Real-Time Pipe Inspection Robot Prototype Development

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

Masters of Engineering

in

Mechatronics

at Massey University, Turitea Campus, Palmerston North

New Zealand

Matthieu Lincoln Jones

2014

Abstract

Concrete pipes are used throughout the world in many different industries to transport wastewater. These pipes are prone to erosion which can sometimes be severe, with a large cost of repair or replacement. This presents the need to inspect the pipes for erosion. Robots that are used to detect cracks and holes in a pipe already exist, but those that are used to inspect for erosion lack the ability to inspect in real-time allowing for high-speed, fully-autonomous inspection. Furthermore, none of these systems provide a stable platform that can traverse a severely eroded pipe while passively resisting rolling. A mechanical platform capable of doing just this was designed through a mathematical study. This concept was then tested by varying key design and environmental factors such as the leg angle, starting orientation, and payload weight and offset to determine the effect on the robot's movements and ability to resist rolling. It was found that the smaller the leg angle the less likely it was for the robot to roll but the more power was required to drive the robot. A leg angle of 20 degrees was found to be a good compromise between these two factors under varying conditions, although further study should be conducted over longer pipe lengths and real operating conditions. A real-time inspection system based on triangulation of a camera image and a laser line on the pipe surface was designed and optimized for implementation on a high speed FPGA. This was then tested and it was found that the inspection system was capable of accurately measuring erosion with a 0.3-0.9mm width resolution and a 0.2-0.6mm depth resolution for pipe diameters of 200-600mm. With a longitudinal resolution of 10mm this system could inspect at five metres per minute, and this could be doubled with suitable compression. This research provides the basis for developing an accurate, real-time pipe inspection robot. It also suggests an approach for developing a prototype capable of being used in varying diameter pipes consisting of an inspection system, an anti-rolling robot platform with position sensing, and a wireless communication system, with 3D result display software. Three research articles have been published from this research. Two of the articles are based on real time image acquisition and processing [1, 2]. The third article is on pipe robot mechanical system design [3].

Acknowledgments

I wish to thank my supervisors, Dr Liqiong Tang and Associate Professor Donald Bailey, for their help and assistance throughout this project, for guiding me through the process and for all the valuable feedback they have given me during this time.

I would also like to thank Arnold Yeoman of Evonik Peroxide Ltd. for the project specification provoking the initial investigation of this project and for the provision of useful resources during the initial stages of this project.

Finally, I would like to thank my family for their support. A special note of thanks to my mother Julianne Jones, for her help, and to my beautiful wife Amy-Jayne for her support, patience, and encouragement throughout this time.

Contents

Abstract i
Acknowledgments iii
List of Figures ix
List of Tables xi
List of Abbreviations xiii
Chapter 1 Introduction 1
1.1 Background 1
1.2 Research topic 3
1.3 Scope of research 4
1.4 Organisation of dissertation 4
Chapter 2 Literature Review 7
2.1 Pipe inspection robotic systems 7
2.1.1 Ground runner robots 7
2.1.2 Wall supported robots 7
2.1.3 Helical or screw robots 8
2.1.4 Inchworm robots 9
2.1.5 Walking leg robots 10
2.2 Measurement methods 10
2.2.1 Closed circuit television CCTV 10
2.2.2 Ultrasonic based systems 11
2.2.3 Thermography 11
2.2.4 Ground penetrating radar 11
2.2.5 Structured lighting 12
2.2.6 Mechanical feeler 12
Chapter 3 Proposed Pipe Profile Capturing System 15
3.1 Structured lighting working principle 15
3.2 CMOS and CCD sensors 17
3.3 Drive system 18
3.3.1 Motor considerations 18
3.3.2 DC motor control 21
3.3.3 Power source 22
3.4 Position sensors 24

3.5 Control & processing system 25
3.6 Communication system 27
3.7 Concept design 28
Chapter 4 Anti-Rolling Robot Platform Design 31
4.1 Anti-rolling mechanism design 31
4.2 Robot motor selection and battery calculations 39
Chapter 5 Image System 43
5.1 Setup 43
5.1.1 Measuring erosion 43
5.1.2 Configuration 1: Camera set vertically 46
5.1.3 Configuration 2: Laser set vertically 47
5.1.4 Configuration comparison 48
5.1.5 Mathematical model development 49
5.2 Algorithm 52
5.2.1 Algorithm development 52
5.2.2 High speed considerations 54
5.2.3 Algorithm implementation 55
5.3 Reference profile 59
5.3.1 Non-linear optimisation 59
5.3.2 Relationship and dependency testing 61
Chapter 6 Communication System and Software Development 63
6.1 Communication system 63
6.1.1 XBee module 63
6.1.2 Using RS232 for serial operation 65
6.2 Data compression techniques 68
6.3 Pipe profile reconstruction 69
Chapter 7 Results and Analysis 73
7.1 Anti-rolling mechanical system testing 73
7.2 Image system testing 76
Chapter 8 Discussion and Conclusions 83
References 89
Appendix A – C# 3D Erosion Display Application 97
C# Program (Program.cs) 97
3D Platform (Game1.cs) 97

GUI Form (Form1.cs) 104
Effects File (effects.fx) 107
Appendix B – Matlab Optimisation Code 113
Optimisation - Main 113
Optimisation – Function 114
Calibration Results Display 116
Relationship Testing 118
Determine X and Y values for pipe radius with no erosion 124
Erosion profile to depth – calculation and use of scale factors k1 and k2 125
Appendix C – FPGA Handle-C Code 127
Camera.hcc 127
Display.hcc 129
Filter.hch 132
Filter.hcc 132
Profile.hch 132
Profile.hcc 133
Peaks.hcc 139
RS232.hch 141
RS232.hcc 141
Appendix D - Anti-rolling Test Results 145
Appendix E – Image System Scale Factors 149
Appendix F – Battery Calculation Data 156

List of Figures

Figure 1.1 – Concrete wastewater pipe with water stains showing erosion concentration 1
Figure 1.2 – Concrete wastewater pipe showing extensive erosion with entire bottom of pipe eroded 1
Figure 2.1 – Ground runner robot [3] 7
Figure 2.2 – Wall supported robot [7] 8
Figure 2.3 – Screw type robot with screw pattern shown [10] 8
Figure 2.4 – Spring type helical robot [11] 9
Figure 2.5 – Inchworm robot with clamp and extension modules [12] 9
Figure 2.6 – Walking pipe robot with two-link legs [15] 10
Figure 3.1 – Structured lighting principle 16
Figure 3.2 – Laser line produced by shinning laser through cylindrical glass prism [25] 17
Figure 3.3 – Dot matrix pattern for determining object profile [26] 17
Figure 3.4 – Operation of DC motor [30] 19
Figure 3.5 – Operation of brushless DC motor [31] 20
Figure 3.6 – H-bridge showing forward and reverse control [32] 21
Figure 3.7 – H-bridge with fly-back diodes and pull up/down resistors [33] 22
Figure 3.8 – Comparison of different battery types (Wh/kg vs Wh/L) [35] 23
Figure 3.9 – FPGA hardware architecture and logic cell [44] 26
Figure 4.1 – Anti-rolling pipe robot concept 31
Figure 4.2 – Concept design of anti-rolling system 32
Figure 4.3 – $\boldsymbol{\theta}$ -Normal force plot for a unit mass 33
Figure 4.4 – Mechanical platform with payload offset 34
Figure 4.5 – Effect on offset distance (D) of leg angle ($\boldsymbol{\theta}$) and coefficient of friction (μ) 35
Figure 4.6 – Effect on offset distance (D) of pipe radius (R) and coefficient of friction (μ) 36
Figure 4.7 – Effect on offset distance (D) of leg angle ($\boldsymbol{\theta}$) for different levels of friction coefficient (μ) 37
Figure 4.8 – Effect on offset distance (D) of payload weight ($\boldsymbol{Wload}$) for different levels of friction coefficient (μ) 38
Figure 4.9 – Effect on offset distance (D) of pipe radius (R) for different levels of friction coefficient (μ) 39
Figure 4.10 – Normal force of tyre against pipe wall 40
Figure 4.11 – Mass/capacity comparison of lithium and lead acid batteries 42
Figure 5.1 – Coordinate definition 43
Figure 5.2 – Pinhole Model [57] 44
Figure 5.3 – Image coordinates definition [57] 45
Figure 5.4 – Two primary configurations of laser and camera 45
Figure 5.5 – Image taken using configuration 1 46
Figure 5.6 – Image taken using configuration 2 47
Figure 5.7 – Comparison of occlusions occurring in configuration 1 and 2 48
Figure 5.8 – New coordinate definition 49
Figure 5.9 – Camera Axes 51
Figure 5.10 – Captured image 53
Figure 5.11 – Reference profile (green) and laser stripe (red) 54
Figure 5.12 – Difference between reference and laser 54

Figure 5.13 – Terasic DE0 FPGA development board [58] with Terasic D5M camera [59] 55
Figure 5.14 – Processing logic for measuring pipe profile 56
Figure 5.15 – Bayer pattern [58] 57
Figure 5.16 – Red component 58
Figure 5.17 – Green component (shown as red for comparison) 58
Figure 5.18 – Red minus green 58
Figure 5.19 – Smoothing (red) and profile (green) 58
Figure 5.20 – Processing logic to flag occlusions and derive erosion profile 58
Figure 5.21 – Image data (black) along with curve generated from initial parameters (red) 60
Figure 5.22 – Image data (black) along with curve generated from optimised parameters (red) 61
Figure 6.1 – Basic ZigBee network topology [59] 63
Figure 6.2 – API packet structure [53] 64
Figure 6.3 – RS232 Procedure 65
Figure 6.4 – RS232 transmission [61] 67
Figure 6.5 – RS232 lines including hardware handshaking [62] 67
Figure 6.6 – Pipe profile reconstruction procedure 70
Figure 6.7 – Pipe erosion 2D colour map 70
Figure 6.8 – 3D profile of pipe showing height colour map 71
Figure 6.9 – Screenshot of 3D display application 72
Figure 7.1 – Rig developed for testing anti-rolling system 73
Figure 7.2 – Definition of anti-rolling test parameters 75
Figure 7.3 – Test rig used for testing image system 76
Figure 7.4 – Reference profile from test setup (red) 77
Figure 7.5 – Cross section with of example object with measurements 77
Figure 7.6 – Generated profile of above object (blue) 78
Figure 7.7 – Generated erosion profile of example object (green) 79
Figure 7.8 – Object used for testing 79
Figure 7.9 – Generated erosion map of pipe and object – colour map disabled 80
Figure 7.10 – Normalised erosion profile of test object (green) in mm 81

List of Tables

Table 3.1 – H-bridge switch positions and outcomes 21
Table 3.2 – Table showing common XBee modules and specifications 28
Table 5.1 – Initial parameters and optimised parameters along with total error squared for each 61
Table 7.1 – Anti-rolling test results 74
Table 7.2 – Real measurements vs profile calculated measurements 81

List of Abbreviations

AC	Alternating current
BLDC	Brushless DC
CCD	Charge-coupled device
CCTV	Closed circuit television
CMOS	Complementary metal-oxide-semiconductor
DC	Direct current
FPGA	Field programmable gate array
IC	Integrated circuit
IMU	Inertial measurement unit
MOS	Metal-oxide-semiconductor
MOSFET	Metal-oxide semiconductor field effect transistor
PAN	Personal area network
PCB	Printed circuit board
PWM	Pulse width modulation
RLE	Run length encoding
UART	Universal asynchronous receiver/transmitter
VGA	Video graphics array

Chapter 1 Introduction

1.1 Background

Concrete pipes are used throughout the world as a medium for carrying wastewater for many reasons such as their relatively low cost, robustness, and ease of installation. Often these pipes last for years with little to no maintenance or repair, however, this may not always be the case as concrete is susceptible to acidic erosion. In many applications this is not a concern but many industrial plants have a small amount of acid that may leach into or run off with the wastewater. The introduction of acid can cause the cement compounds to dissolve in the pipes when the pH of the wastewater is below 6.5. If this is left unchecked it can lead to severe cases of erosion which may damage the structural integrity of the pipe or lead to the contents of the pipe leaching into the surroundings and ultimately into the water table which can be a major environmental concern. In general, these wastewater pipes are not usually run full, which means that the majority of the erosion is concentrated on the bottom third of the pipe. In some cases the erosion can be so severe that the entire bottom of the pipe has been eroded exposing the wastewater directly to the environment.

Figure 1.1 – Concrete wastewater pipe with water stains showing erosion concentration

Figure 1.2 – Concrete wastewater pipe showing extensive erosion with entire bottom of pipe eroded

Once erosion is found there are many different approaches that can be taken in order to alleviate the problem. One solution is to simply replace the concrete pipes with new ones. This is often expensive as it requires extensive excavation to access the pipes as well as the cost of the replacement pipes. In such cases it is common to replace the pipes with alternative products, such as inserting plastic pipes within the concrete ones or relining the concrete pipes often with acid resistive lining. It is also possible to simply lay new pipes and discontinue the use of the old ones which may not require quite so much excavation. All of these methods, however, have their relative advantages and disadvantages including the cost, the complexity of the installation, the available sizes of pipes, the life expectancy of the pipes, and their effectiveness of retaining acidic wastewater under different operating environments. Generally the method of repair chosen will depend on the actual extent of the erosion in the pipe and the rate of erosion. Therefore, pipe inspection and erosion evaluation is necessary before any decisions are made about the future of the wastewater systems.

Unfortunately, inspection of the pipes is often difficult as the pipes are generally buried underground and are difficult to access. This type of inspection can be facilitated by the use of a robot platform that traverses the pipe while performing an inspection. There are many different designs for pipe robots that are currently being used such as bottom crawlers, wall supported robots, helical or screw robots, inchworm robots or walking leg robots. The bottom crawler design is very common for many different robots as they drive along on a surface much like a car with wheels or tracks. In situations where there is severe erosion in a pipe these robots may not be able to navigate through the pipe because of the obstacles in their path. Even if they could navigate these difficult obstacles it would be virtually impossible for the robot to maintain a constant frame of reference for the inspection system. Helical robots have a similar problem as they require contact with the pipe throughout a full 360 degrees so any severe erosion may prove difficult. Walking robots may be prone to getting their legs stuck in cavities created by the erosion and inchworm robots are very slow with jerky movements so are not suitable for a high speed inspection system. In contrast, a wall supported pipe robot could be designed such that the wheels contact the pipe above the main area of erosion in the pipe allowing them to traverse the pipe without issue. Wheeled robots also tend to be one of the fastest because of the low rolling resistance. The main difficulty with designing a wall supported robot is to provide a means that will prevent them from rolling as they travel down the pipe as this can cause two main problems: if the robot rolls the wheels may enter the area of severe erosion causing it to get stuck; also, if the robot rolls, then there is not a consistent frame of reference for the inspection system making it difficult to provide accurate information on the erosion in the pipe. Active measures can be taken to detect and counteract any roll but this complicates the design and calibration of the inspection system. An alternative is to develop a passive system capable of resisting this roll to provide a stable platform for a pipe inspection system.

There are many different inspection methods being proposed for inspecting pipes including CCTV inspection, ground penetrating radar, ultrasonic sensors, laser transducers, or infra-red thermography. Although these are all different techniques for measuring the erosion the data transfer process used is very similar. The pipe inspection system generally consists of a robot platform performing the inspection and either transferring the data back to a host PC by use of an umbilical cable or storing the data locally until the end of the inspection run then transferring the data to a PC. Some modern systems incorporate wireless transmission but the large volume of data requires a large bandwidth and can make the inspection slow. Regardless of the transmission medium the data is processed on the PC to extract the relevant information. Data processing often requires a large

amount of human interaction to determine the extent of the erosion and also has to be done offline. Some systems such as the KANTARO [1] implement smart systems capable of performing a limited amount of processing in real-time to detect features such as cracks. This allows the robot to reduce the amount of data that needs to be captured and processed. However, this does not provide any advantage when the entire pipe is likely to be eroded as the whole pipe needs to be processed.

Developing a pipe inspection robot that could inspect a pipe in real-time without any human interaction would generate large benefits to the speed and cost of inspection. It would also create the possibility for an operator to re-inspect a section of pipe while the robot is still in the vicinity instead of having to perform another inspection run. It would also allow the robot to regulate its own inspection resolution based on the extent of erosion detected to improve the inspection speed. To be able to inspect a pipe in real-time a processor must be used that is capable of processing the large amount of data in a short amount of time. This is traditionally done on a PC but this reduces the flexibility of the system. The transmission bandwidth also dictates the inspection speed due to the large volume of data that needs to be transmitted and this is especially problematic with wireless communications. For embedded real-time processing a low power, high speed processor such as an FPGA is required with a processing algorithm optimised for the inspection system.

This research designed and tested a stable robot platform using passive measures to resist rolling as it travels through a pipe. A real-time inspection system based on triangulation of a laser and camera was also designed, and a high speed processing algorithm was developed for implementation on an FPGA embeded processor. This system was tested and the results proved the validity of this system for the given application. A 3D display application was also developed for viewing the results. This research provides the basis, and suggests the design and implementation, for developing a fully-autonomous pipe inspection robot prototype capable of providing a fast and accurate inspection of erosion in real-time and conveying these results to a user.

1.2 Research topic

The aim of this research was to develop a robotic platform capable of performing an inspection on a pipe to determine the extent of any erosion present. This system would comprise a sensor for measuring the erosion, an autonomous platform for traversing the pipe, a processing unit and algorithm designed to process the data and provide information on the erosion, and a PC interface including a communication methodology and a user interface. The objectives were:

- Design a hardware device that is capable of obtaining an erosion profile accurate enough to give a close approximation of the volume of the erosion for the purpose of determining how much re-lining agent is required, and hence, the cost of this approach. Based on the requirements provided by the relevant industrial company the lateral resolution across the pipe had to be at least one measurement every one to two millimetres and depth resolution must be within one millimetre. However, it was sufficient to take one profile every five to twenty millimetres;
- Develop a methodology that was able to capture the extent of the pipe erosion in real-time and meet the industrial requirements on wastewater pipe inspection accuracy;
- Design a stable mechanical platform that allowed the measurement system to traverse the pipe and obtain meaningful information of the location of the erosion as well as the extent;

- Design a high-speed processing system that could process the data obtained in real-time and provide quantitative data on the level of erosion in the pipe eliminating the need for offline human interaction.

1.3 Scope of research

Based on the research topics and the consideration of the project time frame, the scope of this research was limited to:

1) Investigate and develop a feasible method that is able to capture the internal profile of a wastewater pipe and use the profile data to evaluate the pipe erosion;
2) Design a prototype pipe inspection robotic mechanism that was able to provide a stable platform for pipe inspection devices to accurately locate the erosion position and orientation along the pipe circumference;
3) Develop a high speed data processing system that was suitable for pipe inspection and real-time data acquisition and processing.

1.4 Organisation of dissertation

This dissertation contains eight chapters and six appendices. The development of the pipe inspection methodology, data processing unit, robot mechanical system, control and communication system, and the software design is presented in detail in the main chapters. The appendices include supporting information such as the developed code, test results, image system parameters, and calculation data.

Chapter 1 provides the background to the research along with the aims, objectives, and scope of the research.

Chapter 2 provides a review of the current systems focussing on the measurement method and the robot platform used to transport the inspection system.

Chapter 3 explores different design aspects for each of the key components required to develop a working system including the measurement system, the drive system, position sensing, the control and processing system, and the communication system.

Chapter 4 discusses the development of the mechanical system to provide a stable platform capable of traversing the pipe in order to facilitate accurate results to be obtained.

Chapter 5 focusses on the entire image system. This includes the setup and acquisition of the data as well as the necessary processing. An algorithm for processing the data in real-time is developed and implemented on an FPGA and a method of calibration is also developed.

Chapter 6 is primarily concerned with the communication between the processing unit and a PC as well as the PC software developed for the user interface and to reconstruct and present the results.

Chapter 7 provides details on the integration of the main systems to perform testing of the prototype and these results are presented and an analysis discussed.

Chapter 8 concludes the main section of this dissertation with a discussion on the research and the conclusion drawn from this.

Appendix A contains the code developed in C# for the reconstruction and display of the results.

Appendix B presents the code developed in Matlab for the mathematical modelling and optimisation of the image system, including the code used to generate the scale factors for the image coordinates.

Appendix C contains the Handle-C code developed for the Terasic DE0 FPGA. This includes the code to capture data from the camera, the code to display the data via a VGA interface, the code to process the data including the smoothing, and peak detection, and the code to transmit the data via the on-board RS232.

Appendix D holds the anti-rolling test results.

Appendix E is the scale factors that are applied to the erosion profile from the calibrated image system in order to determine the actual level of erosion in mm.

Appendix F details the battery and motor calculations and the data used for the calculations.

Chapter 2 Literature Review

2.1 Pipe inspection robotic systems

2.1.1 Ground runner robots

There are many different configurations that can be used for in-pipe robots. The simplest of these is one that is commonly used for many different types of robots where wheels or tracks are used to drive along the bottom of the pipe (see Figure 2.1 – Ground runner robot Figure 2.1) such as [2]. These operate much the same as a car or similar vehicle. The problem with these is that the concrete which makes up the pipe walls is largely made up of cement compounds and stones. When the cement compounds have completely dissolved these stones are left on the bottom of the pipe. These, along with other stones brought to the surface and other debris such as the exposed steel reinforcements used in the concrete can make the bottom surface very rough and difficult for a robot to navigate. This also makes it hard for the robot to maintain a stable frame of reference for the inspection system.

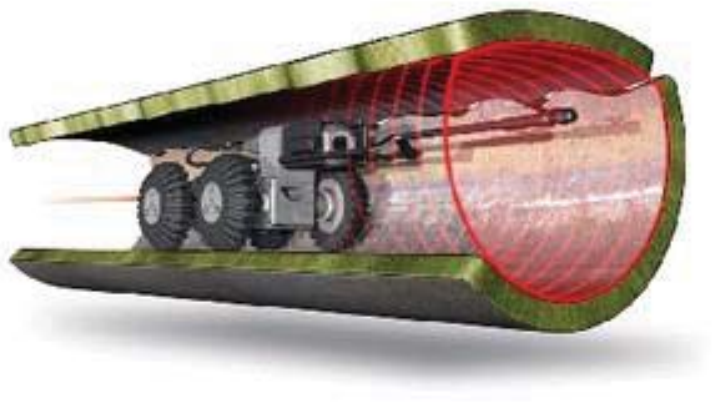


Figure 2.1 – Ground runner robot [3]

2.1.2 Wall supported robots

One way to overcome the difficulties of a common ground runner is to have the wheels or tracks supported on the pipe wall such as [4, 5, 1]. If positioned appropriately the wheels or tracks would avoid the majority of the erosion that is likely to be present. If this is of particular concern then tracks can be used to ensure that the robot does not get stuck in any holes. These robots have the possibility of twisting as they travel down the pipe but there are two ways to counteract this. The first is to use an active system such as the KANTARO [1] robot which uses an inclination sensor to measure the tilt and counteract the tilt using an actuator. The KANTARO robot drives near the bottom of the pipe, however, so it is very prone to being bumped around on any rough surfaces or rocks in the bottom of the pipe. The alternative is to design the robot in such a way that it resists this rotation in the first place. This passive measure reduces the need for active countermeasures. Some of these wall-supported style of robots also have suspension systems so that they can easily negotiate changes in pipe diameter and other irregularities [6]. This configuration is one of the simplest of the in-pipe robots and also one of the most efficient. It is also suitable for carrying heavy payloads such as large batteries for extended run times. These attributes make it ideal for long range inspections [6].

Figure 2.2 – Wall supported robot [7]

2.1.3 Helical or screw robots

The screw type robots propel themselves by rotating a set of angled wheels about the axis corresponding to the direction of propulsion which causes the wheels to move in a screw pattern as shown in Figure 2.3 [8]. This type of robot is ideal for vertical applications as they pull themselves through the pipe and the three or more point contact prevents the robot from falling. Some robots such as [9] have less than three points of contact but these are not designed for vertical use. The wheels are generally sprung loaded so that the robot can deal with differing diameters, however, they do not work particularly well when there is sudden erosion or when the erosion is too deep as it requires contact with the pipe through 360 degrees.

Figure 2.3 – Screw type robot with screw pattern shown [10]

Another type of helical pipe robot has a coil-spring like mechanism with wheels located around the periphery of the coil [11]. This type of design offers greater flexibility with the same advantages of a standard helical robot but the downside is that the complexity of the design does make it prone to getting stuck in some situations [6].

Figure 2.4 – Spring type helical robot [11]

2.1.4 Inchworm robots

Inchworm type robots generally consist of two clamp sections and an extension system [12]. They manoeuvre their way along a pipe by repeatedly clamping the rear section against the pipe wall, extending the front section forward, clamping this, and then retracting the back section towards the front. An example of this is shown in Figure 2.5.

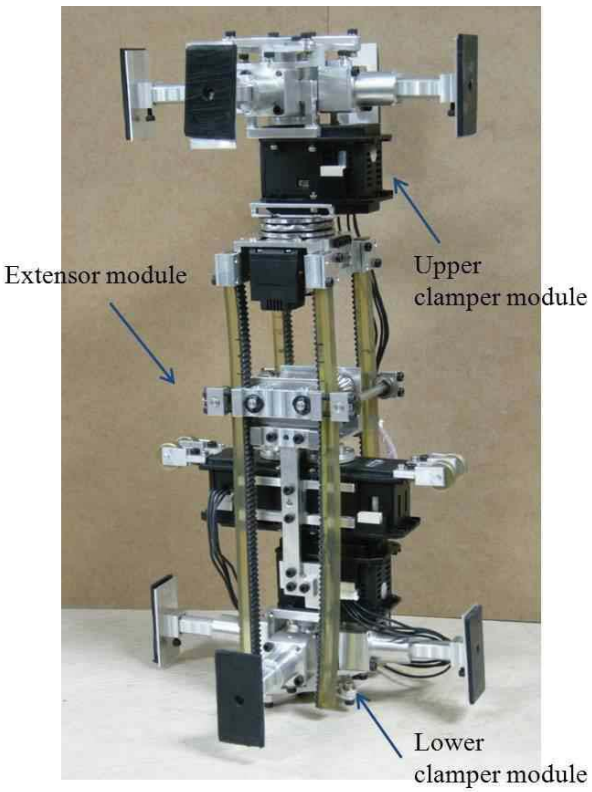


Figure 2.5 – Inchworm robot with clamp and extension modules [12]

This structure means that they can easily deal with changes in pipe diameter and they are also very effective for vertical pipes. Several robots based on this configuration have also been developed to utilise compressed air [13, 14] instead of electric motors which are used in most robots. This is particularly useful in hazardous areas where sparks from electric motors could be fatal such as in gas

pipes. However, they do not cope well with bends as they are linear in nature and their movement can be very disjointed and slow due to the clamping and unclamping cycles.

2.1.5 Walking leg robots

Walking pipe robots generally consist of two sets of four legs attached to the main body of the robot with revolving joints known as the hip joint [15]. These legs have an additional joint resembling a knee joint making them a two-link leg. Moving these two joints in different combinations allow the robot to walk much the same as any other creature walks. The multiple sets of legs provide stability within the pipe and allow them to walk through any inclination – even vertical pipes – without slipping. The low contact area causes little damage to the inside of the pipe but the mechanism to facilitate the legs is complex and they can be prone to their legs becoming stuck in holes [6].

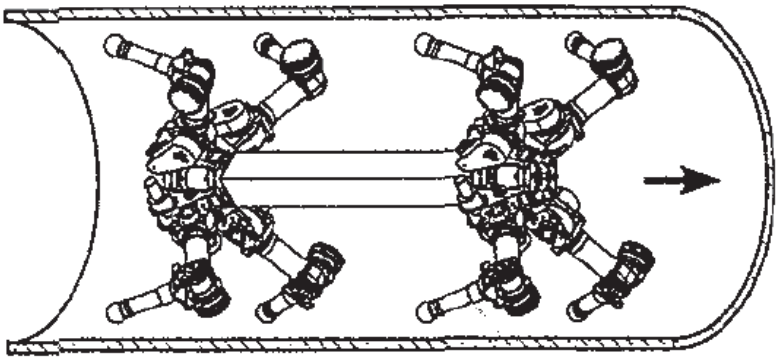


Figure 2.6 – Walking pipe robot with two-link legs [15]

2.2 Measurement methods

2.2.1 Closed circuit television CCTV

There are many different methods that could be used for measuring the erosion in a pipe. One method that has been used extensively in earlier systems as well as some more recent ones is CCTV (Closed Circuit Television) [16, 1]. A platform travels down the pipe with a camera and light source attached to it which is used to record footage of the pipe interior. The footage can be very useful as it provides a large amount of detail and allows a user to see the inside view of the pipe and its condition. This requires human interpretation of the entire footage. This has several advantages over computerised interpretation such as humans are more able to make inferences in order to identify foreign objects and possibly even cracks using less information than a computer and they can also deal with unexpected circumstances far better. However, it also has many disadvantages such as it requires a large amount of time (approximately three hours of footage for every kilometre of pipe) to go through all of the footage, it is prone to human error, and there is a classification of the level of erosion made subject to the individual assessor instead of an objective analysis providing quantitative data [16].

The disadvantages of human interpretation has led to CCTV inspection becoming less common, and, in recent years, efforts have been made to develop systems using many different sensing methods along with computer processing and analysis to combat this. Some of the sensing methods used include ultrasonic sensors, infra-red thermography, ground penetrating radar, and laser based systems [16].

2.2.2 Ultrasonic based systems

Ultrasonic based systems [16, 17, 18] basically use a transducer to send an ultrasonic (high frequency sound wave) pulse out and measure the signal once it has bounced back. The elapsed time between emission and reception [16] can be used to calculate the distance travelled by making use of the speed of sound which is usually known. The amplitude of the returned signal can also be used to give useful information on the surface of the point inspected. Multiple points around the pipe can be measured by rotating the transducer and taking multiple readings. This method can be very useful for inspecting pipes and many different systems have been developed based on this. The main disadvantages, however, are that there is a large amount of data generated, the different materials within the pipe (such as concrete, exposed steel reinforcement rods, stones, soil, and other foreign objects) with non-homogeneous properties may affect the measurements, rough surfaces which commonly occur when there is erosion may cause coupling issues, further problems associated with wave generation and producing guided waves, and finally the speed of the inspection is a trade-off with resolution as the rotation speed must decrease as more points are measured. Common systems can take 50 measurements with a rotating speed of 120rpm [16]. This means that each measurement takes approximately 0.01s.

2.2.3 Thermography

Thermography works by measuring the temperature differences across objects. A passive heat source can be used where the sun is used (during the day) as an energy source to heat the ground above the pipes or vice versa (for night inspections) where the ground is used as the heat source and sky as a heat sink. Alternatively, an active heat source can be used where infrared tube lights are used to heat the pipe. In either case an infrared scanner is used to measure the spatial distribution of the heat and how this changes over time. This can often be done from outside the pipe allowing large areas to be covered at one time. Different techniques can be used to improve the feature detection resolution as well as minimising sensitivity to illumination. Large areas can also be inspected in relatively short periods and the inspection is an area testing technique as opposed to point test techniques which require measuring separate points to obtain information on the area. The type of material being tested does not affect the measurements and this technique is superior for detecting surface defects such as deteriorated pipeline insulation, leaks, voids, or cracks. However, there is a large amount of data generated using this method which may or may not require human interpretation as well as sophisticated processing to analyse [16].

2.2.4 Ground penetrating radar

Ground penetrating radar (GPR) transmits pulses of high frequency electromagnetic waves through an antenna, typically in the range of 100MHz to 1GHz [16]and measures the reflections using a receiver (which can be a separate antenna or the same one used for transmitting). This process works much the same as for the ultrasonic technique and the scan is carried out in much the same way. However, with GPR the scan can be performed from inside or outside the pipe and information about the pipe surroundings can also be gathered if necessary. More than one antenna can be used to give detail of the surroundings, the structure of the pipe and the pipe-surrounding interface, and the pipe detail with increased precision operating on differing frequencies. The actual choice of frequency is a trade-off between resolution and penetration depth. The relative permittivity of concrete is very close to that of air so it can be difficult to identify cracks, but other detail such as wet soil, obstacles, or metal pieces are easily detected. GPR is often chosen because of its high inspection speeds when

compared with other methods and the fact that the antenna does not need to be in contact with the pipe surface and does not even have to be within the pipe [16, 19].

2.2.5 Structured lighting

Structured lighting approaches use a light source projecting a structured light pattern onto the surface of an object and a light sensitive sensor which detects the position of the light and uses this to measure the physical characteristics of an object using triangulation [20, 21]. Since the precision of the generated pattern is very important it is common to use lasers as the light source instead of standard lighting systems and this is called laser profilometry. The structured light pattern can be anything from a single dot or an array of dots to a solid line or other such patterns and the pattern used will depend on the application [21]. There are many different configurations for using laser based systems but one of the simplest is to have a laser diode and a light sensor on the same optical path and rotate these around the circumference of the pipe in much the same way as the ultrasonic method. This is called a single spot method and uses a single laser dot projected on the surface. An alternative method is to use another pattern such as a laser line shone around the circumference of the pipe and a camera to image the entire circle at once [22, 20, 16]. Common CCTV systems can also be modified to be used for this purpose [23, 20]. This method has the advantage that it is quicker to acquire all of the data points but it also tends to be less accurate and requires more complicated calibration and calculations to determine the pipe profile. Using laser based systems has proved to be efficient for pipe inspection as they provide an accurate and relatively fast inspection (depending on the methodology). It also has the advantage that the system can be extended to use the camera to give valuable CCTV footage of the pipe allowing for greater versatility.

The KANTARO [1] robot makes use of a smart system consisting of a laser and a smart camera for on-board processing. However, this processing is limited to landmark or feature detection in order to increase the inspection speed and decrease the required data storage. When a landmark is detected then the inspection system records the type of landmark such as a crack, or hole as well as the images corresponding to this area for further processing. This would not provide substantial benefit when the entire length of pipe is likely to be eroded and is better suited to applications of infrequent defects.

2.2.6 Mechanical feeler

Most pipe inspection robots deploy non-contact inspection methods, whereas the mechanical feeler method relies on contact with the inspection surface to obtain its data. For this method a mechanical probe or feeler is extended towards the eroded surface. A sensor then measures the distance that that probe extends and this gives the inspection profile which can then be used to determine the level of erosion. The robot has to stop for each sample so that the feeler are not dragged along the surface when they are extended. This can make the inspection very slow and jerky with the robot having to start and stop all the time. The main advantage of this method is that it uses a very simple form of inspection to give an absolute position for the location of the erosion at that point. However, while the inspection principle is simple, the actual mechanism can become very complex with lots of moving parts and components that all have to be carefully maintained and calibrated. Furthermore, in cases where there is severe erosion it may be necessary to completely lift the feeler array out of the way to avoid obstructions in the pipe. This further increases the complexity of the design and would depend on the stroke of the feelers. With a longer stroke the feelers may be able to get out of the way but this will generally decrease the resolution of the sensor measurements. Because of the physical size of each inspection element the special resolution is

likely to be quite poor. This can lead to local extrema being measured and may not be a good representation of the surrounding erosion. This type of measurement can be very useful, however, when coupled with a non-contact method of inspection. This allows the erosion profile to be verified and can be very useful when the alternative inspection has blind spots.

Chapter 3 Proposed Pipe Profile Capturing System

A pipe inspection robot must contain an inspection method for measuring the erosion in the pipe. In addition, it is necessary to have some form of a drive system to move the robot through the pipe. To improve the usefulness of the information gained it is also necessary to have a position-sensing element to provide feedback on the location of the erosion and the data gained from this, along with the erosion level data, which must either be sent via appropriate communication to a base station or be stored locally. The selection of these components may greatly affect the ability of the robot to meet the demands of the application.

For this application it is imperative to select an inspection method and a processing system that is capable of providing real-time acquisition and processing of the erosion profile. It would also be beneficial to reduce the volume of data so that this information can be transmitted while inspecting for a user to view in real-time. It is also important to choose the components with the goal of increasing the processing accuracy and speed while reducing the overall cost of the system. The size and complexity of the robot will play a large part in adaptability of the robot for use in different pipes and scenarios so it is important to reduce these where possible. Finally, the run time of such a system directly affects the usefulness and flexibility of the system so it is beneficial to increase this as much as possible without sacrificing other design criteria.

Thermography is not suitable for this type of inspection as it is an area inspection method and would therefore be slow and cumbersome for continuous inspection. Furthermore, it would be difficult to use this type of inspection to evaluate the erosion inside a pipe. Similarly, GPR is less than ideal due to the difficulties involved in selecting a balance between penetration depth and resolution for detecting the erosion from outside the pipe. One of the most common approaches is using structured lighting as this has many advantages over other methods such as CCTV, ultrasonic, and mechanical feeler methods. The data produced can be processed autonomously to evaluate the erosion as well as to reduce the data volume for storage or transmission and since a camera is used it can also double as a CCTV inspection providing a video feed along with the processed data. Depending on the image processing, a structured light approach can be relatively fast while still providing a high level of detail. It is a non-contact method and is easy to implement for an in-pipe inspection. For these reasons it was chosen as the best inspection method for this application.

When inspecting the bottom of a pipe using structured lighting it is important to have a stable frame of reference and this would be difficult to provide using a helical robot as it is constantly rotating. A helical robot may also have difficulty navigating through sections of pipe with severe erosion such as when the entire bottom of the pipe is missing. A wheeled approach provides a much smoother motion than either inchworm or walking robots and it also tends to be a lot faster with less stopping and starting. This makes it the preferred method for this application since the robot does not need to navigate any large inclines or vertical climbs.

3.1 Structured lighting working principle

Structured lighting is a well-known technique for measuring the height of an object and works on the basic principle of triangulation where the position of two nodes is known and is used to find the position of the third node. The sensor, or camera, is used to measure the position of the laser in the image. This can be used to calculate the height of the object by first determining the offset of the position from the known position without the object present. This gives a measurement in pixels which

must be converted into real-world measurements and scaled to compensate for any distortions introduced in the system depending on the set-up and operating parameters. Figure 3.1 shows how this principle works.

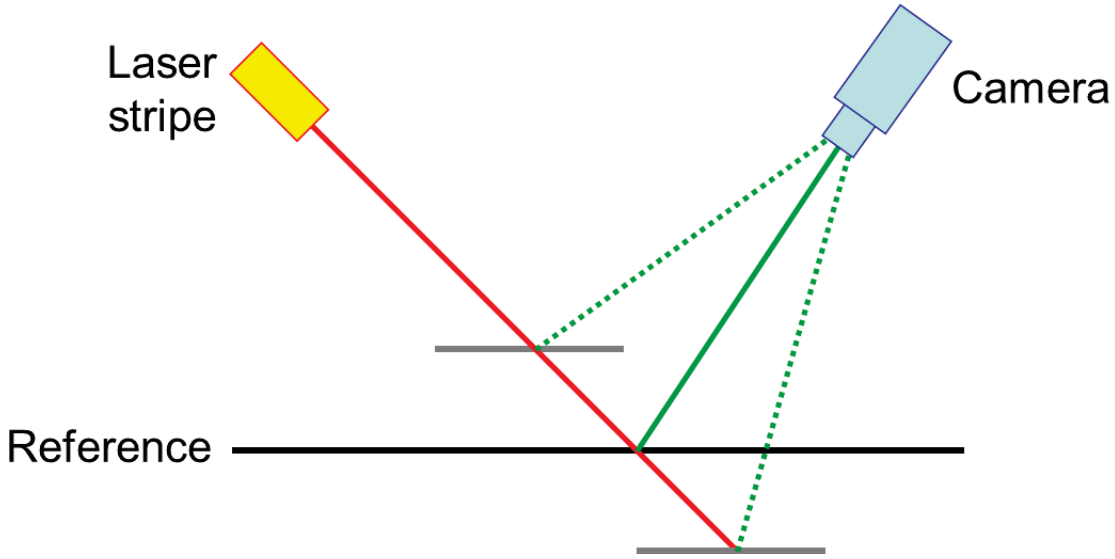


Figure 3.1 – Structured lighting principle

The light source projects a pattern onto the scene as shown in red. If there is no object present the position of the light in the image will be along the solid green line and appear at a given position in the image. This is the reference position. When the object is present the camera no longer sees the light along the same path. Instead, it sees it along the top-most dotted green path which corresponds to a different position within the image. The difference between these two positions depends on the height of the object. In the case of erosion, it will be the same except the height will be inverted – the position in the image will shift down instead of up and will correspond to the bottom-most dotted green line

Many different light patterns can be used such as a dot matrix or even a simple line, depending on the information required. A single dot is not very useful unless its position is constantly changed to scan an entire cross section of pipe. Unless a rotating single spot scanning method is used [16] it is better to use a line or circle projected around the circumference of the pipe. This can be easily accomplished by refracting a laser through a cylindrical prism as shown in Figure 3.2. Other patterns, such as a dot matrix [24] or checker board pattern, can be used to yield greater detail such as in Figure 3.3, but this is not necessary for the purpose of scanning a pipe for erosion as consecutive profiles will be measured along the length of the pipe. Only the height across the pipe laterally is of interest for each individual cross-sectional profile so it is sufficient to use a simple laser line.

