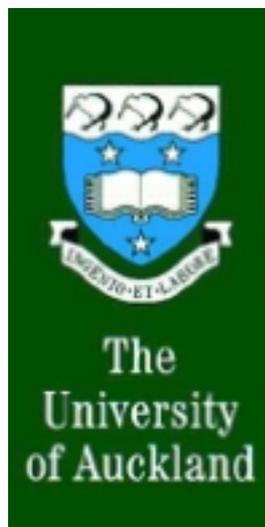


A PATH FOLLOWING SYSTEM FOR AUTONOMOUS ROBOTS WITH MINIMAL COMPUTING POWER



Approved by _____

Recommended for Acceptance: _____

Accepted: _____

A PATH FOLLOWING SYSTEM FOR AUTONOMOUS ROBOTS WITH MINIMAL COMPUTING POWER

by

Andrew James Thomson, BSc

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

Master of Science in Computer
Science

UNIVERSITY OF AUCKLAND

2001

ABSTRACT

Classical mobile robot control systems are not suitable for use in industrial environments. The high cost of such systems, both to acquire and maintain them, prohibits their adoption. This thesis proposes a new mobile robot architecture and investigates the motion control subsystem of that architecture. This motion control subsystem is based on following a marked path using visual servoing techniques to reduce computational overhead. Two pieces of information are extracted from each frame: the horizontal position of the path, relative to the centre of the image, and the gradient of the path. These pieces of information are then passed into a proportional steering system, which uses them to steer towards the path. The use of marked paths rather than a model of the environment ensures that the downtime, caused by changes to the environment, is minimised. The lack of a model of the robot should allow the control system to easily be ported to different robot hardware.

ACKNOWLEDGMENTS

The author wishes to thank Dr. Jacky Baltes for his support and encouragement during the period of this research.

TABLE OF CONTENTS

Abstract.....	i
Acknowledgements	ii
List of Figures	v
Introduction	1
Chapter 1: Motivation.....	2
1.1 Introduction	2
1.1.1 Problems with Existing Mobile Robots.....	3
1.2 Reducing Complexity	3
1.2.1 Landmarks with Explicit Paths	4
1.2.2 Landmarks without Explicit Paths	5
1.2.3 Marked Paths	6
1.3 Control Architecture	7
1.3.1 Path Following.....	8
1.3.2 Localisation	9
1.3.3 High-Level Behaviors	9
Chapter 2: Literature Review.....	10
2.1 Path Following.....	11
2.1.1 Road Vehicles.....	11
2.1.2 Smaller Robots	12
2.1.3 Problems with These Methods	12
2.2 Visual Servoing.....	13
Chapter 3: Description of algorithm.....	15
3.1 Hardware Description	15
3.2 Image Processing.....	17
3.3 Steering Control.....	22
3.4 Example 1: A Straight Line Path	23
3.5 Example 2: A Curved Path.....	25
Chapter 4: Results.....	28
4.1 Description of experimental setup.....	28
4.2 Ten Pixel Results.....	31
4.3 Twenty Pixel Results.....	37
4.4 Thirty Pixel Results	43
Chapter 5: Conclusion	48
Future Work	49

Bibliography..... 50

LIST OF FIGURES

<i>Number</i>	<i>Page</i>
1.1 A landmark system with explicit paths	5
1.2 A landmark system without explicit paths.....	6
1.3 A marked path system.....	7
1.4 The proposed robot control system	8
3.1 One of the CITRs autonomous robots	15
3.2 Colour image of the path.....	19
3.3 Greyscale image of the path.....	19
3.4 Image after applying the threshold	19
3.5 Image processing detail	20
3.6 Inverted image after applying thresholding	23
3.7 First pass for the point P1.....	23
3.8 First pass for the point P2.....	24
3.9 The path approximation between P1 and P2	24
3.10 Colour image of the path.....	25
3.11 Inverted image after applying thresholding	25
3.12 First and second passes for the point P1	26
3.13 First pass for the point P2.....	26
3.14 The path approximation between P1 and P2	26
4.1 The evaluation track.....	28
4.2 The evaluation track with the robot showing scale.....	29
4.3 A plot of the test path.....	30
4.4 The track approximation	31
4.5 Diagram of the processing region size at 10 pixels.....	32
4.6 Graph of the average error against the offset weighting.....	32
4.7 Plot of the robots path with an offset weighting of 100%.....	33
4.8 Plot of the robots path with an offset weighting of 50%.....	35
4.9 Plot of the robots path with an offset weighting of 5%.....	35
4.10 Graph of the average error s of the test runs against the speed of the robot	36

4.11	Graph of the average error s of the test runs against the number of steering commands issued	37
4.12	Diagram of the processing region size at 20 pixels.....	37
4.13	Graph of the average error against the offset weighting.....	38
4.14	Plot of the robots path with an offset weighting of 75%.....	40
4.15	Plot of the robots path with an offset weighting of 5%.....	40
4.16	Graph of the average error s of the test runs against the speed of the robot.....	41
4.17	Graph of the average error s of the test runs against the number of steering commands issued.....	42
4.18	Diagram of the processing region size at 30 pixels.....	43
4.19	Graph of the average error against the offset weighting.....	44
4.20	Plot of the robots path with an offset weighting of 85%.....	45
4.21	Plot of the robots path with an offset weighting of 5%.....	45
4.22	Graph of the average error s of the test runs against the speed of the robot.....	46
4.23	Graph of the average error s of the test runs against the number of steering commands issued.....	47

INTRODUCTION

A mobile robot control system based on following a marked path can overcome several of the problems of traditional mobile robots. To be useful in an industrial environment a mobile robot must be inexpensive, reliable and easy to adapt to any changes in the working environment.

Traditional mobile robots, typically based around models of the environment or learning techniques, such as neural networks, are not suitable. They require expensive sensors and considerable processing resources, which are expensive and often determine the design of the robot. If the environment changes either the model must be replaced, requiring considerable time and expertise, or the robot must be retrained, again requiring time and expertise.

Control systems based on marked paths do not suffer from these limitations as badly. They can be implemented using minimal sensors and processing resources, and if the environment changes the paths can simply be altered, requiring little or no reprogramming of the robots.

The goal of this thesis is to investigate a simple path following robot, which is designed to be a flexible base for more complex robot behaviours.

Chapter 1

MOTIVATION

1.1 Introduction

Robotic systems have long been used to improve the speed and efficiency of manufacturing tasks. In the automotive industry robots perform the more repetitive tasks, such as welding, or potentially hazardous tasks like spray painting [KUKA2000][CRS1999]. In the electronics industry robots perform tasks requiring high precision, like manufacturing integrated circuits, or repetitive tasks such as assembling circuit boards. Many other industries use robots to replace humans in repetitive or hazardous situations.

However, in all of these cases robotic systems are used in small, usually self contained, sections of the production lines. Humans are still required to fetch components from warehouses and, in some cases, shift partially assembled product sections between stations on the production line. These tasks are just as repetitive and, sometimes, hazardous as the tasks performed by robots.

While conveyer belts can be used to transport components their paths are difficult to alter if the production line changes and they require a large amount of floor space, particularly if several belts converge on one section of the production line. Mobile robots, on the other hand,

do not require much floor space and their paths can more easily be altered if circumstances change. The main problem with mobile robots is that they are expensive to obtain and maintain.

1.1.1 Problems with Existing Mobile Robots

Most mobile robots are designed using a five-step process:

1. Define a problem
2. Create a set of requirements to solve the problem
3. Generate a problem specification
4. Design a solution to the specification
5. Design a robot to implement the solution

While this design process produces robots that are good at solving the problem, they are expensive, time consuming to produce and do not tend to be very adaptable. Typical solutions use models of the environment and the robot in order to perform the desired task. These models require accurate information about the state of the robot and the environment and thus require considerable processing resources to maintain these models. The major source of the expense of these robots is the sensors and processors required to update these models. The cost of the other materials to construct the robots is not as significant.

1.2 Reducing Complexity

As the control system of the robot determines the sensors and processing resources required, reducing the complexity of the control

system should reduce the cost of the robot. Several methods can be used to reduce the complexity of the control system, though the easiest is to simplify or remove one or both of the models used. The model of the environment can be simplified by one of several methods.

1.2.1 Landmarks with Explicit Paths

In this method, the robot holds a basic map detailing how turn in order to move between consecutive pairs of landmarks (see Figure 1.1). As each landmark is unique, the robot can localise itself at each stage of the path. However, if the robot misses a landmark it is unable to recover. If the environment changes it must be reprogrammed.

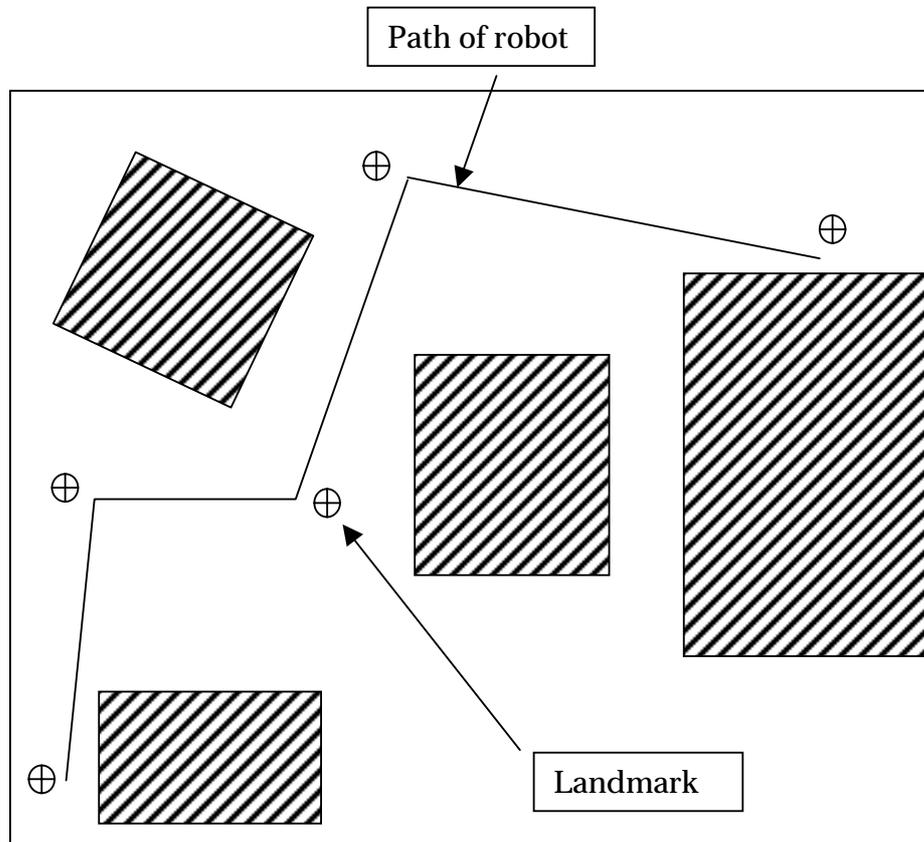


Figure 1.1: A landmark system with explicit paths.

