikipedia.org/wiki/Hough_transform#mw-head. [Accessed June 2011].
- [27] D. Somon, "Kalman Filtering," *Embedded Systems Programming*, vol. 21, no. 6, pp. 72-79, June 2001.
- [28] G. Welch and G. Bishop, "An Introduction to the Kalman Filter," in *UNC-Chapel Hill, TR 95-041*, Chapel Hill, Department of Computer Science, University of North Carolina, July 2006, pp. 1-16.
- [29] G. A. Carpenter and S. Grossberg, *Pattern Recognition by Self-Organizing Neural Networks*, Cambridge, MA: MIT Press, 1991.

- [30] B. M. Wilamowski, "Challenges in Applications of Computational," in *Proceedings of the 2010 IEEE International Symposium on Industrial Electronics*, Bari, Italy, 2010.
- [31] W. S. McCulloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.
- [32] N. J. Luwes, *Artificial Intelligence Machine Vision Grading System*, Bloemfontein: Thesis DTech, School of Electrical and Computer Systems Engineering, CUT, Free State, 2010.
- [33] B. J. Kotze, *Use of Genetic Algorithms for the Optimisation of Test Patterns for Digital Circuits*, Bloemfontein: Dissertation MTech, School of Electrical and Computer Systems Engineering, CUT, Free State, March 2001.
- [34] B. Siciliano and O. Khatib, *Handbook of Robotica*, Berlin: Springer, 2008.
- [35] E. P. Boje, *Intelligent AGV with Navigation, Object detection and Avoidance in an Unknown environment*, Bloemfontein, Free State: Dissertatin MTech, School of Electrical and Computer Systems Engineering, CUT, Free State, November 2006.
- [36] H. G. Lubbe, *Intelligent Automated Guided Vechiles(AGV) with Genetic Algorithm Decision Making Capabilities*, Bloemfontein, Free State: Dissertation MTech, School of Electrical and Computer Systems Engineering, CUT, Free State, January 2007.
- [37] H. Nguyen, N. Pezeshkian, M. Raymond, A. Gupta and J. Spector, "Autonomous Communicaton Relays for Tactical Robots," in *The 11th International Conference on Advanced Robotics (ICAR 2003)*, Coimbra, Portugal, 30 June - 3 July 2003.
- [38] J.-S. Lee, "A Command Filtering Framework to Collision Avoidance for Mobile Sensory Robots," in *IEEE International Symposium on Industrial Electronics 2007 (ISIE 2007)*, Vigo, Spain, 4-7 June 2007.
- [39] [Online]. Available: <http://www.surmafloor.co.uk/antislipsurfaces.htm>. [Accessed April 2012].

- [40] M. A. Sotelo, F. J. Rodriguez, L. Magdalena, L. M. Bergasa and L. Boquete, "A Colour Vision-Based Lane Tracking System for Autonomous Driving on Unmarked Roads," in *Autonomous Robots 16*, Manufactured in the Netherlands, Kluwer Academic Publishers, 2004, pp. 95-116.
- [41] T. Goedemé, M. Nuttin, T. Tuytelaars and L. Van Gool, "Omnidirectional Vision Based Topological Navigation," *International Journal of Computer Vision*, vol. 74, no. 3, pp. 219-236, September 2007.
- [42] J. Park, W. Rasheed and J. Beak, "Robot Navigation using Camera by Identifying Arrow Signs," in *IEEE Computer Society*, Washington, 2008.
- [43] U. Zakir, E. A. Edirisinghe and A. Hussain, "Road Sign Detection and Recognition from Video Stream," in *Advances in Brain Inspired Cognitive Systems, Lecture Notes in Computer Science, Volume 7366*, Berlin Heidelberg, Springer, 2012, pp. 411-419.
- [44] B. Kotze, G. Jordaan and H. Vermaak, "A Reconfigurable AGV with Omnidirectional Sensing," *Journal for New Generation Sciences*, vol. 8, no. 3, pp. 76-85, 2010.
- [45] B. Kotze, G. Jordaan and H. Vermaak, "Development of a Reconfigurable Automatic Guided Vehicle Platform with Omnidirectional Sensing Capabilities," in *Proceedings of the 2010 IEEE International Symposium on Industrial Electronics*, Bari, Italy, 2010.
- [46] *Users Guide, Image Acquisition Toolbox, Version 1*, The Mathworks, March 2003.
- [47] *Users Guide, Image Processing Toolbox, Version 4*, The Mathworks, May 2003.
- [48] B. J. Kotze, "Development of an Automated Guided Vehicle platform for research purposes and teaching," in *Proceedings of: 13th Annual Research Seminar, Faculty of Engineering & Information Technology, CUT, Free State, SESSION 4*, Bloemfontein, 14 October 2010.