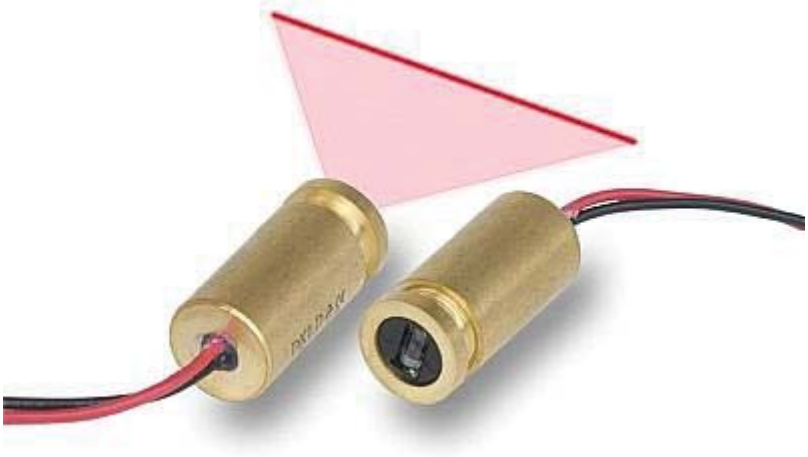


Figure 3.2 – Laser line produced by shinning laser through cylindrical glass prism [25]

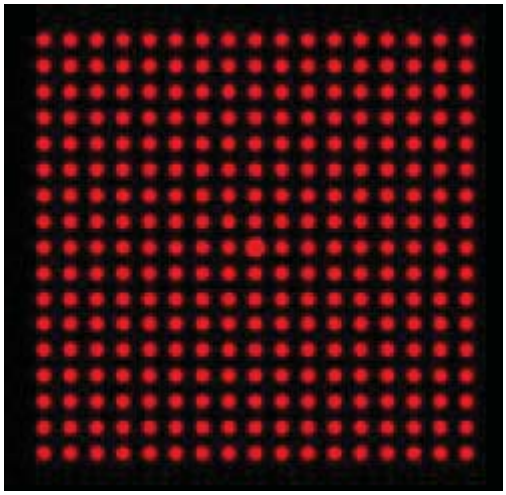


Figure 3.3 – Dot matrix pattern for determining object profile [26]

3.2 CMOS and CCD sensors

The most common technique for measuring the deformation is using a camera to image the scene. Modern digital cameras can either be CMOS (Complementary Metal-Oxide-Semiconductor) or CCD (Charge Couple Device) sensors. These two different technologies both have their own strengths and weaknesses, making them suited to different applications. CCD image sensors are basically made up of an array of MOS capacitors. As photons hit these capacitors a charge is built up that is proportional to the light intensity at that point in the image. Once the sensor has been exposed to the image, the charges on the capacitors are read by shifting the charge from one capacitor to another neighbouring capacitor – much like a shift register. The charge on the final capacitor in the row is shifted into charge amplifier which converts this into a voltage which can be sampled and digitised. CMOS sensors, on the other hand, are basically made up of an array of photodiodes that each have their own charge-to-voltage converters, and often include amplifiers, noise correction, and digitisation circuits. This means that each sensor element has several transistors next to it but it also means that each pixel can be read out individually. [27]

It was originally thought that CMOS sensors would cost less to manufacture than CCD sensors because CMOS sensors could be produced on just about any silicon production line. While this did not prove to be the case, CMOS sensors do still tend to be cheaper because of the cost of related circuitry involved with CCD sensors (such as the timing generation, biasing, analogue signal processing, digitisation, interface and feedback circuitry).

The process used in CCD sensors to transfer the charge in a bucket-brigade fashion is one that consumes a large amount of power. CCD sensors can consume up to 100 times more power than that of a CMOS sensor [27]. This, coupled with cost savings, makes CMOS sensors more suitable for high volume consumer applications especially where high quality images are not the main priority such as in mobile phones.

In the past, CCD sensors could provide higher resolution, higher quality images than CMOS sensors and as such were often used over CMOS sensors in applications where high quality images were required. The reason CMOS sensors yielded poorer quality images was because of the inferior light sensitivity of the CMOS sensors due to the larger real-estate required on the chip for the additional transistors required for each sensing element. In more modern times, however, this is not so much the case. A large amount of investment has been put into CMOS sensor development largely because of their high volume uses. This has led to better light sensitivity and overall CMOS sensors now tend to be on par with their CCD counterparts in terms of quality and resolution. [27, 28, 29]

Since CMOS sensors have an individual amplification circuit for each element, CMOS sensors tend to be more prone to uniform noise [29]. Again, however, with the developments of newer CMOS sensors these differences are again disappearing. Having individual amplification circuits and the ability to read pixels out individually gives CMOS sensors the unique ability to only read the necessary pixels. This is often done using a method called ‘windowing’ where only the area of interest on the sensor is read out. This allows increased frame rates for smaller areas.

CMOS sensors also tend to be faster than CCD sensors. This is mainly because all of the camera functions are placed on the image sensor so signal and power traces can be shorter with less inductance, capacitive, and propagation delays. [27]

3.3 Drive system

3.3.1 Motor considerations

There are four main types of motors used for robotic applications: geared DC motors, brushless DC motors, stepper motors, or servos. Of these, DC motors are one of the easiest to use as they just require an appropriate voltage to be applied across the two terminals. The magnitude of the voltage determines the speed of the motor and the polarity controls the direction the motor turns. DC motors are generally controlled using a DC driver which takes a PWM from a control circuit and generates an analogue supply voltage which is applied to the motor. The operation of a DC motor is shown in Figure 3.4. It works off the basis of mechanical commutation using brushes. However, the use of brushes also has disadvantages such as brushes often cause sparks so they cannot be used in many pipe environments where built up gases could ignite. Furthermore, the brushes are prone to wearing so they may not last as long as other motors. DC motors often induce a large amount of noise to nearby electronics.

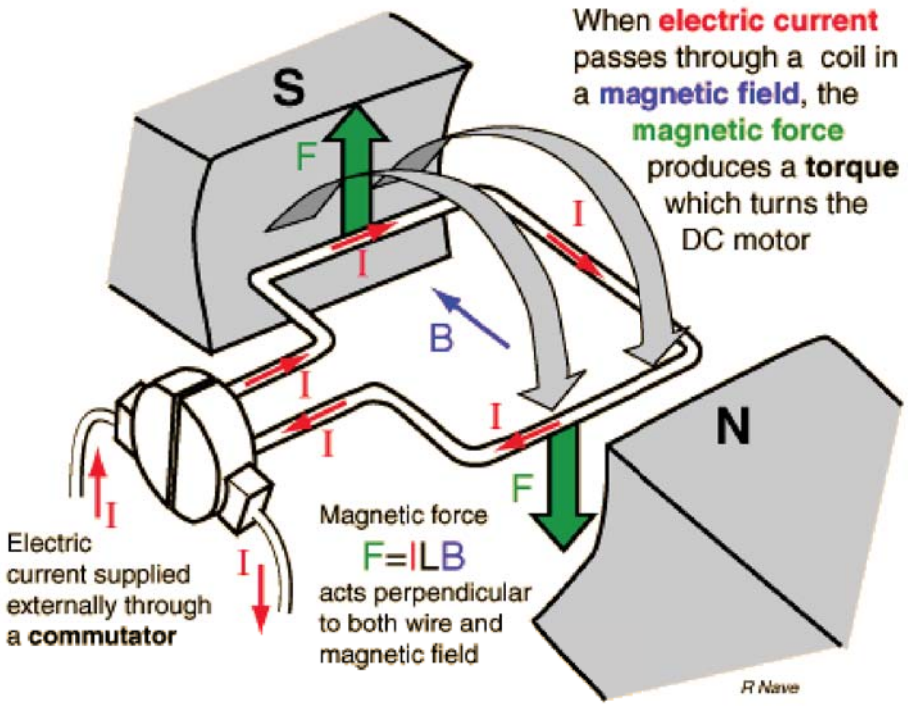


Figure 3.4 – Operation of DC motor [30]

Brushless DC (BLDC) motors are nothing like brushed DC motors. In fact, they actually operate more like an AC motor with a DC sine wave which is simply a square wave. The basic principle of a BLDC motor is shown in Figure 3.5. There are two varieties for these types of motors – outrunner or inrunner BLDC motors. Inrunner BLDC motors have a rotating permanent magnet in the rotor with stationary motor windings on the rotor housing, whereas outrunners are the opposite of this with fixed coils in the centre and rotating magnets on the motor housing. Outrunner BLDC motors tend to be slower speed with higher torque than inrunner motors, although, they are still generally faster than brushed DC motors. In either case, they have to be electronically commutated. This is usually done by use of a sophisticated brushless driver. Brushless motors are often preferable over brushed motors because of their superior power-to-weight ratios, compact sizes, and lack of brushes reducing noise, sparks and maintenance required while increasing efficiency. However, they are generally more expensive and their complex control circuits often double the price of the motor.

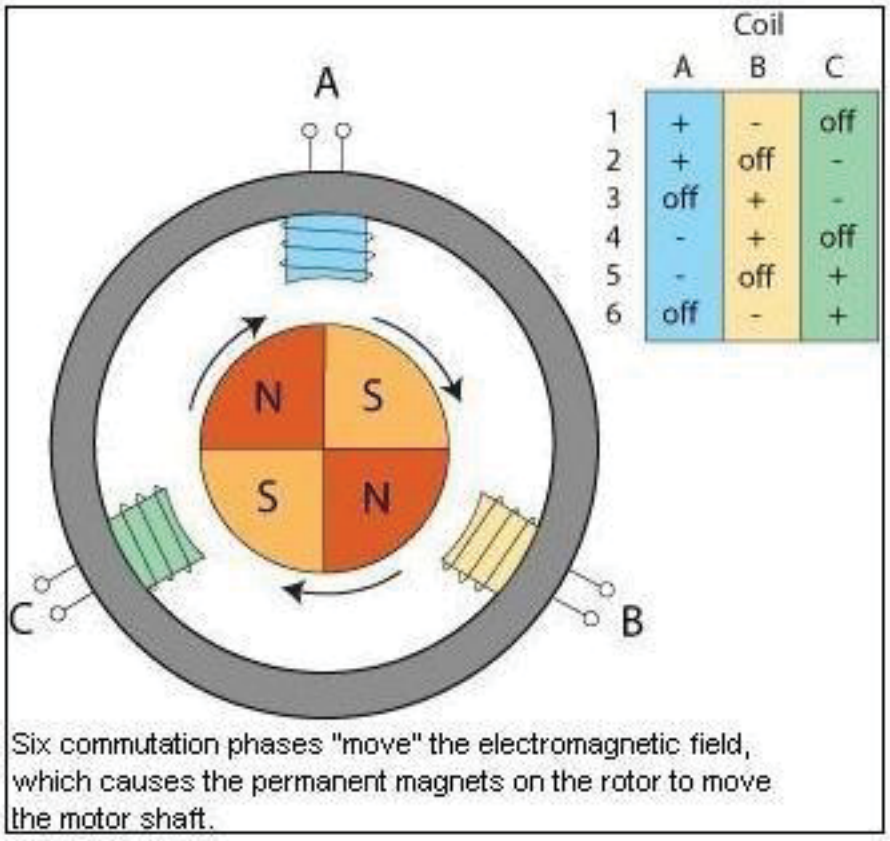


Figure 3.5 – Operation of brushless DC motor [31]

Stepper motors are a type of brushless DC motor with a slightly different construction. They consist of toothed electromagnets around the outside with a centre gear made of iron. When the magnets are energised in turn, the gear's teeth align with the energised magnet's teeth. The motor is constructed in such a way that when this happens the gear's teeth are slightly offset from the next magnet so that when the first electromagnet is turned off, and the second on, the gear rotates slightly. This is considered as one step. Stepper motors have many of the same advantages and disadvantages of normal brushless motors except that they require additional control. This control allows the motor to rotate by a precise angle by rotating by the corresponding number of steps. However, since stepper motors can slip, this attribute cannot be used for accurate position sensing so steppers do not yield any significant benefits for pipe robot applications.

While stepper motors allow limited ability to control position using open loop mechanisms, servos use a closed loop mechanisms to achieve greater accuracy and reliability. They basically consist of a motor (the actual type of motor is not critical but is often a simple brushed DC motor) and some form of encoder or position sensor which is used to provide feedback on the position of the motor. This requires additional control complexity and the additional components mean that they cost more but they have the advantage of being able to optimise the speed, power, and accuracy of the overall system. Many servos only allow for a limited angle of rotation and must be modified in order to allow for continuous rotation. Systems can be designed that incorporate position sensors and motors to gain the benefits of servos without the need for modifying servos and this often allows for better system integration and control for some robotic applications.

3.3.2 DC motor control

H-bridges are commonly used to control DC motors and the basic design consists of 4 switches labelled A1, A2, B1, and B2 in Figure 3.6. Switches A1 and B2 connect the motor to the high supply voltage while B1 and A2 connect the motor to the low supply voltage. This allows the motor to be controlled in either direction by closing one set of the switches at a time, either A1 and A2, or B1 and B2, as illustrated in Figure 3.6.

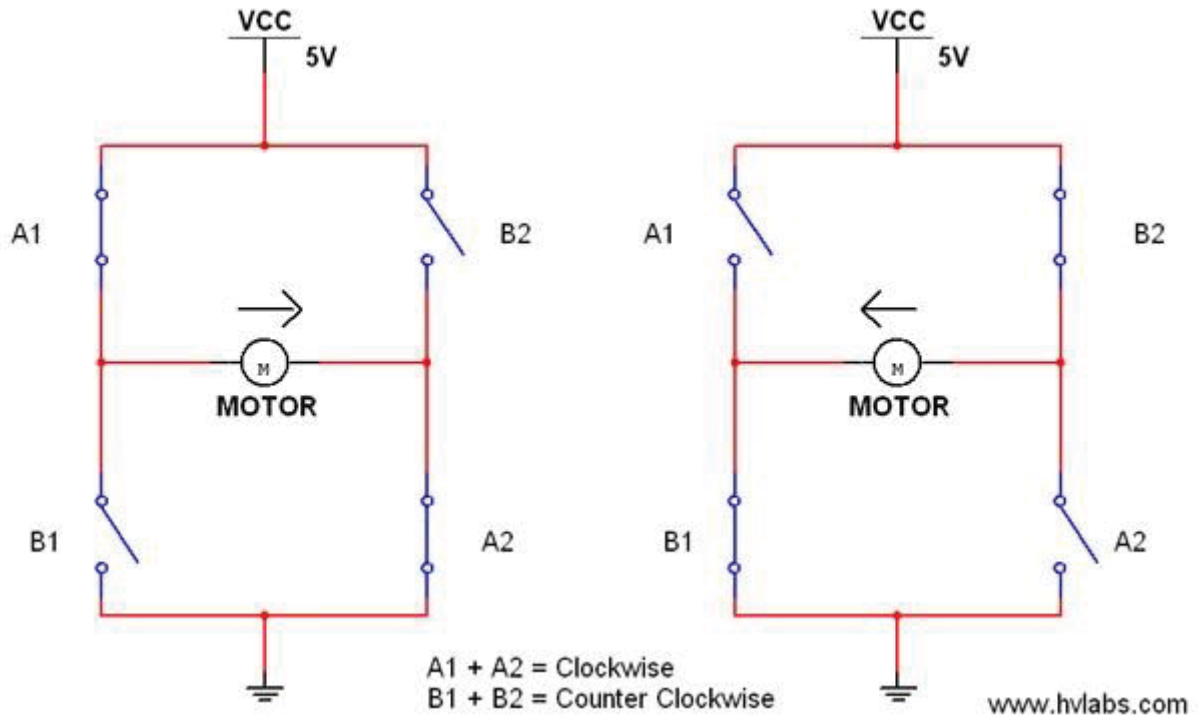


Figure 3.6 – H-bridge showing forward and reverse control [32]

Not only does this allow for bidirectional control of the motor but it also allows other functions to be performed such as breaking the motor or allowing the motor to free run to a stop by disconnecting it completely from the supply. Breaking is achieved by closing both A1 and B2 or by closing both B1 and A2 as this shorts the motor terminals and will bring the motor to a brisk stop. The set of switches A1 and B2, or B2 and A2, should never be closed at the same time as this will cause the supply to be shorted which is known as shoot-through. A list of combinations of switch positions and the results of these are shown in Table 3.1.

Table 3.1 – H-bridge switch positions and outcomes

Switch A1	Switch A2	Switch B1	Switch B2	Result
1	1	0	0	Motor moves right
0	0	1	1	Motor moves left
0	0	0	0	Motor free runs
1	0	0	1	Motor brakes
0	1	1	0	Motor brakes
1	0	1	0	Supply short circuits
0	1	0	1	Supply short circuits
1	1	1	1	Supply short circuits

For DC motor control a DC motor driver in the form of a solid-state integrated circuit (IC) is generally used. Most DC motor drivers are based on the H-bridge design to some extent and may be either a half-bridge for omnidirectional control or a full H-bridge for bidirectional control of the motor. The ICs generally consist of MOSFETs which are used as the switches with pull-up and pull-down resistors to hold the output when the switches are not being driven. Fly-back diodes are normally required externally to protect the IC from reverse current caused by the inductive load of the motor as shown in Figure 3.7. Schottky diodes are generally used for this purpose as they have a low forward voltage drop and fast switching which means they are fast to react. A capacitor across the motor is also recommended to suppress the effects of electromagnetic interference caused by the motor.

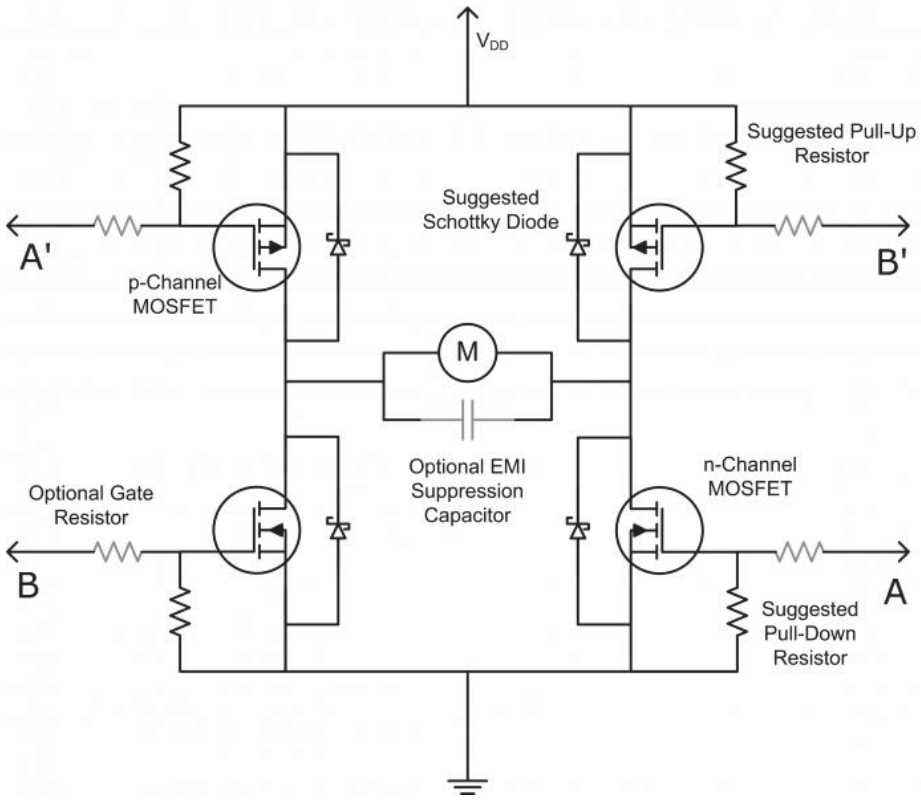


Figure 3.7 – H-bridge with fly-back diodes and pull up/down resistors [33]

Motor drivers can control the speed of the motor by varying the voltage applied to the motor. Some motor driver ICs have a PWM and a direction input while others have two PWM inputs which control each of the two H-bridge channels. The second method has the advantage that full motor control, including breaking, can be achieved with only two inputs in the same was as shown in Table 3.1 where A1 and A2 are always switched together, and B1 and B2 are always switched together. The IC uses the PWM input to determine the motor voltage based off a reference voltage. This reference voltage is the maximum voltage that can be applied to the motor and is often higher than the circuit power to suit the needs of the motor. Additionally, a large electrolytic capacitor should be used on the motor power supply to smooth out the spikes caused by the motor turning off and on.

3.3.3 Power source

One of the main problems still facing the development of new roaming robotic systems is the power source. This is especially the case for in-pipe robots where continual operation is critical. One option is to use an umbilical cable attached to the robot to supply power from an external source. This is

often coupled with additional cables to facilitate communications and control. While this type of power source generally means there is an endless supply of power available, it also limits the distance of pipe that can be inspected as the friction of the cable along the bottom of the pipe increases with distance along the pipe [34]. It also introduces complexities involved with paying out the cable at the correct speed and recovering the cable or entire robot after the inspection.

The alternative to this is to use a rechargeable battery. In early days this was often not an option, however, with the developments in battery technology, batteries are available that weigh less and provide more power than previously. Batteries provide the advantage that the range is limited only by how long the battery will last. Moreover, it makes the system less messy with fewer, or possibly no, cables and, as a result, often more manoeuvrable. Using batteries does have some significant disadvantages, however. The obvious disadvantage is that batteries don't last forever. In order to gain longer usage a battery with a higher capacity must be used and this means more weight. Sometimes the extra weight of the battery means that the motor size and power consumption is increased so the increase in range is disproportionate. This comes with increased battery cost, as well as any associated costs with upsizing the motors. Another disadvantage of using a battery is that batteries are not perfect and each cycle of the battery can be different. In fact, the capacity of each battery decreases with increased use. This means that a large factor of safety (FOS) must be built in to the battery calculation to compensate for this. This also helps to ensure that the robot will not run out of power part way through an inspection assuming the robot is within its specified inspection range although the prevention of this outcome can never be guaranteed entirely and other precautions may need to be made such as having a backup battery or recovery line.

When batteries are the chosen method for powering the robot there are still several options. There are a number of different types of rechargeable batteries available. Some of the most common ones include lead acid batteries, nickel metal hydride batteries (NiMH), nickel cadmium batteries (Ni-Cd), lithium ion batteries or lithium polymer batteries (otherwise known as lithium ion polymer batteries).

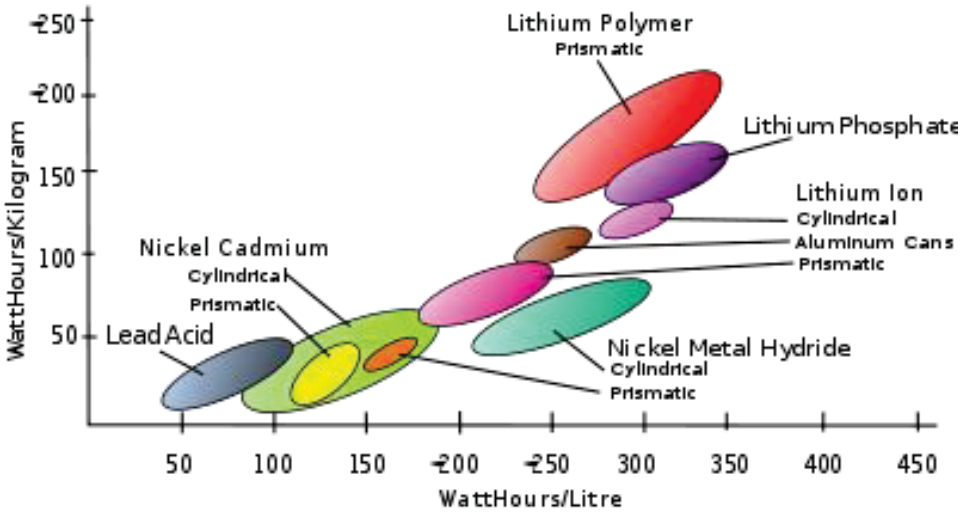


Figure 3.8 – Comparison of different battery types (Wh/kg vs Wh/L) [35]

Lithium based batteries are largely becoming the battery of choice in many modern applications. This is because they have much greater capacity-to-weight and capacity-to-volume ratios than other batteries as shown in Figure 3.8. This means that they can provide more power for longer, with a much

smaller foot print both in terms of physical size and weight. The disadvantage of these, however, is that they tend to cost a lot more, and their maximum capacity is limited to approximately 8Wh in most cases due to manufacturing limitations. They can also be less stable if treated incorrectly so care must be taken to ensure they are handled and used correctly. In many small robots lithium batteries are the ideal choice as they allow the robot to be mobile without a large amount of additional weight. However, due to their capacity limitations they are not as attractive for larger robots or robots that may need extended run times. In these cases the ideal solution may be to choose a lead-acid or similar battery which can provide much larger capacities.

3.4 Position sensors

In order for the profile data to be meaningful, it is important to know exactly where in the pipe the image was taken. Without this knowledge it would be impossible to stitch all of the individual data sets back together to provide an accurate map of the pipe. The simplest way of implementing this would be to acquire and process an image at a set time interval. This, coupled with the length of the pipe and the number of images taken, could be used to provide an approximation of the position of each image. This may provide enough insight into the overall condition of the pipe, but in most cases this would not be sufficient. If the robot encounters slippery areas within the pipe, which is very likely given that the pipes are likely to be slippery by nature, then the image density in that area will be higher as the slippage causes the robot to traverse that section more slowly. This will cause the map of the pipe to be considerably distorted. This leads to the need of a position sensing mechanism that can provide more accurate information on the absolute position of the robot within the pipe.

Positioning methods such as using a camera to determine the position of a robot are used extensively [36, 37], however, this would not be appropriate as an angle large enough to provide an accurate position measurement could not be realised without the line-of-sight of the camera being obscured by the pipe and the ground above it. Similarly, other optical reliant methods [37] such as using cameras mounted on the robot to detect movement aren't applicable as the pipe will be too dark to allow them to function effectively. The use of Global Positioning System (GPS) is also limited due to the lack of line-of-sight from the robot to the necessary satellites. Ultrasonic and radar approaches are similarly eliminated due to range restrictions. Ultrasonic methods are best used for short range applications typically less than two metres, and radar is best suited for longer range applications such as those over one kilometre if accuracy is important [38].

One such solution would be to use two encoders and comparing the data between these. These encoders would need to be placed on the non-driving wheels in order to detect slippage, otherwise only the motor speed would be determined and this would not be any more useful than the previous approach. By comparing the data between the two non-driving wheels the effects of slippage could be significantly reduced and this would allow for more useful position information. By using an encoder on each of the non-driving wheels the effects of free-wheeling is also reduced as the likelihood of this occurring (due to a pit in the pipe wall, etc.) on both wheels at the same time is low. To further reduce this the motor speed could also be measured to ensure the non-driving wheel speed is not in excess of the motor speed.

Since the robot is designed to inspect only pipes that are straight it would also be possible to use a laser positioning system. This would require a laser to be placed on the robot and a reflector at either end of the pipe. The position would then be determined by pulsing the laser and measuring the time

it takes for the pulse to be reflected back. This is often used to provide a position in two or three dimensions [38], but in this case it is only necessary to determine the position in one dimension along the length of the pipe. This method is immune to any effects of slippage or free-wheeling and is accurate to within a few millimetres [38]. Using this type of positioning system does have its benefits but it also comes with a financial trade-off. If it is necessary to increase the accuracy to sub-millimetre accuracy then it would have to be coupled with another approach, further increasing the cost, and accuracy, of the system.

One final approach is to use some form of an Inertial Measurement Unit (IMU) to measure the acceleration and velocity of the robot as it travels through the pipe. An IMU typically consist of a combination of the following: an accelerometer, a gyro sensor to determine the angular velocity, a magnetometer which acts as a digital compass, and an altimeter. Since the direction and altitude are unimportant the latter two can be ignored. Furthermore, if the angular rotation of the platform is to remain constant then the gyro sensor can also be ignored; otherwise this is needed to compensate the measurements of the accelerometer. By using the information from the accelerometer the position can be determined by extracting the acceleration and velocity profiles from the accelerometer over the run-time of the motors. This approach on its own may not be enough to provide an accurate position measurement due to the limitations of the accelerometer and the proneness to noise. However, coupled with one of the aforementioned approaches, in particular the encoder approach, this could lead to a relatively low cost solution with increased accuracy. [39, 40, 41]

3.5 Control & processing system

There are many different approaches that could be taken for processing the data and controlling the robot. This would vary depending on whether the processing was to be performed on the robot itself or on a remote node which would still require some form of processor on the robot to facilitate the data communication and interpretation. The advantage of doing the processing on the robot is that it can drastically reduce the bandwidth required to transfer the data to a remote processor. This increases the communication medium possibilities as well as decreasing the time for transmission, and hence, allowing the system to inspect the pipe at higher speeds.

Typically some form of microcontroller embedded on the robot would be used to implement the control of the robotic platform due to the programming simplicity, versatility and cost. However, implementing any image or signal processing can become difficult on a microcontroller because of the memory restrictions and the speed at which it can process the signal.

In contrast, a dedicated circuit or application-specific integrated circuit (ASIC) can be designed to optimise the processing speeds for a given application. This often makes them the implementation mechanism of choice for digital signal processors (DSPs) [42]. The limitation of these dedicated circuits is that they reduce the flexibility and re-configurability of the logic. This has led to the growing popularity of solutions such as Field Programmable Gate Arrays (FPGAs) which combine the flexibility of the reprogrammable controllers with the application specific dedication of the ASICs. This hugely aids in the development process of processing systems as the design can be changed indefinitely, within reason.

Both FPGAs and microcontrollers are programmable. The main difference is that microcontrollers are software implementations whereas FPGAs are hardware implementations. This means that the design of an FPGA program is based directly on the implementation of the hardware which is made up of a large number of logic cells with interconnects such as in Figure 3.9. These logic cells typically consist of a look-up table (LUT) with a latch on the output. The LUT consists of an arrangement of AND and OR gates as shown with the AND section fixed and the OR section programmable. This allows any arbitrary output combination of the inputs [43].

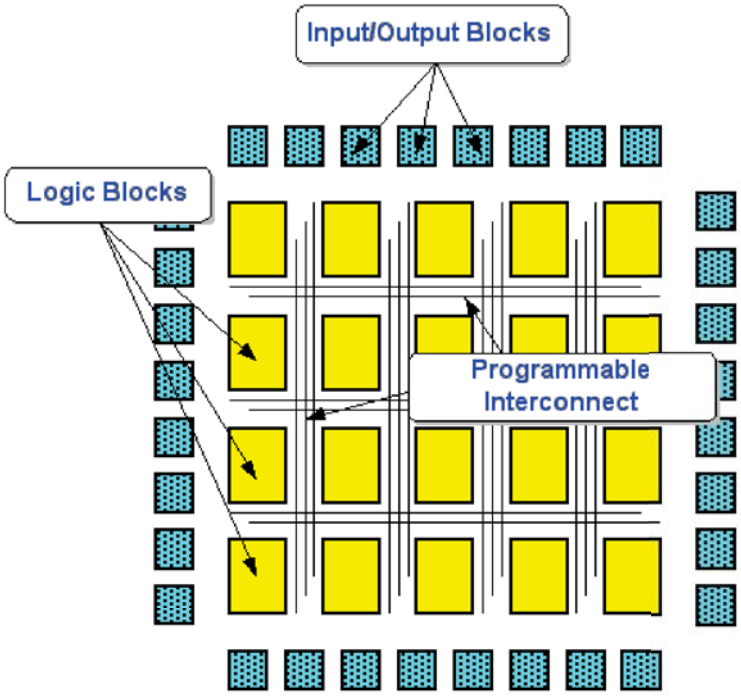


Figure 3.9 – FPGA hardware architecture and logic cell [44]

Using FPGAs for digital signal processing has many advantages over a software processor. One such advantage is the ability to implement parallel processing. Some multi-core software processors are able to do parallel processing to a certain extent but with a hardware implementation the ability to parallel process is vastly increased. This is particularly useful in applications such as processing images as it allows each area of the image to be processed at the same time, whether this is row, column, a given window size or even individual pixels in some cases. This dramatically improves processing times compared to a serial processor which would have to loop through each pixel individually. It can also improve the memory requirements of the system as the whole image does not need to be buffered to memory as it can be processed as it is streamed from the sensor.

The clock speeds of serial processors can vary significantly from very low right up to a few gigahertz. Microcontrollers usually have a clock speed in the 100 to 600 megahertz range, whereas, FPGAs are generally rated for clock speeds of 30 to 100 megahertz and typical applications will be in the 30 to 60 megahertz range [45]. Despite the clock cycle advantage of microcontrollers, FPGAs have a significant efficiency advantage over microcontrollers [45]. They also tend to consume less power to process a given task in the same time frame as their microcontroller counterparts. This makes them ideal for embedded systems that require large processing. As such, FPGAs are becoming increasingly popular for image processing in embedded systems [46, 47].

3.6 Communication system

There are many different forms of communication that can be used for a mobile robot and these can be divided into two main categories: wireless and non-wireless. Non-wireless methods involve running a hard-line from the robot to the communication centre. This would usually be some form of copper wire such as an Ethernet, serial, or USB cable but fibre-optics can also be used and have been proven to have many advantages over copper such as the speed and bandwidth as well as reduced weight and friction [34]. When a mobile robot is already using an umbilical cable for the power it makes sense to couple this with the communication medium especially since a hard-line generally offers greater speeds and bandwidths over wireless options. However, in cases where a battery is chosen as the appropriate power source some form of wireless communication would be preferable to maintain the flexibility of the system.

There are also many types of wireless communications and these can generally be classified under one of the following: radio frequency (RF) wireless, Wi-Fi point-to-point access, Wi-Fi TCP/IP network, GSM Cellular, Bluetooth, or Satellite communication. Obviously any system such as mobile networks or satellite communications would be out of the question as the concrete pipes and depth of the robot below ground would impede the functionality of the communications.

In order to embed a wireless communication module within a battery-operated mobile robot it is necessary to try to reduce both the cost and, more importantly, the power-consumption of the unit. Bluetooth and Zigbee are two radios that have emerged to target these needs. Bluetooth is generally used for short range transmissions up to 10m (Bluetooth Class II) with very low power consumption (approximately 1mw) but this can be extended up to 100m (Bluetooth Class I) with a power consumption of approximately 100mW [48, 49].

Zigbee is a radio specification designed on the IEEE 802.15.4 standard which is designed for lower cost, lower power consumption applications than Bluetooth. This comes at the cost of bandwidth as the Zigbee protocol allows for data rates up to 250Kbps whereas Bluetooth is up to 1Mbps [48]. Zigbee communication is usually within the 10-100m range but can be extended up to 3km by increasing the RF transmitting power [50]. Although Zigbee is built on the IEEE 802.15.4, which is primarily a point-to-point communication standard, it facilitates a mesh network which allows the range to be further increased by adding additional nodes [51].

XBee is one manufacturer of Zigbee radios and they also manufacture Wi-Fi, and 802.15.4 based radios. Similar Bluetooth radios are also available on the market.

Table 3.2 shows a comparison of the most common range of these radios. The series one modules are those based straight off the 802.15.4 standard and only allow point-to-point or star topology [51, 52]. They are designed to work straight out of the box but are not compatible with the series two modules. The series two modules, on the other hand, are based on the Zigbee protocol [53]. This does require a small amount of configuration out of the box but it allows for more varied network structures. The PRO model in each series is designed for applications that require longer ranges at the cost of higher power consumption.

Table 3.2 – Table showing common XBee modules and specifications

Product	Frequency	Power Output	Maximum Range	RF Data Rate	Protocol	Multi-point	Mesh
XBee WiFi	2.4 GHz	<120 mW		<65 Mbps	802.11b/g/n	Yes	
Bluetooth Bee*	2.4 GHz	10-100 mW	10-100m	1 Mbps	Bluetooth	Yes	
XBee ZB	2.4 GHz	1.25/2 mW	120 m	250 kbps	Zigbee		Yes
Xbee-PRO ZB	2.4 GHz	63 mW	3.2 km	250 kbps	Zigbee		Yes
XBee ZB SMT	2.4 GHz	3.1/6.3 mW	1.2 km	250 kbps	Zigbee		Yes
XBee-PRO ZB SMT	2.4 GHz	63 mW	3.2 km	250 kbps	Zigbee		Yes
XBee 802.15.4	2.4 GHz	1 mW	90 m	250 kbps	802.15.4	Yes	
XBee-PRO 802.15.4	2.4 GHz	63 mW**	1.6 km	250 kbps	802.15.4	Yes	
XBee-PRO xsc	900 MHz	100 mW	24 km***	9.6 kbps	Proprietary	Yes	
XBee-PRO 900	900 MHz	50 mW	10 km***	156 kbps	Proprietary	Yes	
XBee-PRO DigiMesh 900	900 MHz	50 mW	10 km***	156 kbps	DigiMesh		Yes
XBee DigiMesh 2.4	2.4 GHz	1 mW	90 m	250 kbps	DigiMesh		Yes
XBee-PRO DigiMesh 2.4	2.4 GHz	63 mW**	1.6 km	250 kbps	Proprietary		Yes

* Not XBee brand but used for comparison

** International variants of 2.4 GHz XBee-PRO modules are limited to a 10 mW transmission (TX) power

*** Range with high gain antennas

For short range, low data rate systems, the Zigbee modules are superior for embedded systems because of their low power consumption. For systems that require larger transmission ranges, the PRO series would also work well in an embedded system although they do consume more power. In general, a power consumption of less than 100mW is insignificant in comparison to the motor consumption and if power consumption is a major concern the Zigbee modules can be configured to turn off to save power when they are not needed [54]. If longer ranges or higher data rates are necessary then other solutions must be considered such as Wifi which is built on the IEEE 802.11 standard and can realise data rates up to 600 Mbps.

Furthermore, when Zigbee devices are used, the end node consumes considerably less power than the router and host nodes [54]. This characteristic can be utilised to reduce power consumption on the robot by having the device on the robot as the end device. The host node could then be at the receiver end and make use of a power source other than the robot, hence, decreasing the power consumption of the robot.

3.7 Concept design

The design concept consisted of a structured lighting measurement system mounted on a wheeled robot platform. Geared DC motors were chosen to drive the wheels due to the their ability to reduce

the complexity of the drive chain while providing sufficient torque to drive under any conditions and allowing for good speed control using simple electronics.

While lithium batteries tend to be more expensive than alternatives their superior power density makes them more ideal for powering robotic applications where size and weight are important, such as for pipe robots. Not having an umbilical cable for power increases the flexibility and range of the robot. Using lithium batteries also means that it would be easy to have spare batteries to swap out for extended use.

This design concept uses encoders for position sensing. This is a cheap and effective solution which does not require any additional sensing elements such as a reflector when using a laser system. Ideally encoders would be used on each of the four wheels so that the feedback from these could be cross examined. Alternatively, an IMU could be used in conjunction with an encoder on each of the non-driving wheels to reduce the cost and complexity of the design, while maintaining an accurate position measurement.

For battery operated robots the power supply is limited so it is important that the electronics and communications use as little power as practical. On top of this, the type of communication is limited by the operating environment of the robot which for this application is inside a pipe which rules out communication methods such as GSM or satellite communication. A wireless serial based Xbee module was chosen for the communication as they have a very low power consumption with relatively large range; however this does come at the cost of bandwidth which is an issue if the raw video footage needs to be transmitted.

Using a structured lighting approach for measuring the erosion yields a large volume of image data that needs to be processed. For this to be processed in real-time it is necessary to do this on the robot itself to reduce the bandwidth required to transmit the data. While FPGAs have lower clock speeds than most microcontrollers they have an efficiency advantage and their parallel processing characteristics make them far superior for image processing. Additionally, they tend to have more memory than microcontrollers for storing the image data and they have low power consumption. These characteristics make them the ideal choice for embedded image processing applications such as this.

Chapter 4 Anti-Rolling Robot Platform Design

4.1 Anti-rolling mechanism design

In the case where the erosion is concentrated on the bottom of the pipe, it is difficult to come up with a robot design that can traverse along the bottom of the pipe. This is especially true when the erosion is severe, since it would be extremely difficult to provide a consistent frame of reference which is necessary for any system incorporating a device such as a camera. Without a consistent frame of reference it is difficult to determine where the camera is pointing and, thus, the information returned from the camera is not very useful. The alternative is to develop a robot that traverses the pipe by means of contact with the pipe walls. This can also be difficult as the pipe is circular so a design such as this may have a tendency to roll within the pipe. Therefore, if a design such as this is to be considered it is necessary to design a robot with anti-roll characteristics.

A concept was developed for the mechanical platform of a pipe robot with anti-rolling properties. This design consisted of a rectangular body spanning the width of the pipe, supported at each end by a set of legs with wheels attached. These legs are attached to the platform through the platform's central axis which allows them to be fixed at an angle to the horizontal axis. This concept is shown in Figure 4.1.