1.2.2 Landmarks without Explicit Paths

Rather than programming the robot with the path to follow between the landmarks, this method simply provides the robot with an ordered list of the landmarks to drive to and leaves the robot to find its own way (see Figure 1.2). While the processing of the environment model is easier, this method does require that the robot can detect two or more landmarks at any point on the path, requiring either a good sensor system or a large number of landmarks. As with the

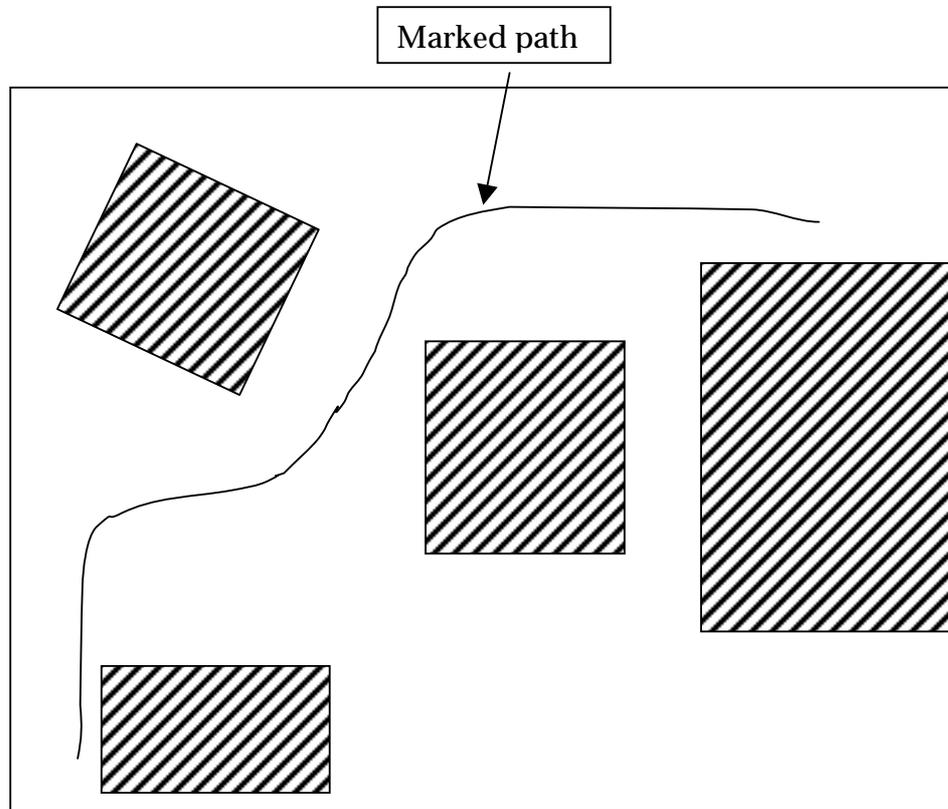


Figure 1.3: A marked path system.

reduce development time. If the robot loses the path a reasonably simple localisation routine can allow it to recover. The path can be easily altered to reflect changes in the environment without requiring any changes to the robots software.

1.3 Control Architecture

If the motion control subsystem is simplified, it makes sense to simplify the entire robot control system. In an industrial environment, a mobile robot would mainly be used to transport items between two

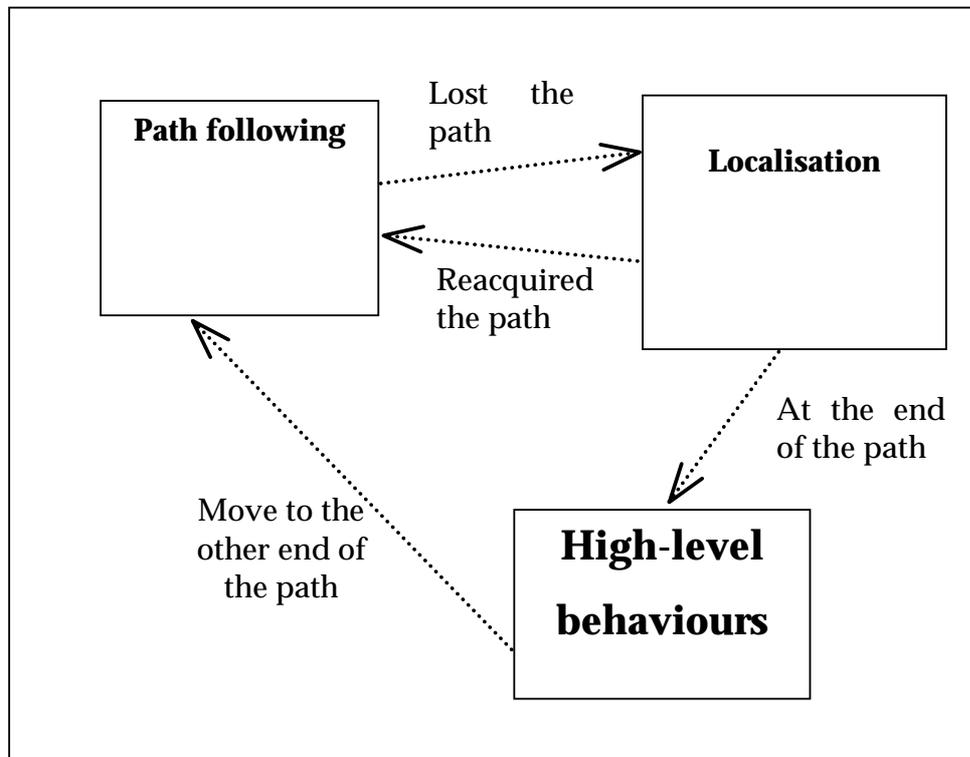


Figure 1.4: The proposed robot control system.

stages in the production line. Thus, the most complex robot behaviours are only used at each end of the path.

With this in mind, the control system for the robot can be split into three sections (see Figure 1.4).

1.3.1 Path following

A system of paths marked on the floor of the work area provides the most flexible replacement for a model of the environment. The path following system can be made extremely simple, as the robot would

not be required to perform complex tasks while moving. This thesis describes one possible path following system.

1.3.2 Localisation

If, for some reason, the robot loses track of the path some form of localisation system is required in order to reacquire it. The localisation system would also be responsible for determining whether the robot is lost or it has reached the end of the path.

1.3.3 High-level behaviours

The most complex robot behaviours would be used at each end of the robot's path to load or unload the robot or, in the case of a warehouse robot, to retrieve a particular item on a shelf.

LITERATURE REVIEW

In recent years a great deal of time and effort have been spent on developing systems to enable an autonomous robot to follow a marked path using a vision system. Not surprisingly, the majority of this research has been towards modifying, or designing from scratch, a full-sized road vehicle so that it can drive on ordinary roads without human supervision. Due to the large amount of space available in an ordinary road vehicle, high performance computers can be used to perform complex image processing and, typically, to maintain a mathematical model of the vehicle and the environment [PW1993] [WSGM1999].

Research into autonomous driving using smaller robots typically follows one of two approaches. In the first approach a mathematical model of the vehicle and its surroundings is generated, tested in simulation, and then applied to a robot built specifically for the purpose [HN1997] [KMMT1996]. In the second approach a combination of a visual servoing system and a kinematic model is used, again the robot is typically designed around the solution technique [MKS1999]. Due to the size of these robots, the processing resources available are quite limited so simpler models and

techniques, such as visual servoing, are used to reduce the processing load.

The following sections contain a brief overview of the research done in path following, including autonomous driving, and the research in visual servoing that can be applied to path following for autonomous robots.

2.1 Path Following

Path following research can be separated into two main categories: research involving road vehicles and research involving smaller robots. The following sections give a brief overview of the research in both categories.

2.1.1 Road Vehicles

Due to the large cargo spaces available in the vehicles considerable processing resources can be applied to the path following problem, thus path following techniques using real road vehicles tend to use complex processing algorithms.

While most of these systems use visual data as inputs, several other sensors, including range finders and odometers, provide additional feedback about the vehicle and the surrounding environment. Some systems are even capable of functioning, when given an accurate model of the environment, using only range finding [PW1993] or location data [WSGM1999].

The sensor data is then fed into the control system of the vehicle where most of the processing takes place. Typically the control systems are designed by Artificial Intelligence researchers and are thus based on systems such as Neural Networks [Pom1992], internal mathematical models [SN1992] or Fuzzy Logic [PW1993]. Hybrid solutions have also been trialed, combining, for example, Neural Networks and reinforcement learning [OLC2000].

As the control systems use established techniques, much of the research concentrates on high-level processing, identifying stoplights and intersections, rather than on the actual driving [SB1998] [OLC2000].

2.1.2 Smaller Robots

As the size of these robots prohibit the use of high performance computational equipment the algorithms used for path following are simpler than those used in the full sized road vehicles. However, for the most part, the same basic approaches are used though there is a greater emphasis on simulation [MKS1999] [RH1993].

The image-processing steps for these robots typically use Kalman filters [MKS1999] and other simple functions for extracting information about the curvature of the path. The control system is typically a fuzzy logic system [LSL1999] if one exists, see the next section, a kinematic model of the vehicle is also used [MKS1999] [HN1997].