- [49] P. Rüdiger, "Optical Intensity," in *Encyclopedia of Laser Physics and Technology*, RP Photonics, 2012.
- [50] "MATLAB® Version 7.8 (R2009a), Help Demos, [Software Accessed]".
- [51] [Online]. Available: <http://www.mathworks.com/matlabcentral/fileexchange/1836>. [Accessed Jan 2011].
- [52] J. Li, X. Dai and Z. Meng, "Automatic Reconfiguration of Petri Net Controllers for Reconfigurable Manufacturing Systems With an Improved Net Rewriting System-Based Approach," *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 1, pp. 156-167, January 2009.
- [53] *Data Sheet, PIC16F87X, 28/40-Pin 8-bit CMOS FLASH Microcontrollers*, U.S.A.: Microchip, 1999.
- [54] QUANTACH, "Products - 802.11b/g serial and Ethernet wireless solutions," 2008. [Online]. Available: http://www.quatech.com/catalog/airbornedirect_80211bg.php. [Accessed 6 April 2009].
- [55] [Online]. Available: <http://sine.ni.com/nips/cds/view/p/lang/en/nid/208010>. [Accessed June 2012].
- [56] *Sabertooth 2x10 User's Guide*, Dimension Engineering, February 2007.
- [57] B. J. Kotze, "Navigation of a Automatic Guided Vehicle utilizing Image Processing and a Low Resolution Camera," in *Proceedings of: 14th Annual Research Seminar, Faculty of Engineering & Information Technology, CUT, Free State, SESSION 3*, Bloemfontein, 13 October 2011.
- [58] MATLAB®, "Chroma-based Road Tracking Demo, Computer Vision System Toolbox," in *Help*, Published with MATLAB® 7.12, 2007.
- [59] *Getting Started Guide, NI-DAQ(TM) mx for USB Devices*, National Instruments, December 2007.
- [60] D. Sawics, *Hobby Servo*.

- [61] *General specification for Tetrax DC drive motor*, January 2011.
- [62] MATLAB®, “Traffic Warning Sign Recognition Demo, Computer Vision System Toolbox,” in *Help*, Published with MATLAB® r 7.12, 2007.
- [63] Wikimedia Foundation, Inc., “PNG - From Wikipedia, the free encyclopedia,” 20 June 2012. [Online]. Available: <http://en.wikipedia.org/wiki/PNG>. [Accessed July 2012].
- [64] “Microsoft Windows XP, Professional Version 2002, Service Pack 3, [Software Accessed]”.
- [65] C.-C. Hsu, M.-C. Lu and K.-W. Chin, “Distance Measurement Based on Pixel Variation of CCD Images,” in *Proceedings of the 4th International Conference on Autonomous Robots and Agents*, ©2009 IEEE, Wellington, New Zealand, 10-12 February 2009.
- [66] K. Wolf, “Geometry and dynamics in refracting systems,” *European Journal of Physics*, vol. 16, pp. 14-20, 1995.
- [67] D. Schön, *The Reflective Practitioner, How Professionals Think In Action*, Basic Books, 1983.
- [68] D. Scaramuzza and R. Siegwart, “Appearance-Guided Monocular Omnidirectional Visual Odometry for Outdoor Ground Vehicles,” *IEEE TRANSACTIONS ON ROBOTICS*, vol. 24, no. 4, pp. 1015-1026, October 2008.
- [69] *Guardian Sunguard*, Advanced Architectural Glass.
- [70] W. Colblentz, “Infra-Red Reflection Spectra,” *The American Physical Society*, vol. 23, no. 3, p. 248, 1906.
- [71] Perspex South Africa, “Properties of 'Perspex'®,” [Online]. Available: www.perspex.co.za. [Accessed October 2012].
- [72] *Understanding Brewer Science's Bottom Anti-Reflective Coatings*, Rolla MO, USA: Brewer Science Inc., 2002.

- [73] J. Maddocks and R. Williams, *VHDL Image Processing Source Modules REFERENCE MANUAL*, Somerset, UK: HUNT ENGINEERING, July 2006.
- [74] B. A. Draper, J. R. Beveridge, A. W. Böhm, C. Ross and M. Chawathe, "Accelerated Image Processing on FPGA's," *IEEE Transactions on Image Processing* , vol. 12, no. 12, pp. 1543-1551, January 2004.
- [75] [Online]. Available: <http://sine.ni.com/nips/cds/view/p/lang/en/nid/205894>. [Accessed September 2012].