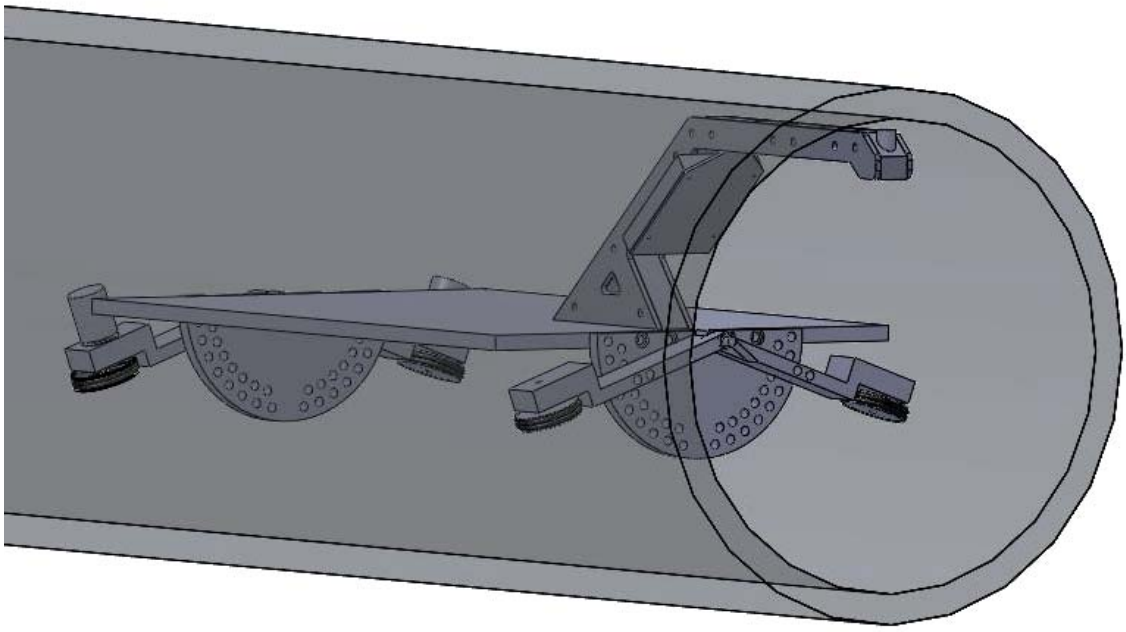


Figure 4.1 – Anti-rolling pipe robot concept

Figure 4.2 shows this concept. The centre of the coordinate system is defined as the centre of the rectangular platform which also coincides with the centre of the pipe. The x-axis is defined as the horizontal axis while the y-axis is vertical. The z-axis is along the direction of the pipe out of the page. The radius of the pipe is defined as R, and the gravitational force on the platform is defined as W. To simplify the analysis, the gravitational force and the corresponding shift in centre of gravity due to the

legs and attached wheels is ignored. F_a and F_b are the normal forces acting on the left and right wheel respectively. The angle θ is defined as the angle of the legs from the horizontal. For the right-hand-side leg this angle is clockwise, and for the left-hand-side it is the same magnitude but in an anti-clockwise direction. In the concept designed, the legs pivoted in the centre of the platform allowing this angle to be fixed at different angles if necessary.

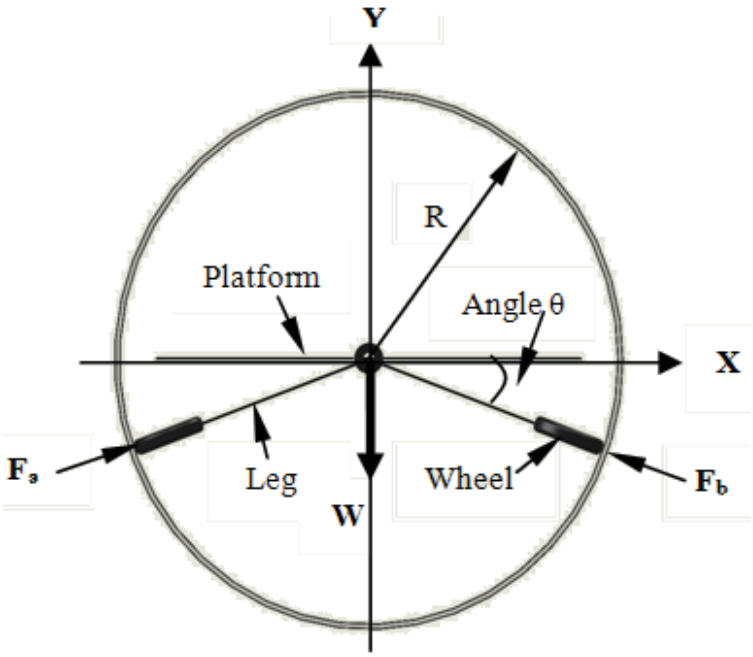


Figure 4.2 – Concept design of anti-rolling system

The design of the platform is symmetric about both the y and z axes. This ensures that when the platform is placed in the pipe horizontally the centre of gravity always coincides with the y-axis. Under ideal conditions, the centre of gravity should coincide with the coordinate system origin. At the very least it should lie somewhere on the y-z plane, and preferably below the top surface of the rectangular platform.

The symmetric design ensures that when the platform has no payload and is in a static state the equilibrium condition is met such that the horizontal component of the normal forces are equal and opposite and the summation of the vertical components are equal and opposite to the gravitational force W, as described by:

$$(\boldsymbol{F_a} - \boldsymbol{F_b}) \cos \boldsymbol{\theta} = \boldsymbol{0} \tag{4.1}$$

$$(\boldsymbol{F_a} + \boldsymbol{F_b}) \sin \boldsymbol{\theta} - \boldsymbol{W} = \boldsymbol{0} \tag{4.2}$$

When this equilibrium condition is met, such as in Figure 4.2, it is clear that there is no rolling torque generated by the influence of the normal forces F_a and F_b or the gravitational force, W. Any rolling motion that was to exist would be as a result of a force with a magnitude great enough to overcome the friction force between the wheels and the pipe wall. This friction force depends on both the normal force as well as the coefficient of friction, μ. The coefficient of friction is determined using the characteristics of the materials of the pipe and wheels as well as characteristics pertaining to the operating conditions. Once the influence of these factors is determined there is very little room for

variation of this coefficient. Therefore, the only real way to change the friction force is to vary the normal force.

From Equations 4.1 and 4.2 we can ascertain that

$$\boldsymbol{F_a = F_b = \frac{W}{2 \sin \theta}} \tag{4.3}$$

This shows that the normal forces that result from the interaction of the wheels with the pipe wall change with the angle θ. The rational domain for this angle is $0 \leq \theta \leq 90$ although an angle of 90 degrees would never be practical or achievable as the left and right legs would be acting in the same physical domain. Figure 4.3 shows the normal force, F_a, plotted against the rational domain of θ for a unit mass. This shows that as the angle of θ decreases the normal force increases such that when θ is less than 20 degrees and approaches zero, the normal force approaches infinity.

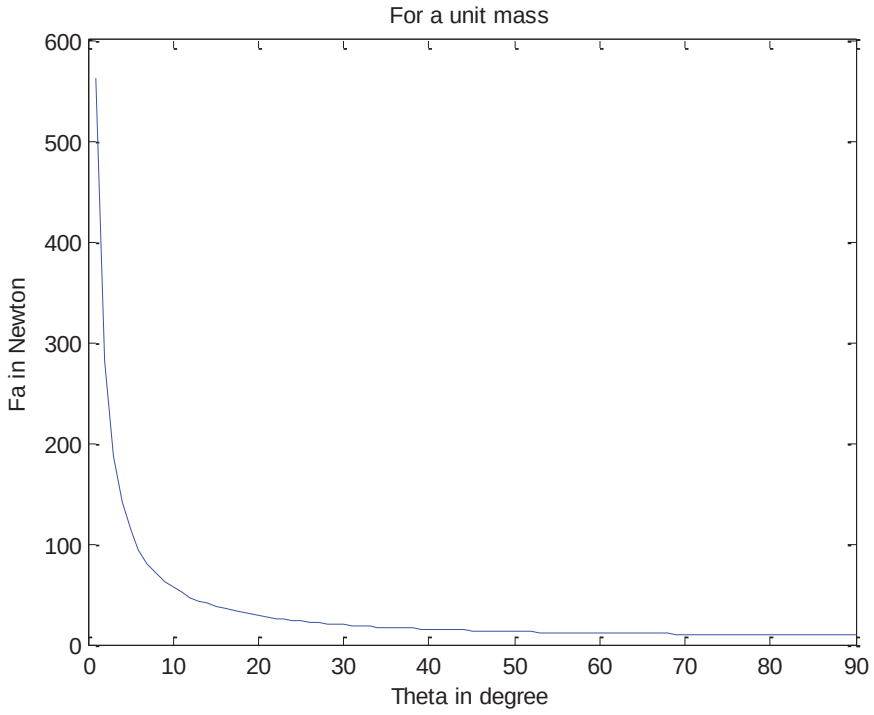


Figure 4.3 – θ -Normal force plot for a unit mass

Figure 4.3 shows that the angle of the legs has a large influence on the magnitude of the normal forces. This means that for a constant value of μ, the angle, θ, can be adjusted to vary the friction force of the robot wheels on the pipe wall. This can be used to prevent the robot from rolling by increasing θ.

When the platform is placed in the pipe at an angle other that horizontal the normal forces on either wheel are still equal and opposite as the centre of gravity is still in the y-z plane due to the symmetrical design. Only the horizontal and vertical components of these normal forces have actually changed. This means that, at an angle, the robotic platform is no more likely to rotate than when it is perfectly

horizontal. If the centre of gravity was to move, however, by placing a payload offset on the platform, then this may no longer be the case, depending on the magnitude and offset of this payload as illustrated in Figure 4.4.

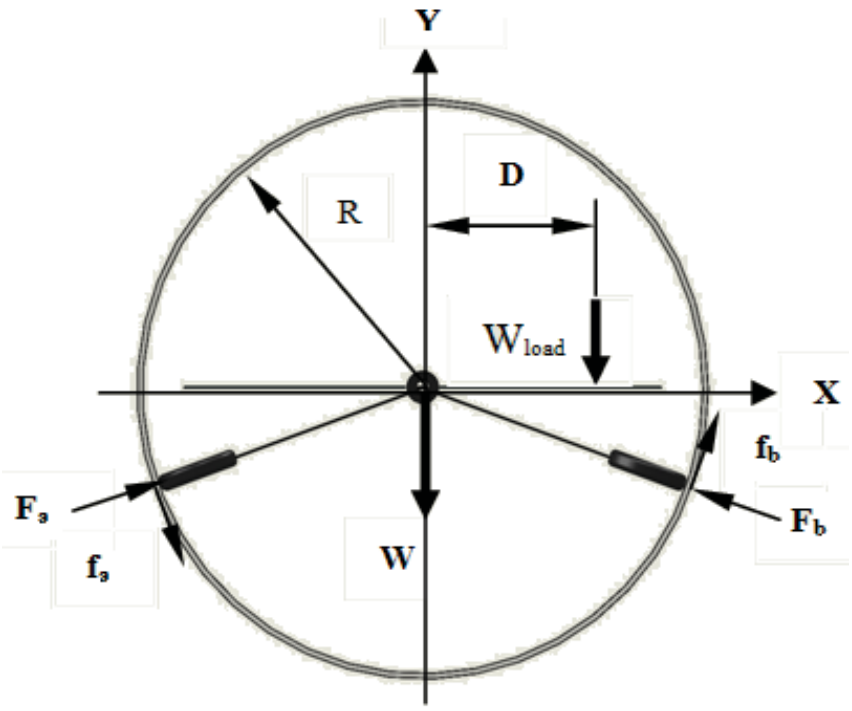


Figure 4.4 – Mechanical platform with payload offset

This is the same as before, but with an additional gravitational force, W_{load}, which corresponds to the payload weight which has been placed on the platform, offset by D on the x-axis. Also, f_a and f_b are the friction forces applied on the wheels of the robot. In this case, in order for the robot to be in an equilibrium state, the following conditions must be met such that the net external forces and torques acting on the robot sum to zero.

$$(F_a - F_b)\cos\theta + \mu(F_a + F_b)\sin\theta = 0 \tag{4.4}$$

$$(F_a + F_b)\sin\theta - \mu(F_a - F_b)\cos\theta - W - W_{load} = 0 \tag{4.5}$$

$$DW_{load} - (f_a + f_b)R = 0 \tag{4.6}$$

Therefore:

$$F_a = \frac{F_b(\cos\theta - \mu\sin\theta)}{\cos\theta + \mu\sin\theta} \tag{4.7}$$

$$F_b = \frac{(W + W_{load})(\cos\theta + \mu\sin\theta)}{2\sin\theta\cos\theta\,(1 + \mu^2)} \tag{4.8}$$

$$D = \frac{\mu R(W + W_{load})}{W_{load}\sin\theta(1 + \mu^2)} \tag{4.9}$$

This shows that for a robot with a fixed weight and payload and a pipe with a known radius, the only factors that affect the offset allowed without the robot rolling is the angle θ and the coefficient of friction, μ.

The coefficient of friction for the wheel in contact with the pipe wall is determined using knowledge about the materials of each and the operating environment. In this case the wheel is made of a smooth material and is in contact with a concrete pipe which is more than likely to be damp and slippery. The potential values for μ in this case range from 0.1 to 0.4 based off the considerations made in [55, 56].

Equation 4.9 can be re-written as a function of μ and θ:

$$\boldsymbol{D = f\left(\frac{\mu}{\sin\theta(1+\mu^2)}\right)} \tag{4.10}$$

where $\theta \neq 0$ and the radius R of the pipe, the robot weight W, and the offset payload weight W_{load} make a unit offset distance. Figure 4.5 shows the effect on the offset distance that both μ and θ have. This indicates that D increases with decreasing θ and increasing μ. This means that for a given application, the likelihood of the robot rolling is lower if there is a higher coefficient of friction, or a smaller leg angle.

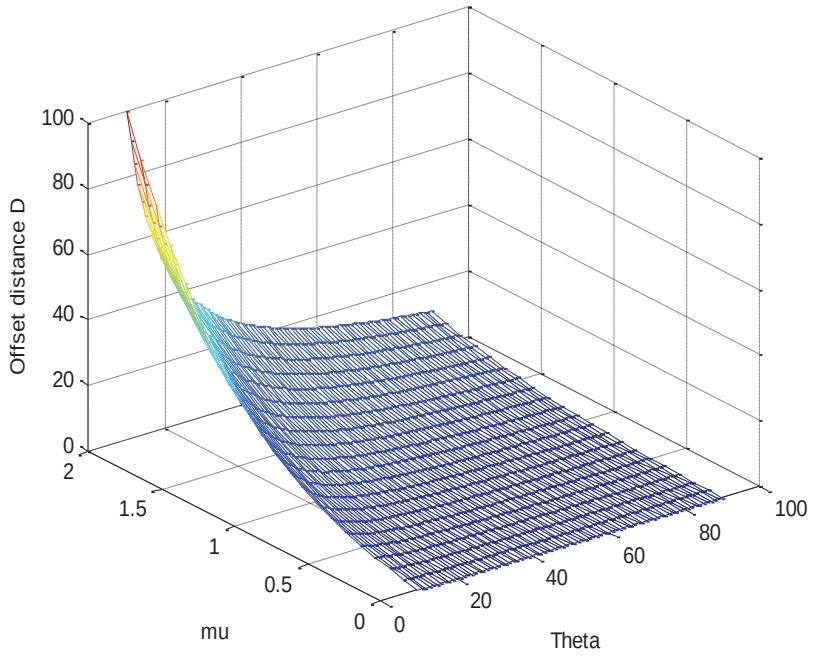


Figure 4.5 – Effect on offset distance (D) of leg angle (θ) and coefficient of friction (μ)

From both Figure 4.3 and Figure 4.5 it is apparent that if the likelihood of the robot rolling is to be reduced, it should be designed such that the angle of the legs, is made as small as possible. However, following Equations 4.15 through 4.22, it is also clear that the smaller this angle the larger the power required to drive the robot as the power to move at a constant speed is proportional to the normal force. A reasonable compromise between these two is choosing a leg angle of 20 degrees.

Once the robot has been designed with a specific weight, and payload, and the leg angle has been fixed, the only remaining elements affecting the roll, from Equation 4.9, are the radius of the pipe and the coefficient of friction. D can be expressed as a function of pipe radius R and μ such as:

$$D = f\left(\frac{\mu R}{1+\mu^2}\right) \tag{4.11}$$

where the robot weight, payload, and leg angle make a unit offset distance. The effect of the offset distance D of both the coefficient of friction and pipe radius are shown in Figure 4.6. It is clear that the offset distance is proportional to the pipe radius.

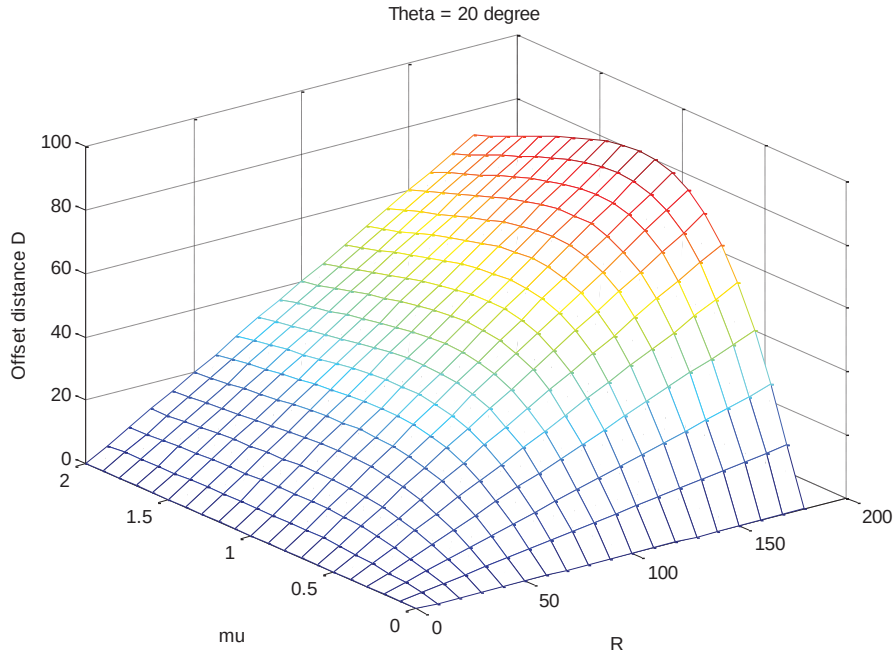


Figure 4.6 – Effect on offset distance (D) of pipe radius (R) and coefficient of friction (μ)

For an anti-rolling system it is clear that the offset distance D of any payload is limited by other design factors such as the leg angle θ, the actual weight of the payload, and the radius of the pipe. Even once the pipe radius is known and the robotic platform has been designed, the specific value of the coefficient of friction is still unknown, and it is even possible for the coefficient to change as the robot moves through the pipes and encounters differing levels of dampness induced slipperiness and other such factors. The effect of the offset distance on each of these design factors can be studied for various levels of μ to gain insight into the ability of the platform to maintain its anti-rolling properties.

Figure 4.7 shows the effect of different values of μ coupled with the leg angle θ. Here the pipe radius was chosen to be 155mm, the platform weight to be approximately 5kg, and the payload weight to be 2kg. These are based on the robotic platform developed for testing this system in Chapter 7.

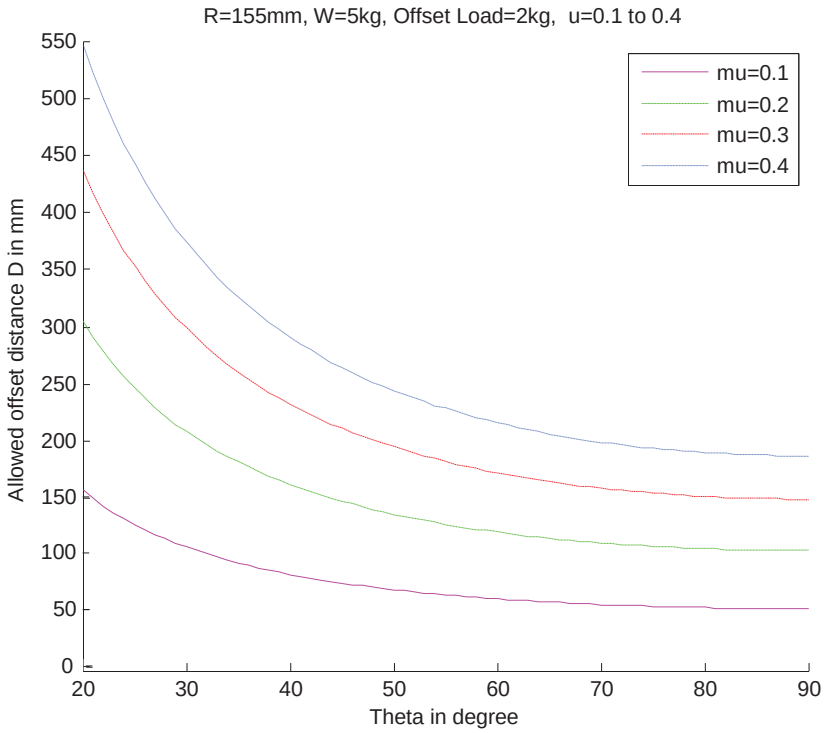


Figure 4.7 – Effect on offset distance (D) of leg angle (θ) for different levels of friction coefficient (μ)

Figure 4.7 shows that even the smallest value of μ from the probable range for μ allows the offset distance to be 155mm for a leg angle of 20 degrees. These means that a two kilogram payload could be placed on either side of the platform at the very edge and still not cause the robot to roll. Since a smaller value of μ corresponds to the slipperiest conditions this means that the robot should be able to remain stable under any conditions within the pipe. It also shows that when the magnitude and offset of a given payload is known, the leg angle can be optimised to reduce the power required based on the lowest value of μ.

A similar analysis can also be performed to identify the effect that different payload weights will cause on the offset distance. This can be particularly useful to consider during the design stage as it allows for additional payloads to be added to the robot in future, such as additional power packs. Figure 4.8 shows the effect on the offset distance that different payloads have for a platform with a fixed leg angle of 20 degrees.

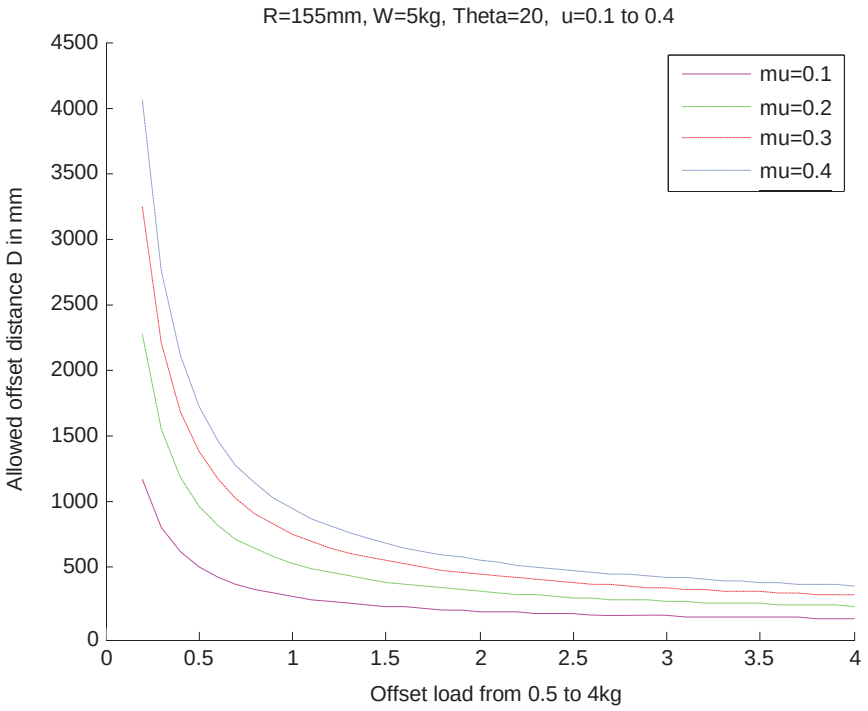


Figure 4.8 – Effect on offset distance (D) of payload weight (W_{load}) for different levels of friction coefficient (μ)

This example shows that when the pipe radius is 155mm it makes no difference where the payload is placed on the robot as long as it is no more than two kilograms. After this point, more careful consideration must be made as to the actual placement of the payload to avoid rolling. This does not mean the larger weight cannot be used, even for the most severe case where μ is 0.1, but other design factors must be included to avoid rolling.

The final design factor to consider is the effect of the pipe radius R for different levels of μ. This shows that as the pipe radius increases, so does the allowed offset distance. Even for a low value of μ, the payload can still be placed near the outside of the pipe without inducing any roll, irrespective of the actual radius of the pipe.

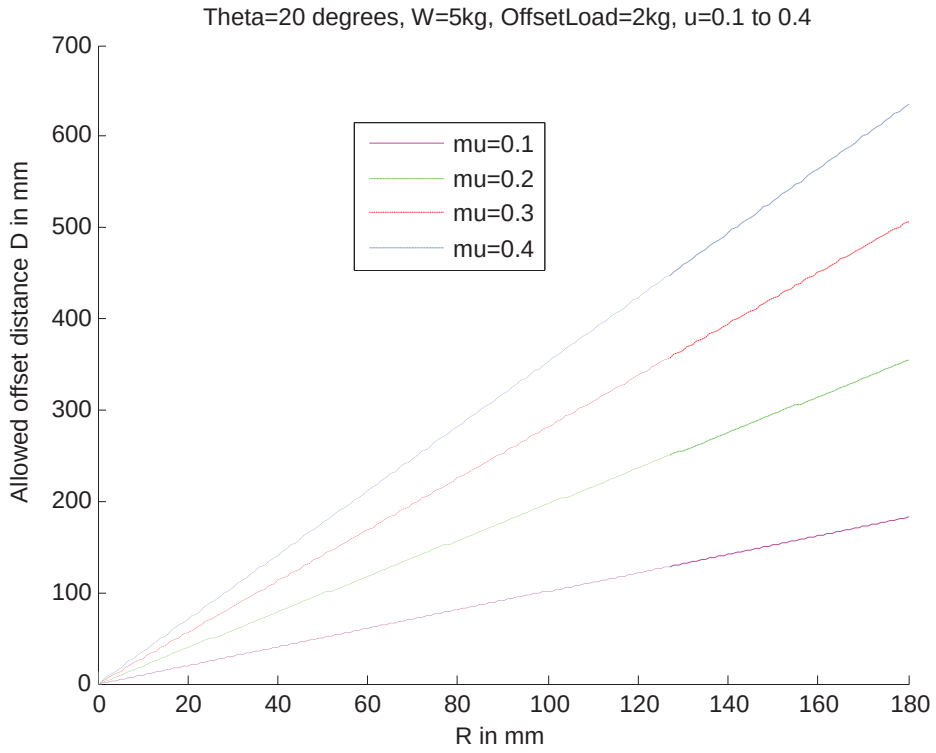


Figure 4.9 – Effect on offset distance (D) of pipe radius (R) for different levels of friction coefficient (μ)

By considering the requirements of the robot such as the weight, payload, and operating environments, it is clear that it should be possible to develop a platform with anti-rolling properties by carefully choosing the aspects of the design such as the leg angle, and the position of the payload. This theory was tested and has been presented in Chapter 70 along with the results and an analysis of these.

4.2 Robot motor selection and battery calculations

The motor and battery requirements of the robot can be calculated given the physical properties of the robot as well as the operating requirements such as the required running time. In general, the longer the robot must be able to run, the greater the capacity of the battery must be. But using a larger battery means that the weight of the system is increased so larger motors may be needed. With larger motors and more weight there will be larger current draw so the same size battery will no longer last as long. As such there is a trade off in selecting the correct motors and battery for a given application and it is important to find the right balance of these.

In order to calculate the required battery and motor sizes the forces involved must first be analysed. These forces are made up of three main components: the force the robot must be able to overcome in order for the wheels to roll (f_{roll}) – otherwise known as friction force, the force the robot must overcome to travel up any incline (f_{hill}), and the air resistance the robot encounters as it moves (f_{air}), as shown in the following equation:

$$\boldsymbol{F = f_{roll} + f_{hill} + f_{air}} \tag{4.12}$$

For the given configuration above, the normal force of the tyres against the wall of the pipe can be calculated and this can be used to calculate the friction force necessary to overcome in order to move using the following equations:

$$f_{roll} = \mu N \tag{4.13}$$

$$mg = N_y = N \cos\theta \tag{4.14}$$

where μ is the rolling resistance coefficient (normally within the range of 0.010 to 0.015), N is the sum of normal force, m is the mass of the system, g is the acceleration due to gravity (9.81 ms^{-1}), N_y is the vertical component of the normal force, and θ is the angle of the legs from horizontal as illustrated in Figure 4.10.

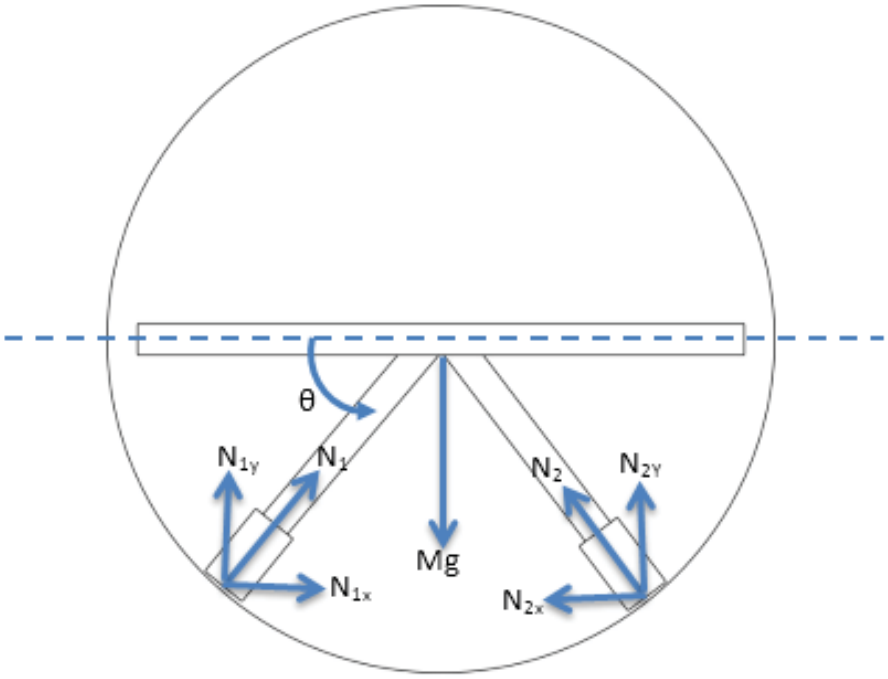


Figure 4.10 – Normal force of tyre against pipe wall

This yields the following equation for the rolling force:

$$f_{roll} = \frac{\mu mg}{\cos\theta} \tag{4.15}$$

The force that must be overcome to travel up an incline of angle of φ can be calculated as:

$$f_{hill} = mg \sin\varphi \tag{4.16}$$

Finally, the force from air resistance can be calculated using the following equation:

$$f_{air} = \frac{1}{2} C_d A \rho v^2 \tag{4.17}$$

where C_d is the drag coefficient, A is the cross-sectional area of the robot, ρ is the density of air (approximately 1.2 kg.m^{-3}) and v is the velocity of the robot. These forces can be used to calculate the power required for forward motion at a constant velocity (P_v) using Equation 4.18:

$$\boldsymbol{P_v = (f_{roll} + f_{hill} + f_{air})v} \tag{4.18}$$

where v is the velocity of the robot. In order to calculate the torque requirements of the motor the total power is required. This means calculating the power required to accelerate the robot (P_a) and this can be calculated using the following:

$$\boldsymbol{P_a = \frac{E_k}{t}} \tag{4.19}$$

$$\boldsymbol{E_k = \frac{1}{2}mv^2} \tag{4.20}$$

where E_k is kinetic energy, t is the time taken to accelerate, m is the mass of the robot and v is the velocity. Rewrite the power required to accelerate the robot as:

$$\boldsymbol{P_a = \frac{1}{2}mva} \tag{4.21}$$

$$\boldsymbol{P = P_a + P_v} \tag{4.22}$$

where a is the acceleration, to give the required maximum torque (τ) of the motors as:

$$\boldsymbol{\tau = \frac{P}{w}\frac{100}{e}} \tag{4.23}$$

where w is the angular velocity (rad s^{-1}), and e is the efficiency (%). The angular velocity is:

$$\boldsymbol{w = \frac{rpm}{60}2\pi} \tag{4.24}$$

This can be used to size the required motor. The total power required can also be used to size the battery as

$$\boldsymbol{P = IV} \tag{4.25}$$

where I is current and V is voltage. Substituting in Equation 4.23 and rearranging, the current required can be specified as:

$$\boldsymbol{I = \frac{\tau we}{100V}} \tag{4.26}$$

This can be used to calculate the required capacity of the battery using the following equation:

$$\boldsymbol{C = It} \tag{4.27}$$

where t is the required operation time of the robot.

Figure 4.11 shows the trade-off between weight and capacity for both lithium and lead-acid batteries. It also shows the required battery capacity that would be necessary to power a robot with a given weight (including the battery weight) for three hours (all parameters given in Appendix F).

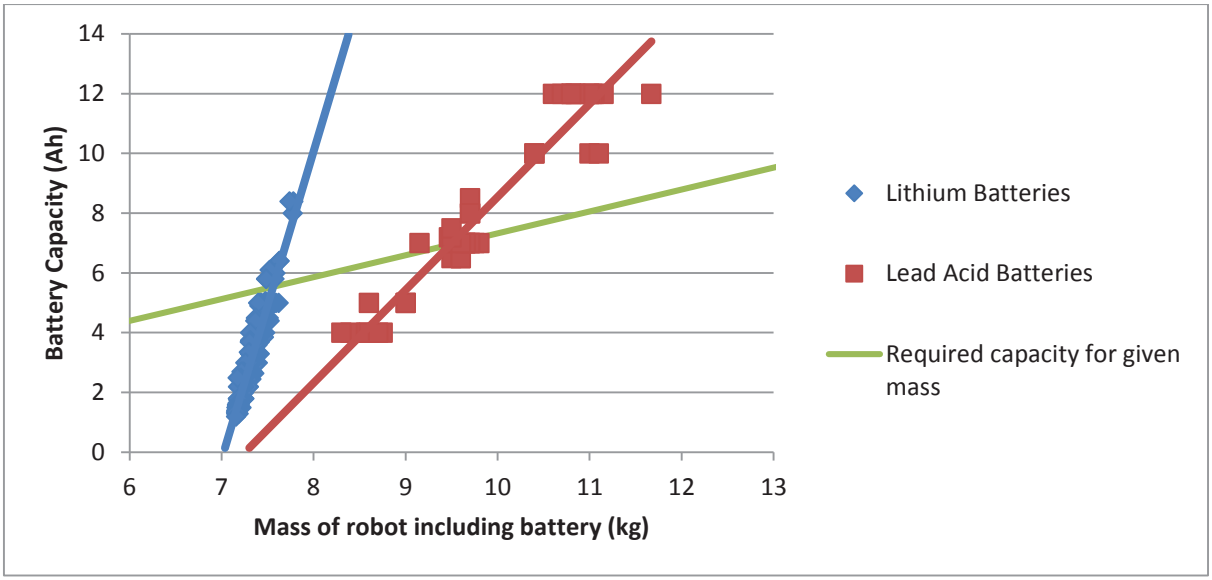


Figure 4.11 – Mass/capacity comparison of lithium and lead acid batteries

This shows, for the given parameters, that the optimum solution would be to use either a 5500mAh lithium battery or a 7Ah lead-acid battery. The advantage of using the lithium option is that the actual size of the battery will be smaller as well as lighter than its lead-acid counterpart. This means that the overall size of the robot can be reduced so that it is able to fit in smaller pipes. The disadvantages, however, are that they are more expensive, and they are currently limited in capacity to around 8000mAh. This means that if the payload of the robot, the run time, or other factors such as the maximum incline were to be increased, a bigger battery will be necessary and this may not be possible with lithium batteries. This would likely be the case if the size of the robot was increased for larger pipes leading to a much greater robot mass.

Chapter 5 Image System

5.1 Setup

5.1.1 Measuring erosion

Structured lighting approaches are usually implemented with a reference which is a planar surface. This makes measuring the profile of an object relatively simple by measuring the level of distortion of the light pattern from the reference plane. To apply the same technique to measure the profile of a pipe wall, a curved reference plane must be used as this represents the pipe wall without any erosion. This significantly increases the complexity of measuring the profile and before this can be done the effect of using a curved reference must first be understood by determining where the laser would appear in the image. Let the coordinates be defined as in Figure 5.1, with the x along the length of the pipe, y across the width of the pipe, and z vertically. Also, let r be the radius of the pipe, and θ and φ be the angle of the camera and laser respectively. Assume that where the laser intersects the pipe along the bottom (at the origin) is imaged to the centre of the camera's field of view.

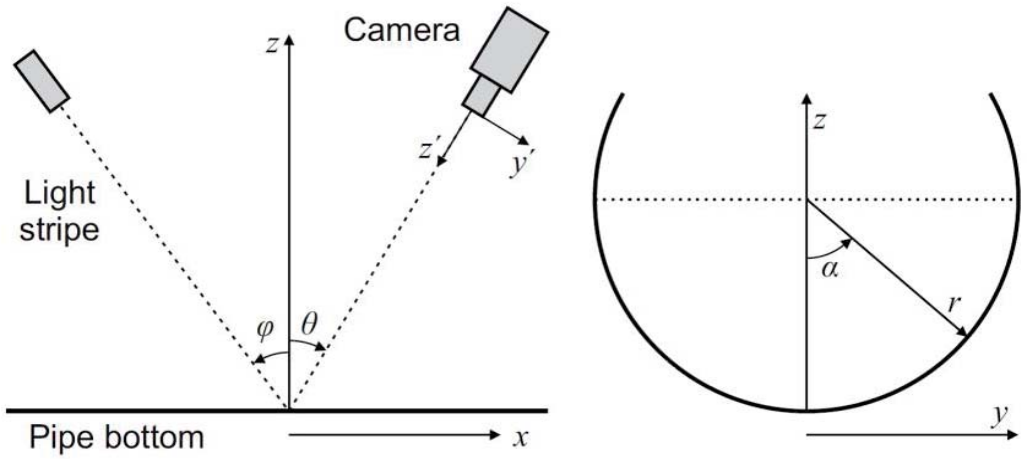


Figure 5.1 – Coordinate definition

The intersection of the laser stripe with the circular pipe wall will form an ellipse in the image. The position along this ellipse can be parameterised by α to give the points along the laser stripe as

$$\boldsymbol{y = r \sin \alpha} \tag{5.1}$$

$$\boldsymbol{z = r(1 - \cos \alpha) - d} \tag{5.2}$$

$$\boldsymbol{x = -z \tan \varphi} \tag{5.3}$$

where d is the vertical depth of any erosion at the point corresponding to α.

It is necessary to determine where these points would lie within the image and in order to do this these points can first be transformed into camera centred coordinates. For this, the camera position must be subtracted off and the coordinates rotated to align the axes with those of the camera. This transformation gives

$$x' = y \tag{5.4}$$

$$y' = (x - x_c)\cos\theta - (z - z_c)\sin\theta \tag{5.5}$$

$$z' = -(z - z_c)\cos\theta - (x - x_c)\sin\theta \tag{5.6}$$

where x_c is the distance along the x axis between the laser and the camera and z_c is the height of the camera from the origin. In this way the camera is located at (x_c, 0, z_c) and

$$x_c = z_c \tan\theta \tag{5.7}$$

Substituting equations 5.1, 5.2, 5.3 and 5.7 into 5.5 and 5.6 gives

$$y' = (r\cos\alpha - r + d)(\tan\varphi\cos\theta + \sin\theta) \tag{5.8}$$

$$z' = (r - r\cos\alpha - d)(\tan\varphi\sin\theta - \cos\theta) + \frac{z_c}{\cos\theta} \tag{5.9}$$

A pinhole model projects a scene onto an image plane through a small aperture that acts like a lens. This causes the image to be an inverted and reversed reconstruction of the scene as shown in Figure 5.2.

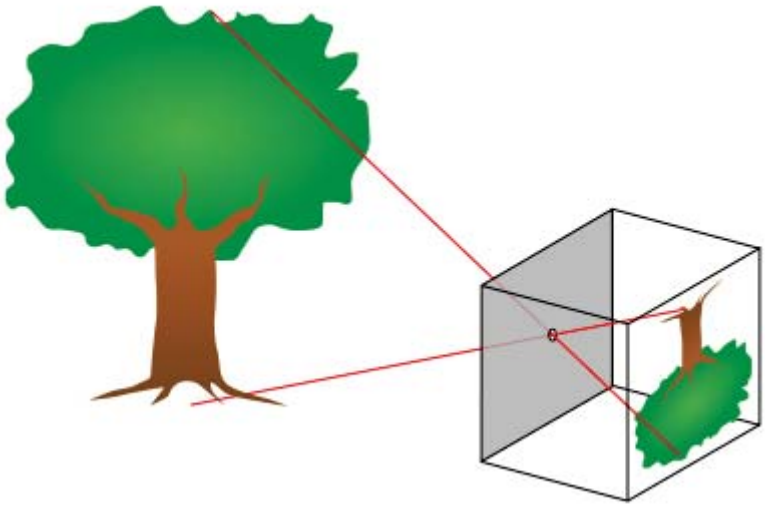


Figure 5.2 – Pinhole Model [57]

By using a pinhole model to represent the camera the projection of the laser stripe onto the sensor gives the image coordinates as

$$\hat{x} = f\frac{x'}{z'} \tag{5.10}$$

$$\hat{y} = f\frac{y'}{z'} \tag{5.11}$$

where f is the effective focal length of the lens (in pixels) and the origin for the image coordinates is defined to be the centre of the image. Also, $\hat{x}$ and $\hat{y}$ are the x and y coordinates in the image as defined by Figure 5.3.