2.1.3 Problems with These Methods

While all of the above methods are capable of functioning in an industrial environment, they are not suited to one. Many of the methods, [OLC2000] [Pom1992] and [PW1993] for example, rely on one or more desktop computers to perform the image processing and vehicle control computations. Those solutions using models of the environment [SN1992] require even more computing power. Such computational devices cannot easily be mounted on a mobile robot platform using current technology.

Systems using learning techniques to locate the path [Pom1992] must be trained in the actual environment under the full range of expected lighting conditions, an expensive and time consuming process. A robotic system which uses a model of the environment not only requires considerable time and effort to setup, it also requires the same effort whenever the environment changes.

Therefore, while these approaches are useful for environments such as roads, which do not change very often, they are not suitable for industrial environments, particularly where the environment may change every few months.

2.2 Visual Servoing

Visual servoing, the process of obtaining information directly from an image, is popular on smaller robots where there is not enough processing power to reconstruct 3-dimensional information from the

image. It is typically used to obtain the information required by kinematic models [MKS1999] [HN1997], though simple rote learning is also used [MMMM1994] [KMMT1996].

Most of the path following techniques using visual servoing are designed and tested in simulation, actual implementation on real robots is usually treated as an afterthought, if at all.

DESIGN

3.1 Hardware Description

As with all of the robots in use at the Centre for Information Technology and Robotics (CITR), the autonomous vision based robots

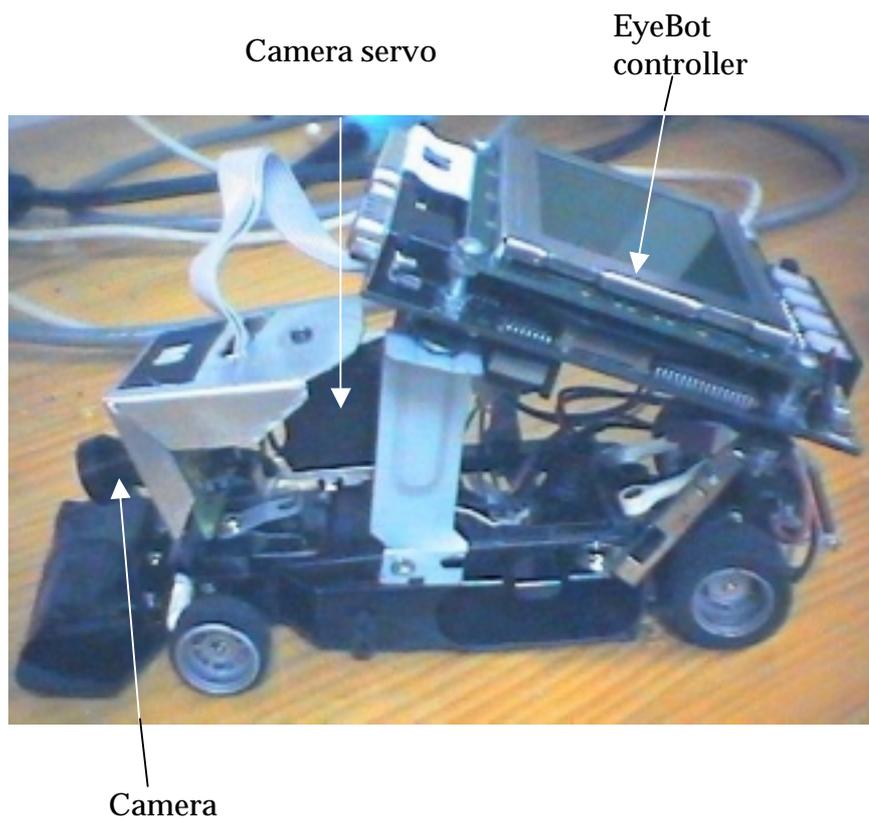


Figure 3.1: One of the CITR's autonomous robots.

(see Figure 3.1) are designed as general-purpose systems and are constructed from commercially available components. While custom-built robots designed around specific problems would have a higher performance, evaluating competing solutions to a given problem would be difficult and the resulting robots would be expensive and difficult to produce.

The chassis of the robot is the chassis of a 1:24 scale radio controlled car, which provides a drive motor capable of multiple speeds and a steering servo capable of multiple positions. The chassis does not have any form of odometry feedback for position or velocity information and the feedback from the steering servo is coarse, restricting the number of usable steering positions (to 21 for the purposes of this thesis).

The microcontroller unit is an EyeBot controller produced by Thomas Braunl of the University of Western Australia [Brau1998]. The controller is based around the Motorola MC68332 32bit microcontroller and has one megabyte of RAM and 512 KB of FLASH RAM for running and storing the supplied operating system (RoBIOS) and user programs. The controller has a number of input and output connectors for: a camera, several servos, touch sensors, infrared distance measuring sensors, motors and a serial connection for exchanging data with a desktop computer.

The camera (called an EyeCam) is designed by the University of Western Australia to work with the EyeBot controller it is a 24 bit

CMOS camera with a usable resolution of 80x60 pixels. The standard camera routines built into RoBIOS can run the camera at 3.8 frames per second, which is too low for the work at the CITR, new drivers were developed which improve the frame rate to 15 frames per second, or to 30 frames per second with a small amount of synchronisation noise on the right hand edge of each frame. While the camera is mounted on a panning servo it unnecessary to pan the camera as when the path is no longer in view a localisation module would take control.

3.2 Image Processing

As most people, when driving, make steering changes based on how local road conditions differ from the vehicles current position and orientation, it was decided to construct a simple image processing algorithm to capture this information. Firstly, we make two assumptions about the path to be followed:

Assumption 1: there is a high contrast between the path and the floor

Assumption 2: the robot is capable of driving the path

The first assumption is to guarantee that the camera is capable of easily discriminating the path from the floor, a condition that would be met in an industrial or warehouse environment. The second assumption is to ensure that the robot is capable of following the path without having to use complex reversing manoeuvres. As a robot using this path following system would be replacing a human

operated vehicle, which would be larger than the robot, there would be enough free space in the environment to allow such paths.

Unfortunately due to the green tint in the colour images from the camera (see Figure 3.2) it proved too difficult to accurately detect the path near the middle of the image. Fortunately, RoBIOS provides an efficient routine to convert the 24-bit colour image to a 4-bit grey scale (see Figure 3.3). A threshold, chosen to select the maximum amount of track and a minimum amount of the surroundings, is then applied to the grey scale image to produce a black and white image (see Figure 3.4). In practice thresholding the entire image proved too time consuming so only those portions of the image which are processed have the threshold applied. Due to the position of the camera, any part of the track in the upper half of the image is over 15 centimetres in front of the robot, and is considered too far away to be useful.



Figure 3.2: Colour image of the path.



Figure 3.3: Greyscale image of the path.



Figure 3.4: Image after applying thresholding.

Two pieces of information are extracted from the image: the offset of the path from the centre of the image, and the gradient of the path. Figure 3.5 shows the feature extraction in more detail. Rather than processing the entire region, which would require too much computing time, only the outer edges of the processing region are checked to locate the path. These edges are offset from the edges of the image due to the presence of a one pixel black border and the presence of noise caused by synchronisation problems if we use the 30 frames per second camera driver.

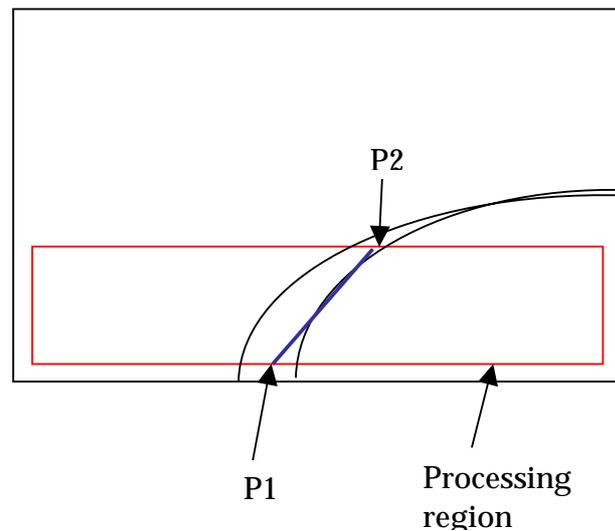


Figure 3.5: Image processing detail.

The point P1 is tested for first, across the lower edge, then up the left edge and finally up the right, if P1 does not exist the track is assumed absent. Point P2 is located by checking the top edge, then the left and right edges, if P1 was found on one of the side edges that edge is not checked for P2.

The positions of P1 and P2 are computed by taking the average position of the white pixels on the edge. One result of this is that if two parallel lines are present P1 and P2 will be between the lines, which would allow the robot to perform simple autonomous road driving if the field of view from the camera was wide enough. The line between P1 and P2 gives the trend of the path in the region of interest.

$$Offset = \frac{(P1x - 40) + (P2x - 40)}{40}$$

Equation 3.1:
Computing the offset

The offset of the path is computed by combining the differences between the horizontal positions of P1 and P2 and normalising the result so that it lies between -1, for the left-hand side of the image, and 1, for the right hand side (see Equation 3.1). The gradient is

$$Gradient = \frac{\left(\frac{(P1x - P2x)}{(P1y - P2y)} \right)}{40}$$

Equation 3.2:
Computing the
gradient.

computed and then scaled so that a gradient of zero indicates a vertical line (in the image), a negative gradient indicating that the line slopes to the left and a positive gradient indicating a slope to the right (see Equation 3.2). Note that the gradient function assumes that the path is half the image wide and that there are several special cases for

when $(P1x-P2x)$ or $(P1y-P2y)$ are zero. A horizontal line is represented by a gradient of 1 (the maximum positive gradient), as there is no way of knowing which direction it is leaning in.

3.3 Steering Control

While the image-processing step can be used on a different robotic platform with little or no modification, the steering control system must be modified. Changing from a car-like robot with multiple steering positions, as used for these experiments, to a wheelchair-like robot, where the drive motors also change the robots direction, requires enormous changes in the steering system if a kinematic model of the robot is used. A bang-bang controller is portable but tends to over steer and is thus unsuitable for high-speed applications.

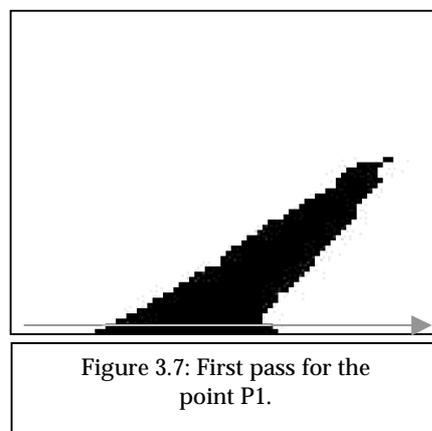
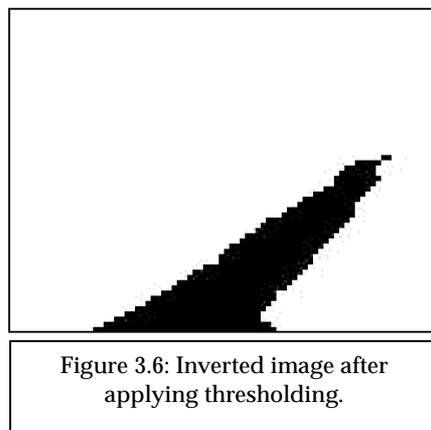
The solution used here is to use a proportional steering controller, which will steer based on a combination of the outputs of the image processing step. The only changes required when changing the chassis type are to the motor control functions and to the mapping between the outputs of the image processing step and the commands to the robot.

Obviously if the path is on the left side of the image the robot should turn left, similarly if the path is on the right side it should turn right, likewise it should turn in the direction the path is leaning in the image. However if the path is leaning to the right but the path is on the left side of the image the robot should turn either slightly left or

drive straight ahead, depending on the position and slope of the path. To ensure this behaviour, gradient and offset information from the image processing step is weighted, combined and normalised to have a value between -1 and 1. This normalised value is then used to determine which of the 21 pre-set steering positions, 10 to either side plus straight-ahead, to use in order to return the path to the centre of the image.

5.4 Example 1: A Straight Line Path

Beginning with the path in Figure 3.2, the thresholded image looks like Figure 3.6 (which has been inverted for clarity).



The first task is to locate the point P1. This is done by examining the bottom edge of the processing region, which is one pixel above the last row of the image (see Figure 3.7). This examination pass discovers 33 pixels of the track in a continuous line between (20, 59) and (53, 59). Averaging these positions gives a position for P1 at (37, 59), after

rounding fractional values to the nearest integer, as shown in Figure 3.8.

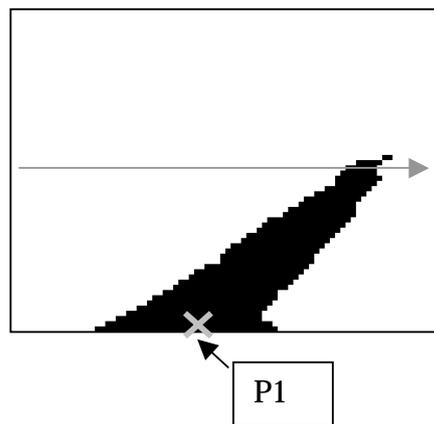


Figure 3.8: First pass for the point P2.

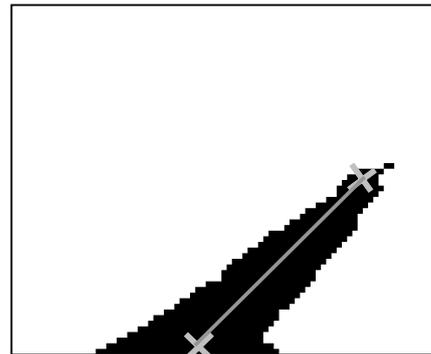


Figure 3.9: The path approximation between P1 and P2.

The next step is to locate P2 by firstly checking the top line of the processing region (see Figure 3.8). This examination pass discovers six pixels of the track in a continuous line between (62, 29) and (68, 29). Averaging these positions gives a position for P2 at (65, 29), and an approximation for the path which runs between P1 and P1 (see Figure 3.9).

The next tasks are to compute the gradient and position offset of the path approximation. The offset is the normalised combination of the offsets for P1 and P2 (see Equation 3.1), which in this case are -3 and 25 respectively, giving an offset of 0.55. The gradient is computed as in Equation 3.2 giving a gradient of 0.02.

The gradient and offset values are then combined to compute the steering angle required to follow the path. If the weightings on the offset and gradient values are equal, the output of the steering function is 0.29 or roughly 30% to the right.

3.5 Example 2: A Curved Path

Beginning with the path in Figure 3.10, the thresholded image looks like Figure 3.11 (which has been inverted for clarity)

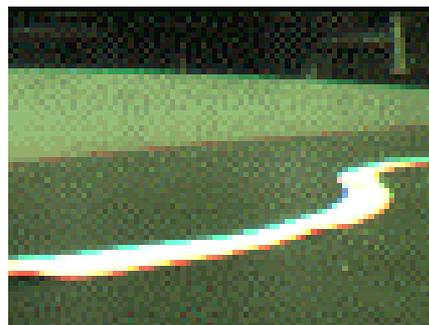


Figure 3.10: Colour image of the path.

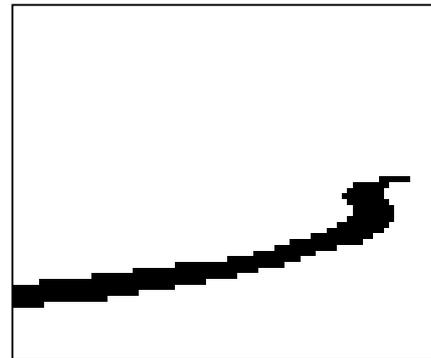


Figure 3.11: Inverted image after applying thresholding.

The first task is to locate the point P1. As the path does not cross the lower edge of the processing region, the left side of the region is checked. This examination pass discovers six pixels of the track in a continuous line between (4, 50) and (4, 56). Averaging these positions gives a position for P1 at (4, 53) as shown in Figure 3.12.

The next step is to locate P2 by firstly checking the top line of the processing region (see Figure 3.13). This examination pass discovers four pixels of the track in a continuous line between (70, 29) and (74,

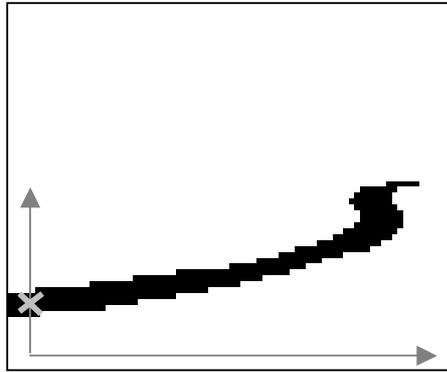


Figure 3.12: First and second passes for the point P1.

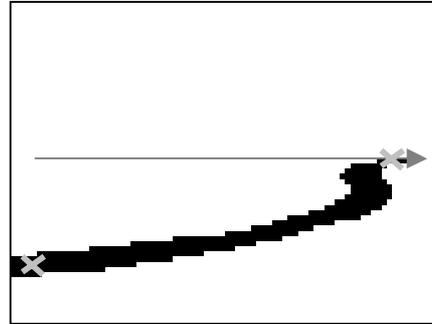


Figure 3.13: First pass for the point P2.

29). Averaging these positions gives a position for P2 at (72, 29), and an approximation for the path which runs between P1 and P1 (see Figure 3.14).

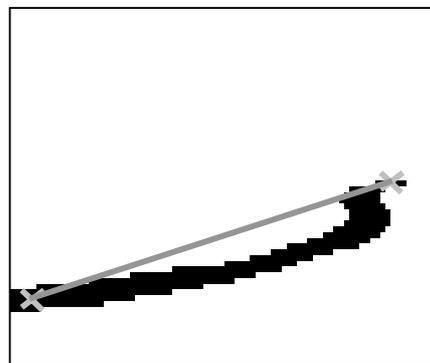


Figure 3.14: The path approximation between P1 and P2.

The next tasks are to compute the gradient and position offset of the path approximation. The offset is the normalised combination of the offsets for P1 and P2, which in this case are -36 and 32 respectively,

giving an offset of -0.1. The gradient is computed using Equation 3.2 giving a gradient of 0.07.

The gradient and offset values are then combined to compute the steering angle required to follow the path. If the weightings on the offset and gradient values are equal, the output of the steering function is -0.01 or approximately straight-ahead.

RESULTS

4.1 Experimental Setup

In order to evaluate the effects changing the weightings on the gradient and offset information have on the performance of the algorithm a test path was set up on a 2.74 by 1.52 metre playing field. The track was made from three centimetre wide curves cut from white paper and secured to the field surface with masking tape (see Figure 4.1). The white paper on the green background provided a high contrast path, which was easy to distinguish by thresholding the



Figure 4.1: The evaluation track

image. The masking tape was removed from the robot's image by the threshold (see Figure 2.4).

Using the field allowed us to use a calibrated camera to track the robot with a high precision. However the walls of the field proved to be a mixed blessing, while they prevented the robot from leaving the test area (forcing someone to chase after it), they were quite close to the track which skewed the results, reducing the maximum errors measured. If the robot became stuck against a wall it was turned so that it would be able to see the track given it's current steering angle.

The track was designed to have several challenging areas for the robot, including the S-bends on the right-hand side and the sharp turns on the left-hand side after a relatively long straight section where the robot builds up speed. The curve at the top-left of Figure



Figure 4.2: The evaluation track with the robot showing scale.

4.1 was especially difficult as it is close to the walls and the radius of

the curve is approximately half the length of the robot (see Figure 4.2), a condition unlikely to be encountered by a real road vehicle.