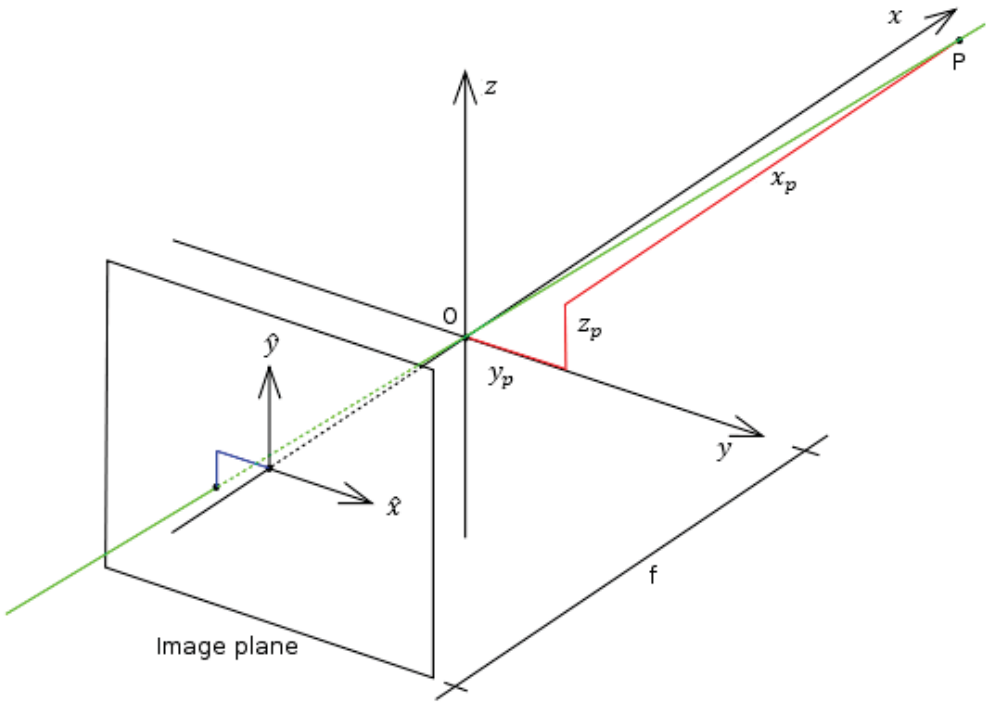


Figure 5.3 – Image coordinates definition [57]

This clearly shows that the lateral position within the image, y, is mapped to the horizontal position, $\hat{x}$, within the captured image. However, since the surface of the pipe is curved, the mapping is distorted by the scale factor z'. This scale factor is constant across the image if

$$\tan\varphi \sin\theta - \cos\theta = 0 \tag{5.12}$$

which requires that the principle axis of the camera be perpendicular to the plane containing the laser stripe, i.e.

$$\theta + \varphi = 90^\circ \tag{5.13}$$

Unfortunately, configurations such as these are impractical as the large angle makes occlusions resulting from large erosion more difficult to handle.

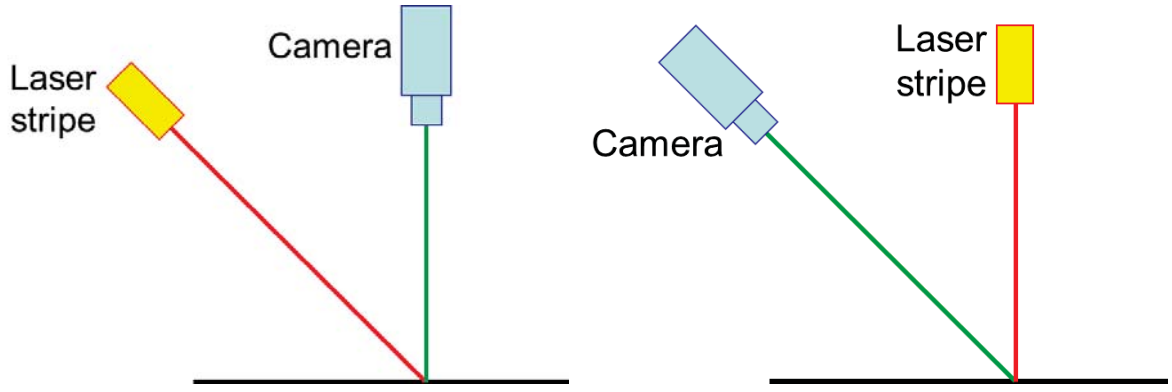


Figure 5.4 – Two primary configurations of laser and camera

Instead, there are two primary configurations that are of interest as shown in Figure 5.4. Both of these configurations will have a non-uniform distortion introduced by the z' term that will need compensation. The first of these configurations has the camera pointing directly downwards (θ=0) with the laser on an angle, and the other has the laser vertical (φ=0) and the camera on an angle.

5.1.2 Configuration 1: Camera set vertically

This configuration minimises the perspective distortion when looking at a horizontal plane as the principle axis of the camera is perpendicular to the plane. It is for this reason that this is the typical arrangement used for structured lighting approaches. An example of an image taken in this configuration in shown in Figure 5.5.

Figure 5.5 – Image taken using configuration 1

With this configuration the laser intersects the pipe at different positions along the pipe (x≠0). This is due to the combination of having the laser mounted at an angle and the curved surface of the pipe wall.

Since θ=0 in this configuration we can substitute this into Equation 5.10 and eliminate α to give

$$\hat{x} = f\frac{y}{\sqrt{r^2 - y^2} - r + d + z_c} \tag{5.14}$$

$$\hat{y} = f\tan\theta\frac{\sqrt{r^2 - y^2} - r + d}{\sqrt{r^2 - y^2} - r + d + z_c} \tag{5.15}$$

Assuming that the erosion represented by d in the denominator, is small relative to the other terms Equation 5.14 can be inverted to find the erosion as a function of $\hat{x}$ for given y values. This can be used to substitute into Equation 5.15 to give the mapping between the position in the image and the lateral position within the pipe. The presence of the y term in the denominator means that for increasing values of $|y|$ the magnification increases. This is because the pipe wall is closer to the image.

The offset of the laser in the image resulting from the erosion can be found by

$$\Delta\hat{y} = \frac{df\tan\theta}{\sqrt{r^2 - y^2} - r + z_c} \tag{5.16}$$

The shift of the line in the image due to erosion is approximately proportional to the actual depth of the erosion, although the scale factor is again dependent on the lateral position of the laser within the image.

5.1.3 Configuration 2: Laser set vertically

The second configuration positions the laser vertically with the camera on an angle. This provides the advantage that the laser always intersects the pipe at a fixed position along the length of the pipe regardless of any erosion that may be present (ie. x=0). An example of an image captured using this configuration is shown in Figure 5.6.

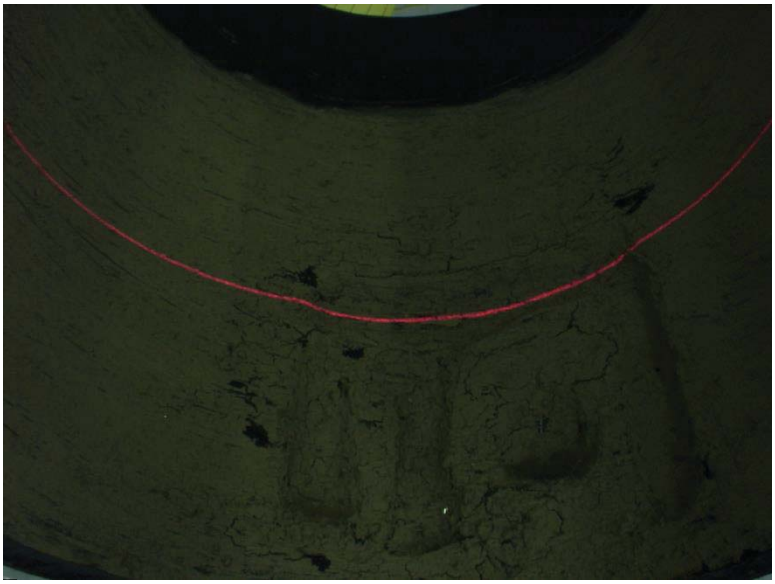


Figure 5.6 – Image taken using configuration 2

Since φ=0 in this configuration we can substitute this into Equation 5.10 and eliminate α to simplify to

$$\hat{x} = f\frac{y\cos\theta}{\left(\sqrt{r^2 - y^2} - r + d\right)cos^2\theta + z_c} \tag{5.17}$$

$$\hat{y} = f\tan\theta\frac{\left(\sqrt{r^2 - y^2} - r + d\right)\sin\theta\cos\theta}{\left(\sqrt{r^2 - y^2} - r + d\right)cos^2\theta + z_c} \tag{5.18}$$

Again, there is a position dependant magnification of the image. This magnification factor is less than the factor for the previous configuration because the camera is angled. Increasing the angle of the camera within the confines of the pipe allows the distance between the camera and the bottom of the pipe to be increased, which reduces the overall magnification while increasing the field of view of the camera.

As before, the shift of the line in the image is approximately proportional to the depth of the erosion

$$\Delta\hat{y} = \frac{df \sin\theta \cos\theta}{\left(\sqrt{r^2 - y^2} - r\right) cos^2\theta + z_c} \tag{5.19}$$

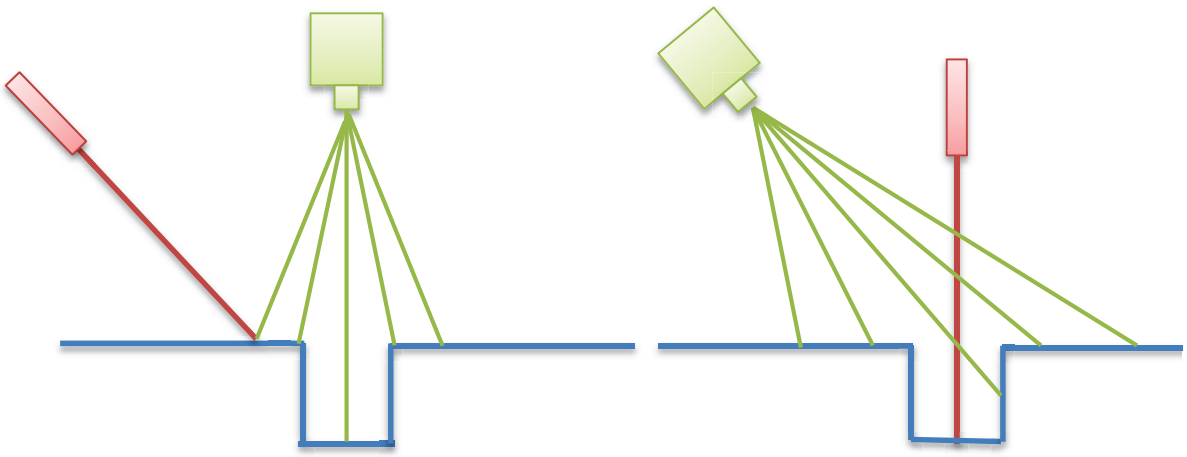
with the scale factor dependant on the lateral position within the image.

5.1.4 Configuration comparison

With both of the configurations there is a non-linear relationship between the lateral position in the image and the lateral position in the pipe. Similarly, when there is erosion present there is also a non-linear relationship between the depth of the erosion and the vertical position in the image. In the second configuration with the laser pointing vertically this distortion is less; however, this still needs to be corrected for in both cases.

An advantage with the second configuration is that with the camera on an angle the optical path length can be extended enabling a narrower field of view to be used to cover the same width across the pipe. This typically has two main advantages: firstly lenses with longer focal lengths are less susceptible to lens distortion, and secondly the actual focus is less critical. This also means that for any given camera lens, the field of view across the pipe is larger as can be seen by comparing Figure 5.5 with Figure 5.6. This means that the inspection system can inspect a larger portion of the pipe with a single camera. Because of this a single camera in this configuration may be all that is necessary since the erosion is typically limited to the bottom third of the pipe.

Both configurations are prone to occlusions especially when the angle between the camera and laser is large. Occlusions occur when there is deep erosion especially when this occurs suddenly. With the first configuration the camera is pointing vertically downwards and this means that the laser will often still be present in the image on the occluding surface enabling the line to still be detected. With the second configuration with the camera on an angle the laser may not be visible in the image at all and this must be accounted for in the processing.



Occlusion resulting from vertical laser

Occlusions resulting from vertical camera

Figure 5.7 – Comparison of occlusions occurring in configuration 1 and 2

On balance, the advantages of the vertical laser with an angled camera make it more suitable for this application.

5.1.5 Mathematical model development

The mathematical model that has been developed so far is useful for analysing the system and different setups but any further than that and its usefulness is diminished as the model is only sufficient to provide a loose approximation of the system. This is because of the many assumptions made about the precise positioning of the camera and laser in order to simplify the analysis. The assumptions that were made are:

- The laser is mounted perfectly vertical;
- The laser is mounted in such a way that it is projected exactly perpendicular to the direction of motion of the inspection platform (no rotation about z-axis);
- The camera is mounted perfectly in the centre of the pipe;
- The camera is rotated only about one axis (θ) and not the other two axes.

It is very unlikely that any of these parameters will be exactly zero. Rather, it is more likely that there will be a very small error, which will affect the image coordinates to a varying degree. It is necessary to consider all of these errors however small as they can impinge significantly on the overall accuracy of the system.

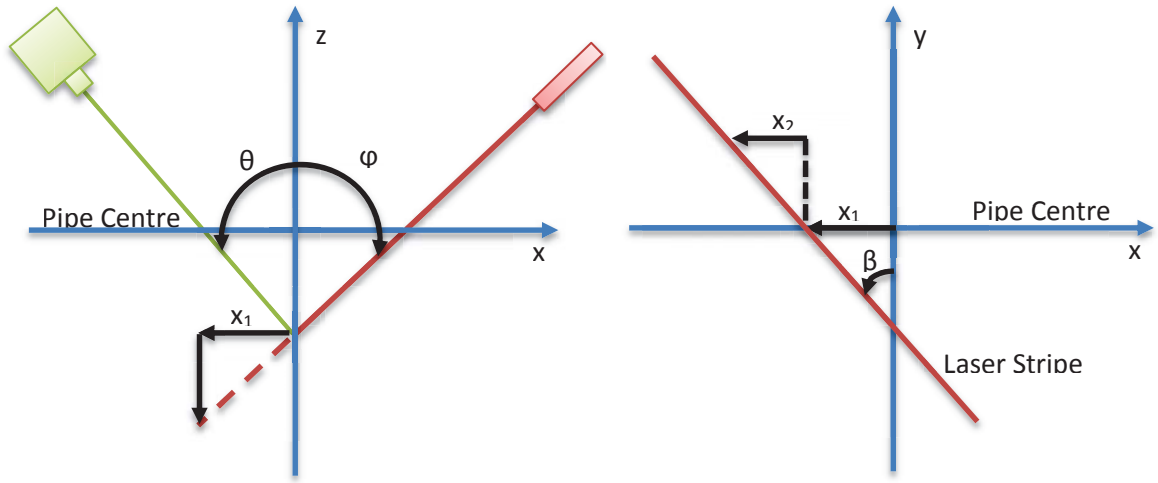


Figure 5.8 – New coordinate definition

The same coordinate system is used as is shown in Figure 5.1, and many of the parameters remain the same, although it is important to note a few key differences. From Figure 5.8

$$x_1 = -(z - r)\tan\varphi \tag{5.20}$$

$$x_2 = -y\tan\beta \tag{5.21}$$

where x_1 is the offset due to the erosion, x_2 is the offset due to the rotation of the laser, β is the counter-clockwise rotation of the laser on the x-y plane and φ is the clockwise rotation of the laser line on the x-z plane.

From this we can derive the x coordinates of the points around the cross-section of pipe as

$$\boldsymbol{x = -y \tan \beta - (z - r) \tan \varphi \cos \beta} \tag{5.22}$$

This takes into account the potential error in the alignment of the laser. With access to precision engineering methods it is possible to get the laser to be almost vertical relative to the inspection platform; however it is far more difficult to align the laser perfectly perpendicular to the x-axis due to most laser modules having cylindrical housings. A small error in this angle could easily propagate into a large error at the widths of the pipe.

From Figure 5.8 y and z can be derived using the same method as previously to give

$$\boldsymbol{y = r \sin \alpha + d \sin \sigma} \tag{5.23}$$

$$\boldsymbol{z = -r \cos \alpha - d \cos \sigma} \tag{5.24}$$

An importance difference here is that the level of erosion, d, has been defined as the distance along the line of sight of the camera and σ is the angle of the line of sight of the camera for a given point in the image. This can be defined as

$$\boldsymbol{\sigma = \alpha - \tan^{-1} \frac{r \sin \alpha}{r \cos \alpha + zc}} \tag{5.25}$$

By defining the erosion in this way the non-linear distortion of the image in the vertical direction is eliminated. This makes it far easier to accurately measure the level of erosion as each column of the image will always refer to the same physical x position. The position of the laser in each column will then give the actual erosion at each of those points.

To determine the effect on the image, it is necessary to transfer these points into camera centred coordinates. To do this the camera must be rotated through the rotational matrix R given as

$$\boldsymbol{R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x \\ 0 & \sin \theta_x & \cos \theta_x \end{bmatrix}} \tag{5.26}$$

$$\boldsymbol{R_2 = \begin{bmatrix} \cos \theta_y & 0 & \sin \theta_y \\ 0 & 1 & 0 \\ -\sin \theta_y & 0 & \cos \theta_y \end{bmatrix}} \tag{5.27}$$

$$\boldsymbol{R_3 = \begin{bmatrix} \cos \theta_z & -\sin \theta_z & 0 \\ \sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}} \tag{5.28}$$

$$\boldsymbol{\mathrm{R} = R_1 R_2 R_3} \tag{5.29}$$

This incorporates the three potential rotations of the camera where θ_x is the clockwise rotation on the y-z plane, θ_y is the counter-clockwise rotation of the camera on its x-z plane (formally θ), and θ_z is the clockwise rotation of the camera on its x-y plane. The axes of the camera are shown in Figure 5.9.

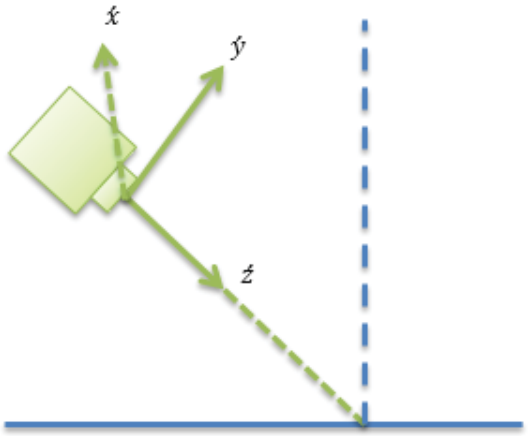


Figure 5.9 – Camera Axes

The offset of the camera in the x, y, and z directions must also be taken into account. These offsets are x_c, y_c, and z_c respectively such that the camera is centred at (x_c, y_c, z_c). The distance x_c is defined as the distance between the camera centre and the laser centre along the x axis. Similarly the distance z_c is defined as the height of the camera above the origin along the z axis. In this way it is not necessary to have separate parameters for the position of the laser as x_c is relative to the lasers position and the actual height of the laser is inconsequential. This gives the camera centred coordinates as

$$\begin{bmatrix} \acute{y} \\ \acute{x} \\ \acute{z} \end{bmatrix} = \mathbf{R} \begin{bmatrix} x - x_c \\ y - y_c \\ z_c - z \end{bmatrix} \tag{5.30}$$

It is important to note that after translation and rotation, the x axis becomes the y axis of the camera, the y axis becomes the x axis of the camera, and the z axis becomes the -z axis of the camera.

Finally, to give the image coordinates a pinhole model can again be used

$$\hat{x} = -f \frac{\acute{x}}{4\acute{z}} \tag{5.31}$$

$$\hat{y} = f \frac{\acute{y}}{2\acute{z}} \tag{5.32}$$

The scale factors of 0.25 in the $\hat{x}$ term and 0.5 in the $\hat{y}$ term are necessary to compensate for the column binning and skipping every second column and row as is explained later in 0.2.3. This gives the final expressions which can then be used to accurately map the points along the curvature of the pipe to an image, given accurate parameters.

The original definition of the erosion meant that there was a non-linear distortion in both the vertical and horizontal directions of the image. This would make calculating the erosion from this very difficult. Each individual point in any image would correspond to a different x, and y scale factor, making it necessary for a look-up table of scale factors the size of the image to be calculated and stored. If the erosion was found to be in a particular pixel, the scale factor in the corresponding cell of the table would be used to give the scaled level erosion. For a 640x480 image 307,200 values would need to be calculated, stored, and accessed, thus taking up valuable processing time and memory.

Using the new definition the level of erosion can be calculated by rearranging Equation 5.32 to give

$$\boldsymbol{d} = \frac{\Delta\hat{\boldsymbol{y}}}{\boldsymbol{k_1}\Delta\hat{\boldsymbol{y}} + \boldsymbol{k_2}} \tag{5.33}$$

where the erosion, d, is in millimetres, $\Delta\hat{y}$ is the difference between the actual data at that with no erosion, and $k_1$and k_2 are constants for each column. This means that a lookup table with only two values for each column needs to be used significantly reducing the complexity.

By making the same assumption as previously done (that is, to eliminate the parameter that are expected to be close to zero), k_1 and k_2 can be simplified to the following

$$\boldsymbol{k_1} = \frac{-\cos\boldsymbol{\sigma}}{\boldsymbol{r}\cos\boldsymbol{\alpha} + \boldsymbol{z_c} + (\boldsymbol{z_c} + \boldsymbol{r})\tan^2\boldsymbol{\theta}} \tag{5.34}$$

$$\boldsymbol{k_2} = \frac{\boldsymbol{f}\tan\boldsymbol{\theta}\cos\boldsymbol{\sigma}\left(\boldsymbol{z_c} + (\boldsymbol{z_c} + \boldsymbol{r})\tan^2\boldsymbol{\theta} + \boldsymbol{r}\right)}{\boldsymbol{2}(\boldsymbol{r}\cos\boldsymbol{\alpha} + \boldsymbol{z_c} + (\boldsymbol{z_c} + \boldsymbol{r})\tan^2\boldsymbol{\theta})^2} \tag{5.35}$$

The error introduced by making these assumptions is minimal and, in general, is less than a tenth of a millimetre. Without doing sub-pixel interpolation the accuracy of the erosion measurements is limited to a pixel. With the test setup described in Chapter 70 this accuracy corresponds to approximately 1mm, therefore for the purpose of this study, this error is insignificant.

Converting the erosion from pixels to millimetres could be performed either on the embedded processor on the inspection platform or once the data has been transferred to a PC. The advantage of performing the processing on the embedded system is that it generally reduces the range of values as the column can range from 1-480 which corresponds to an erosion of approximately -240 to 240mm in the centre of the image. The level of erosion will generally be limited to a certain range within this. Also, the erosion will almost always be positive (corresponding to a level lower than the pipe inner surface) unless there are miscellaneous objects or large stones in the pipe. By reducing the range of values, the data bits that need to be transmitted can be reduced. The disadvantage of performing this operation on the embedded processor is the increased processing time and complexity as well as increased memory requirements to hold the pre-calculated constants.

5.2 Algorithm

5.2.1 Algorithm development

Image processing is used to determine the level of erosion across a cross-section of pipe. This is done by extracting the position of the laser stripe in the image. This can be compared to a reference state with no erosion to determine the actual level of erosion in pixels. This can then be converted to millimetres instead of pixels to give useful and qualitative information for determining the extent of the erosion present. This conversion is non-trivial as it depends on both the horizontal and vertical positions within the image.

In general, the pixel in each column with the greatest red component will correspond to the centre of the laser stripe in that column. It is possible, however, that the background may have a greater red intensity in areas where the reflectivity of the inspection surface is low. Since the pipes of interest are made of concrete the background is grey in colour. This means that the red, green, and blue

components will all have relatively the same intensity and this will generally hold for the eroded sections of the pipe as well. By subtracting either the green or blue components from the red the background is effectively eliminated avoiding any scenarios where the background may be detected instead of the laser in the image. This will have little effect on the laser stripe itself.

The laser has a certain thickness in pixels in the image depending on the setup and it is necessary to extract the centre of the laser stripe. This can be done by filtering the image with an averaging filter the same width as the laser stripe in the image, which smooth's out the line and produces a more defined peak corresponding to the centre of the laser line enabling a single pixel wide profile to be extracted more reliably.

The profile can be extracted by determining the row number that corresponds to the maximum pixel value in each column, thus giving an accuracy to the nearest pixel. However, this can also be found to sub-pixel accuracy if necessary by interpolating between the pixels either side of the centre of the line.

As mentioned previously, it is possible for occlusions to occur and this must be incorporated into the algorithm. Potential occlusions are identified by comparing the intensity of the maximum value in each column to a pre-set threshold which indicates the minimum expected intensity value. If it is below this threshold, and the position of this maximum in the image is too far from the expected location of the laser based on the surrounding data points, it is flagged as a potential occlusion. The way in which these flagged occlusions are handled depends on the user and the purpose of the inspection but these will generally have to be more closely inspected by a human operator.

The reference level, which is that of the pipe without any erosion, can be subtracted from the resulting profile and the erosion profile can be determined using the Equation 5.33. This effectively converts the erosion from pixels to millimetres. This process is shown in Figure 5.10, Figure 5.11, and Figure 5.12.

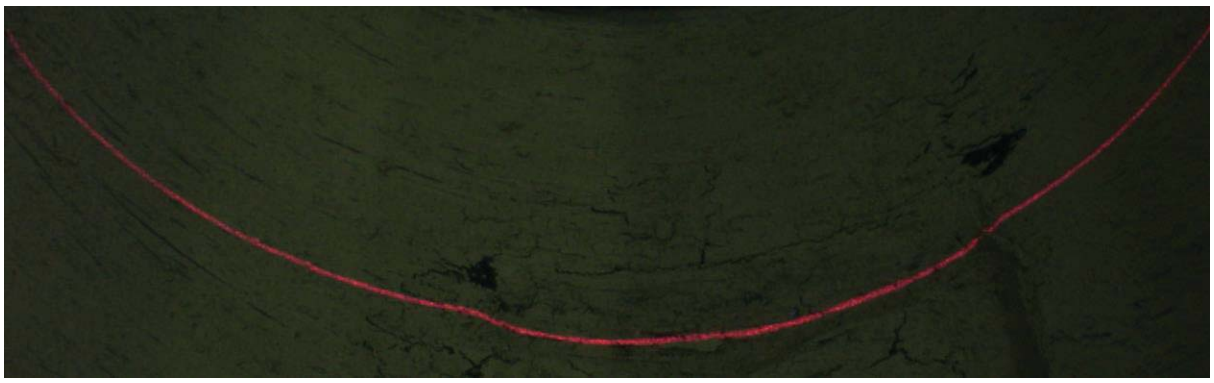


Figure 5.10 – Captured image

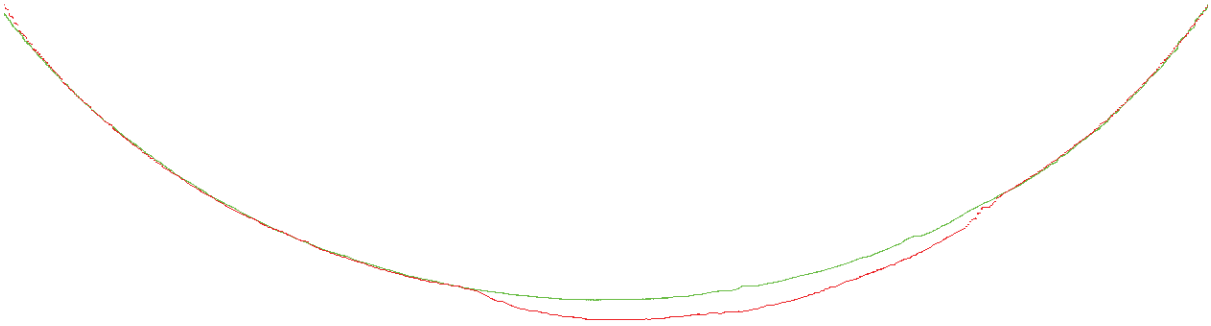


Figure 5.11 – Reference profile (green) and laser stripe (red)

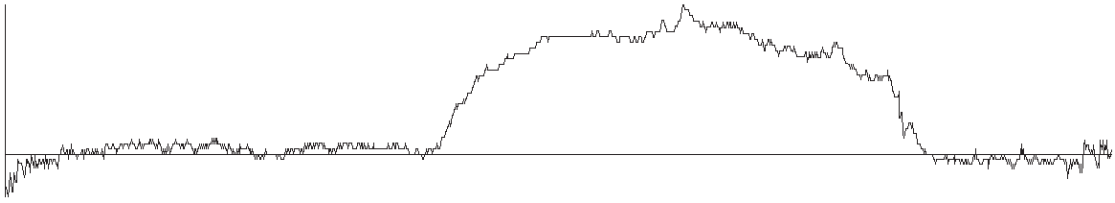


Figure 5.12 – Difference between reference and laser

Figure 5.10, shows the captured image of the pipe. Figure 5.11 shows the reference curve in green along with the profile extracted from the captured image in red. This line is simply the laser line from the image. The difference between the reference line and the laser is shown in Figure 5.12. This represents the level of erosion across the pipe measured in pixels and is the erosion profile in pixels.

5.2.2 High speed considerations

The algorithm adopted has been designed for high speed implementation. This is achieved by keeping the image processing relatively simple and allowing it to be applied in real-time. Processing the image in real-time means that it is not necessary to write the whole image to memory and then read it back while processing. Instead, the processing can be performed on the data as it is streamed from the camera so the processing on a frame can be completed almost as soon as the frame itself is captured.

To aid the image processing, the algorithm has been optimised to perform the processing in parallel. This means that the processing, such as the averaging filter, can be applied to each column in unison instead of consecutively. This significantly reduces the latency, although at the expense of a small amount of memory to maintain the processing state for each column.

This algorithm allows the volume of data that needs to be stored, processed, and transmitted to be reduced early on in the processing cycle. This requires that the processing must be done in real-time, on the robot itself. A conventional processor (such as a microcontroller) is not suitable for processing on the robot. A small, low power processor is required that can handle the large volume of data and processing required. A solution to both of these problems is to convert the image from a 2D image to a 1D profile using an FPGA. The FPGA that was used for this is an Altera Cyclone III on a Terasic DE0 development board. Using an FPGA means that the processing is done in hardware which can utilise parallel processing, which is not possible with software processors.

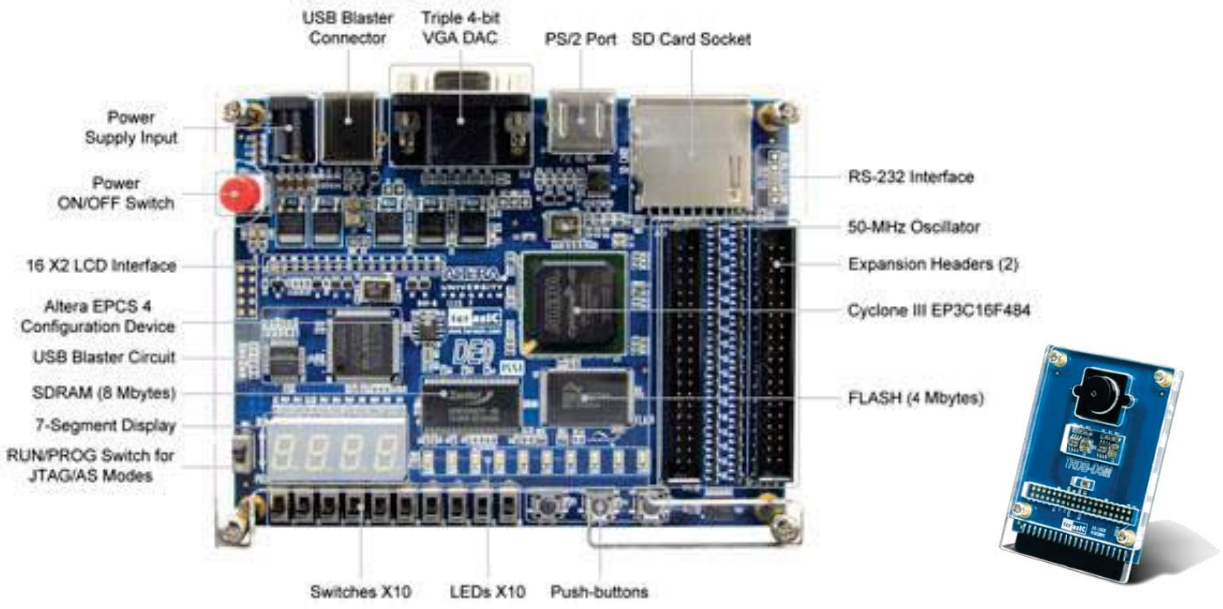


Figure 5.13 – Terasic DE0 FPGA development board [58] with Terasic D5M camera [59]

If the data is stored locally on the robot itself, such as by writing to an SD card which is supported using the DE0, then the maximum frame rate is 49fps. This means that if a profile is taken every ten millimetres then one metre of pipe can be scanned every two seconds. If the data is transmitted serially, on the other hand, the frame rate would be limited to approximately 17fps with a baud rate of 115200. This means that with the same resolution it would take six seconds to scan one metre of pipe. This could be reduced by as much as half if suitable compression techniques are employed, leading to comparable scan speeds.

5.2.3 Algorithm implementation

The VGA output of the FPGA development board was used to display the output from the camera for different stages of the image processing. This is useful as it means that the whole image does not need to be transmitted to a PC for debugging and testing. This requires the use of a 25 MHz clock, however, the camera has a maximum pixel clock of 96 MHz and it is preferable to run this as high as possible in order to increase the frame rate. Fortunately, the DE0 has the ability to run separate domains at the same time with different clock speeds, and it has an on-board clock of 50 MHz. A 25 MHz clock domain was set up to run the VGA by dividing this system clock by two, and an 86 MHz clock domain was set up for the camera and image processing using a PLL (phase locked loop). A clock of 86 MHz was used for this instead of the maximum 96 MHz due to the limitations imposed by the ribbon cable connecting the camera to the FPGA. Using different clock domains does introduce some difficulties. For example, memory can only be read from or written to from one clock domain. Transferring data from one domain to another can be achieved either by using channels or dual-port memory. Channels work by allowing one parallel branch to output data onto the channel and another branch to read data from the channel. They are useful for synchronising different clock domains as the channel will only transfer if both branches are ready. This means that if the output branch is ready to transfer before the input branch is ready to read a new value it will wait until the input branch is ready before continuing. In this way, channels can be exploited to allow the system to work at the maximum possible speed by

allowing the processing to occur in a faster domain and still be synchronised to the VGA output (as the processing may take several clock cycles).

Instead of capturing an image from the camera and placing it in a frame buffer before processing, the images can be processed directly as they are streamed from the camera. This significantly reduces the memory requirements as a whole image does not need to be stored in order to process it. Instead only a few rows of the image may be required. It also improves the latency as all of the processing has been performed on each frame by the time it has finished reading out from the camera. The processing logic for implementing the developed algorithm in this manner is outlined in Figure 5.14, and the code written for implementing in Handle-C can be found in Appendix C.

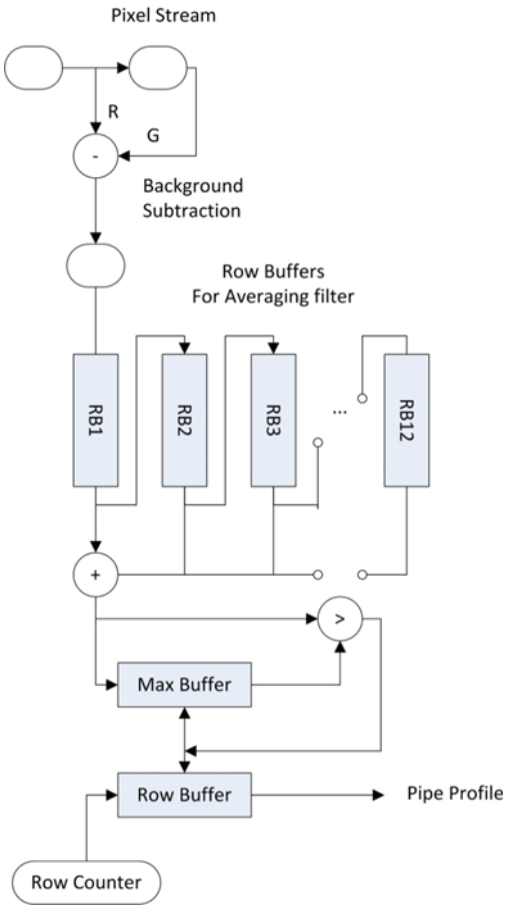


Figure 5.14 – Processing logic for measuring pipe profile

The D5M camera has a CMOS sensor with a resolution of 2592x1944 pixels with 12 bits per pixel output. The entire image does not need to be read out. Instead, a rectangular window of just the relevant area can be read out to increase the frame rate. In this case the entire width is required to maximise the inspection coverage, however it is not necessary to use the entire height of the image as it is very unlikely that the erosion would extend to this extent. Column binning is also used to increase the sensitivity in low light conditions. This works by combining the values in adjacent cells so the horizontal resolution is halved but the sensitivity is increased.

The D5M camera is a single-chip camera which obtains a colour image using a Bayer pattern colour filter such as the one shown in Figure 5.15. Every pixel represents only one colour, and since it is the red pixels that are of interest, every second row can be skipped. The green pixels are then subtracted

from the adjacent red pixels, to eliminate the background, using a simple horizontal filter which produces an output every second clock cycle. This leaves an effective image resolution from the red pixels of 648x972 resulting in a lateral resolution of approximately 0.5mm, and a depth resolution of better than 0.5mm.

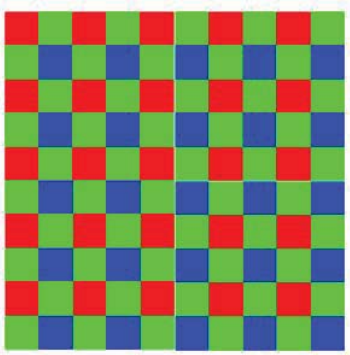


Figure 5.15 – Bayer pattern [58]

The next stage is to smooth the image vertically, and find the maximum value in each column. Instead of loading the whole image and processing the columns individually, the parallel characteristics of the FPGA can be exploited to process all of the columns at the same time as the data comes from the camera. In this setup the laser line was approximately 12 pixels wide so a 12x1 vertical smoothing filter was used which requires buffering the previous 11 rows to process with the current one. Instead of actually averaging the pixel values it is sufficient to simply sum the values within the window. This effectively has the same result and removes the need to divide by 12.

The search for the maximum value in each column requires maintaining two additional buffers: one to store the maximum value found so far in each column, called the max buffer; and the other to store the position, or row number, of this maximum within the column, called the row buffer. Once the first 12 rows have been smoothed, the max buffer and the row buffer are initialised to zero. For the subsequent windows, the current value is compared to the value stored in the max buffer for that column and if it is larger than the max buffer, both the max buffer and the row buffer values are replaced with the current row's value and position. At the end of the image the max buffer contains the maximum filtered red component for each column and the row buffer holds the row number of that maximum for each column. At this point the row buffer contains a profile of the pipe. This profile must be shifted left by two and up by twelve pixels to account for the processing latency of the image processing.

The resulting images from the above steps are shown in Figure 5.16, Figure 5.17, Figure 5.18, and Figure 5.19. These images show the red component, the green component, the result from subtracting the green from the red, and finally, the result of the averaging filter (in red) with the extracted profile overlaid (in green).

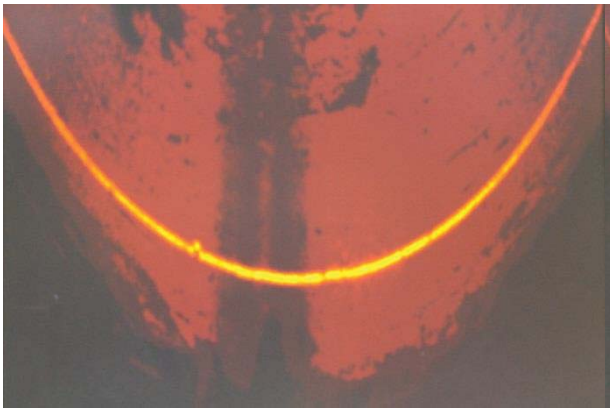


Figure 5.16 – Red component

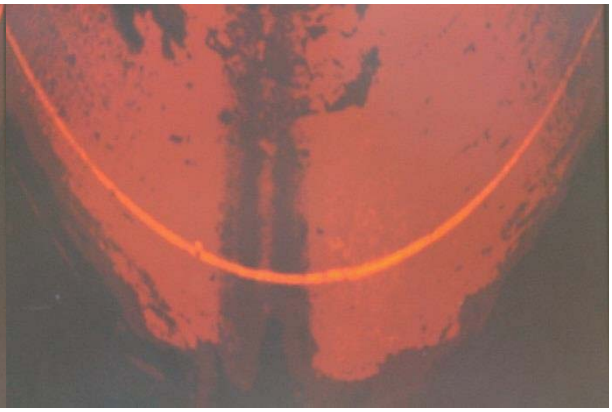


Figure 5.17 – Green component (shown as red for comparison)

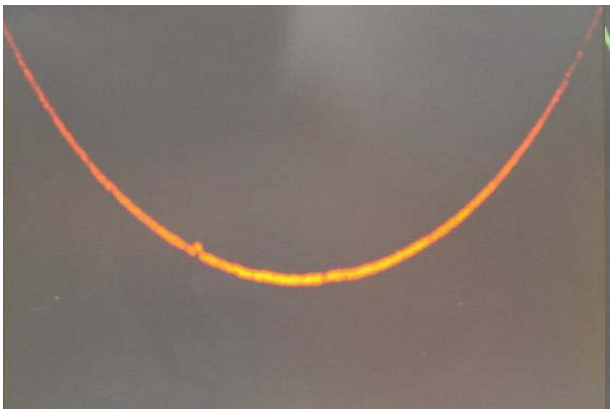


Figure 5.18 – Red minus green

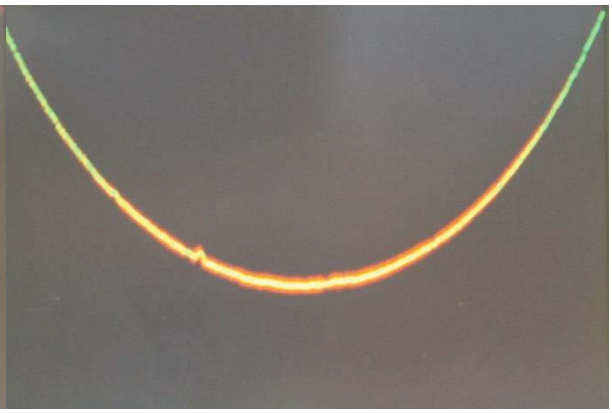


Figure 5.19 – Smoothing (red) and profile (green)

At this point the data was transmitted to a PC using an RS232 protocol and the remaining processing was performed on the PC. This could, however, also be included on the FPGA if necessary following the logic shown in Figure 5.20.