Figure 4.3 shows an example plot of the test path as seen from the video tracking system. In order to generate the error information, simple straight-line approximations were made to the sections of the test path (see Figure 4.4).

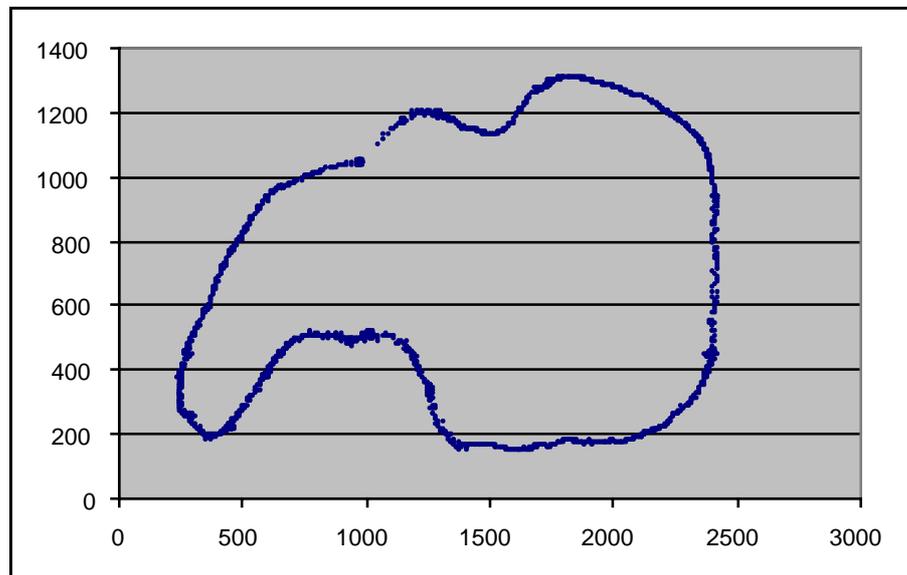


Figure 4.3: A plot of the test path.

Three different sizes of processing region were tested: ten, twenty and thirty pixels, which corresponded to distances of 11cm, 13cm and 17cm in front of the robot. Each test run consisted of five laps in order to obtain a reasonable average performance for the values of the weights, which were changed for each test run.

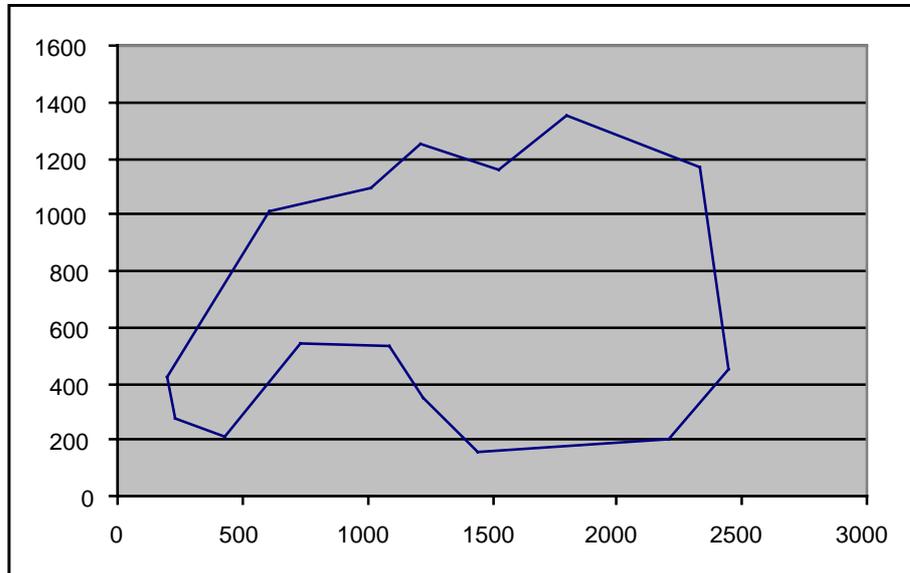


Figure 4.4: The track approximation.

In order to test the impact of other factors on the error, the maximum and average speeds for each test run were recorded as were the number of steering commands issued. All of the test runs were performed with a 60% power setting on the drive motor, though the actual speed of the robot depended on the battery level at the time of the experiment. A test run at an 80% power level was attempted, but was aborted due to fears of damage to the robot.

4.2 Ten Pixel Results

With the processing region at ten pixels in height, the robot can see a section of the path that is approximately 2.5 centimetres in length beginning approximately 8.8 centimetres in front of the bumper (see Figure 4.5). Due to this short distance, it was assumed that the offset information would be far more important than the gradient

information and thus the best results would be obtained when there was little or no weighting on the gradient. A graph of the average error against the offset weight would therefore show a steady, or possibly exponential, decrease as the offset weight increased.

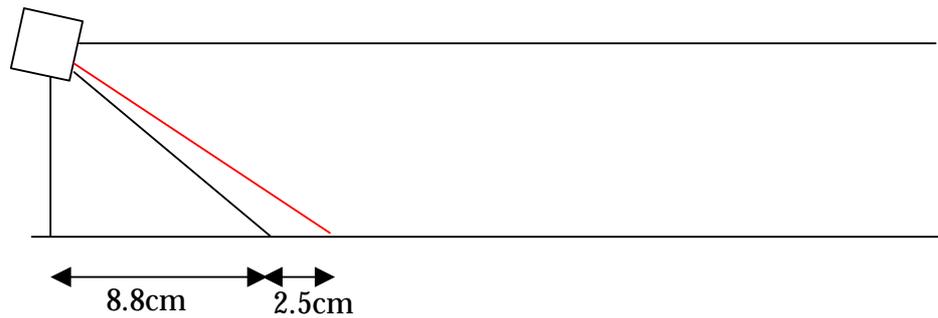


Figure 4.5: Diagram of the processing region size at 10 pixels.

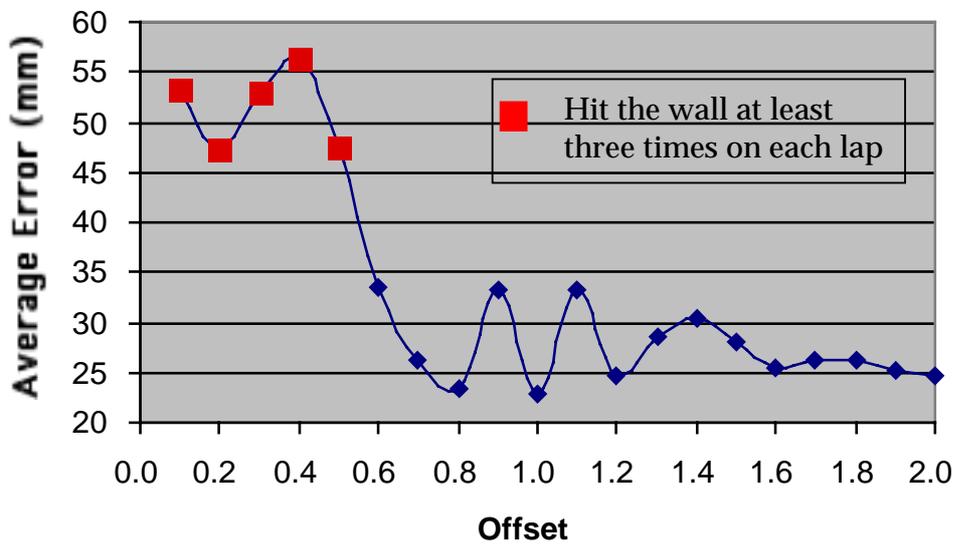


Figure 4.6: Graph of the average error against the offset weighting.

However the average errors for when the offset weight is greater than 0.6 (30%) are virtually identical (see Figure 4.6). The best results were obtained when the weights on the gradient and offset were about equal, though the difference between those errors and those obtained with higher offset weightings is less than one centimetre. Figure 4.7 shows a plot of all five laps with an offset weighting of 2.0 (100%), Figure 4.8 shows a similar plot with an offset weighting of 1.0 (50%).

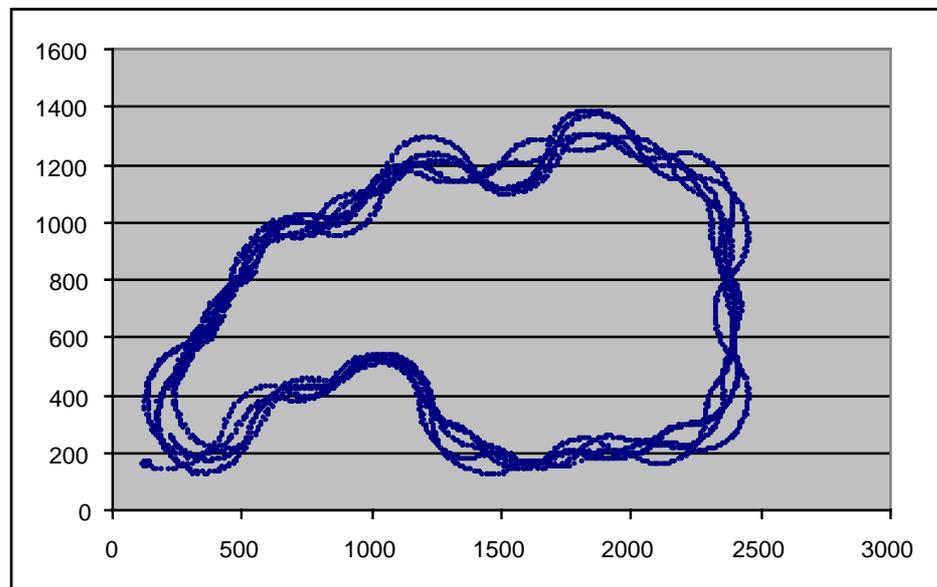


Figure 4.7: Plot of the robots path with an offset weighting of 100%.

When the weighting on the offset drops below 0.6 the errors increase dramatically, as expected. The restricted distance and length of track that the robot is capable of viewing in this configuration is not enough to obtain meaningful information on the trend of the path. Even though a plot of the robots path when the offset weight is 0.1 (5%) bears little resemblance to the actual path (see Figure 4.9), the

average error is less than six centimetres, probably due to the relatively small distance between the path and the walls.

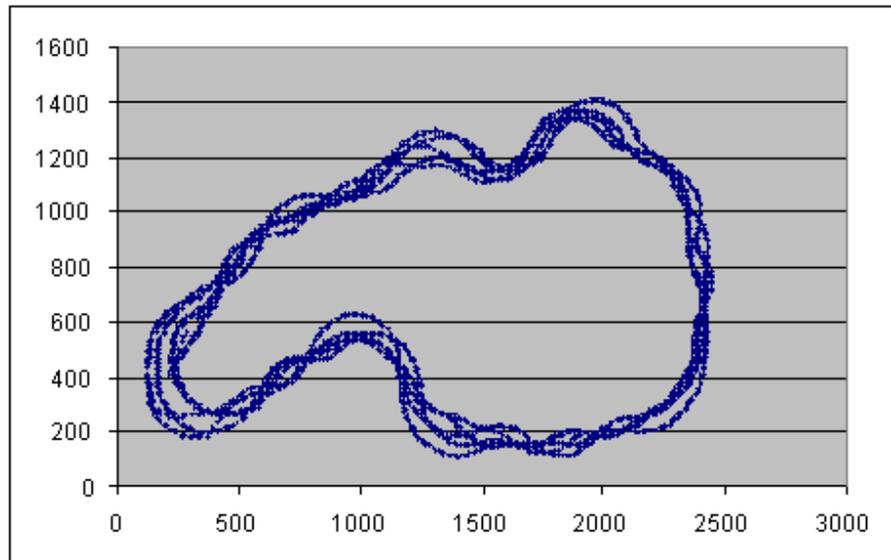


Figure 4.8: Plot of the robots path with an offset weighting of 50%.

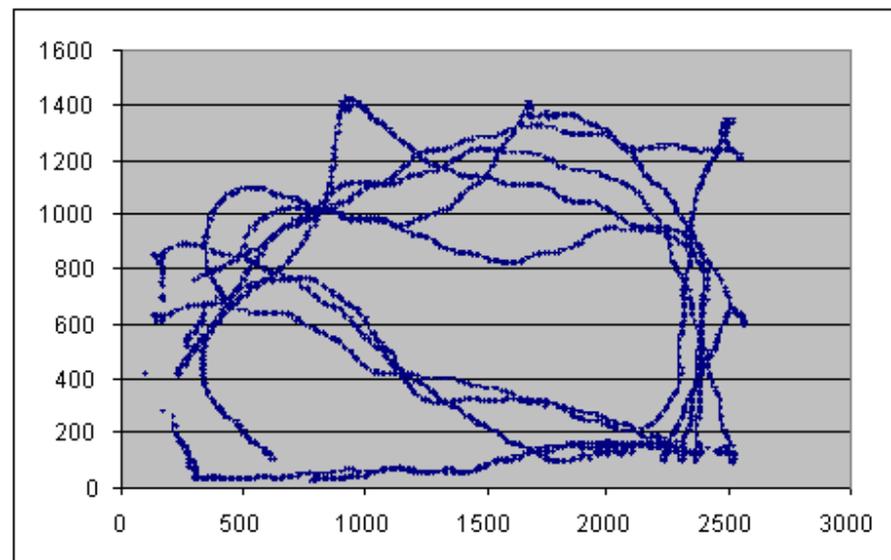


Figure 4.9: Plot of the robots path with an offset weighting of 5%.

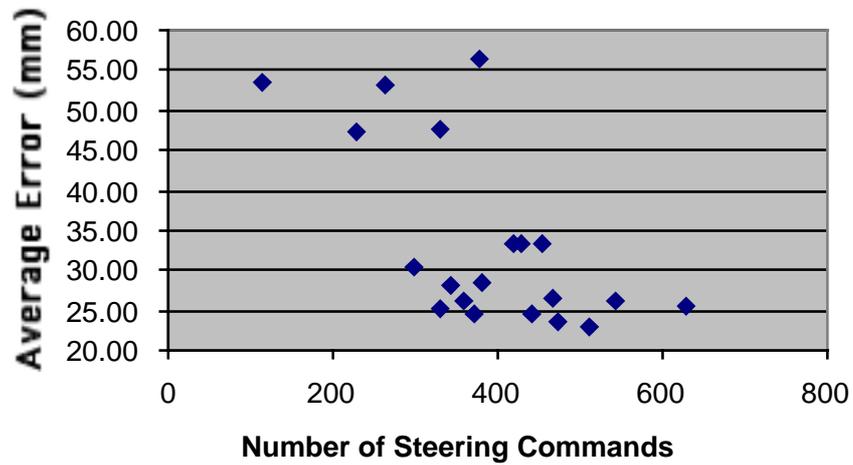


Figure 4.11: Graph of the average errors of the test runs against the number of steering commands issued.

From these results, it can be seen that the most important factor in the performance of the robot was the weighting on the offset information. The weighting on the gradient information seems less important, though more relevant than the speed of the robot.

4.3 Twenty Pixel Results

With the processing region at 20 pixels in height, the robot can see a

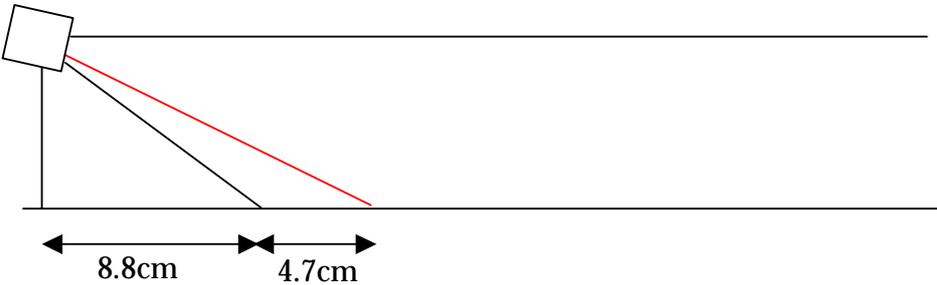


Figure 4.12: Diagram of the processing region size at 20 pixels.

region of the path approximately 4.7 centimetres in length, beginning at approximately 8.8 centimetres from the front bumper (see Figure 4.12). It was expected that in this configuration the gradient information would become more significant, due to the increased distances visible, and so the lowest error would occur when the gradient and offset weights were about equal

However, the results obtained were similar to those obtained for the ten pixel high processing region. A graph of the results (see Figure 4.13) shows an almost horizontal distribution of results when the offset weighting is greater than 0.2 (10%). Also contrary to expectations is the fact that the minimum error with a twenty pixel high region is greater than the minimum error for a ten pixel high region, with the greater amount of information available it was

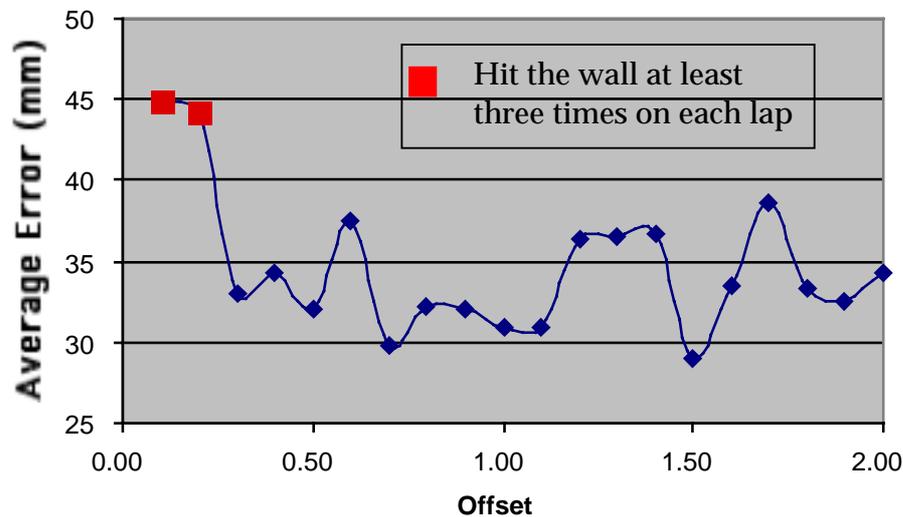


Figure 4.13: Graph of the average error against the offset weighting.

expected that the errors would decrease as the size of the processing region increased.

The plot of the robots path when the offset value is 1.5 (see Figure 4.14) is similar to the paths in Figures 4.4 and 4.5. Again, while the plot of the robots path when the offset weight is 0.1 (see Figure 4.15) bears little resemblance to the actual path, the measured error is relatively low.

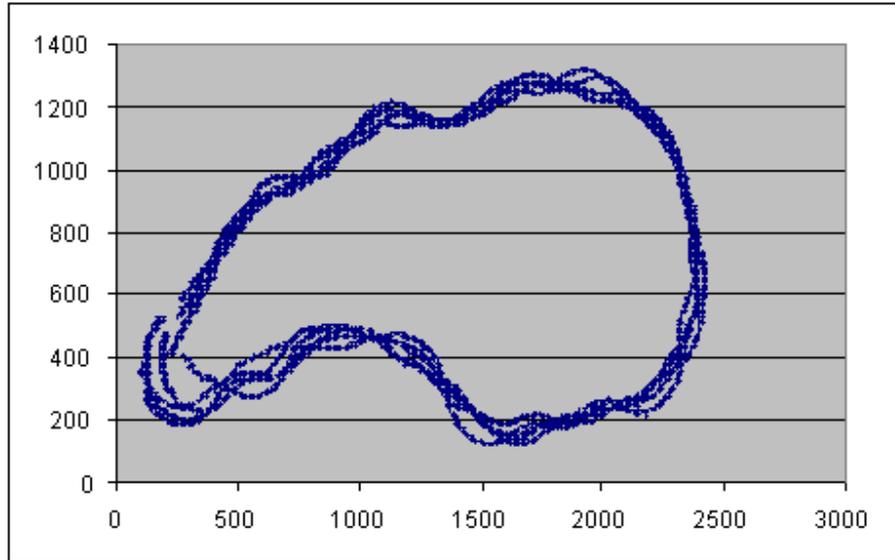


Figure 4.14: Plot of the robots path with an offset weighting of 75%.