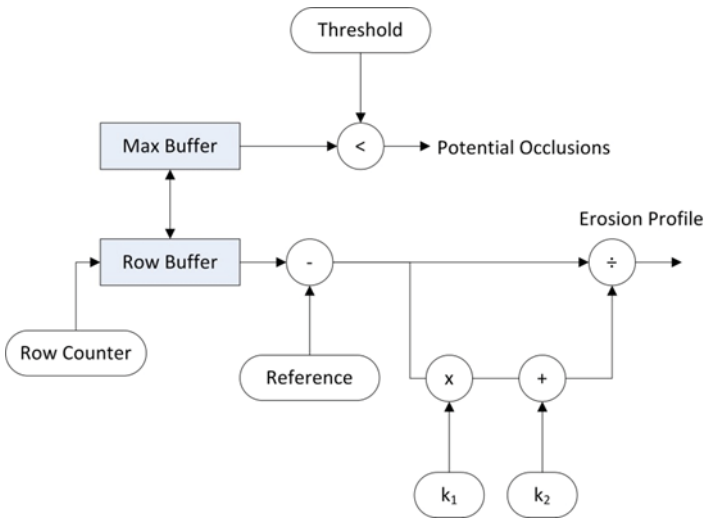


Figure 5.20 – Processing logic to flag occlusions and derive erosion profile

The values in the max buffer and row buffer can be used for comparing against a pre-set intensity threshold and expected position to identify and flag potential occlusions. The erosion profile can then be determined by subtracting the reference profile which represents what the profile would look like without any erosion. The erosion can then be calculated in millimetres by taking the output multiplied by k_1 plus k_2 and dividing into the output as per Equation 5.33 where k_1 and k_2 can be pre-calculated using Equations 5.34 and 5.35 and stored locally. Due to the processing complexity these calculations would not be implemented on the FPGA but would have to be calculated on a PC and transferred to the FPGA for use.

5.3 Reference profile

The developed algorithm calculates the erosion profile by subtracting the reference curve from the pipe profile. This requires a reference curve that accurately portrays a cross-section of pipe without any erosion with the given setup. This reference curve could be derived by simply taking an image of a non-eroded section of pipe and using the resulting profile. However, this is prone to error and this will be reflected in the erosion profile for the pipe. Also, it is not always possible to find a non-eroded section of pipe. An alternative to this method is to generate a reference curve from the mathematical model using measured system parameters. However, since it is impossible to accurately measure these parameters it is useful to take an image of a section of pipe which has little erosion and use it to calibrate the reference curve given from the mathematical model.

The calibration uses the initial approximations of the parameters and optimises these to fit the curve given from the image. The output parameters are considered as the calibrated parameters and are used to determine the reference profile by applying them to the mathematical model. It is only necessary to perform this calibration once for each setup or when the parameters may have changed significantly. This means that the calibration does not need to be performed in real-time; rather it could be processed utilizing the processing power of a PC and then transferred to the FPGA to be used in real-time.

5.3.1 Non-linear optimisation

Non-linear optimisation is used to estimate the values of the model parameters for the calibration of the system. The aim of the optimisation is to minimise the sum of the squared error, and since the model is non-linear so is the optimisation. The error is defined as the difference between the data from the image representing the non-eroded pipe and the data generated using the mathematical model. For any given y value across the pipe, the erosion at that point is represented by the data in the corresponding column, or $\hat{x}$ value, using the definition where the erosion is along the line of sight of the camera. This means that the optimisation is only concerned with the data within the same column and this simplifies the calibration. This gives the following equation to minimise

$$\boldsymbol{error} = [\hat{\boldsymbol{y}}_{\boldsymbol{m}} - \hat{\boldsymbol{y}}]^{\boldsymbol{2}} \tag{5.36}$$

where $\hat{y}_m$ is the y value calculated from the model and $\hat{y}$ is the y value within the image with the corresponding $\hat{x}$ value.

For any given image the $\hat{x}$ values are known and consist of consecutive integers corresponding to each column. In contrast, the model derived $\hat{x}_m$ and $\hat{y}_m$ values are parameterised by the angle α around the pipe. It is necessary to have the $\hat{y}_m$ values from the model corresponding to the $\hat{x}$ values from the

image so it is necessary to find α as a function of $\hat{x}$. In this way, the $\hat{x}$ values from the image can be used to find the correct α values to use in order to calculate the corresponding $\hat{y}_m$ values. Equation 5.32 can be used to calculate the $\hat{y}_m$ values and can also be written as

$$\hat{y}_m = f(f, r, \alpha, \varphi, \beta, \theta_x, \theta_y, \theta_z, x_c, y_c, z_c) \tag{5.37}$$

Since Equation 5.31 is hard to invert it is solved numerically to be in the form

$$\alpha = f(f, r, \varphi, \beta, \theta_x, \theta_y, \theta_z, x_c, y_c, z_c) \tag{5.38}$$

By substituting Equation 5.37 into 5.36, an equation for the error is derived which can be minimised to give the reference curve. This equation has ten degrees of freedom so the optimisation routine is time consuming. This makes a PC more appropriate for the calibration process than an embedded processor and Matlab is an ideal platform for accomplishing this. The code developed in Matlab for this process is shown in Appendix B.

Figure 5.21 shows the data generated from the model using the initial conditions (shown in black) overlaid on the actual image data from a non-eroded section of pipe (shown in red). This shows a significant amount of error which will not allow for successful erosion detection. This is contrasted to the data generated using the optimised parameters (in red) overlaid on the same image data (in black) on Figure 5.22.

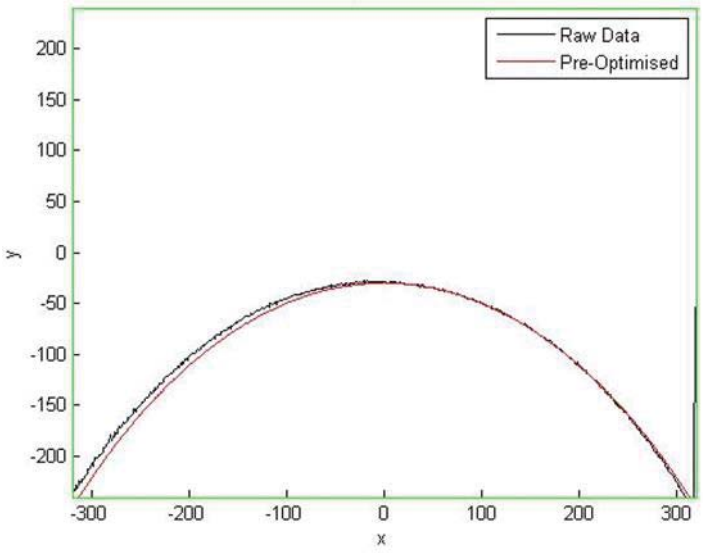


Figure 5.21 – Image data (black) along with curve generated from initial parameters (red)

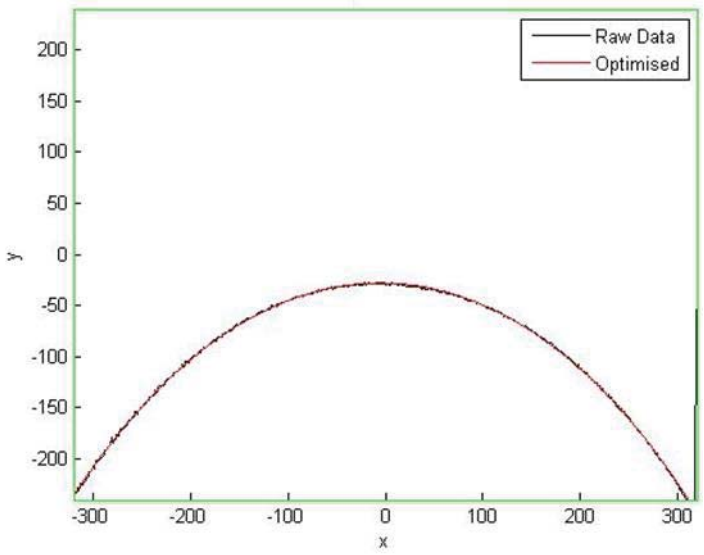


Figure 5.22 – Image data (black) along with curve generated from optimised parameters (red)

Table 5.1 shows the initial conditions used to generate the profile in Figure 5.21 as well as the optimised parameters to generate the profile in Figure 5.22. The sum of squared error residuals is also shown for each case. From this it can be clearly seen that the optimisation has significantly reduced the average error of the system. There may still be a small amount of error, however this is insignificant and will still allow the inspection to successfully identify areas and extent of erosion within a small tolerance governed by the success of calibration.

Table 5.1 – Initial parameters and optimised parameters along with total error squared for each

Parameter	Initial	Optimized
f	3326	3325.9998
φ	0	4.8574
β	0	0.4956
x_c	-244	-244.0021
y_c	0	0.0004
z_c	125	125.0003
θ_x	0	-0.3498
θ_y	40	43.0297
θ_z	0	-0.1959
Total Error Squared	14319.9743	747.0813

5.3.2 Relationship and dependency testing

The goal, when preforming an optimisation, is to find a global minimum where the error of the entire system is minimised. It is, however, possible to find, and get trapped in, a local minimum which is not the desired outcome. This brings about the need to test for this, and this test can be done by exploring the effect on the outcome of inducing a small change on the individual input parameters of the optimisation. If the outcome is the same set of parameters then it is likely that the optimisation has, in fact, converged to a global minimum. On the other hand, if the parameters have now changed it is likely that either of the results was a local minimum. Different sets of changes could then be combined to further explore this possibility. Because of the large degree of freedom of this system this could encompass a large amount of possibilities and it may be hard to determine whether it is a local and

global minimum. Whether a local or global minimum is converged on depends entirely on the initial conditions so this process must be performed every time the system is calibrated.

As long as the error between the image data and the generated profile is small it may not matter if the optimisation has converged on a local or global minimum. If the generated profile closely matches the image data then it should be sufficient for use as the reference profile as long as the image was taken from a clean section of pipe with very little erosion. This should be equivalent to simply using the image data as a worst case scenario.

It is also possible that there is more than one optimal set of parameters. This may be due to relationships between parameters within the optimisation equation making the use of some parameters redundant. If this is the case it is necessary to determine the relationships and eliminate the unnecessary parameters if possible. An example of such a relationship might be if either φ or θ_y could be altered to achieve the same optimum. In this scenario there is obviously only one degree of freedom described by the two parameters so the two parameters should be combined to create one parameter representing the angle between the laser and the camera.

It is possible to test for these relationships by fixing each of the parameters in turn with the optimised value with a small error. This parameter is then excluded from the optimisation. If the optimisation converges to the same minimum it is likely that there is indeed a relationship between the excluded parameter and one of the remaining parameters. This remaining parameter can be determined via further evaluation.

This test was performed on the mathematical model developed in Chapter 5.1.5, however, there was no conclusive evidence that any relationships exist. It is possible that complex relationships exist in this model, such as between sets of parameters and not just individual parameters. One method for eliminating the effects of any potential relationships is by performing an optimisation of only the key parameters that are likely to have a significant influence. These optimised parameters are then fixed and a further optimisation performed to tune the remaining parameters. The parameters excluded initially in this case are those close to zero so the initial optimisation involves f, r, φ, θ_y, x_c and z_c; and the secondary optimisation involves β, θ_x, θ_z, and y_c.

Chapter 6 Communication System and Software Development

6.1 Communication system

6.1.1 XBee module

Zigbee based XBee modules are an ideal solution for embedded communications as discussed in Chapter 3. ZigBee networks are called personal area networks (PAN) and each network must have a unique 16-bit identifier called the PAN ID. These networks are made up of nodes and these nodes can be one of three types: a coordinator, router or end device. An example of a ZigBee network is shown in Figure 6.1.

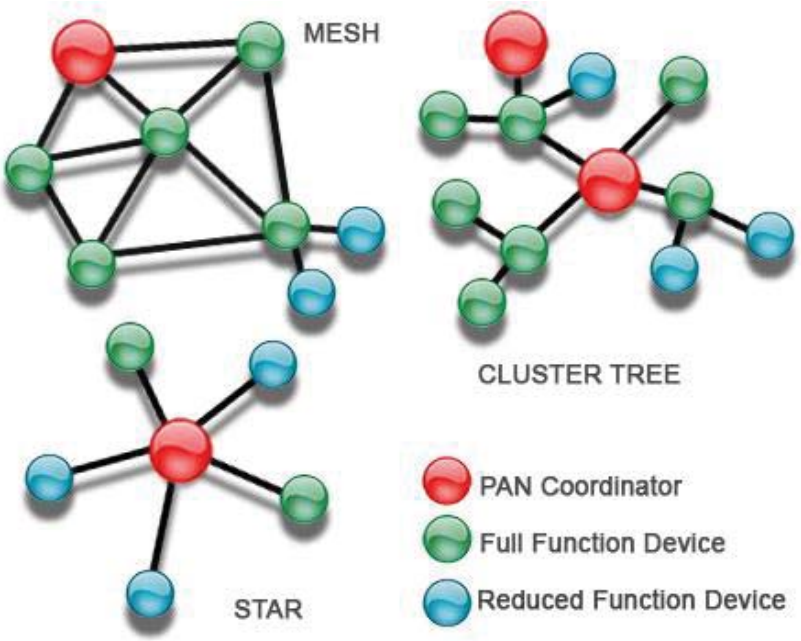


Figure 6.1 – Basic ZigBee network topology [59]

There is only one coordinator for each network and this is responsible for selecting the PAN ID and channel. Once a coordinator has started a PAN it can allow other router and end device nodes to join the network. The coordinator can send and receive data and it can also assist in routing the data through the mesh network. Usually the coordinator will be mains powered as opposed to battery powered as it must be able to allow joins or route data.

Router nodes are much like the coordinator except they cannot start a new PAN. Once they have connected to an existing PAN they can also allow other routers and end devices to join. Once again, routers can send and receive data transmissions as well as assisting in the routing of data between nodes. Routers are usually mains powered devices for the same reasons as the coordinator. That is, they cannot sleep as they allow other devices to join the network and are key for routing data between nodes.

End devices, on the other hand, cannot allow other devices to join a network and are not involved in routing data between nodes. Rather, once they have joined a PAN they can only send or receive data. A unique characteristic of an end device is that they can go to sleep. When it does sleep, the end device's parent, which is the device (either router or coordinator) that allowed the end device to join

the network, must collect and buffer all data packets intended for that device until it is awake again. This characteristic of end devices makes them ideal to be battery powered.

The simplest PAN has two nodes: a coordinator and an end device and acts much the same as an 802.15.4 point-to-point network. In this case the end device would be associated with the inspection platform and would be battery operated. The coordinator would be located at the control end attached to either a laptop or a local area network linked to the control PC. Additional router nodes could be added to the network in order to extend the range of the network from end device to coordinator or to accommodate corners in the pipe network, etc. This could be extended further in the future for similar inspection systems to allow multiple inspection devices to work and communicate at the same time to form interactive nodes inspecting separate sections simultaneously.

Unlike series one XBees, series 2/2.5/ZB XBees are not designed to work straight out of the box. They first require a small amount of configuration. This configuration is done by setting the value of given registers on the XBee, either by using a serial interface or by using the specifically designed X-CTU software [60]. The configuration consists of setting a PAN ID (although this is not strictly necessary as the coordinator can obtain a free PAN ID and channel automatically), configuring the devices as either coordinator, router or end device, setting the node identifier (name used to identify each node), and configuring the network type as either transparent (AT) mode or as application programming interface (API) mode.

In AT mode, communication is limited to point-to-point communication between two XBees. Any given node can only send data to the node whose serial number is programmed into its destination address register. While this register can be altered using the appropriate AT commands, the transmitting node must be connected directly to the receiving node for this to work. In contrast, API mode can send data from any node in the network to any other node in the network. However, this means that the data must be structured into packets as shown in Figure 6.2. The specific operation of API mode is outlined in [53].

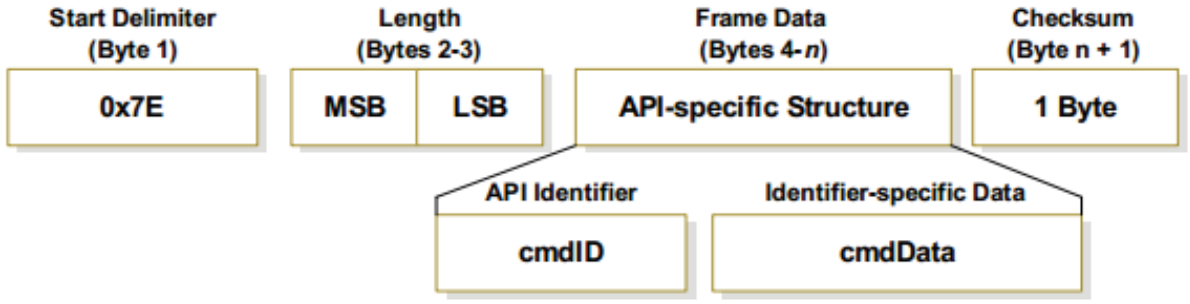


Figure 6.2 – API packet structure [53]

Another consideration to be made when choosing an appropriate XBee module is the type of antenna as this depends on the operating environments and affects the overall range of the wireless system. The XBee modules have three main antenna options; a PCB chip antenna, a ¼ monopole integrated whip antenna, or a U.FL antenna connector which allows a dipole or other external antenna to be connected. The whip and chip antennas are ideal for embedded applications and can be housed in plastic cases as they have no problem radiating through plastic. If a metal case is used then an externally mounted antenna would be preferable. Whip antennas yield a larger wireless range than chip antennas as they have a larger gain. In the case of a series 2 XBee the gain of the chip antenna is

-1.5dB and the gain using a whip antenna is 1.5dB. This difference of 3dB means that the whip antenna will have a receiver and transmitter power greater than that of the chip antenna by a factor of two. Using a whip antenna and a standard series two XBee, a wireless range of over 100m could easily be achieved and this is more than sufficient for this purpose.

6.1.2 Using RS232 for serial operation

XBees are essentially serial-based wireless communication and, as such, can be used for a drop in replacement for serial cables. Since data is inherently parallel and is used in this manner (e.g. a Byte is a parallel data structure of 8 bits) the data needs to be separated before transmitting and reassembled on the receiver end. This is done through the use of a Universal Asynchronous Receiver/Transmitter (UART) which is a hardware component used to translate data between parallel and serial. The XBees all have built in UARTs for accomplishing this. UARTs are usually used in conjunction with a serial communication standard. The most common of these in conjunction with using a PC interface is RS232.

RS232 uses a pair of wires to transmit data serially. The outgoing data is labelled TX for transmission, and the incoming data is labelled RX for receiving. A minimum of three wires are needed for two-way communication. These are TX, RX, and ground. The TX and RX lines are crossed over between two components which allows them to talk to each other.

As RS232 is asynchronous there is no clock signal. This means that the transmitter is not synchronised to the receiver and both ends of the link must be set up the same in order for the receiver to decode the data stream. The RS232 procedure works as illustrated by Figure 6.3.

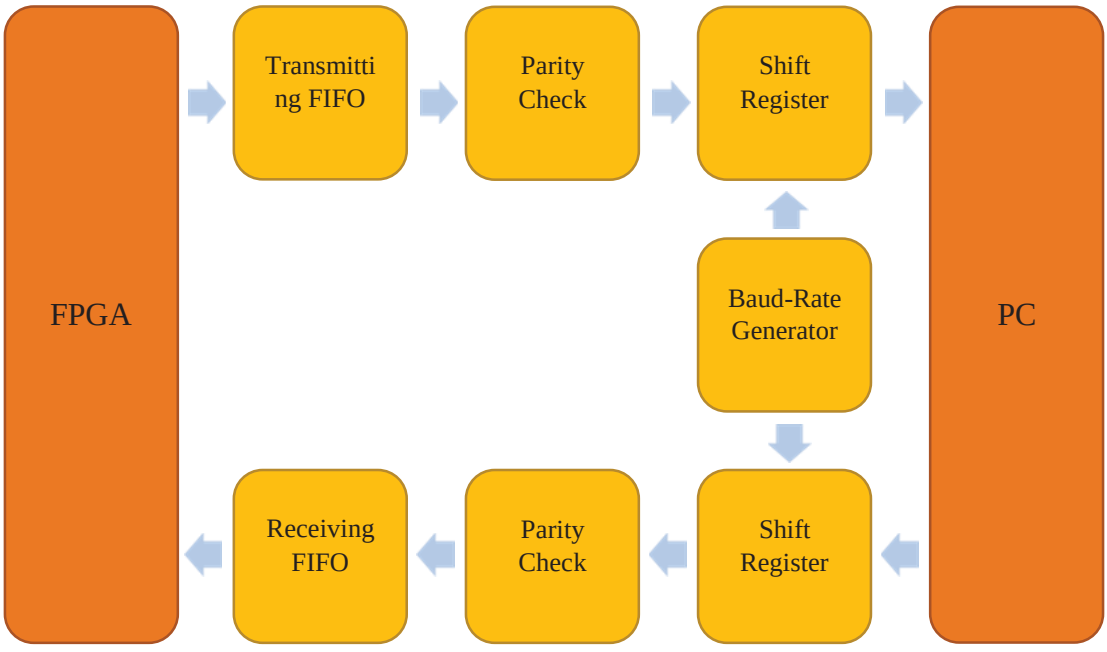


Figure 6.3 – RS232 Procedure

To allow this type of asynchronous communication to operate efficiently a Baud rate is used. This rate governs the speed at which a signal can change state and is effectively the “symbols per second” rate. The Baud rate must be set the same on each device in order for these devices to communicate. This

tells the device at what speed to send or receive data. A Baud rate of 9600 gives a frequency of 9600Hz and a bit period of 104.17us. Although this is a common rate to use by default, the Baud rate can commonly range from 1200 to 115200 for RS232.

The structure of an RS232 transmission is shown in Figure 6.4. During periods in-between transmissions the TX line is held high. To begin transmission the transmitter sends a low logic zero for one bit duration. This indicates to the receiver that data is about to follow. This start bit allows the receiver to synchronise to the incoming data by creating its own sampling clock at sixteen times the Baud rate. This allows the receiver to sample the incoming stream in the middle of each bit but it also allows the receiver to re-synchronise its sampling clock to match any discrepancies between the receiver and transmitter by detecting rising and falling edges of the data.

The start bit is then followed by the data word with a length ranging from five to eight bits long. This length must be known by both the transmitter and receiver before communication is established. The most common word length is 7 or 8 bits as 7 bits is required to represent an ASCII character and the eighth bit extends the character set for graphical symbols. Word lengths of 5 or 6 bits are less common but can still be used. This length generally defaults to 8 bits.

The data bits can then be followed by a parity bit. This is used as a crude form of error detection. It is not necessary to use this bit, however when it is used, it can be either odd or even parity. When using even parity, the number of bits whose value is one are counted up. If this is odd then the parity bit will be set to one to make the total count of ones even, but if the count is already even then the parity bit will be set to zero. When using odd parity, the opposite of this is true and the parity bit will be set so that the total number of ones is odd instead of even. The parity bit is then used at the receiver end to determine if any errors have occurred during transmission using the same process. The problem with this type of error detection is that it can only detect an odd number of errors. If there is an even number of errors they will cancel each other out and not be detected. When transmitting in noisy environments or over large distances it may be useful to use other forms of error detection such as cyclic redundancy check (CRC) to overcome this.

CRCs work by calculating a check value for a block of data based on the remainder of a polynomial division of the data. This calculation is re-evaluated when the data is received and if it is a different value then actions can be taken to deal with the corrupted data. CRCs are particularly useful as they are easy to implement, simple to calculate and analyse mathematically, and are particularly good at detecting common errors caused by noise during transmission.

Finally, there is the stop bit. This essentially provides a period of time before the next start bit can be sent and enables this start bit to be seen. The stop bit is always high and can have a length of 1, 1.5, or 2 bits.

For this type of communication to work there are several settings that need to be known on both the transmitter and receiver end. This includes the Baud rate, the data word length, the parity mode, and the length of the stop bit. Different settings on either end of the transmission will lead to errors and incorrect transmission.

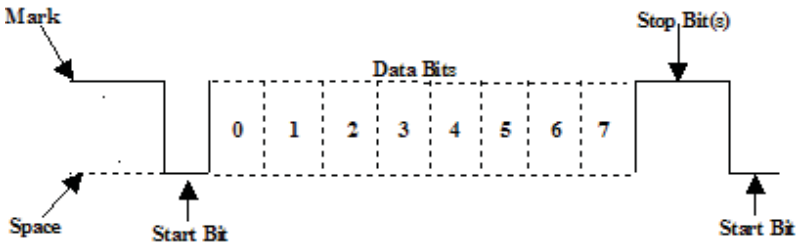


Figure 6.4 – RS232 transmission [61]

Sometimes it is necessary to use flow control to stop data being transmitted when the PC is not ready for it and vice versa. Flow control can also be called handshaking and can be in the form of software or hardware control. Software flow control is often called Xon/Xoff as the receiver sends an Xoff character (ASCII character 17) when it is unable to receive any more data and the transmitting device must wait for an Xon character (ASCII character 19) before continuing with the transmission. This form of handshaking does not require any additional lines, whereas, hardware handshaking does. It operates in a similar manner but uses separate lines to indicate the status of the receiver. There are four additional lines that can be used for hardware flow control but only one pair of these lines is used depending on the communication software in use. These pairs are DTR (Data Terminal Ready) and DSR (Data Set Ready), or RTS (Request To Send) and CTS (Clear To Send). The reason that there are two methods that can be used is that RTS/CTS was never designed for implementing flow control. It was derived for systems with half-duplex modems that only use one line for both RX and TX and as such cannot transmit in both directions at once. Extra lines were therefore necessary to coordinate which device was sending and which was receiving. With the use of full-duplex modems the use of these lines was adapted to implement flow control in the same way as DTR/DSR. These lines are crossed over from one device to another as shown in Figure 6.5. The DTR is set on one device when that device is ready to receive data and the second device detects this on its DSR line which would be set. Hardware flow control is superior to software flow control but is not often used, and often handshaking is not even necessary.

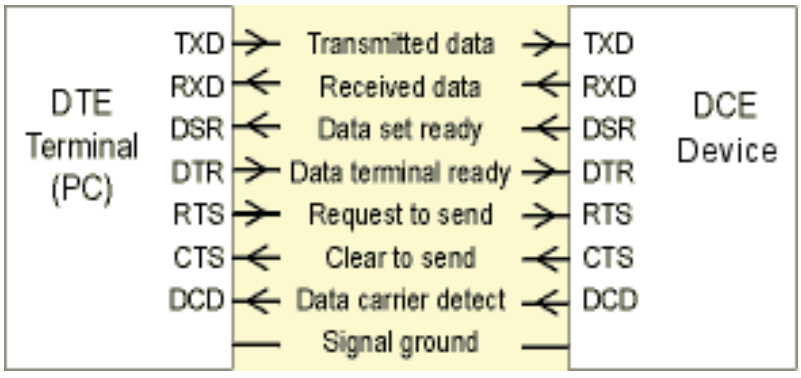


Figure 6.5 – RS232 lines including hardware handshaking [62]

Generally RS232 operates in the range of -15v to +15v although a maximum voltage of ±25v is specified and voltages of ±5v, ±10v, ±12v are commonly used. Higher voltages and a large range are used so that longer transmission lengths can be realised when using a cable. This is because the resistance of the cable reduces the voltage of the signal the further it has to travel. Voltages of -15 to -3 correspond to a logic of zero while voltages of +3v to +15v correspond to logic one, or high. Voltages in the range

of -3v to +3v are not valid for RS232 transmissions. Since most integrated circuits do not operate in this range, hardware, such as the MAX232 chip [63] used on the DE0 FPGA board, must be used to convert the voltage levels.

6.2 Data compression techniques

Data compression is an effective way of reducing the data volume in order to increase the speed of transmitting an erosion profile. Data compression can be categorised into two main groups; lossy compression, and lossless compression. Where lossy compression is concerned, data is compressed by removing information. This may consist of removing data that is redundant or data that is not necessary to include based on the limitations of human perception (such as in images, or audio where data is removed but they still appear or sound the same to humans). Other times, however, lossy compression involves reducing the quality or resolution of the data in order to decrease the data volume. Lossless data compression, on the other hand, is the method of reducing the data without losing any information at all. This is generally done by exploiting statistical redundancies in the data.

There are many different compression techniques that can be used to compress data and the techniques used often depend on the type of data being compressed or the purpose for which the data will be used. For example, when encoding images the limitations of the human visual system are exploited to remove data that is not essential to our perception of the image. Generally, a combination of lossy and lossless compression is used to meet the needs of the system. Lossy compression is first used to reduce the data as much as possible within the given constraints, and then lossless compression is used to further reduce the data without compromising any of the essential information.

The data resolution constraints of concern for this application are the lateral measurement resolution and the depth resolution. These have been previously defined as one measurement every 1-2mm laterally with a depth resolution of 1mm. The developed system gives 648 measurements over an angle of approximately 120 degrees, so for a pipe diameter of 300mm this yields a lateral resolution of one measurement every 0.4mm laterally. This is well within the specifications so could be reduced. However, when the application is extended to 600mm diameter pipes this resolution would reduce to 0.8mm and this is not far from the specification range. This means that lossy compression could be used to reduce the data; however, the level of compression is dependent on the pipe diameter. This form of compression simply involves reducing the data points. For the case where the lateral resolution is 0.4mm this could be done by simply skipping every two to four data points. A better method is to use interpolated points at even intervals across the profile. Often it will be favourable to use the highest resolution possible in order to give more valuable data on the pipe condition. This may be useful to allow the level of compression to be determined by the user in order to trade-off between the speed and accuracy of the inspection depending on the needs of the user.

The depth resolution is specified as 1mm but the actual resolution is, again, dependent on the radius of the pipe being inspected. For a 300mm diameter pipe this resolution is 0.3mm reducing to 0.6mm for a 600mm diameter pipe. Although this is well within the specification, the increased accuracy may often be useful so lossy compression may, again, not be the best solution. However, it may be advantageous to be able to select the level of lossy compression based on the pipe diameter and the user's needs (further work) for both the lateral and depth resolutions.

Lossless compression, on the other hand, is generally always helpful as there is never any loss of information. The form of compression that is most appropriate will depend on the data being encoded. The proposed compression technique for this application is a two-stage process based on run length encoding (RLE) and entropy encoding.

Run length encoding works by analysing the data and determining how long an individual number repeats consecutively. When this length is larger than two, the length of the run and that data value are stored instead of the individual values. It follows that this form of encoding works best when there are large runs with the same value. This is likely to be the case in areas where there is no erosion or where the erosion is small enough that it is considered insignificant and the erosion profile can be reduced to large runs of zero. Also, depending on the level of accuracy of the depth measurements, this may also occur when there is consistent level of erosion across the pipe. However, with an accuracy of better than 1mm this is not likely to be the case as there will be a lot of noise introduced in the data due to quantisation error.

Entropy encoding is used to compress data by replacing each unique data value with a unique code. The length of this code is variable and the smallest code is assigned to the data value that appears most frequently. This form of compression works best when the majority of the data falls in a small set of values. This is likely to be the case when the level of erosion falls within expected levels and is fairly consistent across the pipe. One of the most common forms of entropy encoding is Huffman coding which uses a table to store the values and corresponding codes. This table must be used for decoding as well as for encoding so it must either be predefined based on the likelihood of a value appearing derived from the expected level of erosion, or calculated off the actual statistics of it appearing during inspection and transmitted along with the data.

A combination of RLE and Huffman coding would be useful for encoding data for this application but the actual form of this compression would depend on many factors such as the extent of the erosion, the level of debris in the bottom of the pipes, and the selected accuracy of the depth measurement.

6.3 Pipe profile reconstruction

The process used for reconstructing the pipe profile is shown in Figure 6.6. The first stage is to actually obtain the data. This is done by reading the appropriate serial port ensuring that the settings such as the Baud rate, word length, parity bit, and stop bit length are all correctly set. This data then needs to be written to a buffer before writing the data to a file. This is done to ensure that the writing to file process does not delay the RS232 receiver. This is particularly important when there is no flow control as the receiver may get out of sync with the transmitter.

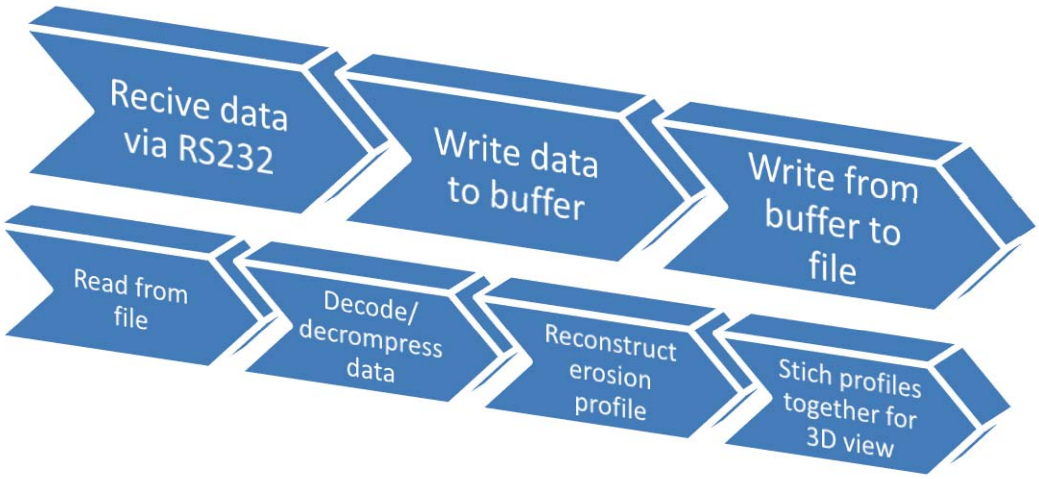


Figure 6.6 – Pipe profile reconstruction procedure

Once the data has been captured it then needs to be decoded to get the data back to the original state. This includes decoding any compression that has been performed on the data as well as adding the reference profile back on. Once this has been done the data should represent the profile of the pipe with erosion once again. This data can them be used to give the user a view of the erosion in the pipe. One way in which this could be done would be to use a 2D height map where colours are used to signify the level of erosion at that point as shown in Figure 6.7.

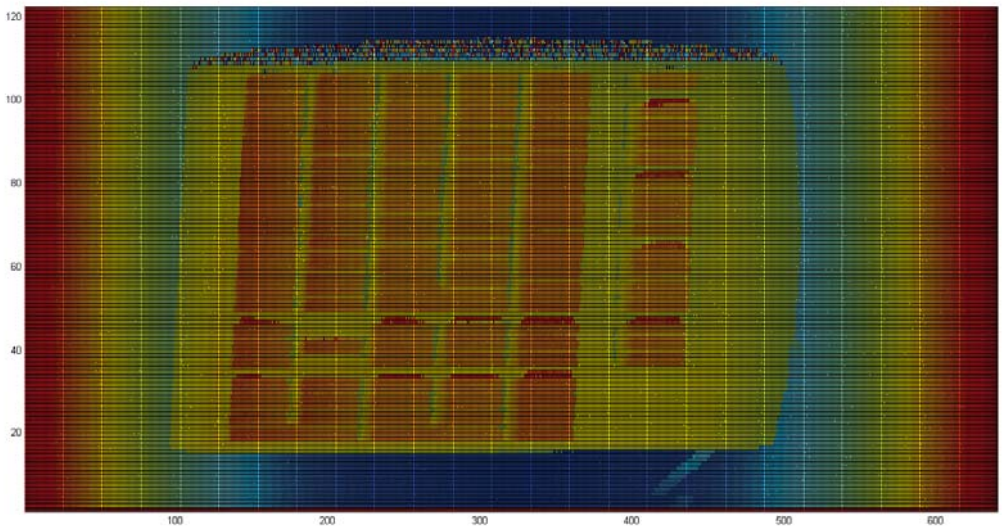


Figure 6.7 – Pipe erosion 2D colour map

Another method is to use a 3D representation where the user can actually see the reconstruction of the pipe with the erosion present by stitching the different profiles together and using these to form a surface in three dimensions. This method allows the user to see how the pipe would appear to the

human eye but does not necessarily make it easy to identify the actual extent of any erosion present. The chosen approach was to use a combination of these two methods to allow the user not only to visualise the erosion but to easily identify the extent of such erosion as well. This approach combines the two methods by providing a 3D map of the erosion where the surface colour is dependent on the depth of any erosion from the expected point. When there is no erosion the surface is a blue colour, however, as the level of erosion increases the colour of the profile fades from blue into red, where bright red represents very severe erosion. An example of the experiment is shown in Figure 6.8 where the object placed in the pipe is a computer keyboard.

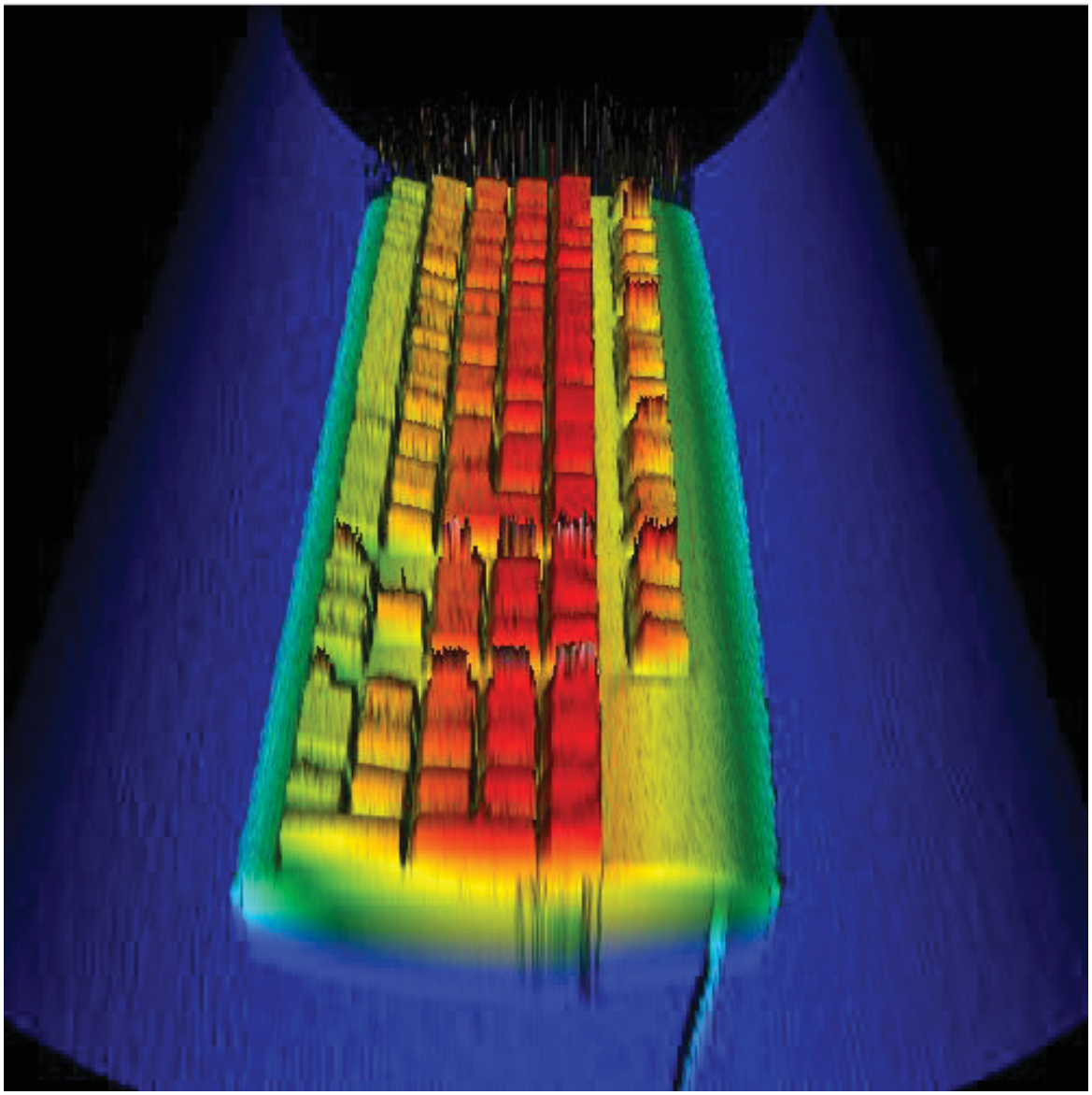


Figure 6.8 – 3D profile of pipe showing height colour map

The developed program for providing height colour-map profile was done in C#. The graphical component to provide the 3D capabilities was done using XnA which is the C# equivalent of directX. This program enables many capabilities such as allowing the user to “walk through” the pipe. This essentially moves along the pipe so that the entire profile can be viewed as if the user was actually walking through the pipe. It also allows the user to change the angle, magnification, and position of

the viewpoint in order to view specific details and have a closer (or broader) inspection of specific areas. The code for this program is attached in Appendix A and a screenshot is shown in Figure 6.9.