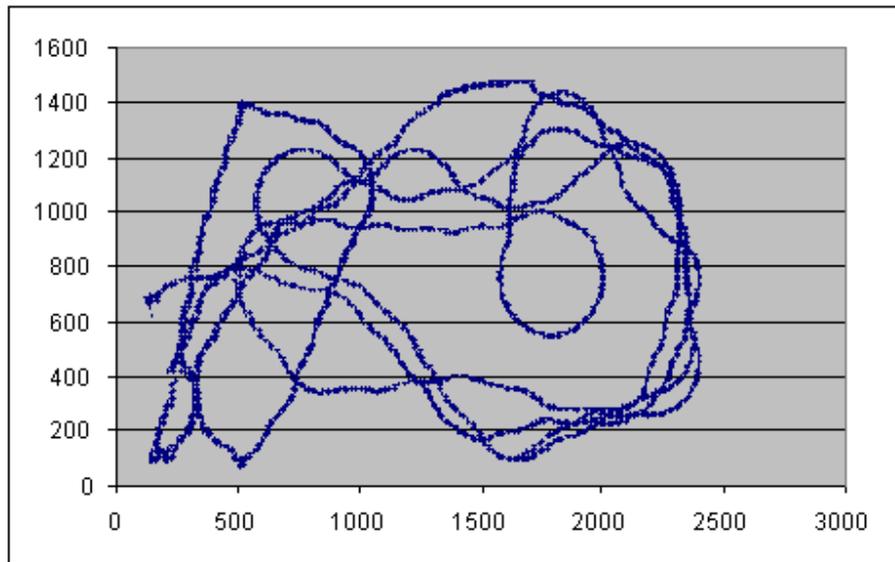


Figure 4.15: Plot of the robots path with an offset weighting of 5%.

As with the ten-pixel results, the average and maximum speeds of the robot have little or no effect on the errors (see Figure 4.16). Though, as with the ten-pixel case, contact with the walls can lower the average

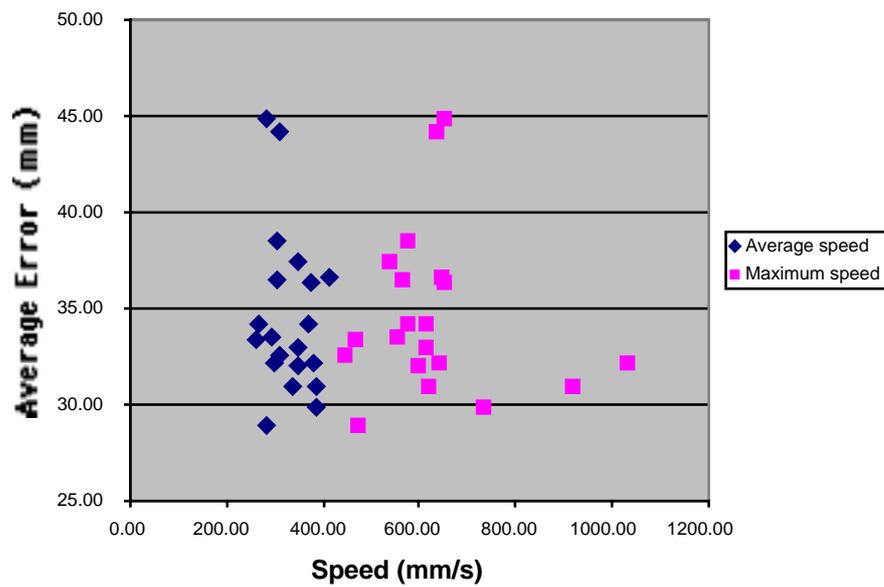


Figure 4.16: Graph of the average error s of the test runs against the speed of the robot.

speed value reducing its usefulness on test runs with high errors.

The relationship between the number of steering commands issued and the average error behaved as predicted, with the smallest errors associated with the highest number of commands (see Figure 4.17).

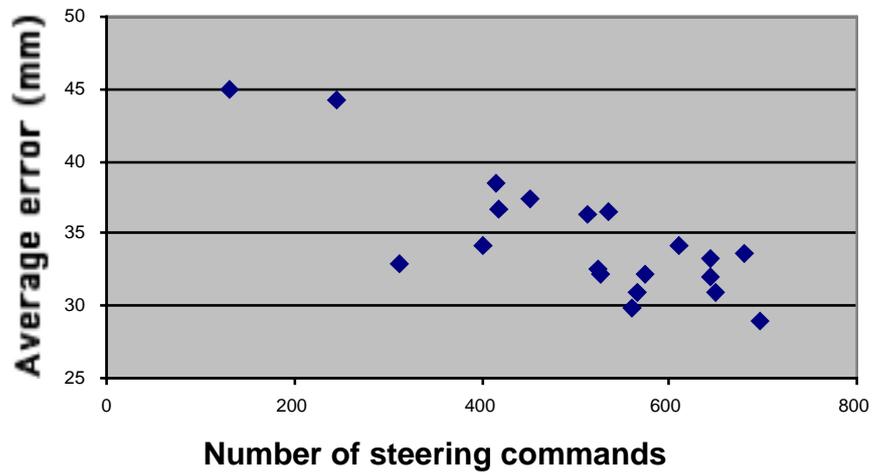


Figure 4.17: Graph of the average error s of the test runs against the number of steering commands issued.

4.4 Thirty Pixel Results

With the processing region at 30 pixels in height the robot can see a section of the path approximately 8.3 centimetres in length beginning at approximately 8.8 centimetres in front of the bumper (see Figure 4.18). With this size it was expected that the gradient information would be more important than the offset information.

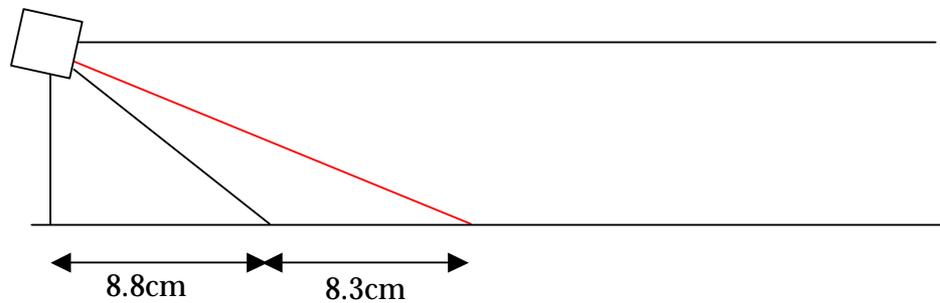


Figure 4.18: Diagram of the processing region size at 30 pixels.

The results obtained, however, were more similar to those expected for the twenty pixel high processing region (see Figure 4.19) with a minimum error at an offset weight of 1.7 (85%). This is also the only tested region size where a 100% offset weighting does not produce one of the more accurate results.

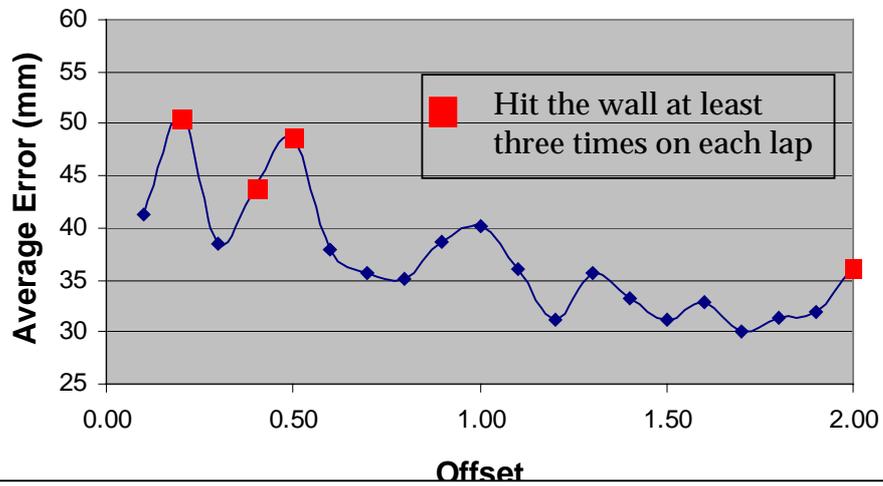


Figure 4.19: Graph of the average error against the offset weighting.

While the most accurate test runs produced plots similar to those of the ten and twenty pixel high processing regions (see Figures 4.4, 4.5, 4.11 and 4.20). The plot when the offset weight is 0.1 bears almost no resemblance to the actual path at all (see Figure 4.21) though the average error is quite low.

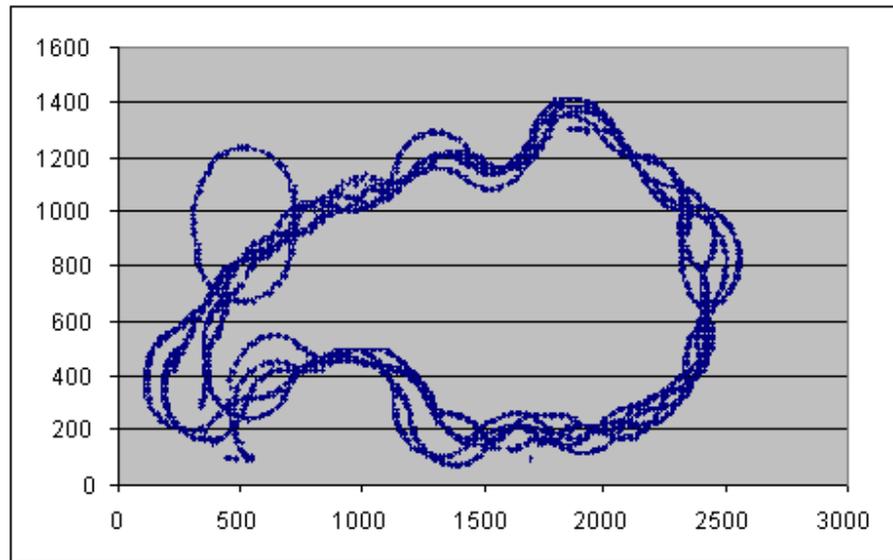


Figure 4.20: Plot of the robots path with an offset weighting of 85%.