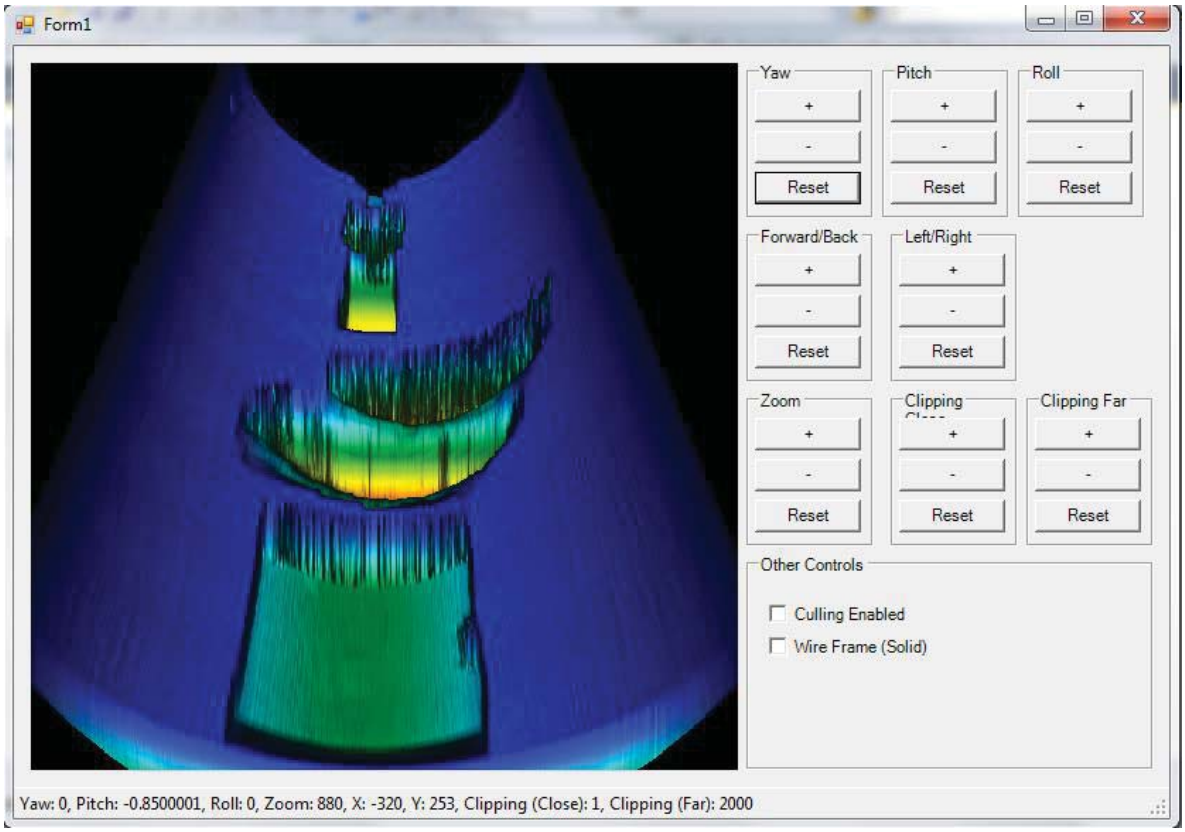


Figure 6.9 – Screenshot of 3D display application

Chapter 7 Results and Analysis

7.1 Anti-rolling mechanical system testing

In reality, the performance of the robot will not match the mathematical model exactly. Factors such as misaligned wheels will produce a torque that will cause the robot to corkscrew as it travels down the pipe. In order to test the practical validity of the anti-rolling system, a test rig was developed resembling that described in Chapter 4, and shown in Figure 7.1. The weight of the test rig was approximately 5kg, the radius of the pipe was 155mm, and the platform of the test rig spanned the width of the pipe with dimensions of 400x300x10mm, and was made of aluminium. Each set of legs attached to the platform via a semi-circle bracket that allowed the legs to be fixed at different angles with a pivot point at the centre of the circle. Each bracket had two sets of tapped holes 10 degrees apart, and each row offset by 5 degrees such that the legs could be set in increments of 5 degrees. Smaller intervals or multiple layers of holes could be used for finer angle adjustment if necessary. The design of the legs allowed them to overlap such that they could be adjusted from 0 to 80 degrees and once the angle was selected the legs were fastened to the bracket.

The motors used are geared DC motors with a shaft offset from the centre of the motor via the gearing. This means that the motors could be attached directly to the wheels and mounted on the legs without any power transmission medium. This greatly simplifies the design, although it may not be possible with larger motors in the same size pipe with the same size wheels. With a larger wheel size there would be more room for mounting the motor but additional gearing may be required. For this test a wheel size of 20mm was used. The front wheels were attached to the legs via two shafts and bushings.

Figure 7.1 – Rig developed for testing anti-rolling system

The initial stage of testing was to adjust the leg angle and evaluate the effect of this on the stability of the robot over a two metre length of pipe by recording the robot's movements. The platform was placed in the pipe at various angles to simulate what would happen if the rig was skew to begin with

and the robot was driven manually by switching a 5A power supply. With a large leg angle the robot could drive freely but was very unstable. As the angle decreased it was evident that the robot had to work harder and the motors drew more current. As the leg angle approached zero it become virtually impossible for the robot to drive under its own power. Upon inspection, it was also evident that as this angle approached zero it was physically more difficult to manually induce roll on the robot thus supporting the findings in Chapter 40. This shows that as the leg angle decreases, the normal forces on the robot wheels caused by gravity increases, significantly increasing both the power required to drive the robot and the ability of the robot to resist rolling within the pipe. Although the platform was able to maintain its starting orientation under a wide range of leg angles, a leg angle of 20 degrees was selected as a good compromise between power requirements and anti-rolling characteristics. This allow for additional load to be added to the platform without making it unstable. It also aids the robot in avoiding the worst of the erosion in the pipe, especially in cases where the bottom of the pipe may no longer exist.

Pipe inspection robots have to carry additional components such as the measurement system, processing units, data recording and communication devices, and batteries. This makes it necessary to test the platform with an additional payload. Since the payload may not always be able to be placed evenly on the robot it is also necessary to explore the behaviour of the robot with a payload offset from the centre of the platform, creating a torsion on the robot. Aluminium blocks were used for the test weights and gradually increased from 0.5kg to 2.5kg. The weights were placed 75mm from the centre of the platform creating an approximate torque of 0.4 to 1.8Nm. The offset distance was limited by the physical space inside the pipe and the size of the aluminium blocks. A summary of the results for the anti-rolling test are shown in Table 7.1 below and in Appendix D.

Table 7.1 – Anti-rolling test results

Starting Angle (Degrees)	Finishing Angle (Degrees)	Offset Weight (kg)	Offset Distance (mm)	Rotation (Degrees)
356	356	0	0	0
3	3	0	0	0
1	1	0	0	0
13	13	0	0	0
28	28	0	0	0
56	56	0	0	0
27	26	0.5	75	-1
27	27	1.2	75	0
24	23	2.5	75	-1

The starting angle is the clockwise angle of the rig relative to the horizontal axis through the centre of the pipe, which the test rig starts off in the pipe. The finishing angle is the angle that the rig finishes in at the opposite end of the pipe once travelling through it, and the rotation is the difference between these two measurements (positive rotation is clockwise). The offset weight and distance describe the position and magnitude of any weight placed on the rig to off-balance it, relative to the central axis of the pipe along the direction of motion of the rig. This definition is shown in Figure 7.2.

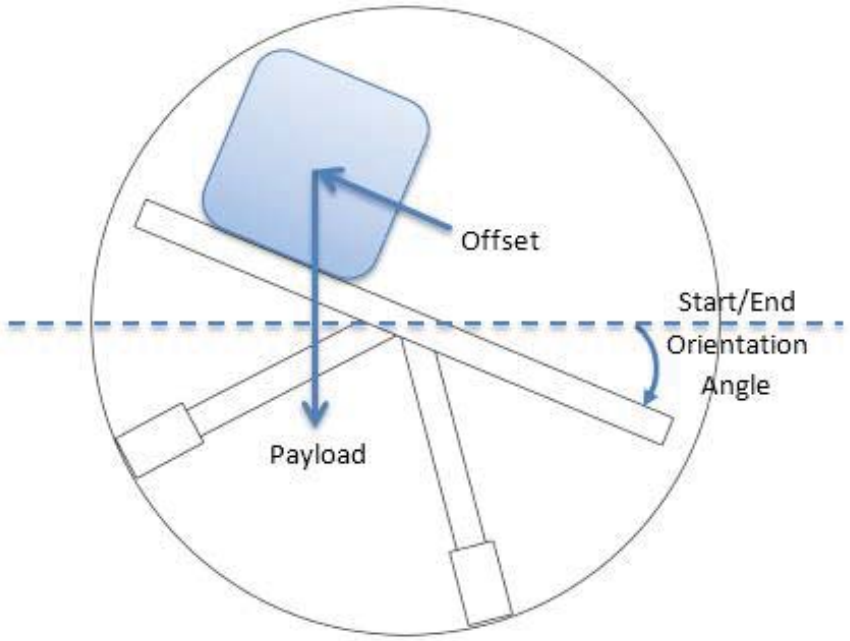


Figure 7.2 – Definition of anti-rolling test parameters

It is clear from the test results that with no offset load the robotic platform is stable, regardless of the initial angle. Furthermore, even with an offset weight there is little to no rotation. This shows that if the weight is balanced evenly on the robot, the robot will not rotate as it travels through the pipe. Given that the accuracy of the angle measurement is limited to one degree it is possible that this will also be the case if the weight is offset slightly, however, further studies would have to be conducted to fully verify this. In general, it would be preferable to have the weight evenly distributed to produce an even load on the robot axes.

If the robot was not able to resist rolling then further steps would have to be taken. This could include decreasing the leg angle to increase the friction force on the wheels. However, this is clearly at the expense of power consumption and would not be appropriate where long inspection runs are required. Alternatively, passive or active measures could be used to self-stabilise the platform. This may include using a sensor to measure the roll and an actuator to compensate for it. However, this would greatly increase the complexity of the design especially in the calibration of the inspection system. In other situations where the robot is not as likely to roll it may be appropriate to increase the leg angle to decrease the power consumption of the robot. It is recommended that the leg angle not exceed 45 degrees as a high leg angle can cause the movement of the inspection robot to become unstable.

As the offset weight was increased to 2.5kg the platform was still able to maintain its starting orientation but the motors reached their limit preventing heavier weights from being tested. However, the expected payload for this application, including the image system, electronics, and the battery, weigh less than 2.5kg, and this weight is more than likely going to be centred on the robot. Under these conditions the test clearly shows that the robot platform will be able to hold its initial orientation with a leg angle of 20 degrees, to provide a stable platform for the inspection system.

From the test results it is obvious that larger payloads will require larger motors to be considered for driving the inspection robot, especially in cases where the leg angle is required to be close to, or less than, 20 degrees. However, larger motors mean more weight, increasing the required size of the battery and in turn increasing the overall weight of the system. If the required capacity of the battery is beyond the capabilities of a lithium battery then heavier alternatives such as lead-acid batteries must also be considered. This shows that a small increase in the required payload capacity can create a large increase in the overall size and weight of the robot and this must be considered for a given application.

7.2 Image system testing

The test rig used for the anti-rolling testing above was further developed for testing the proposed image system. This included mounting a laser, camera, and FPGA to the rig as shown in Figure 7.3.

Figure 7.3 – Test rig used for testing image system

Measurements were used to provide a reference profile based on the calibration technique presented in Chapter 50. The profile is shown in Figure 7.4.

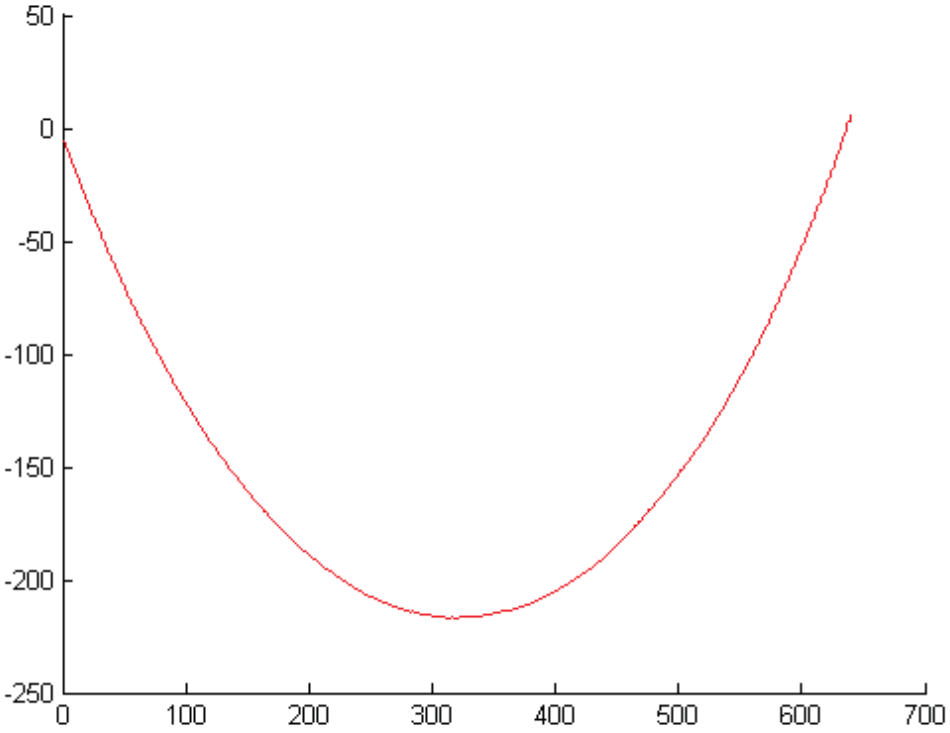


Figure 7.4 – Reference profile from test setup (red)

Different objects such as a computer keyboard were placed on the bottom of the pipe and the generated profiles at set intervals were recorded. These profiles were then used to create erosion profiles by subtracting the above reference from the measured profiles as described in Chapter 40. In this case, the erosion profiles actually show the height of the object above the reference instead of the depth of the erosion below the reference as would be the case in an eroded section of pipe. However, this is more difficult to simulate accurately, and the process works in exactly the same way.

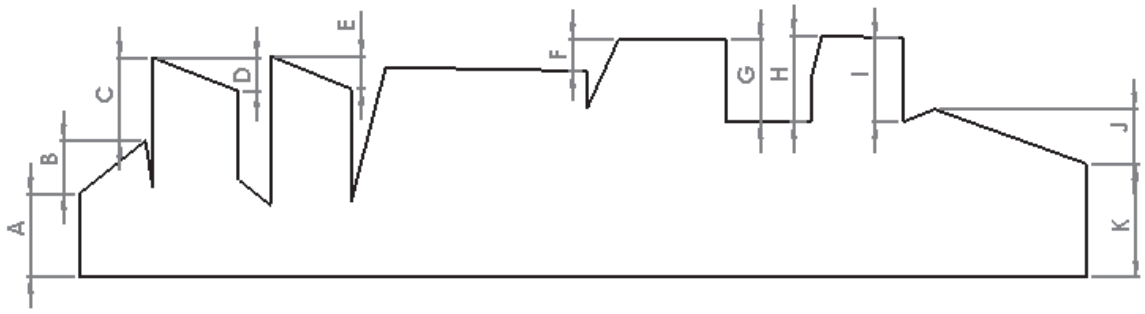


Figure 7.5 – Cross section with of example object with measurements

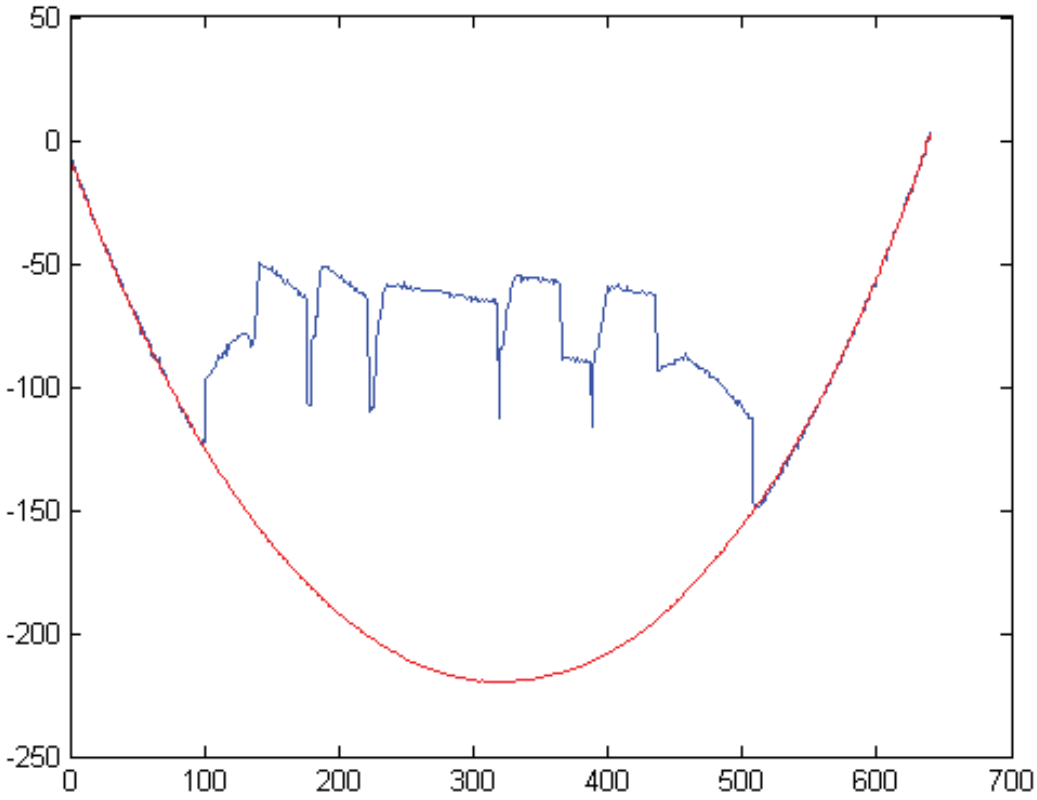


Figure 7.6 – Generated profile of above object (blue)

Figure 7.5 shows the cross section of the keyboard that was used for testing the image system with the measurement definition shown. The actual measurements corresponding to this are shown in Table 7.2. Figure 7.6 shows the generated profile of the object, and Figure 7.7 shows the result (green) from subtracting the profile (blue) from the reference curve (red) and multiplying by the scale factors. This represents the erosion profile of the cross-section of the object in millimetres.

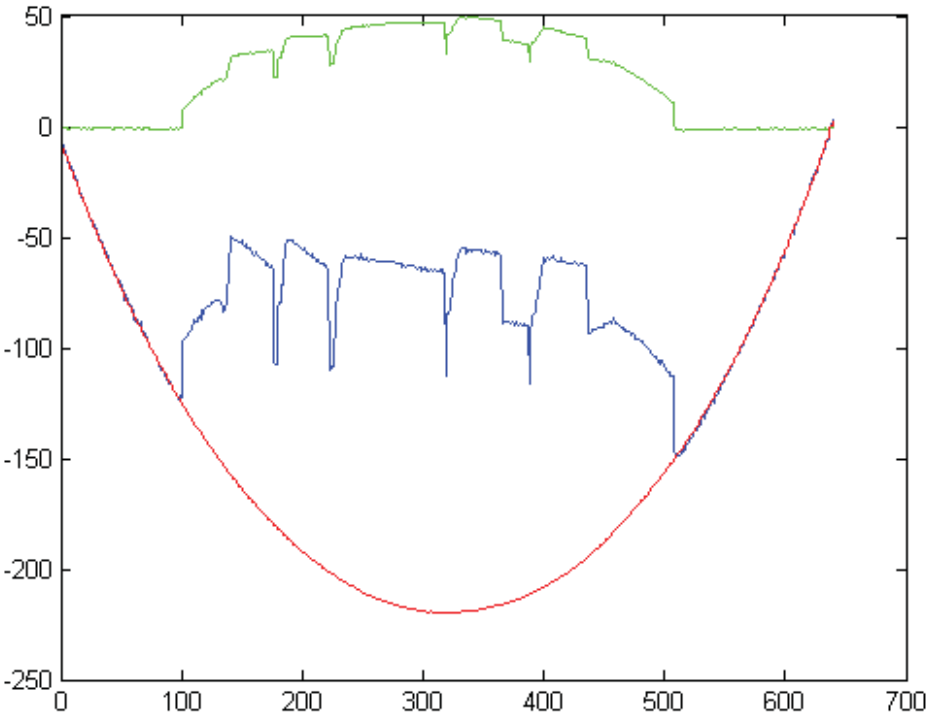


Figure 7.7 – Generated erosion profile of example object (green)

Once the erosion profiles are generated along the length of the object they can be stitched together, using the developed software, into an erosion map for the pipe with the object in it. This is shown in Figure 7.8 and Figure 6.8 which shows the extent of the erosion using a height colour map. Figure 7.8 shows the actual keyboard used to create this map. The spacing between measurements for this map was 5mm.

Figure 7.8 – Object used for testing

Figure 7.9 – Generated erosion map of pipe and object – colour map disabled

The generated erosion map of the object closely resembles the object itself, although the effect of occlusions occurring can be clearly seen at the falling edges, and this appears as random noise. This is because the laser cannot be seen in the image, but the pixel with the greatest red intensity is still chosen as the laser position even though the red intensity at this point will be low. The effect of these occlusions can be minimised by scanning the pipe in both directions, or simply flagging the data point as an occlusion if the intensity is below an average value and the position is outside the expected threshold of the points around it as described in Chapter 4.

The erosion map of the keyboard in the pipe was constructed using a longitudinal resolution of five millimetres. This means that a profile was taken every five millimetres along the pipe. This is a relatively large interval but it is clear that this is still sufficient to pick up a large amount of detail such as the individual keys on the keyboard. This shows that in most cases a finer resolution will not be needed. In fact, in most cases a larger gap between profiles could be used depending on the level of detail and the required accuracy of the inspection.

Equation 4.33 can be used to determine the level of erosion in millimetres from the erosion profile. This requires first calculating the set of constants, K1 and K2, using Equations 4.34 and 4.35. These are included in Appendix E, and remain the same for each set of calibration parameters. The erosion

profile in millimetres for the above test object was calculated using these values and is shown in Figure 7.10. This has also been normalised to exclude the curvature of the pipe under the object to provide an accurate representation of the object. This would not generally be required but due to the test setup it is unavoidable. The actual values shown in Figure 7.5 are compared with those obtained from Figure 7.10 and shown in Table 7.2.

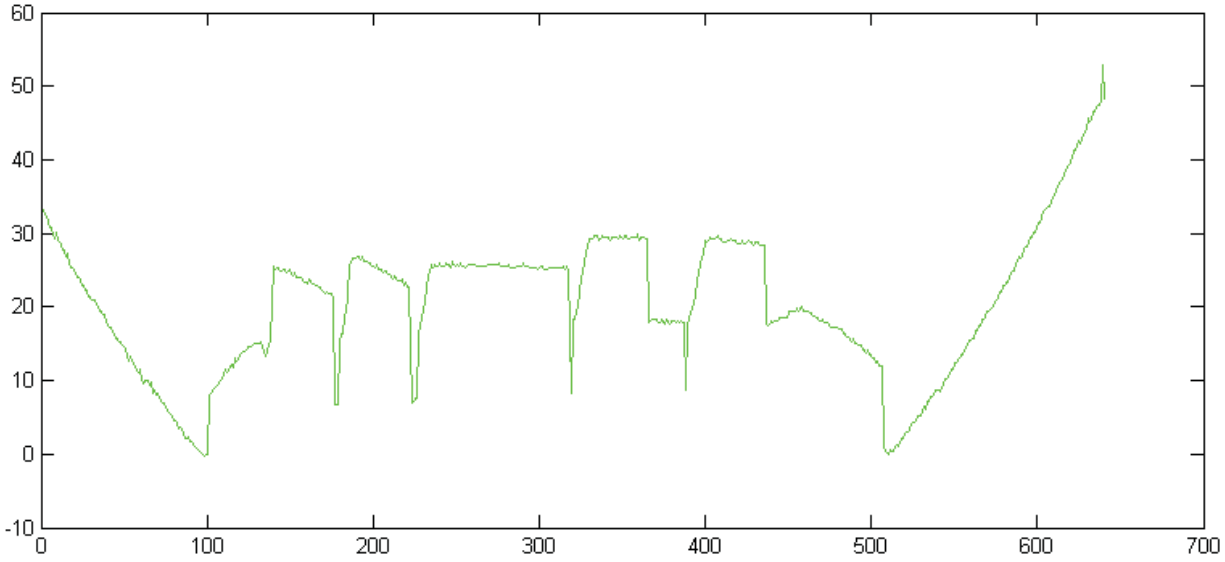


Figure 7.10 – Normalised erosion profile of test object (green) in mm

Table 7.2 – Real measurements vs profile calculated measurements

Dimension	Real Dimensions (±0.05mm)	Profile Dimensions (±0.005mm)	Difference (±0.05mm)
A	8.4	8.37	0.0
B	5.5	5.53	0.0
C	8.5	8.47	0.0
D	3.3	3.41	0.1
E	3.3	3.36	0.1
F	3.2	3.2	0.0
G	8.5	8.41	0.1
H	8.8	8.83	0.0
I	8.6	8.53	0.1
J	4.1	4.27	0.2
K	11.0	10.88	0.1

This shows that the accuracy of the image system is well within the required tolerance, and can provide an accurate representation of the erosion in the pipe so long as there are not too many large occlusions. It also shows that any large objects that may be present in the pipe may affect the accuracy of the inspection as this may create cavities below the object that cannot be detected. These objects could include large chunks of concrete or rocks lodged in the pipe, exposed pieces of the steel reinforcements used in the pipe walls, or any other foreign objects.

During testing it was difficult to determine the exact processing time as this largely depended on the transmission of the data which was limited by the baud rate of the RS232 transmission. However, using RS232, baud rates of up to 115.2Kbaud can be attained. The processing system easily keep up with these speeds but it did cause problems for the software on the PC side as the serial terminal buffers were being overloaded by the volume of data. This requires the software to improve its technique for reading and saving the data in order for high speeds to be fully realised.

The maximum inspection speed based on the data transmission can be determined using:

$$\boldsymbol{speed} = \frac{\mathbf{1}\ \boldsymbol{line}}{\mathbf{640}\ \boldsymbol{samples}} * \frac{\mathbf{1}\ \boldsymbol{sample}}{\mathbf{10}\ \boldsymbol{bits}} * \frac{\mathbf{115200}\ \boldsymbol{bits}}{\mathbf{1}\ \boldsymbol{sec}} * \frac{\mathbf{60}\ \boldsymbol{sec}}{\mathbf{1}\ \boldsymbol{min}} * \frac{\mathbf{5}\ \boldsymbol{mm}}{\boldsymbol{line}} * \frac{\mathbf{1}\ \boldsymbol{m}}{\mathbf{1000}\boldsymbol{mm}} \qquad (7.1)$$

Using RS232 transmission with a baud rate of 115.2Kbaud would enable approximately five and a half metres of pipe to be inspected per minute without any compression techniques employed, and with the five millimetre longitudinal resolution.

Chapter 8 Discussion and Conclusions

Concrete pipes are used in many different applications and are prone to erosion over time raising the need for automated inspection and evaluation in order for more efficient maintenance of these large sections of pipe. Current systems all rely on post evaluation, either in the form of human interpretation or computerised post-processing of the image data to detect erosion. The aim of this research was to devise a solution that could both inspect a pipe and determine the level of erosion in real-time, completely eliminating the need for human intervention or post-processing of the data. The measurement of the erosion was required to have a lateral resolution across the pipe of at least one measurement every one to two millimetres with a depth resolution of better than one millimetre. The longitudinal resolution along the pipe was only required to be one measurement every five to twenty millimetres.

In order to achieve this it was necessary to develop a robot platform that could successfully navigate the erosion while providing a consistent frame of reference. If this is not the case then the area of the pipe that is being examined will change between frames causing inaccurate data when it is pieced back together. In the case of a pipe inspection robot it is difficult to develop such a design with a robot that travels along the bottom of the pipe as this may be a rough surface and there may possibly be obstructions in cases of severe erosion. The most suitable design for this application was a wall supported pipe robot that drives along the wall of the pipe avoiding the worst of the erosion, however, this type of robot can be prone to rolling. This roll can be measured using appropriate sensors and then counteracted but this complicates the design of the robot as well as the design and calibration of the inspection system. For these reasons this research aimed to develop a stable robot platform with anti-rolling characteristics suitable for use with the developed measurement system.

A mechanical platform was designed to passively combat any rotation. This design consisted of four legs at an angle from horizontal with wheels that contacted the pipe wall. A study of this design showed that the ability of the platform to resist rotational movement was largely dependent on the leg angle as this provides the normal force of the wheel on the pipe wall. The smaller the leg angle is from horizontal the harder it is for the platform to rotate. This was even more evident if the centre of gravity of the payload did not coincide with the central axis of the platform as this offset was more likely to cause rotation due to the imbalanced normal forces acting on the wheels. The study clearly showed that the leg angle should be reduced in order to decrease the likelihood of the robot to rotate. This was irrespective of the starting orientation of the platform within the pipe. However, further study also showed that the leg angle has a major effect on the amount of power and the size of the motors required to drive the robot. It is suggested that a leg angle of 20 degrees is a good choice to limit rotation even with an offset payload without having unnecessary motor and power demands.

A test platform was built based on the previously studied design with a leg angle that could be varied at 5 degree intervals. This was used to test the validity of the developed mechanical system. With a large leg angle the robot could move freely but by steadily decreasing this angle it was clear that it became harder for the motors to drive the robot to the point of stalling, and that this required more power. This confirmed what the study suggested would happen. The design was further tested by placing it in a pipe at varying angles with varying payloads and offsets and examining its movements through the pipe. It was apparent that there was little to no rotation of the platform, again confirming what the study suggested. The limitation of this testing was that there was only a limited length of pipe to perform this test in. Also, the pipe used was plastic instead of concrete so the coefficient of

friction could be quite different. The next step of testing would be to use longer pipes with the same characteristics of those to be used in the final application to further confirm the study and provide further results for analysis.

The erosion measurement must be fast and reliable. It also needs to have a high resolution if it is to be used to determine the volume of the erosion. For these reasons a structured light approach based on triangulation of a laser and camera was chosen for the measurement method. This uses a camera to detect the position of a laser that is shone onto the surface of the pipe to detect the presence and severity of any erosion. This is done by processing the camera data to find the position of the laser in each column of the image and converting this to real-coordinates instead of image coordinates.

Standard serial processors are relatively slow for processing images as they have to process the entire image one pixel at a time. This would also require a large amount of memory to store the images while processing. An alternative is to use a hardware processor such as an FPGA which is effectively a reprogrammable logic circuit. The parallel processing ability of such a processor can be exploited to develop a processing algorithm that minimises the time and memory required to process each image. In this case, the images can effectively be processed as they are streamed from the camera, essentially eliminating the processing time of the images making it ideal for real-time processing.

With any processor the image processing algorithm can be optimised to reduce the processing time required. This is even more important on a hardware processor with parallel processing capabilities as multiple tasks can be processed at the same time. An algorithm was developed for processing the image to find the position of the laser in each column. This algorithm processes each column at the same time and works by first subtracting the green component of the image from the red to illuminate any common background noise such as natural light. Each column is then smoothed using a smoothing window and the laser position extracted by finding the brightest pixel in each column. This is done as the image is streamed from the camera, although the processing does lag the image stream by the size of the smoothing window. A reference profile of the pipe without any erosion is subtracted from the laser profile and this is then scaled to give the level of erosion for that cross section of pipe. Any points that are outside the expected range and have intensity below the threshold are marked as occlusions in the data.

Usually a planar reference would be used for the imaging system. In this case the pipe without any erosion was the reference and since this was circular the conversion from image coordinates to real world measurements was non-linear and depended on the position across the image. To simplify this, a mathematical model of the system was optimised to provide a set of scale factors that could be applied to the image coordinates by use of a look-up table on the FPGA or once that data was transmitted to a PC. This optimisation is done as part of the calibration for the camera system, but must be performed on a computer as it is heavily processor intensive. This optimisation is also used to generate the reference profile that is subtracted from the measured profile to give the level of erosion.

The mathematical model of the system has a large number of parameters leading to a complex non-linear optimisation with many degrees of freedom. These parameters were studied to determine if there were any relationships or dependencies between them in order to simply the optimisation and remove redundancies. No such relationships were found, however, this does not mean they didn’t

exist. There is the possibility that there may be complex relationships which are harder to detect and may require further analysis.

For this application an Altera Cyclone III FPGA on a Terasic DE0 board was chosen as the processor coupled with a compatible 5-mega pixel digital camera. This allowed fast, low power, parallel processing of the images. A static line laser was used to provide the structured light and was mounted pointing straight down with the camera at an angle of 40 degrees. This provided the benefit that the optical path length could be extended enabling a narrower field of view to be used to cover the same cross section of pipe. This also meant that the actual focus was less important. Having the laser vertical meant that the laser stripe may not always be present in the image when there were large obstructions so this has to be compensated for in the algorithm but there was also the added benefit that the laser was always at the same x position along the pipe for each column of the image. A larger angle meant that the camera could be mounted further away from the laser, giving a wider field of view but it also makes it more difficult to deal with occlusions resulting from sudden changes in erosion.

With this setup the maximum inspection rate was 49 profiles per second which translates to half a meter per second with a ten millimetre longitudinal resolution. However, this depends on the data communication medium. With the suggested wireless serial connection, the data rate was limited to 17 profiles per second. A compression technique based on run length encoding and entropy coding is suggested to increase the data rate by as much as double allowing the image system to run at full speed over the wireless serial communication link.

The image system was tested by placing several objects in a pipe and scanning them. The scanned image closely resembled the test objects and individual keys on a keyboard could be detected even with a longitudinal resolution of five millimetres. This resolution could be altered depending on the level of detail required, allowing for faster inspections where the detection of finer details is not as critical. The effects from occlusions occurring due to sudden changes were also apparent as the code to handle these had not yet been implemented. This appeared as random noise in the image. By measuring the known dimensions of a cross section of the test objects and comparing these to the corresponding values obtained from the generated profile it was clear that this system was capable of detecting erosion to a very high degree of accuracy. The overall accuracy depends on the frequency and severity of any occlusions resulting in a degree of uncertainty. The most likely scenario where occlusions would be a problem is if there is debris in the pipe causing cavities where the camera cannot see. If this is particularly bad the pipe could be scanned in both directions to limit the effects of this.

Profile values that have a low intensity and that are not near the expected position in the image are likely to be artefacts resulting from occlusions. Values with a low intensity should be flagged as possible occlusions in the image processing algorithm. If the position of these points in the image is then significantly different from the closest points in the reconstructed profile that were not marked as occlusions, then the flagged points should be excluded from the profile and the value at that position in the profile will effectively be interpolated using the surrounding values. This will give a best-guess approximation to the data lost because of the occlusion and the accuracy of the overall profile will depend on the occurrence and severity of such occlusions.

The speed of inspection during testing was limited by the transmission rate of the serial communication link. This was continually transmitting data at seventeen profiles per second without

any data compression techniques being deployed. This is obviously an area for development which would lead to greater inspection speeds. This would also require improvements in the C# application to deal with the increased data rates as it struggled to save the data at the rate it was being received. This would best be implemented in multiple threads with one thread to receive the data and store it in a buffer, and the second thread to read the buffer and save this data to a file. Using a database to save the data would also yield a great improvement over straight data files.

An important aspect of any automated measurement system is the ability to convey quantitative results to the end user in an easy to understand fashion. One common technique for doing this is to display a 2D colour map of the erosion where different colours are used to display the extent of the erosion. This doesn't show a large amount of detail, however, so a 3D approach was developed for users to be able to examine a section of pipe in more detail. This could be used in conjunction with a colour map as well as other statistics such as the volume of erosion or volume per metre of pipe. The data received by the 3D application is the level of erosion across each cross section of pipe. This is added to the known pipe dimensions without erosion and stitched back together to provide the 3D model. This requires information such as the radius of the pipe and the longitudinal resolution of the inspection to be coded into the header of the data file.

This research both designed and tested a measurement system and a robot platform to carry the measurement system. The communication between this robot and a PC was also developed as well as the user interface to view the results. Other key aspects for the development of a prototype have been considered and suggested for implementation as well as improvements that could be made.

This research both designed and tested a measurement system and a robot platform, that can resist roll, to carry the measurement system. The communication between this robot and a PC was also developed as well as the user interface to view the results. While this has provided a solid basis for a pipe inspection robot for the given application there are many areas that could be developed before consolidating a finished solution. One such area that is very important is the position sensing as this will have a large influence on the accuracy of the information provided on the erosion such as the volume of erosion. The suggested approach is to use a combination of encoders on both driving and non-driving axes as well as an inertial measurement unit such as the accelerometer built in to the DE0 Nano FPGA board. This is likely to reduce the effects of slippage and give a more accurate representation of the robot position then a single encoder.

Since there are many different sized pipes used for many different reasons it would be advantageous to be able to use the same robot in different diameter pipes. For the developed robotic platform this would involve increasing the width of the legs in contact with the pipe. This could be done by either using different sets of legs with lengths that match the given pipe, or by allowing the legs to widen using a screw-like linkage mechanism. When using the inspection robot in different diameter pipes the smallest diameter that the robot is to be used in will dictate the allowable payload offset for the robot. This is because the offset weight and distance is proportional to the pipe radius. Failing to comply with this may mean that the robot becomes unstable in smaller diameter pipes. Furthermore, a calibration must be performed each time the pipe radius is changed as this affects the scaling, reference profile, and reconstruction of the profile as the pipe radius is an integral part of the calculations and optimisation used for these.

It would also be beneficial to improve the optimisation used for calibration to reduce the time required to calibrate the image system and allow this to be done more often. This is especially important if the robot is used in different diameter pipes as this will directly affect the calibration parameters. These improvements may build on the current optimisation and limit the parameters to given ranges, as well as splitting the optimisation down into two steps optimising the essential parameters then the non-essential parameters to reduce the degrees of freedom and greatly reducing the complexity of the optimisation. Further researching any possible parameter relationships or redundancies would aid in simplifying this.

A suitable lifeline may also be required for this robot to reduce the risk of the robot getting stuck under severe conditions such as the robot falling into a hole, or the battery dying. This may include an early warning detection system that monitors the robot's vitals as well as preventing the robot from continuing if a large obstruction or hole is detected. This is not always enough, though, and in some cases a retrieval line or further rescue systems may be required and this should be considered for the final prototype.

The final step would be to assemble the entire system and perform field testing, using this to finalise the design and highlight any areas for development. This final system would consist of a mechanical platform based on the tested design coupled with an FPGA based control system and the developed image processing system. The platform would be driven by two geared DC motors and powered by a high capacity lithium-ion battery using position sensing as suggested. The data would be compressed and transmitted using a wireless XBee link to a field device such as a laptop. This may, in turn, be connected to a local network for users to view and analyse the results in real-time.

This robot would be low power and capable of inspecting a pipe at a rate of five to ten metres per minute with a ten millimetre longitudinal resolution. Based off the tested image system the inspection window would have a resolution of 0.3-0.9mm/pixel across the pipe and a depth resolution of better than 0.3-0.6mm for a pipe diameter of 200-600mm. This means that features as small as 1-2mm wide could be detected to better than half a millimetre deep, assuming the camera has line of sight. This is well within the specified requirements. The speed of the inspection could be as fast as twenty meters of pipe per minute for the maximum specified profile spacing and with the suggested compression implemented. The longitudinal resolution used will directly affect the speed of the inspection. The scope of the erosion throughout the pipe may also limit the speed as the more uniform the measured profiles are, the better the performance of the suggested compression technique. The main factor limiting the speed of the inspection is the transmission medium, however, this is a design artefact of the low-power, real-time nature of the inspection robot.

References

[1] M. Jones, D. Bailey and L. Tang, “Prototype of high speed pipe inspection robot,” in *2012 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, Auckland, 2012.

[2] M. Jones, D. Bailey and L. Tang, “Vision system and calibration for pipe inspection,” in *2012 19th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, Auckland, 2012.

[3] L. Tang, D. Bailey and M. Jones, “Rolling prevention mechanism for underground pipe inspection robot with a real time vision system,” *International Journal of Intelligent Mechatronics and Robotics (IJIMR),* vol. 3, no. 3, pp. 60-76, 2013.

[4] A. A. F. Nassiraei, Y. Kawamura, A. Ahrary, Y. Mikuriya and K. Ishii, “Concept design of a fuly autonomous sewer pipe inspection mobile robot "KANTARO",” in *2007 IEEE International Conference on Robotics and Automation*, Roma, Italy, April, 2007.

[5] N. Truong-Thinh, N. Ngoc-Phuong and T. Phuoc-Tho, “A study of pipe-cleaning and inspection robot,” in *2011 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Karon Beach, Phuket, Dec, 2011.

[6] “Thomasnet News,” [Online]. Available: http://news.thomasnet.com/fullstory/Inspection-Crawlers-include-laser-and-sonar-profiling-533499. [Accessed 01 06 2014].

[7] H.-o. Lim and T. Ohki, “Development of pipe inspection robot,” in *ICCAS-SICE*, Fukuoka, Aug, 2009.

[8] J. J. Park, J. W. Moon, H. Kim, S. C. Jang, D. G. Kim, K. Ahn, S. M. Ryew, H. Moon and H. R. Choi, “Development of the untethered in-pipe inspection robot for natural gas pipelines,” in *2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Jeju, Nov, 2013.

[9] I. N. Ismail, A. Anuar, K. Sahari, M. Baharuddin, M. Fairuz, A. Jalal and J. Saad, “Development of in-pipe inspection robot: A review,” in *2012 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (STUDENT)*, Kuala Lumpur, Oct, 2012.

[10] Inuktun Technology, Inc, “Emerald Insight,” [Online]. Available: http://www.emeraldinsight.com/content_images/fig/0490380402006.png. [Accessed 01 06 2014].

[11] S. Kim, C. H. Kim, Y.-g. Bae, H. Na and S. Jung, “NDT inspection mobile robot with spiral driven mechanism in pipes,” in *2013 44th International Symposium on Robotics (ISR)*, Seoul, Oct, 2013.

[12] S. Poozesh, M. Mehrandezh, H. Najjaran and R. Paranjape, “Design and development of a smart vehicle for inspection of in-service water mains,” in *CCECE 2007 Canadian Conference on Electrical and Computer Engineering*, Vancouver, BC, April, 2007.

[13] Dye, “Cunning Turtle - Pipe Inspection Robots,” 05 06 2011. [Online]. Available: http://www.cunningturtle.com/pipe-inspection-robots/. [Accessed 01 06 2014].

[14] T. Nishihara, K. Osuka and I. Tamura, “Development of a simulated model for inner-gas-pipe inspection robot: SPRING,” in *Proceedings of SICE Annual Conference 2010*, Taipei, Aug, 2010.

[15] W. Jeon, J. Park, I. Kim, Y.-K. Kang and H. Yang, “Development of high mobility in-pipe inspection robot,” in *2011 IEEE/SICE International Symposium on System Integration (SII)*, Kyoto, Dec, 2011.

[16] K.-H. Yoon and Y.-W. Park, “Pipe inspection robot actuated by using compressed air,” in *2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Montreal, ON, July, 2010.

[17] J. Lim, H. Park, S. Moon and B. Kim, “Pneumatic robot based on inchworm moion for small diameter pipe inspection,” in *IEEE International Conference on Robotics and Biomimetics*, Sanya, Dec, 2007.

[18] G. Kostin, F. Chernousko, N. Bolotnik and F. Pfeiffer, “Regular motions of a tube-crawling robot: simulation and optimization,” in *Proceedings of the First Workshop on Robot Motion and Control, 1999 RoMoCo*, Kiekrz, 1999.

[19] O. Duran, K. Althoefer and L. D. Seneviratne, “State of the art in sensor technologies for sewer inspection,” *IEEE Sensors Journal,* vol. 2, no. 2, pp. 73-81, 2002.

[20] S. Li-ying, Y. Xiao-dong and L. Yi-bo, “Research on transducer and frequency of ultrasonic guided waves in urban pipe inspection,” in *4th IEEE Conference on Industrial Electronics and Applications ICIEA*, Xi'an, May, 2009.

[21] S. K. Sinha, S. R. Iyer and M. C. Bhardwaj, “Non-contact ultrasonic sensor and state-of-the-art camera for automated pipe inspection,” in *Proceedings of the IEEE Conference on Sensors*, Oct, 2003.

[22] C. Ekes and B. Neducza, “Pipe condition assessments using pipe penetrating radar,” in *2012 14th International Conference on Ground Penetrating Radar (GPR)*, Shanghai, China, June, 2012.

[23] O. Duran, K. Althoefer and L. D. Seneviratne, “Automated pipe defect detection and categorization using camera/laser-based profiler and artificial neural networ,” *IEEE Transactions on Automated Science and Engineering,* vol. 4, no. 1, pp. 118-126, Jan, 2007.

[24] M. Ribo and M. Brandner, “Sate of the art on vision-based structured light systems for 3D measurements,” in *2005 IEEE International Workshop on Robotic and Sensors Environments*, Ottawa, Canada, Oct, 2005.

[25] O. Duran, K. Althoefer and L. D. Seneviratne, “Experiments using a laser-based transducer and automated analysis techniques for pipe inspection,” in *Proceedings of the 2003 IEEE International Conference on Robotics & Automation*, Taipei, Taiwan, September, 2003.

[26] O. Duran, K. Althoefer and L. D. Seneviratne, “A sensor for pipe inspection: model, analysis and image extraction,” in *Proceedings of the 2003 International Conference on Image Processing*, Sept, 2003.

[27] T. Tsubouchi, Y. Kawaguchi, S. Takaki and S. Yuta, “A straight pipe observation from the inside by laser spot array and a TV camera,” in *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000.

[28] “INNOVAM,” [Online]. Available: http://www.innovam.com/en/03016AL.html. [Accessed 01 06 2014].

[29] “Island Optical Systems,” [Online]. Available: http://www.islandos.com.sg/products/laser-diodes/. [Accessed 01 06 2014].

[30] Axis Communications, “CCD and CMOS sensor technology: Technical white paper,” Axis Communications, 2010.

[31] B. S. Carlson, “P1-12: Comparison of modern CCD and CMOS image sensor technologies and systems for low resolution imaging,” in *Proceedings of the 2002 IEEE International Conference on Sensors*, 2002.

[32] G. Koklu, J. Ghaye, R. Beuchat, G. D. Micheli, Y. Leblebici and S. Carrara, “Quantitative comparison of commercial CCD and custom-designed CMOS camera for biological applications,” in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, Seoul, 2012.

[33] “Hyper Physics - DC Electric Motors,” [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/motdc.html. [Accessed 2013 10 13].

[34] “EML2322L - Electric DC Motors (type, function and application to robots),” [Online]. Available: http://www2.mae.ufl.edu/designlab/Class%20Projects/Background%20Information/Electric%20DC%20motors.htm. [Accessed 13 10 2013].

[35] “Controlling DC Motors,” [Online]. Available: http://www.hvlabs.com/hbridge.html. [Accessed 12 07 2014].

[36] “AB-002: Discrete H-bridge for Enhanced Vibration Control,” [Online]. Available: http://www.precisionmicrodrives.com/application-notes-technical-guides/application-bulletins/ab-002-discrete-h-bridge-circuit-for-enhanced-vibration-motor-control-haptic-feedback. [Accessed 12 07 2014].

[37] H. T. Roman and B. A. Pellegrino, “Pipe crawling inspection robots: an overview,” *IEEE Transactions on Energy Conversion,* vol. 8, pp. 576-583, Sept, 1993.

[38] “Wikipedia - Rechargeable Battery,” [Online]. Available: http://en.wikipedia.org/wiki/Rechargeable_battery. [Accessed 24 11 2012].

[39] A. M. Harsha, S. Abeykoon, L. Udawatta, M. S. Dunuweera, R. T. Gunasekara, M. Fonseka and S. P. Gunasekara, “Enhanced position sensing device for mobile robot application using an optical sensor,” in *Proceedings of the IEEE International Conference on Mechatronics*, Istanbul, Turkey, April, 2011.

[40] M. Kondo and K. Ohnishi, “Constructing a platform of robust position estimation for mobile robot by ODR,” The 8th IEEE International Workshop on Advanced Motion Control, March, 2004.

[41] J. S. Poplawski and I. A. Sultan, “Position sensing of industrial robots - a survey,” *Information Technology Journal,* pp. 14-25, 2007.

[42] S. Maeyama, N. Ishikawa and S. Yuta, “Rule based filtering and fusion of odometry and gyroscope for a fail safe dead reckoning system of a mobile robot,” in *Proceedings of the 1996 IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Washington DC, 1996.

[43] H. Chung, L. Ojeda and J. Borenstein, “Accurate mobile Robot dead-reckoning with a precision-calibrated fiber-optic gyroscope,” *IEEE Transactions on Robotics and Automation,* vol. 17, no. 1, pp. 80-84, Feb, 2001.

[44] H. Chung, L. Ojeda and J. Borenstein, “Sensor fusion for mobile robot dead-reckoning with a precision-calibrated fiber optic gyroscope,” in *Proceesings of the 2001 IEEE International Conference on Robotics and Automation*, Seoul, Korea, May, 2001.

[45] R. Tessier and W. Burleson, “Reconfigurable computing for digital signal processing: a survey.,” *Journal of VLSI Signal Processing,* vol. 28, pp. 7-27, 2001.

[46] D. Bailey, Design for embedded image processing on FPGAs, Wiley-IEEE Press, 2011.

[47] “VLSI Egypt - FPGA-911,” [Online]. Available: http://www.vlsiegypt.com/home/?p=226. [Accessed 12 02 2013].

[48] Z. Guo, W. Najjar, F. Vahid and K. Vissers, “A Quantitative analysis of the speedup factors of FPGAs over processors,” in *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field Programmable Gate Arrays*, Ney York, USA, 2004.

[49] M. P. Battaglia, “Parallelism for imaging applications,” in *Conference Record Northcon/93*, Portland, OR, Oct, 1993.

[50] X. Zhang, Y. Li, J. Wang and Y. Chen, “Design of high-speed image processing system based on FPGA,” in *The Ninth Internation Conference on Electronic Measurment & Instruments*, 2009.

[51] A. Goldsmith, Wireless Communications, Cambridge University Press, 2005.

[52] E. Egea-Lopez, A. Martinez-Sala, J. Vales-Alonso, J. Garcia-Haro and J. Malgosa-Sanahuja, “Wireless communications deployment in industry: a review of issues, options and technologies,” *Computers in Industry,* vol. 56, pp. 29-53, 2005.

[53] Q. Li, H. Zhao and P. Liu, “Research on robot network communication system in underground coal mine based on zigbee,” in *3rd International Symposium on Infomation Processing*, 2010.

[54] Digi, “Demystifying 802.15.4 and ZigBee: White Paper,” Digi International Inc., 2007.

[55] *XBee/XBee-PRO RF Modules,* Digi International, 2012.

[56] *XBee ZNet 2.5/XBee-PRO ZNet 2.5 OEM RF Modules,* Digi International, 2008.

[57] G. Horvat, D. Sostaric and D. Zangar, “Power consumption and RF propogation analysis on Zigbee Xbee modules for ATPC,” in *2012 35th International Conference on Telecommunications and Signal Processing (TSP)*, Prague, July, 2012.

[58] “Coefficients of Friction,” [Online]. Available: http://www.roymech.co.uk/Useful_Tables/Tribology/co_of_frict.htm. [Accessed 17 02 2013].

[59] “Friction and Coefficents of Friction,” [Online]. Available: http://www.engineeringtoolbox.com/friction-coefficients-d_778.html. [Accessed 17 02 2013].

[60] “Wikipedia - Pinehole camera model,” [Online]. Available: http://en.wikipedia.org/wiki/Pinhole_camera_model. [Accessed 17 10 2013].

[61] Terasic, “Altera DE0 Board,” [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=165&No=364&PartNo=3. [Accessed 01 11 2014].

[62] Terasic, “5 Mega Pixel Digital Camera Package,” [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=68&No=281. [Accessed 01 11 2014].

[63] “David Gal - Teach,” [Online]. Available: http://white.stanford.edu/teach/index.php/David_Gal. [Accessed 01 06 2014].

[64] “Zigbee Network Application,” [Online]. Available: http://www.icpdas.com/products/GSM_GPRS/zigbee/zigbee_introduction.htm. [Accessed 17 10 2013].

[65] *X-CTU Configuration & Test Utility Software,* Digi International, 2008.

[66] “WCSCNET - Introduction to RS232 Serial Communication,” [Online]. Available: http://wcscnet.com/Tutorials/SerialComm/Page1.htm. [Accessed 08 08 2012].

[67] “Windmill Software LTD - RS232 Serial Port Communication,” [Online]. Available: http://www.windmill.co.uk/rs232-communication.html. [Accessed 08 08 2012].

[68] *MAX232, MAX232I Dual EIA-232 Drivers/receivers,* vol. 2002 Revision, Texas Instruments, 1989.

[69] Hobby King, “Batteries and Accesories,” [Online]. Available: http://www.hobbyking.com/hobbyking/store/__86__85__batteries_accessories-li_poly_all_brands_.html. [Accessed 20 October 2012].

[70] Element14, “Rechargable Batteries,” [Online]. Available: http://nz.element14.com/jsp/search/browse.jsp?N=2106+204086&Ntk=gensearch&Ntt=lead+acid+batteries&Ntx=mode+matchallpartial. [Accessed 25 October 2014].

[71] Radio Spares, “Lead Acid Rechargable Batteries,” [Online]. Available: http://nz.rs-online.com/web/c/batteries/rechargeable-batteries/lead-acid-rechargeable-batteries/?searchTerm=lead+acid+batteries. [Accessed 24 October 2012].

[72] H. T. Roman, B. A. Pellegrino and W. R. Sigrist, “Pipe crawling inspection robots: An overview,” *IEEE Transactions on Energy Conversion,* vol. 8, pp. 576-583, Sept, 1993.

[73] S. Hirose, H. Ohno, T. Mitsui and K. Suyama, “Design of in-pipe inspection vehicles for ø25, ø50, ø150 pipes.,” in *Proceedings of the 1999 IEEE Internation Conference on Robotics & Automation*, May, 1999.

[74] M. Mehrandezh, H. Najjaran, R. Paranjape and S. Poozesh, “Design and development of a smart vehcile for inspection of in-service water mains.,” in *Canadian Conference on Electrical and Computer Engineering*, April, 2007.

[75] I. Hayashi, N. Iwatsuki, K. Morikawa and M. Ogata, “An in-pipe operation microrobot based on the principle of screw-development of a prototype for running in long and bent pipes.,” in *Proceedings of the 1997 International Symposium of Micromechatronics and Human Science*, 1997.

[76] A. Kakogaqa and M. Shugen, “Mobility of an in-pipe robot with screw drive mechanism inside curved pipes,” in *2010 IEEE International Conference on Robotics and Biomimetics*, 2010.

[77] H. Althoff, N. Elkmann, S. Kutzner, J. Saenz and T. Stuerze, “Robotic systems for cleaning and inspection of large concrete pipes.,” in *1st International Conference on Applied Robotics for the Power Industry*, Oct, 2010.

[78] M. Johnson, “Real time pipeline profile extraction using recursive filtering and circle location,” in *Proceedings 2003 Internation Conference on Image Processing*, Sept, 2003.

[79] M. Kurisu, Y. Oosato and Y. Yokokohji, “Development of a laser range finder for 3D map-building in rubble.,” in *Proceedings of the IEEE International Conference on Mechatronics & Automation*, July, 2005.

[80] K. Althoefer, O. Duran and L. D. Seneviratne, “Laser profiler model for robot-based pipe inspection.,” in *Proceedings World Automation Congress*, July, 2004.

[81] H. Higuchi, T. Kaneko, Y. Kawata and A. Yamashita, “Three dimensional measurment of objects's surface in water using the light stripe projection method.,” in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, May, 2004.

[82] S. K. Tso and Y. X. Ma, “A discrete learning algorithm for robot motion control with both position and velocity sensing,” in *IEEE/RSJ International Workshop on Inteligent Robots and Systems*, Osaka, Japan, Nov, 1991.

[83] Y. Arai and M. Sekiai, “Absolute position measurment system for mobile robot based on incident angle detection of infrared light,” in *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, Oct, 2003.

[84] Z. Wang, H. Li, D. Li and W. Zhao, “A direct calibration method for structured light,” in *Proceedings of the IEEE Internation Conference on Mechatronics & Automation*, Niagara Falls, Canada, July, 2005.

[85] Z. Feng, C. Xu, D. Xiao and W. Zhu, “In-pipe profile detection using circular structured light and its calibration technique,” in *Proceedings of the 2008 IEEE International Conference on Information and Automation*, Zhangiajie, China, June, 2008.

[86] X. Kai, L. Wan Yu and P. Zhao-Bang, “The hybrid calibration of linear structured light systems,” in *Proceedings of the 2006 IEEE International Conference on Automation Science and Engineering*, Shanghai, China, Oct, 2006.

[87] D. Q. Huynh, R. A. Owens and P. E. Hartmann, “Calibrating a structured light stripe system: A novel approach,” *Internation Journal of Computer Vision,* vol. 33, no. 1, pp. 73-86, 1999.

[88] Terasic, DE0 User Manual V1.4, Terasic Technologies, 2009.

[89] Terasic, TRDB-D5M 5 Mega Pixel Digital Camera Development Kit V1.2, Terasic Technologies, 2010.

[90] O. Duran, K. Althoefer and L. D. Seneviratne, “Laser profiler model for robot-based pipe inspection,” in *Proceedings of Automation Congress 2004*, Seville, July, 2004.

[91] Z. Chen and W. Xu, “Wireless communication application in mobile robot with machine vision,” in *5th International Conference on Wireless Communications, Networking and Mobile Computing 2009 WiCom*, Beijing, 2009.

Appendix A – C# 3D Erosion Display Application

C# Program (Program.cs)

```
using System;

namespace XnA6
{
#if WINDOWS || XBOX
    static class Program
    {

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        static void Main(string[] args)
        {
                Form1 form1 = new Form1();
                Game1 game = new Game1(form1.getDrawSurface());
                form1.Game = game;
                game.Form = form1;
                form1.Show();
                game.Run();
        }
    }
#endif
}
```

3D Platform (Game1.cs)

```
using System;
using System.Collections.Generic;
using System.Linq;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.GamerServices;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Media;

namespace XnA6
{

    public class Game1 : Microsoft.Xna.Framework.Game
    {
        public struct VertexPositionColorNormal
        {
            public Vector3 Position;
            public Color Color;
            public Vector3 Normal;

            public readonly static VertexDeclaration VertexDeclaration = new
VertexDeclaration
            (
                new VertexElement(0, VertexElementFormat.Vector3,
VertexElementUsage.Position, 0),
                new VertexElement(sizeof(float) * 3, VertexElementFormat.Color,
VertexElementUsage.Color, 0),
                new VertexElement(sizeof(float) * 3 + 4, VertexElementFormat.Vector3,
VertexElementUsage.Normal, 0)
            );
```

```
        }

        private Form1 form;
        public Form1 Form
        {
            get
            {
                return form;
            }
            set
            {
                form = value;
            }
        }

        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
        private IntPtr drawSurface;

        Effect effect;
        GraphicsDevice device;

        //VertexPositionColor[] vertices;
        Matrix viewMatrix;
        Matrix projectionMatrix;
        //private float angle = 0f;
        public float yaw = 0f;
        public float pitch = 0f;
        public float roll = 0f;
        public int zoom = 880;
        public int X = -320;
        public int Y = 0;
        public float ClippingClose = 1;
        public float ClippingFar = 2000;
        public bool Cull = false;
        public bool Wire = false;
        int[] indices;
        private int terrainWidth = 640;
        //-private int terrainWidth = 300;
        //-private int terrainHeight = 300;
        private int terrainHeight = 121;    //length
        private float[,] heightData;
        VertexPositionColorNormal[] vertices;
        VertexBuffer myVertexBuffer;
        IndexBuffer myIndexBuffer;
        private Color[] colourMap = new Color[250];
        private int maxErosionShownLevel = 220;
        private int maxErosionColourMapLevel = 200;

        private double[,] erosionHeights =
        {
              //Profile Data Here
        };

         public Game1(IntPtr drawSurface)
        {
            graphics = new GraphicsDeviceManager(this);
            Content.RootDirectory = "Content";
            this.drawSurface = drawSurface;
            graphics.PreparingDeviceSettings +=
                new
EventHandler<PreparingDeviceSettingsEventArgs>(graphics_PreparingDeviceSettings);
```

```
System.Windows.Forms.Control.FromHandle((this.Window.Handle)).VisibleChanged +=
                new EventHandler(Game1_VisibleChanged);


        }

        private       void       graphics_PreparingDeviceSettings(object       sender,
PreparingDeviceSettingsEventArgs e)
        {

e.GraphicsDeviceInformation.PresentationParameters.DeviceWindowHandle=drawSurface;
        }

        private void Game1_VisibleChanged(object sender, EventArgs e)
        {
            if   (System.Windows.Forms.Control.FromHandle((this.Window.Handle)).Visible
== true)
                System.Windows.Forms.Control.FromHandle((this.Window.Handle)).Visible
= false;
        }

        private void SetUpVertices()
        {
            vertices = new VertexPositionColorNormal[terrainWidth * terrainHeight];
            double[,] eH = erosionHeights;
            double h = 0;
            double hp = 0;
            double he = 0;
            for (int x = 0; x < terrainWidth; x++)
            {
                for (int y = 0; y < terrainHeight; y++)
                {
                    hp  =  Math.Sqrt((terrainWidth  /  2)  *  (terrainWidth  /  2)  -  (x  -
(terrainWidth / 2)) * (x - (terrainWidth / 2)));
                    he = eH[y, x];

                    h = hp - he;
                    vertices[x + y * terrainWidth].Position = new Vector3(x, -(float)h,
-y*10);
                    vertices[x      +      y      *      terrainWidth].Color      =
colourMap[Math.Abs((int)he)];

                }
            }
        }

        private void SetUpIndices()
        {
            indices = new int[(terrainWidth - 1) * (terrainHeight - 1) * 6];
            int counter = 0;
            for (int y = 0; y < terrainHeight - 1; y++)
            {
                for (int x = 0; x < terrainWidth - 1; x++)
                {
                    int lowerLeft = x + y * terrainWidth;
                    int lowerRight = (x + 1) + y * terrainWidth;
                    int topLeft = x + (y + 1) * terrainWidth;
                    int topRight = (x + 1) + (y + 1) * terrainWidth;

                    indices[counter++] = topLeft;
                    indices[counter++] = lowerRight;
```

```
                indices[counter++] = lowerLeft;

                indices[counter++] = topLeft;
                indices[counter++] = topRight;
                indices[counter++] = lowerRight;
            }
        }
    }


    private void LoadHeightData()
    {
        heightData = new float[4, 3];
        heightData[0, 0] = 0;
        heightData[1, 0] = 0;
        heightData[2, 0] = 0;
        heightData[3, 0] = 0;

        heightData[0, 1] = 0.5f;
        heightData[1, 1] = 0;
        heightData[2, 1] = -1.0f;
        heightData[3, 1] = 0.2f;

        heightData[0, 2] = 1.0f;
        heightData[1, 2] = 1.2f;
        heightData[2, 2] = 0.8f;
        heightData[3, 2] = 0;
    }

    private void CalculateNormals()
    {
        for (int i = 0; i < vertices.Length; i++)
            vertices[i].Normal = new Vector3(0, 0, 0);

        for (int i = 0; i < indices.Length / 3; i++)
        {
            int index1 = indices[i * 3];
            int index2 = indices[i * 3 + 1];
            int index3 = indices[i * 3 + 2];

            Vector3 side1 = vertices[index1].Position - vertices[index3].Position;
            Vector3 side2 = vertices[index1].Position - vertices[index2].Position;
            Vector3 normal = Vector3.Cross(side1, side2);

            vertices[index1].Normal += normal;
            vertices[index2].Normal += normal;
            vertices[index3].Normal += normal;
        }

        for (int i = 0; i < vertices.Length; i++)
            vertices[i].Normal.Normalize();
    }

    private void CopyToBuffers()
    {
        myVertexBuffer = new VertexBuffer(device,
VertexPositionColorNormal.VertexDeclaration, vertices.Length, BufferUsage.WriteOnly);
        myVertexBuffer.SetData(vertices);
        myIndexBuffer = new IndexBuffer(device, typeof(int), indices.Length,
BufferUsage.WriteOnly);
        myIndexBuffer.SetData(indices);
    }
```

```
        public void SetUpCamera()
        {
            viewMatrix = Matrix.CreateLookAt(new Vector3(0, zoom, 0), new Vector3(0, 0,
0), new Vector3(0, 0, -1));
            projectionMatrix = Matrix.CreatePerspectiveFieldOfView(MathHelper.PiOver4,
1/*device.Viewport.AspectRatio*/, ClippingClose, ClippingFar);
        }

        public void SetUpColours()
        {

            //Colour fade 1
            int rMax = Color.Aqua.R;
            int rmin = Color.Blue.R;
            int gMax = Color.Aqua.G;
            int gmin = Color.Blue.G;
            int bMax = Color.Aqua.B;
            int bmin = Color.Blue.B;
            int rAve;
            int gAve;
            int bAve;
            int cmapSize = this.maxErosionColourMapLevel;
            int thisFadeSize = cmapSize / 5;
            for (int i = 0; i < thisFadeSize; i++)
            {
                rAve = rmin + ((rMax - rmin) * i / thisFadeSize);
                gAve = gmin + ((gMax - gmin) * i / thisFadeSize);
                bAve = bmin + ((bMax - bmin) * i / thisFadeSize);
                Color tempColour = new Color(rAve, gAve, bAve);
                colourMap[i] = tempColour;
            }

            //Colour fade 2
            rMax = Color.Green.R;
            rmin = Color.Aqua.R;
            gMax = Color.Green.G;
            gmin = Color.Aqua.G;
            bMax = Color.Green.B;
            bmin = Color.Aqua.B;
            for (int i = 0; i < thisFadeSize; i++)
            {
                rAve = rmin + ((rMax - rmin) * i / thisFadeSize);
                gAve = gmin + ((gMax - gmin) * i / thisFadeSize);
                bAve = bmin + ((bMax - bmin) * i / thisFadeSize);
                Color tempColour = new Color(rAve, gAve, bAve);
                colourMap[i+thisFadeSize] = tempColour;
            }

            //Colour fade 3
            rMax = Color.Yellow.R;
            rmin = Color.Green.R;
            gMax = Color.Yellow.G;
            gmin = Color.Green.G;
            bMax = Color.Yellow.B;
            bmin = Color.Green.B;
            for (int i = 0; i < thisFadeSize; i++)
            {
                rAve = rmin + ((rMax - rmin) * i / thisFadeSize);
                gAve = gmin + ((gMax - gmin) * i / thisFadeSize);
                bAve = bmin + ((bMax - bmin) * i / thisFadeSize);
                Color tempColour = new Color(rAve, gAve, bAve);
```

```
                colourMap[i + thisFadeSize*2] = tempColour;
            }

            //Colour fade 4
            rMax = Color.Red.R;
            rmin = Color.Yellow.R;
            gMax = Color.Red.G;
            gmin = Color.Yellow.G;
            bMax = Color.Red.B;
            bmin = Color.Yellow.B;
            for (int i = 0; i < thisFadeSize; i++)
            {
                rAve = rmin + ((rMax - rmin) * i / thisFadeSize);
                gAve = gmin + ((gMax - gmin) * i / thisFadeSize);
                bAve = bmin + ((bMax - bmin) * i / thisFadeSize);
                Color tempColour = new Color(rAve, gAve, bAve);
                colourMap[i + thisFadeSize*3] = tempColour;
            }

            //Colour fade 5
            rMax = Color.White.R;
            rmin = Color.Red.R;
            gMax = Color.White.G;
            gmin = Color.Red.G;
            bMax = Color.White.B;
            bmin = Color.Red.B;
            for (int i = 0; i < thisFadeSize; i++)
            {
                rAve = rmin + ((rMax - rmin) * i / thisFadeSize);
                gAve = gmin + ((gMax - gmin) * i / thisFadeSize);
                bAve = bmin + ((bMax - bmin) * i / thisFadeSize);
                Color tempColour = new Color(rAve, gAve, bAve);
                colourMap[i + thisFadeSize*4] = tempColour;
            }

            for (int i = this.maxErosionColourMapLevel; i < this.maxErosionShownLevel;
i++)
            {
                colourMap[i] = Color.White;
            }

            for (int i = this.maxErosionShownLevel; i < this.colourMap.Length; i++)
            {
                colourMap[i] = Color.Black;
            }
        }

        protected override void Initialize()
        {
            base.Initialize();
        }

        protected override void LoadContent()
        {

            spriteBatch = new SpriteBatch(GraphicsDevice);

            effect = Content.Load<Effect>("effects");

            device = graphics.GraphicsDevice;
            LoadHeightData();
            SetUpColours();
```

```
            SetUpVertices();
            SetUpCamera();
            SetUpIndices();
            CalculateNormals();
            CopyToBuffers();
        }

        protected override void UnloadContent()
        {

        }

        protected override void Update(GameTime gameTime)
        {
            if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
                this.Exit();

            KeyboardState keyState = Keyboard.GetState();
            if (keyState.IsKeyDown(Keys.W))
                Y += 1;
            if (keyState.IsKeyDown(Keys.S))
                Y -= 1;
            if (keyState.IsKeyDown(Keys.Insert))
                yaw += 0.05f;
            if (keyState.IsKeyDown(Keys.Delete))
                yaw -= 0.05f;
            if (keyState.IsKeyDown(Keys.Home))
                pitch += 0.05f;
            if (keyState.IsKeyDown(Keys.End))
                pitch -= 0.05f;
            if (keyState.IsKeyDown(Keys.PageUp))
                roll += 0.05f;
            if (keyState.IsKeyDown(Keys.PageDown))
                roll -= 0.05f;
            if (keyState.IsKeyDown(Keys.Add))
            {
                zoom -= 10;
                SetUpCamera();
            }
            if (keyState.IsKeyDown(Keys.Subtract))
            {
                zoom += 10;
                SetUpCamera();
            }
            base.Update(gameTime);
        }

        protected override void Draw(GameTime gameTime)
        {
            form.Status = ("Yaw: " + yaw.ToString() + ", Pitch: " + pitch.ToString() +
", Roll: " + roll.ToString() + ", Zoom: " + zoom.ToString() + ", X: " + X.ToString() +
", Y: " + Y.ToString() + ", Clipping (Close): " + ClippingClose.ToString() + ", Clipping
(Far): " + ClippingFar.ToString());

            GraphicsDevice.Clear(Color.Black);

            RasterizerState rs = new RasterizerState();
            if(Cull == true)
                rs.CullMode = CullMode.CullCounterClockwiseFace;
            else
                rs.CullMode = CullMode.None;
            if(Wire == true)
```

```
                rs.FillMode = FillMode.WireFrame;
            else
                rs.FillMode = FillMode.Solid;
            device.RasterizerState = rs;

            effect.CurrentTechnique = effect.Techniques["Colored"];
            effect.Parameters["xView"].SetValue(viewMatrix);
            effect.Parameters["xProjection"].SetValue(projectionMatrix);
            Matrix    worldMatrix    =    Matrix.CreateTranslation(X,    0,    Y)    *
Matrix.CreateFromYawPitchRoll(yaw, pitch, roll);
            effect.Parameters["xWorld"].SetValue(worldMatrix);

            Vector3 lightDirection = new Vector3(0f, -1.0f, 0f);
            lightDirection.Normalize();
            effect.Parameters["xLightDirection"].SetValue(lightDirection);
            effect.Parameters["xAmbient"].SetValue(0.1f);
effect.Parameters["xEnableLighting"].SetValue(true);
            effect.Parameters["xEnableLighting"].SetValue(true);

            foreach (EffectPass pass in effect.CurrentTechnique.Passes)
            {
                pass.Apply();
                device.Indices = myIndexBuffer;
                device.SetVertexBuffer(myVertexBuffer);
                device.DrawIndexedPrimitives(PrimitiveType.TriangleList,     0,     0,
vertices.Length, 0, indices.Length / 3);
            }

            base.Draw(gameTime);
        }
    }
}
```

GUI Form (Form1.cs)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace XnA6
{
    public partial class Form1 : Form
    {
        private Game1 game;
        public Game1 Game
        {
            get
            {
                return game;
            }
            set
            {
                game = value;
            }
        }
```

```
public String Status
{
    get
    {
        return status.Text;
    }
    set
    {
        status.Text = value;
    }
}

public Form1()
{
    InitializeComponent();
}

public IntPtr getDrawSurface()
{
    return pctSurface.Handle;
}

private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

private void yawPlus_Click(object sender, EventArgs e)
{
    game.yaw += 0.05f;
}

private void yawMinus_Click(object sender, EventArgs e)
{
    game.yaw -= 0.05f;
}

private void yawReset_Click(object sender, EventArgs e)
{
    game.yaw = 0f;
}

private void pitchPlus_Click(object sender, EventArgs e)
{
    game.pitch += 0.05f;
}

private void pitchMinus_Click(object sender, EventArgs e)
{
    game.pitch -= 0.05f;
}

private void pitchReset_Click(object sender, EventArgs e)
{
    game.pitch = 0f;
}

private void rollPlus_Click(object sender, EventArgs e)
{
    game.roll += 0.05f;
}
```

```
private void rollMinus_Click(object sender, EventArgs e)
{
    game.roll -= 0.05f;
}

private void rollReset_Click(object sender, EventArgs e)
{
    game.roll = 0f;
}

private void zoomPlus_Click(object sender, EventArgs e)
{
    game.zoom -= 10;
    game.SetUpCamera();
}

private void zoomMinus_Click(object sender, EventArgs e)
{
    game.zoom += 10;
    game.SetUpCamera();
}

private void zoomReset_Click(object sender, EventArgs e)
{
    game.zoom = 300;
    game.SetUpCamera();
}

private void forward_backPlus_Click(object sender, EventArgs e)
{
    game.Y += 1;
}

private void forward_backMinus_Click(object sender, EventArgs e)
{
    game.Y -= 1;
}

private void forward_backReset_Click(object sender, EventArgs e)
{
    game.Y = 0;
}

private void left_rightPlus_Click(object sender, EventArgs e)
{
    game.X += 1;
}

private void left_rightMinus_Click(object sender, EventArgs e)
{
    game.X -= 1;
}

private void left_rightReset_Click(object sender, EventArgs e)
{
    game.X = -150;
}

private void clipping_closePlus_Click(object sender, EventArgs e)
{
    game.ClippingClose += 1;
```

```
            game.SetUpCamera();
        }

        private void clipping_closeMinus_Click(object sender, EventArgs e)
        {
            if(game.ClippingClose>0)
                game.ClippingClose -= 1;
            game.SetUpCamera();
        }

        private void clipping_closeReset_Click(object sender, EventArgs e)
        {
            game.ClippingClose = 1;
            game.SetUpCamera();
        }

        private void clipping_farPlus_Click(object sender, EventArgs e)
        {
            game.ClippingFar += 10;
            game.SetUpCamera();
        }

        private void clipping_farMinus_Click(object sender, EventArgs e)
        {
            if (game.ClippingFar >= 10)
                game.ClippingFar -= 10;
            game.SetUpCamera();
        }

        private void clipping_farReset_Click(object sender, EventArgs e)
        {
            game.ClippingFar = 500;
            game.SetUpCamera();
        }

        private void checkBoxCull_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBoxCull.Checked == true)
                game.Cull = true;
            else
                game.Cull = false;
        }

        private void checkBoxWireFrame_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBoxWireFrame.Checked == true)
                game.Wire = true;
            else
                game.Wire = false;
        }

    }
}
```

Effects File (effects.fx)

```
//-----------------------------------------------------
//--                                                 --
//--            www.riemers.net                  --
//--                    Basic shaders                        --
//--          Use/modify as you like                --
//--                                                   --
```

```
//-----------------------------------------------------

struct VertexToPixel
{
    float4 Position         : POSITION;
    float4 Color            : COLOR0;
    float LightingFactor: TEXCOORD0;
    float2 TextureCoords: TEXCOORD1;
};

struct PixelToFrame
{
    float4 Color : COLOR0;
};

//------- Constants --------
float4x4 xView;
float4x4 xProjection;
float4x4 xWorld;
float3 xLightDirection;
float xAmbient;
bool xEnableLighting;
bool xShowNormals;
float3 xCamPos;
float3 xCamUp;
float xPointSpriteSize;

//------- Texture Samplers --------

Texture xTexture;
sampler TextureSampler = sampler_state { texture = <xTexture>; magfilter = LINEAR;
minfilter = LINEAR; mipfilter=LINEAR; AddressU = mirror; AddressV = mirror;};

//------- Technique: Pretransformed --------

VertexToPixel PretransformedVS( float4 inPos : POSITION, float4 inColor: COLOR)
{
       VertexToPixel Output = (VertexToPixel)0;

       Output.Position = inPos;
       Output.Color = inColor;

       return Output;
}

PixelToFrame PretransformedPS(VertexToPixel PSIn)
{
       PixelToFrame Output = (PixelToFrame)0;

       Output.Color = PSIn.Color;

       return Output;
}

technique Pretransformed
{
       pass Pass0
       {
              VertexShader = compile vs_2_0 PretransformedVS();
              PixelShader  = compile ps_2_0 PretransformedPS();
       }
}
```

```
//------- Technique: Colored --------

VertexToPixel ColoredVS( float4 inPos : POSITION, float3 inNormal: NORMAL, float4
inColor: COLOR)
{
	VertexToPixel Output = (VertexToPixel)0;
	float4x4 preViewProjection = mul (xView, xProjection);
	float4x4 preWorldViewProjection = mul (xWorld, preViewProjection);

	Output.Position = mul(inPos, preWorldViewProjection);
	Output.Color = inColor;

	float3 Normal = normalize(mul(normalize(inNormal), xWorld));
	Output.LightingFactor = 1;
	if (xEnableLighting)
		Output.LightingFactor = dot(Normal, -xLightDirection);

	return Output;
}

PixelToFrame ColoredPS(VertexToPixel PSIn)
{
	PixelToFrame Output = (PixelToFrame)0;

	Output.Color = PSIn.Color;
	Output.Color.rgb *= saturate(PSIn.LightingFactor) + xAmbient;

	return Output;
}

technique Colored
{
	pass Pass0
	{
		VertexShader = compile vs_2_0 ColoredVS();
		PixelShader  = compile ps_2_0 ColoredPS();
	}
}

//------- Technique: ColoredNoShading --------

VertexToPixel ColoredNoShadingVS( float4 inPos : POSITION, float4 inColor: COLOR)
{
	VertexToPixel Output = (VertexToPixel)0;
	float4x4 preViewProjection = mul (xView, xProjection);
	float4x4 preWorldViewProjection = mul (xWorld, preViewProjection);

	Output.Position = mul(inPos, preWorldViewProjection);
	Output.Color = inColor;

	return Output;
}

PixelToFrame ColoredNoShadingPS(VertexToPixel PSIn)
{
	PixelToFrame Output = (PixelToFrame)0;

	Output.Color = PSIn.Color;

	return Output;
}
```

```
technique ColoredNoShading
{
	pass Pass0
	{
		VertexShader = compile vs_2_0 ColoredNoShadingVS();
		PixelShader  = compile ps_2_0 ColoredNoShadingPS();
	}
}


//------- Technique: Textured --------

VertexToPixel TexturedVS( float4 inPos : POSITION, float3 inNormal: NORMAL, float2
inTexCoords: TEXCOORD0)
{
	VertexToPixel Output = (VertexToPixel)0;
	float4x4 preViewProjection = mul (xView, xProjection);
	float4x4 preWorldViewProjection = mul (xWorld, preViewProjection);

	Output.Position = mul(inPos, preWorldViewProjection);
	Output.TextureCoords = inTexCoords;

	float3 Normal = normalize(mul(normalize(inNormal), xWorld));
	Output.LightingFactor = 1;
	if (xEnableLighting)
		Output.LightingFactor = dot(Normal, -xLightDirection);

	return Output;
}

PixelToFrame TexturedPS(VertexToPixel PSIn)
{
	PixelToFrame Output = (PixelToFrame)0;

	Output.Color = tex2D(TextureSampler, PSIn.TextureCoords);
	Output.Color.rgb *= saturate(PSIn.LightingFactor) + xAmbient;

	return Output;
}

technique Textured
{
	pass Pass0
	{
		VertexShader = compile vs_2_0 TexturedVS();
		PixelShader  = compile ps_2_0 TexturedPS();
	}
}

//------- Technique: TexturedNoShading --------

VertexToPixel TexturedNoShadingVS( float4 inPos : POSITION, float3 inNormal: NORMAL,
float2 inTexCoords: TEXCOORD0)
{
	VertexToPixel Output = (VertexToPixel)0;
	float4x4 preViewProjection = mul (xView, xProjection);
	float4x4 preWorldViewProjection = mul (xWorld, preViewProjection);

	Output.Position = mul(inPos, preWorldViewProjection);
	Output.TextureCoords = inTexCoords;
```

```
	return Output;
}

PixelToFrame TexturedNoShadingPS(VertexToPixel PSIn)
{
	PixelToFrame Output = (PixelToFrame)0;

	Output.Color = tex2D(TextureSampler, PSIn.TextureCoords);

	return Output;
}

technique TexturedNoShading
{
	pass Pass0
	{
		VertexShader = compile vs_2_0 TexturedNoShadingVS();
		PixelShader  = compile ps_2_0 TexturedNoShadingPS();
	}
}

//------- Technique: PointSprites --------

VertexToPixel PointSpriteVS(float3 inPos: POSITION0, float2 inTexCoord: TEXCOORD0)
{
    VertexToPixel Output = (VertexToPixel)0;

    float3 center = mul(inPos, xWorld);
    float3 eyeVector = center - xCamPos;

    float3 sideVector = cross(eyeVector,xCamUp);
    sideVector = normalize(sideVector);
    float3 upVector = cross(sideVector,eyeVector);
    upVector = normalize(upVector);

    float3 finalPosition = center;
    finalPosition += (inTexCoord.x-0.5f)*sideVector*0.5f*xPointSpriteSize;
    finalPosition += (0.5f-inTexCoord.y)*upVector*0.5f*xPointSpriteSize;

    float4 finalPosition4 = float4(finalPosition, 1);

    float4x4 preViewProjection = mul (xView, xProjection);
    Output.Position = mul(finalPosition4, preViewProjection);