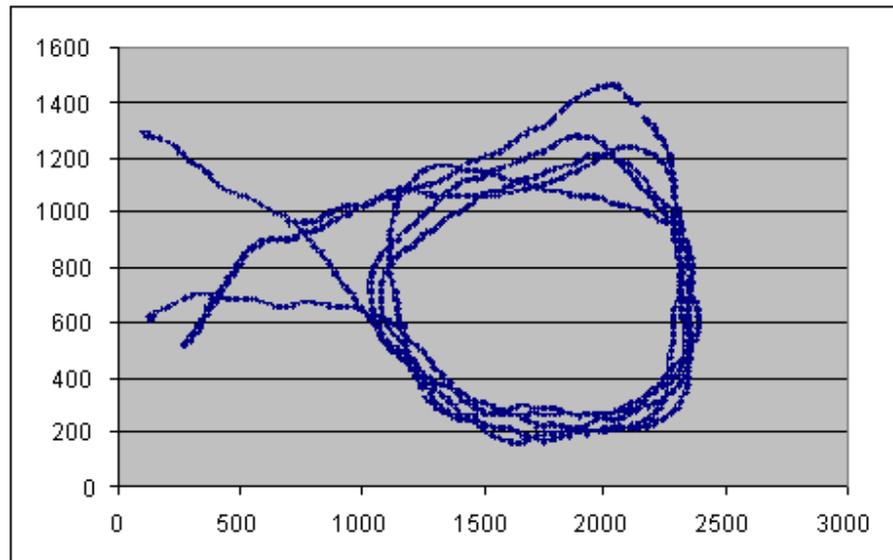


Figure 4.21: Plot of the robots path with an offset weighting of 5%.

As with the ten and twenty pixel high regions, there is no real correlation between the average or maximum speeds of the robot and the average error (see Figure 4.22). There is also no direct relation between the number of steering commands issued and the average error (see Figure 4.23).

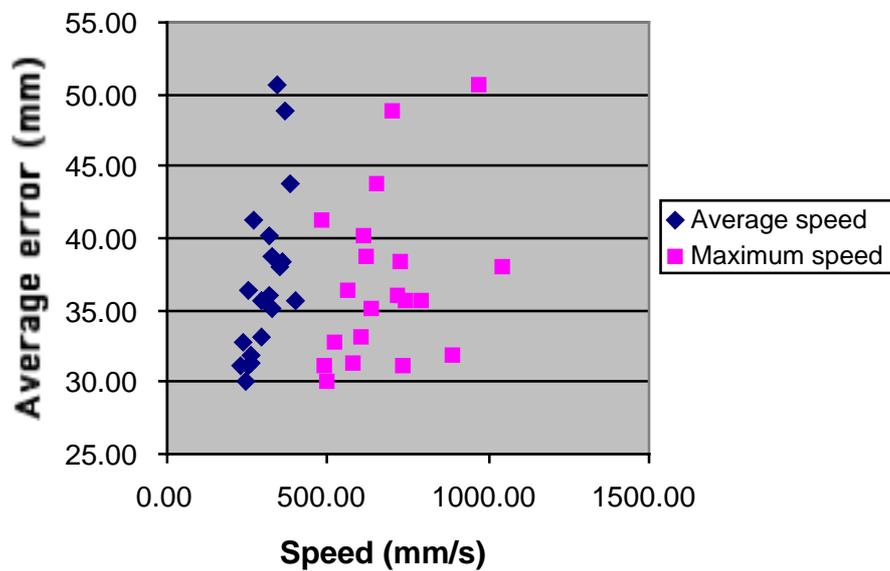


Figure 4.22: Graph of the average error s of the test runs against the speed of the robot.

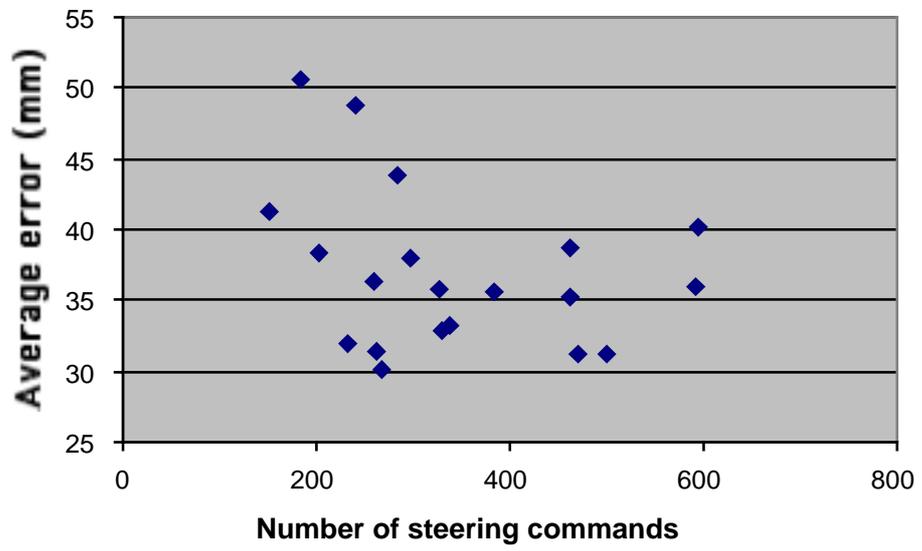


Figure 4.23: Graph of the average error s of the test runs against the number of steering commands issued.

CONCLUSION

Presented in this thesis is a simple vision based path following system. The classical approaches for mobile robot control systems have several drawbacks that make them unsuitable for use in industrial applications. These include the cost and bulk of the sensors and associated processing equipment and the time required to alter the control system if the tasks change.

The mobile robot control system can be split into a number of simple, mostly independent, behaviours. The most critical of these behaviours is the motion control subsystem. Our proposed subsystem is based on following paths marked on the floor of the work area.

This system can be made extremely simple by extracting the position and gradient of the visible section of the path and using that data to directly control the robots steering. As models of the environment and the robot are not necessary, this technique does not require any modification if the paths are altered and should be easily portable to other robotic platforms.

Experiments with this system produced some unexpected results:

Firstly, contrary to expectations the gradient of the path had little effect on the accuracy of the path following.

Secondly, the velocity of the robot also had little effect on the accuracy of the path following.

Finally, the best results were obtained with the smallest processing region size.

Future Work

The errors for several of the results were lowered by the robot hitting the walls surrounding the evaluation path. A greater distance between the path and the walls would enable more accurate error measurements, particularly if a higher robot speed were to be used.

It was discovered, by accident, that the robot was able to drive between two marked lines. These lines had to be reasonably close together, as the camera used did not have a wide field of view. It is possible that this technique could be used as a base for road following tasks.

BIBLIOGRAPHY

- [Brau1998] T. Braunl EyeBot homepage:
<http://www.ee.uwa.edu.au/~braunl/eyebot/>
- [CRS1999] CRS Robotics Corporation homepage:
<http://www.crsrobotics.com/>
- [HN1997] K. Hashimoto and T. Noritsugu. Visual Servoing of Nonholonomic Cart. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, pages 1719-1724, 1997.
- [KMMT1996] M. Kobayashi, Y. Miyamoto, K. Mitsuhashi and Y. Tanaka. A Method of Visual Servoing for Autonomous Vehicles. In *Proceedings of the 4th International Workshop on Advanced Motion Control*, pages 371-376, 1996.
- [KUKA2000] Kuka Robotics homepage:
<http://www.kukarobotics.com/>
- [LSL1999] J. LU, S. Sekhavat and C. Laugier. Fuzzy Variable-Structure Control for Nonholonomic Vehicle Path Tracking. In *Proceedings of the 1999 International Conference on Intelligent Transportation Systems*, pages 465-470, 1999
- [MKS1999] Y. Ma, J. Kosecká and S. Sastry. Vision Guided Navigation for a Nonholonomic Mobile Robot. In *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pages 521-536, 1999.
- [MMMM1994] Y. Masutani, M. Mikawa, N. Maru and F. Miyazaki. Visual Servoing for Nonholonomic Mobile Robots. In *Proceedings of the 1994 IEEE International Conference on Intelligent Robots and Systems*, pages 1133-1140, 1994.

- [OLC2000] S. Oh, J. Lee and D. Choi. A New Reinforcement Learning Architecture for Vision-Based Road Following. In *IEEE Transactions on Vehicular Technology*, Vol. 49, No.3, pages 997-1005, 2000.
- [Pom1992] D.Pomerlau. Progress in Neural Network-based Vision for Autonomous Driving. In *Proceedings of the 1992 Symposium on Intelligent Vehicles*, pages 391-396, 1992.
- [PW1993] F. Pin and Y. Watanabe. Driving A Car Using Reflexive Fuzzy Behaviours. In *IEEE International Conference on Fuzzy Systems*, pages 1425-1430, 1993.
- [RH1993] D. Raviv and M. Herman. Visual Servoing Using Relevant 2-D Image Cues. In *Proceedings of the 1993 Symposium on Intelligent Vehicles*, pages 473-480, 1993.
- [SB1998] G.Salgian and D. Ballard. Visual Routines for Autonomous Driving. In *Proceedings of the 6th International Conference on Computer Vision*, pages 876-882, 1998.
- [SN1992] H. Schneiderman and M. Nashman. Visual Processing for Autonomous Driving. In *Applications of Computer Vision*, pages 164-171, 1992.
- [WSGM1999] H. Weisser, P. Schulenberg, H. Göllinger and T. Michler. Autonomous Driving on Vehicle Test Tracks: Overview, Implementation and Vehicle Diagnostics. In *Proceedings of the IEEE Conference on Intelligent Vehicles '98, Vol. 2*, pages 62-67, 1999.