    Output.TextureCoords = inTexCoord;

    return Output;
}

PixelToFrame PointSpritePS(VertexToPixel PSIn) : COLOR0
{
    PixelToFrame Output = (PixelToFrame)0;
    Output.Color = tex2D(TextureSampler, PSIn.TextureCoords);
    return Output;
}

technique PointSprites
{
	pass Pass0
	{
		VertexShader = compile vs_2_0 PointSpriteVS();
		PixelShader  = compile ps_2_0 PointSpritePS();
```

```
        }
}
```

Appendix B – Matlab Optimisation Code

Optimisation - Main

```
%% Read in data & Optimise
clear all;
close all;
clc;

%Read in data - stored as 'data4optimisation'
file = uigetfile('*.dat');
rawdata = textread(file, '%c');
rawdata_length= size(rawdata,1);
rawdata_length8s = floor(rawdata_length/8);

data = zeros(1,rawdata_length8s);
bit0 = zeros(1, rawdata_length8s/8);
bit1 = zeros(1, rawdata_length8s/8);
bit2 = zeros(1, rawdata_length8s/8);
bit3 = zeros(1, rawdata_length8s/8);
bit4 = zeros(1, rawdata_length8s/8);
bit5 = zeros(1, rawdata_length8s/8);
bit6 = zeros(1, rawdata_length8s/8);
bit7 = zeros(1, rawdata_length8s/8);

bc = 1;

for count=1:rawdata_length8s,
 bit7(bc) = str2num(rawdata(1+(count-1)*8));
 bit6(bc) = str2num(rawdata(2+(count-1)*8));
 bit5(bc) = str2num(rawdata(3+(count-1)*8));
 bit4(bc) = str2num(rawdata(4+(count-1)*8));
 bit3(bc) = str2num(rawdata(5+(count-1)*8));
 bit2(bc) = str2num(rawdata(6+(count-1)*8));
 bit1(bc) = str2num(rawdata(7+(count-1)*8));
 bit0(bc) = str2num(rawdata(8+(count-1)*8));
 data(count) = bit7(bc)*128 + bit6(bc)*64 + bit5(bc)*32 + bit4(bc)*16 +
bit3(bc)*8 + bit2(bc)*4 + bit1(bc)*2 + bit0(bc);
 bc=bc+1;
end

global data4optimisation;
shiftLeft = 2;
shiftUp = 12;
data4optimisation(1:(640 - shiftLeft)) = data((shiftLeft + 1):640) - 240 -
shiftUp;
data4optimisation((641 - shiftLeft):640) = data(1:shiftLeft) - 240 - shiftUp;
disp('Data loaded');



%Equation to calculate alpha's to use for determining y-hat
syms r_s p_s b_s xc_s yc_s zc_s tx_s ty_s tz_s a_s f_s xh_s;
syms y;

y = r_s*sin(a_s);
z = -r_s*cos(a_s);
x = -y*tan(b_s) - (z-r_s)*tan(p_s)*cos(b_s);
```

```
r1 = [1 0 0; 0 cos(tx_s) -sin(tx_s); 0 sin(tx_s) cos(tx_s)];
r2 = [cos(ty_s) 0 sin(ty_s); 0 1 0; -sin(ty_s) 0 cos(ty_s)];
r3 = [cos(tz_s) -sin(tz_s) 0; sin(tz_s) cos(tz_s) 0; 0 0 1];
R = r1 * r2 * r3;
RXYZ = R * [x-xc_s; y-yc_s; z-zc_s];
yd = RXYZ(1);
xd = RXYZ(2);
zd = RXYZ(3);

x_h = -f_s * .25 * xd/zd;
S = solve(x_h-xh_s,a_s);
disp('Alpha(x-hat) expression determined');

global alphaFunction;
%alphaFunction = matlabFunction(S, [r_s, f_s, p_s, b_s, xc_s, yc_s, zc_s,
tx_s, ty_s, tz_s, xh_s]);
alphaFunction = matlabFunction(S);
disp('alphaFunction Saved');

global optimisationIterations;
optimisationIterations = 0;


%Optimisation
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;


clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,     zc_initial,     pi/180*tx_initial,     pi/180*ty_initial,
pi/180*tz_initial];
[po, eo, ef, op] = fminunc('functionForOptimisation', params)
%[po, rn, rd, ef, op] = lsqnonlin('functionForOptimisation', params)
optimisatioTimeTaken = toc(optimisationTime)
send_mail_message('mattyj_27@yahoo.com.au','Optimisation Finished',['po = '
num2str(po(1)) ', ' num2str(po(2)) ', ' num2str(po(3)) ', ' num2str(po(4))
',  '  num2str(po(5))  ',  '  num2str(po(6))  ',  '  num2str(po(7))  ',  '
num2str(po(8))    ',    '    num2str(po(9))    '.    Time    taken:    '
num2str(optimisatioTimeTaken)])
global iterationTimeTaken;
sum(iterationTimeTaken)
```

Optimisation – Function

```
function g=functionForOptimisation(params)
global data4optimisation;
global alphaFunction;
global optimisationIterations;
```

```
global iterationTimeTaken;
y_m = data4optimisation;
%SumErrorSquared = 0;

iterationTime = tic;

radius = 310/2;
    focal = params(1);
    phi = params(2);
    beta = params(3);
    x_c = params(4);
    y_c = params(5);
    z_c = params(6);
    t_x = params(7);
    t_y = params(8);
    t_z = params(9);


%Calculate Alphas
syms xh_n;
Alpha = alphaFunction(beta, focal, phi, radius, t_x, t_y, t_z, x_c, xh_n,
y_c, z_c);

y_h = zeros(1, 640);
x_h = zeros(1, 640);
Error = zeros(1, 640);
alphaFromXhsAll = zeros(2, 640);
%countValid = 0;
for count = 1:640
    alphaFromXhs = subs(Alpha,xh_n,count-321);
    alphaFromXhsAll(1, count) = alphaFromXhs(1);
    alphaFromXhsAll(2, count) = alphaFromXhs(2);

    %Calculate Ys
    alpha = alphaFromXhs(2);
    y = radius * sin(alpha);
    z = -radius*cos(alpha);
    %z = -sqrt(radius^2 - y^2);
    x = -y * tan(beta) - (z - radius) * tan(phi) * cos(beta);
    r1 = [1 0 0; 0 cos(t_x) -sin(t_x); 0 sin(t_x) cos(t_x)];
    r2 = [cos(t_y) 0 sin(t_y); 0 1 0; -sin(t_y) 0 cos(t_y)];
    r3 = [cos(t_z) -sin(t_z) 0; sin(t_z) cos(t_z) 0; 0 0 1];
    R = r1 * r2 * r3;
    RotatedXYZ = R * [x-x_c; y-y_c; z-z_c];                                    %
[x_d, y_d, z_d] = R[y, x, z]; Instead of = R[x, y, z];
    y_d = RotatedXYZ(1);
    x_d = RotatedXYZ(2);
    z_d = -RotatedXYZ(3);
    x_h(count) = focal * .25 * -x_d / z_d;
    y_h(count) = -focal * .5 * y_d / z_d;

    if(isnan(y_h(count)))
       y_h(count) = y_h(count-1);
    end
    Error(count) = y_m(count) - y_h(count);

end

optimisationIterations = optimisationIterations + 1;
```

```
%SumErrorSquared = 0;
SumErrorSquared = sum(Error(1:639).*Error(1:639));

disp(['Iteration:  ' num2str(optimisationIterations)]);
disp(['      Sum of Error Squared:  ' num2str(SumErrorSquared)]);

iterationTimeTaken(optimisationIterations) = toc(iterationTime);
disp(['                          Time    taken    4    iteration:         '
num2str(iterationTimeTaken(optimisationIterations))]);
%keyboard
% hold on;
% plot(-320:319, y_h);
g = SumErrorSquared;
%g = Error(1:639);
```

Calibration Results Display

```
%clear all;
%close all;
clc;

%Read in data - stored as 'data4optimisation'
file = uigetfile('*.dat');
rawdata = textread(file, '%c');
rawdata_length= size(rawdata,1);
rawdata_length8s = floor(rawdata_length/8);

data = zeros(1,rawdata_length8s);
bit0 = zeros(1, rawdata_length8s/8);
bit1 = zeros(1, rawdata_length8s/8);
bit2 = zeros(1, rawdata_length8s/8);
bit3 = zeros(1, rawdata_length8s/8);
bit4 = zeros(1, rawdata_length8s/8);
bit5 = zeros(1, rawdata_length8s/8);
bit6 = zeros(1, rawdata_length8s/8);
bit7 = zeros(1, rawdata_length8s/8);

bc = 1;

for count=1:rawdata_length8s,
 bit7(bc) = str2num(rawdata(1+(count-1)*8));
 bit6(bc) = str2num(rawdata(2+(count-1)*8));
 bit5(bc) = str2num(rawdata(3+(count-1)*8));
 bit4(bc) = str2num(rawdata(4+(count-1)*8));
 bit3(bc) = str2num(rawdata(5+(count-1)*8));
 bit2(bc) = str2num(rawdata(6+(count-1)*8));
 bit1(bc) = str2num(rawdata(7+(count-1)*8));
 bit0(bc) = str2num(rawdata(8+(count-1)*8));
 data(count) = bit7(bc)*128 + bit6(bc)*64 + bit5(bc)*32 + bit4(bc)*16 +
bit3(bc)*8 + bit2(bc)*4 + bit1(bc)*2 + bit0(bc);
 bc=bc+1;
end

shiftLeft = 2;
shiftUp = 12;
realData(1:(640 - shiftLeft)) = data((shiftLeft + 1):640) - 240 - shiftUp;
realData((641 - shiftLeft):640) = data(1:shiftLeft) - 240 - shiftUp;
```

```
%Initial
f_initial = 3226;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

% %Optimised
f_optimised = 3326;
phi_optimised = 4.8574;
beta_optimised = 0.4956;
xc_optimised = -244.0021;
yc_optimised = 0.00037207;
zc_optimised = 125.0003;
tx_optimised = -0.3498;
ty_optimised = 43.0296;
tz_optimised = -0.1959;

params_intial = [f_initial, pi/180*phi_initial, pi/180*beta_initial,
xc_initial, yc_initial, zc_initial, pi/180*tx_initial, pi/180*ty_initial,
pi/180*tz_initial];
[xh_intial, yh_intial] = determineXhYh(params_intial);

params_optimised = [f_optimised, pi/180*phi_optimised,
pi/180*beta_optimised, xc_optimised, yc_optimised, zc_optimised,
pi/180*tx_optimised, pi/180*ty_optimised, pi/180*tz_optimised];
[xh_optimised, yh_optimised] = determineXhYh(params_optimised);



figure;
hold on;
plot(0-320:rawdata_length8s-1-320,realData, 'black');
plot(-xh_intial,yh_intial,'red');
plot(-320, -240:240,'green')
plot(320, -240:240,'green')
plot(-320:320, -240,'green')
plot(-320:320, 240,'green')
hold off;
axis([-320 320 -240 240]);
title('Real Data vs Equation Generated Data');
xlabel('x');
ylabel('y');
legend('Raw Data', 'Pre-Optimised');

figure;
hold on;
plot(0-320:rawdata_length8s-1-320,realData, 'black');
plot(-xh_optimised,yh_optimised,'red');
plot(-320, -240:240,'green')
plot(320, -240:240,'green')
plot(-320:320, -240,'green')
plot(-320:320, 240,'green')
```

```
hold off;
axis([-320 320 -240 240]);
title('Real Data vs Equation Generated Data');
xlabel('x');
ylabel('y');
legend('Raw Data', 'Optimised');
```

Relationship Testing

```
%Intital
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);
global iterationTimeTaken;
sum(iterationTimeTaken)

%f1
f_initial = 3300;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%f2
f_initial = 3400;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
```

```
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%p1
f_initial = 3326;
phi_initial = -1;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%p2
f_initial = 3326;
phi_initial = 5;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%b1
f_initial = 3326;
phi_initial = 0;
beta_initial = -1;
```

```
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%b2
f_initial = 3326;
phi_initial = 0;
beta_initial = 5;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%x1
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -240;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%x2
```

```
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -250;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%y1
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = -1;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%y2
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 5;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
```

```
optimisatioTimeTaken = toc(optimisationTime);

%z1
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 120;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%z2
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 130;
tx_initial = 0;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%tx1
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = -1;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
```

```
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%tx2
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 5;
ty_initial = 40;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%ty1
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 35;
tz_initial = 0;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%ty2
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 45;
tz_initial = 0;

clc;
```

```
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%tz1
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = -1;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);

%tz2
f_initial = 3326;
phi_initial = 0;
beta_initial = 0;
xc_initial = -244;
yc_initial = 0;
zc_initial = 125;
tx_initial = 0;
ty_initial = 40;
tz_initial = 5;

clc;
disp('Starting optimisation...');
optimisationIterations = 0;
optimisationTime = tic;
params = [f_initial, pi/180*phi_initial, pi/180*beta_initial, xc_initial,
yc_initial,      zc_initial,      pi/180*tx_initial,      pi/180*ty_initial,
pi/180*tz_initial];
[po0, eo0, ef0, op0] = fminunc('functionForOptimisation', params);
optimisatioTimeTaken = toc(optimisationTime);
```

Determine X and Y values for pipe radius with no erosion

```
function [xh, yh] = determineXhYh(params)

global alphaFunction;

radius = 310/2;
    focal = params(1);
    phi = params(2);
```

```
    beta = params(3);
    x_c = params(4);
    y_c = params(5);
    z_c = params(6);
    t_x = params(7);
    t_y = params(8);
    t_z = params(9);

syms xh_n;
Alpha = alphaFunction(beta, focal, phi, radius, t_x, t_y, t_z, x_c, xh_n,
y_c, z_c);

y_h = zeros(1, 640);
x_h = zeros(1, 640);

for count = 1:640
    alphaFromXhs = subs(Alpha,xh_n,count-321);

    alpha = alphaFromXhs(1);
    y = radius * sin(alpha);
    %z = -sqrt(radius^2 - y^2);
    z = -radius*cos(alpha);
    x = -y * tan(beta) - (z - radius) * tan(phi) * cos(beta);
    r1 = [1 0 0; 0 cos(t_x) -sin(t_x); 0 sin(t_x) cos(t_x)];
    r2 = [cos(t_y) 0 sin(t_y); 0 1 0; -sin(t_y) 0 cos(t_y)];
    r3 = [cos(t_z) -sin(t_z) 0; sin(t_z) cos(t_z) 0; 0 0 1];
    R = r1 * r2 * r3;
    RotatedXYZ = R * [x-x_c; y-y_c; z-z_c];                                    %
[x_d, y_d, z_d] = R[y, x, z]; Instead of = R[x, y, z];
    y_d = RotatedXYZ(1);
    x_d = RotatedXYZ(2);
    z_d = RotatedXYZ(3);
    y_h(count) = -focal * .5 * y_d / z_d;
    x_h(count) = focal * .25 * -x_d / z_d;
end

%[xh, yh] = [x_h, y_h];
xh = x_h;
yh = y_h;
```

Erosion profile to depth – calculation and use of scale factors k1 and k2

```
eH1 = 51;   %simulated erosion height

global alphaFunction;

% radius = 310/2;
% focal = 3326;
% phi = 4.8574/180*pi;
% beta = 0.4956/180*pi;
% x_c = -244.0021;
% y_c = 0.0004;
% z_c = 126.0003;
% t_x = -0.3498/180*pi;
% %t_y = 40.25/180*pi;
% t_y = 43.0297/180*pi;
% t_z = -0.1959/180*pi;

radius = 310/2;
```

```
focal = 3326;
phi = 0/180*pi;
beta = 0/180*pi;
x_c = -244.0021;
y_c = 0.0004;
z_c = 126.0003;
t_x = 0/180*pi;
%t_y = 40.25/180*pi;
t_y = 40/180*pi;
t_z = 0/180*pi;

syms xh_n;
Alpha = alphaFunction(beta, focal, phi, radius, t_x, t_y, t_z, x_c, xh_n,
y_c, z_c);

ds = zeros(1, 640);
k1s = zeros(1, 640);
k2s = zeros(1, 640);
as = zeros(1, 640);

for count = 1:640
    alphaFromXhs = subs(Alpha,xh_n,count-321);
    a = alphaFromXhs(2);
    as(count) = a;
    A = a - atan( ( radius * sin(a) ) / ( radius*cos(a) + z_c ) );

    t2t = tan(t_y) * tan(t_y);
    ka = -cos(A);
    kb = radius*cos(a);
    kc = (z_c + radius) * t2t;
    kd = focal*tan(t_y)*cos(A);
    ke = radius*cos(a);
    kf = ke + z_c + kc;
    k1 = ka / ( kb + z_c + kc );
    k2 = ( kd*(z_c + kc + radius) ) / ( 2*kf*kf );

    dy = -ErosionProfiles(eH1,count);
    d = dy / ( k1*dy + k2 );

    dy2 = ReferenceData[count]+145;
    d2 = dy2 / (k1*dy2 + k2);


    k1s(count) = k1;
    k2s(count) = k2;
    ds(count) = -d+d2;
end

figure();
plot(ds, 'green');
```

Appendix C – FPGA Handle-C Code

Camera.hcc

```
#include "DE.hch"
#include "Filter.hch"
#include "Profile.hch"
#include "RS232.hch"
interface bus_in (unsigned 1 pin) clock_pin () with {
   data = {DE_CLOCK_50}, clockport = 1};
// Connect to PLL within the FPGA to derive 86 MHz clock for camera
interface altpll(unsigned 5 clk with {clockport = 1})
        pll_(unsigned 2 inclk = 0 @ clock_pin.pin) with {
           busformat = "B[N:0]",
   properties = {
                        {"inclk0_input_frequency", "20000"},    // 50 MHz input clock
                        {"port_clk0", "PORT_USED"},
                        {"clk0_multiply_by", "43"},                          // 50 * 48 / 25 = 96 MHz
                        {"clk0_divide_by", "25"},
                        {"clk0_duty_cycle", "50"},            // 50% duty cycle
                        {"clk0_phase_shift", "0"},
                        {"compensate_clock", "CLK0"},
                        {"bandwidth_type", "AUTO"},
                        {"intended_device_family", "Cyclone III" },
                        {"lpm_hint", "CBX_MODULE_PREFIX=pll"},
                        {"lpm_type", "altpll"},
                        {"operation_mode", "NORMAL"},
                        {"pll_type", "AUTO"},
                        {"width_clock", "5"}},
           bind = 1};
set clock = internal pll_.clk[0] with {rate = 86};

extern chan unsigned 2 cselect;    // Channel for colour selection bits
extern chan unsigned 1 cselect2;    // Channel for colour selection bits
extern chan unsigned 1 test_select;
extern chan unsigned 1 ready_for_new_word;
//extern chan unsigned 1 transfer_transmit_busy;

void main( void ) {
   DE_D5M camera;         // Camera interface
   unsigned 1 hbo;        // Previous value of HBlank (for detecting end of row)
   signal unsigned 12 pv, cdiff;   // Current pixel value
   signal unsigned 16 filtered;
   signal unsigned 16 test;
   signal unsigned 13 diff;
   unsigned 2 clrbits;
   unsigned 1 clrbits2;
   unsigned 1 testing;
   unsigned 12 rr;

   //unsigned 1 ready;
   unsigned 1 busy;
```

```
  Serial serial;

  par {
    D5M_Driver( camera, 0 );       // Set up the camera driver on interface 0
    DE0_DRAMController(86,320);          // Start the DRAM controller

    seq {
      RS232_New_Serial(serial, /*4*/7, 3);
      RS232_Driver(serial, 86);
      //transmit_profile();
    }

    do par {
      cselect ? clrbits;
      cselect2 ? clrbits2;
      test_select ? testing;
      //ready_for_new_word ? ready;
      //transfer_transmit_busy ? busy;
    } while(1);

    //do {
      transmit_profile();

    //} while (1);

    v_average( camera.sync.sy && camera.sync.sx && camera.y[0], camera.x, cdiff, filtered );   //filter
the image using averaging
    peaks(camera.sync.sy  @  (camera.sync.sx  &&  camera.y[0]),  camera.x,  camera.y,  filtered);
//calculate the peaks and hence the profile

    seq {
      DE0_DRAMInitialised();     // Allow RAM to initialise
      D5M_EndFrame( camera );    // Wait for the next frame

      do par {
        if (!camera.sync.sy || !camera.sync.sx || camera.y[0] != 1) delay;  // Blanking - no pixel data
        else par {
          if(camera.x[0] == 0)    //only every second pixel starting from second pixel - ie. red
            rr = camera.pixel;
          else par {
            if(clrbits2 == 0) //selecting if clocks have switched over
              diff = (0@rr) - (0@camera.pixel);
            else
              diff = (0@camera.pixel) - (0@rr);
            cdiff = diff[12] ? 0 : (diff <-12); //Calculates red - green (& controls overflow)

            switch (clrbits) {
              case 0: pv = rr;    //display just the red
                    break;
              case 1: pv = camera.pixel;  //display just the green
```

```
                        break;
                    default: par {
                            if (!clrbits[0])    //Display red - green
                               pv = cdiff;
                            //else if(testing) pv = filtered[12] ? 0xFFF : filtered[11:0];   //Display (red - green)
filtered
                            else pv = filtered[15] ? 0xFFF : filtered[14:3];
                        }
                        break;
                }
                /*if(testing) DE0_DRAMCamera( pv[7:0] );          // Save image data to RAM FIFO
                else */DE0_DRAMCamera( pv \\4 );          // Save image data to RAM FIFO

              }
            }
            hbo = !camera.sync.sx;
            if (!hbo && !camera.sync.sx && camera.sync.sy && camera.y[0] == 1)    // Start of blanking
(end of row)
                DE0_DRAMTrigger( 0@(camera.y \\ 1), 1 );   // Send row to memory
            else delay;
          } while(1);
      }
   }
}
```

Display.hcc

```
#include "DE.hch"
#include "Profile.hch"

set clock = external_divide DE_CLOCK_50 2 with {rate = 25};
macro expr CLOCK_FREQ = 25000000;

chan unsigned 2 cselect;    // Channel for colour selection bits
chan unsigned 1 cselect2;    // Channel for colour selection bits
chan unsigned 1 test_select;

void main( void ) {
   DE_VGA vga;          // The VGA signal interface
   DE_I2C camera_control;
   signal unsigned 8 dd;
   static unsigned 4 zero = 0;
   unsigned 10 sw;
   unsigned 12 clr;
   unsigned 3 key;

   par {
      DE_VGADriver(25, VGA_640_480, vga, 0, 0 );  // Run the driver
      D5M_CameraControl(25, camera_control, 0);
      display_profile(vga.x, vga.y, !vga.sync_disp.sy, !vga.sync_disp.sx, dd[7:4]@0x00, clr); //run the
code to display the profile
      DE_Debounce( key );
```

```
    /*do {
       unsigned 3 k;

       do { k = key; } while (!(~k & key));    //wait for key press
       switch (~k & key) {
          case 1:
             //sendData ! 1;
             break;
          //case 2:
             //break;
          //case 3:
             //break;
          default:
             delay;
             break;
       }
    } while (1);*/

    do par {
       if (sw[3]) par {
          vga.r = clr[11:8];     // Display Profile
          vga.g = clr[7:4];
          vga.b = clr[3:0];
       }
       else par {
          vga.r = dd[7:4];     // Dont display profile
          vga.g = zero;
          vga.b = zero;
       }

    } while (1);

    do par {
       //unsigned 10 sw;
       DE_ReadSw( sw );    // Read switch state
       cselect ! sw[1:0];  // Send 2 LSBs
       cselect2 ! sw[2];
       test_select ! sw[4];
    } while (1);
    seq {
       //-- 640x480 window
       /*
       D5M_Write( camera_control, 3, 479, 1 );     // Set window to 2560x960
       D5M_Write( camera_control, 4, 639, 1 );
       D5M_Write( camera_control, 35, 0x00, 1 );   //Column mode: x2 column binning and skipping
1 column
       D5M_Write( camera_control, 1, 786, 1 );     // Centre image in FOV: 1944/2 - 960/2 + 54 = 546
       D5M_Write( camera_control, 2, 992, 1 );      //                  2592/2 - 2560/2 + 16 = 32
       */
```

```
        //-- 1280x480 window
        /*
        D5M_Write( camera_control, 3, 479, 1 );    // Set window to 1280x480
        D5M_Write( camera_control, 4, 1279, 1 );
        D5M_Write( camera_control, 35, 0x11, 1 );  //Column mode: x2 column binning and skipping
1 column
        D5M_Write( camera_control, 1, 786, 1 );    // Centre image in FOV: 1944/2 - 480/2 + 54 = 786;
        D5M_Write( camera_control, 2, 775, 1 );    //                  2592/2 - 1280/2 + 16 = 775;
        */

        //-- 2560x480 window
        /*
        D5M_Write( camera_control, 3, 479, 1 );    // Set window to 2560x960
        D5M_Write( camera_control, 4, 2559, 1 );
        D5M_Write( camera_control, 35, 0x11, 1 );  //Column mode: x2 column binning and skipping
1 column
        D5M_Write( camera_control, 1, 786, 1 );    // Centre image in FOV: 1944/2 - 480/2 + 54 = 786;
        D5M_Write( camera_control, 2, 32, 1 );     //                  2592/2 - 2560/2 + 16 = 32
        */

        //-- 2560x960 window
        //*
        D5M_Write( camera_control, 3, 959, 1 );    // Set window to 2560x960
        D5M_Write( camera_control, 4, 2559, 1 );
        D5M_Write( camera_control, 35, 0x11, 1 );  //Column mode: x2 column binning and skipping
1 column
        D5M_Write( camera_control, 1, 546, 1 );    // Centre image in FOV: 1944/2 - 960/2 + 54 = 546
        D5M_Write( camera_control, 2, 32, 1 );     //                  2592/2 - 2560/2 + 16 = 32
        //*/

        D5M_Write( camera_control, 32, 0xC040, 1 ); // Flip rows and columns
        D5M_Write( camera_control, 9, 500, 1 );    // Set exposure time
        D5M_Write( camera_control, 53, 0x3F, 0 );  // Set camera gain

        do { delay; } while (vga.sync_disp.sy);  // Wait for a frame to be captured first!
        par {
          do {                  // Trigger a fetch of row for display
            if (vga.x == 384)      // Trigger part way through previous row
              if (vga.y == vga.maxy)
                DE0_DRAMTrigger( 0, 0 );          // First row
              else if (vga.sync_disp.sy && vga.y != vga.visy)  // Remainder
                DE0_DRAMTrigger( 0@(vga.y+1), 0 );
              else delay;
            else delay;
          } while (1);
          do {
            if (vga.sync_disp.sx && vga.sync_disp.sy)
              DE0_DRAMDisplay( dd );     // Read pixel from frame buffer FIFO
            else delay;
          } while (1);
        }
```

```
    }


  }

}
```

Filter.hch

```
typedef mpram {
  rom unsigned 12 r[640];
  wom unsigned 12 w[640];
} row_buff_12;

macro proc v_average( x_valid, x, p_in, p_out );
```

Filter.hcc

```
#include "filter.hch"

macro proc v_average( x_valid, x, p_in, p_out ) {
  row_buff_12 rb[10] with {block = "M9K"};
  unsigned 12 rbo[10];

  do par {
    par (i = 0; i < 10; i++ ) {
      if (x_valid && !x[0])
        rbo[i] = rb[i].r[x[10:1]];    // Read row buffers into register
      else if (x_valid && x[0])
        ifselect( i == 0)
          rb[0].w[x[10:1]] = p_in;      // First row buffer come from input pixel
        else
          rb[i].w[x[10:1]] = rbo[i-1];  // Write into next row buffer
      else delay;
    }
    if (x_valid && x[0])
      p_out = ((0@p_in) + ((0@rbo[0]) + (0@rbo[1]))) +
          (((0@rbo[2]) + (0@rbo[3])) + ((0@rbo[4]) + (0@rbo[5]))) +
          (((0@rbo[6]) + (0@rbo[7])) + ((0@rbo[8]) + (0@rbo[9])));
    else delay;

  } while (1);
}
```

Profile.hch

```
typedef mpram{
  rom unsigned 16 r[640];
  wom unsigned 16 w[640];
} max_buff_16;

typedef mpram{
  rom unsigned 10 r[640];
  wom unsigned 10 w[640];
```

```
} row_buff_10;

typedef mpram{
   rom unsigned 9 r[640];
   wom unsigned 9 w[640];
} row_buff_9;

macro proc peaks(valid, x, y, value);
macro proc xfer_for_transmit(rb);
macro proc xfer_profile(rb,mb);
macro proc transmit_profile();
macro proc display_profile(disp_x, disp_y, vb, hb, clr_in, clr_out);
```

Profile.hcc

```
#include "Profile.hch"
#include "RS232.hch"

//--Reference profile
//rom unsigned 10 mem[640] = {
//#include "referenceCurve.txt"
//};

chan unsigned 1 transfer;
unsigned 1 xfer;

macro proc peaks(valid, x, y, value ) {
   row_buff_10 rb with {block = "M9K"};
   max_buff_16 mb with {block = "M9K"};
   unsigned 16 mbo;
   unsigned 16 r_value;

   do {

     while(valid[1]) par {    //within vertical frame
        if(valid[0]&& x[0])    //Green pixel
        {
           if(r_value > mbo) //compares saved read value with max and replaces mb and rb if
necessary (with red value and row) //-- May have less noise with < as opposed to <= - why???
              par {
                 mb.w[x[10:1]] = r_value;
                 rb.w[x[10:1]] = y\\1;
              }
           else delay;
        }
        else if (valid[0] && !x[0]) //Red pixel - saves red value, looks up max for that x
           par {
              r_value = value;
              //r_value = value[15] ? 0xFFFF : 0@value[14:0];    //--May cause some noise - why????
              mbo = mb.r[x[10:1]];
           }
        else delay;
```

```
      }

      //Transfer to transmit_buff only if serial is ready for it
      if(xfer)
      {
         xfer = 0;
         xfer_for_transmit(rb);
         transfer ! 1;
      }
      else delay;

      //Transfer profile to display and clear rb and mb buffers
      xfer_profile(rb,mb);

      do { delay; } while (!valid[1]); //wait for next frame
   } while (1);
}

//Global RAM used to transfer peaks between clock domains (Camera -> Display)
mpram {
   wom unsigned 10 w[640]; //Write port accessible from camera
   rom unsigned 10 r[640]; //Read port accessible from display
} transfer_buff with {block = "M9K"};

chan unsigned 1 transfer_busy; //used to signal that transfer is in place

//Transfer profile to display
macro proc xfer_profile(rb,mb) {
   unsigned 10 address;

   address = 0;
   transfer_busy ! 1;  //Indicate transfer taking place

   do par{
      //transfer_buff.w[address] = rb.r[address] - mem[address];   //Populate transfer buffer
(subtracts reference profile)
      transfer_buff.w[address] = rb.r[address];
      rb.w[address] = 0;  //-- dont need if use >= instead of just > in line 13
      mb.w[address]=0;    //Reset max buffer
      address++;
   } while(address != 641);

   transfer_busy ! 0;  //Indicate transfer complete
}

//Global RAM used to transfer profile for transmitting
mpram {
   wom unsigned 8 w[640];
   rom unsigned 8 r[640];
} transmit_buff with {block = "M9K"};
```

```
//Transfer profile for serial
macro proc xfer_for_transmit(rb) {
   unsigned 10 address, temp;

   address = 0;
   do par {
      seq {
         temp = rb.r[address];
         transmit_buff.w[address] = temp[7:0];
      }
      address++;
   } while(address != 641);
}

extern chan unsigned 8 tx_chan;

//Send profile over serial
macro proc transmit_profile() {
   unsigned 8 stop_word, word;
   unsigned 1 go;
   unsigned 10 x;
   ram unsigned 8 data[640] with {block="M9K"};

   //Initialisation
   par {
      stop_word = 85;
      xfer = 1;
   }

   do {
      transfer ? go;
         //Load profile from transmit_buff
         x=0;
         do {
            unsigned 8 v;
            v = transmit_buff.r[x];
            data[x] =  v;
            x++;
         } while(x!=640);

         //Transmit data
         x = 0;
         do {
            unsigned 8 d;
            d = data[x];
            if(d == stop_word) d--; else delay;
            tx_chan ! d;
            x++;
         } while (x != 640);
         tx_chan ! stop_word;
```

```
        xfer = 1;
    } while (1);
}

/*mpram {
    wom unsigned 8 w[800];
    rom unsigned 8 r[800];
} transmit_buff with {block = "M9K"};

macro proc xfer_for_transmit(rb) {
    unsigned 10 address, address2, temp;
    unsigned 8 remainder;
    unsigned 3 c;

    //Intialise
    par{
        c = 0;
        address = 0;
        address2 = 0;
    }

    do par {
        switch (c) {
            case 0:
            temp = rb.r[address];
            par {
                transmit_buff.w[address2] = temp[9:2];
                remainder = temp[1:0]@((unsigned 6) 0);
                c++;
                address2++;
            } break;
            case 1:
            temp = rb.r[address];
            par {
                transmit_buff.w[address2] = remainder[7:6] @ temp[9:4];
                remainder = temp[3:0]@((unsigned 4) 0);
                c++;
                address2++;
            } break;
            case 2:
            temp = rb.r[address];
            par {
                transmit_buff.w[address2] = remainder[7:4] @ temp[9:6];
                remainder = temp[5:0]@((unsigned 2) 0);
                c++;
                address2++;
            } break;
            case 3:
            temp = rb.r[address];
            par {
                transmit_buff.w[address2] = remainder[7:2] @ temp[9:8];
```

```
            remainder = 0;
            address2++;
          }
          par {
            transmit_buff.w[address2] = temp[7:0];
            c=0;
            address2++;
          } break;
          default:
          delay;
          break;
        }
        address++;
    } while(address != 641);
}

extern chan unsigned 8 tx_chan;

//Send profile over serial
macro proc transmit_profile() {
    unsigned 8 stop_word, word;
    unsigned 1 go;
    unsigned 10 x;
    ram unsigned 8 data[800] with {block="M9K"};

    //Initialisation
    par {
      stop_word = 85;
      xfer = 1;
    }

    do {
      transfer ? go;
        //Load profile from transmit_buff
        x=0;
        do {
          unsigned 8 v;
          v = transmit_buff.r[x];
          data[x] =  v;
          x++;
        } while(x!=800);

        //Transmit data
        x = 0;
        do {
          unsigned 8 d;
          d = data[x];
          if(d == stop_word) d--; else delay;
          tx_chan ! d;
          x++;
        } while (x != 800);
```

```
        tx_chan ! stop_word;

        xfer = 1;
    } while (1);
}*/


//Display Profile
macro proc display_profile(disp_x, disp_y, vb, hb, clr_in, clr_out) {
    unsigned 1 busy, new_data, hbo;
    unsigned 10 x, y;
    ram unsigned 10 profile[640] with {block="M9K"};

    par {
        do {
            transfer_busy ? busy;   //Receive busy flag
            if (!busy) new_data = 1;    //Just transferred new profile
        } while(1);

        do {
            do { delay; } while (busy); //Wait if profile is transferring
            if (new_data) {
                x = 0;
                do {
                    unsigned 10 v;
                    v = transfer_buff.r[x];
                    par {
                        profile[x]=v;
                        x++;
                    }
                } while (x != 640);
            }
            else delay;

            par {
                x=0;
                y=0;
            }
            do { delay; } while (vb);

            do par {
                hbo = hb;
                if(!hb) par {
                    if (y == profile[x]) {
                        clr_out = clr_in[11:8] @ 0xf @ clr_in[3:0];
                    }
                    else {
                        clr_out = clr_in;
                    }
                    x++;
                }
```

```
            else if (!hbo) par { x = 0; y++; }
            else delay;
         } while (!vb);
      } while (1);
   }
}
```

Peaks.hcc

```
/*#include "Profile.hch"

macro proc peaks( valid, x, y, value ) {
   row_buff_10 rb with {block = "M9K"};
   max_buff_16 mb with {block = "M9K"};
   unsigned 16 mbo;
   unsigned 16 r_value;

   do {
     while(valid[1]) par {    //within vertical frame
         if(valid[0] && !x[0])    //Green pixel
         {
            if(r_value >= mbo) //compares saved read value with max and replaces mb and rb if
necessary (with red value and row)
               par {
                  mb.w[x[10:1]] = r_value;
                  rb.w[x[10:1]] = y[10:1];
               }
            else delay;
         }
         else if (valid[0] && x[0]) //Red pixel - saves red value, looks up max for that x
            par {
               r_value = value;
               mbo = mb.r[x[10:1]];
            }
         else delay;
     }
     xfer_profile(rb);//clear/transfer buffers
     do { delay; } while (!valid[1]); //wait for next frame
   } while (1);
}

//Global RAM used to transfer peaks between clock domains
mpram {
   wom unsigned 10 w[640]; //Write port accessible from camera
   rom unsigned 10 r[640]; //Read port accessible from display
} transfer_buff with {block = "M9K"};

chan unsigned 1 transfer_busy; //used to signal that transfer is in place

//Transfers Peaks, and resets it for the next frame
macro proc xfer_profile(rb) {
   unsigned 10 address;
```

```
   address = 0;
   transfer_busy ! 1;  //Indicate transfer taking place
   do par{
      transfer_buff.w[address] = rb.r[address];
      rb.w[address] = 0;
      address++;
   } while(address != 641);
   transfer_busy ! 0;  //Indicate transfer complete
}

//Display Profile
macro proc display_profile(disp_x, disp_y, vb, hb, clr_in, clr_out) {
   unsigned 1 busy, new_data, hbo;
   unsigned 10 x, y;
   ram unsigned 10 profile[640] with {block="M9K"};

   par {
      do {
         transfer_busy ? busy;   //Receive busy flag
         if (!busy) new_data = 1;    //Just transferred new profile
      } while(1);

      do {
         do { delay; } while (busy); //Wait if profile is transferring
         if (new_data) {
            x = 0;
            do {
               unsigned 10 v;
               v = transfer_buff.r[x];
               par {
                  profile[x]=v;
                  x++;
               }
            } while (x != 640);
         }
         else delay;

         x=0;
         y=0;
         do { delay; } while (vb);

         do par {
            hbo = hb;
            if(!hb) par {
               if (y == profile[x]) {
                  clr_out = clr_in[11:8] @ 0xf @ clr_in[3:0];
               }
               else {
                  clr_out = clr_in;
               }
```

```
            x++;
          }
          else if (!hbo) par { x = 0; y++; }
          else delay;
        } while (!vb);
      } while (1);
    }
}
*/
```

RS232.hch

```
typedef struct {
   unsigned 3 baud_rate;
   unsigned 2 stop_bits;
   unsigned 3 parity;
   unsigned 2 word_length;
} Serial;

macro proc RS232_New_Serial(Serial, baud_rate, word_length);
macro proc RS232_Driver(Serial, Clk_Rate);
```

RS232.hcc

```
#include "RS232.hch"
#include "DE.hch"
#include "Profile.hch"

unsigned 10 tx_sr;

interface bus_out () TX_pin (unsigned 1 OutPort = tx_sr[0])
   with {data = {"U21"}};

macro proc RS232_New_Serial(serial, baud_rate, word_length)
{
   par {
      serial.baud_rate = baud_rate;
       /*BAUD_RATE:
         000 -> 0 -> 115200
         001 -> 1 -> 57600
         010 -> 2 -> 38400
         011 -> 3 -> 19200
         100 -> 4 -> 9600
         101 -> 5 -> 4800
         110 -> 6 -> 2400
         111 -> 7 -> 1200*/
      serial.stop_bits = 1;   //Fixed
      /*STOP_BITS:
         00 -> 0 -> 1
         01 -> 1 -> 1
         10 -> 2 -> 1.5
         11 -> 3 -> 2*/
      serial.parity = 0;  //Fixed
```

```
      /*PARITY:
        000 -> 0 -> None
        001 -> 1 -> Odd
        010 -> 2 -> Even
        011 -> 3 -> Mark
        100 -> 4 -> Space
        101 -> 5 -> None
        110 -> 6 -> None
        111 -> 7 -> None*/
      serial.word_length = 3;
       /*WORD_LENGTH:
        00 -> 0 -> 5
        01 -> 1 -> 6
        10 -> 2 -> 7
        11 -> 3 -> 8*/
   }
}

chan unsigned 8 tx_chan;

macro proc RS232_Driver(Serial, Clk_Rate)
{
   unsigned 20 clk_div, count;
   unsigned 8 data;
   unsigned 6 bits;

   switch (Serial.baud_rate) { //Select divisor based on board rate
      case 0:
         //clkdiv = 8;//7; //115200 Baud
         //clk_div = 747;
         clk_div = Clk_Rate*10000/1152;
         break;
      case 1:
         //clkdiv = 16;//15;   //57600 Baud
         //clk_div = 1493;
         clk_div = Clk_Rate*10000/576;
         break;
      case 2:
         //clkdiv = 24;//23;   //38400 Baud
         //clk_div = 2240;
         clk_div = Clk_Rate*10000/384;
         break;
      case 3:
         //clkdiv = 48;//47;   //19200 Baud
         //clk_div = 4479;
         clk_div = Clk_Rate*10000/192;
         break;
      case 4:
         //clkdiv = 96;//95;   //9600 Baud
         //clk_div = 8958;
         clk_div = Clk_Rate*10000/96;
```

```
      break;
   case 5:
      //clkdiv = 192;//191; //4800 Baud
      //clk_div = 17917;
      clk_div = Clk_Rate*10000/48;
      break;
   case 6:
      //clkdiv = 384;//383; //2400 Baud
      //clk_div = 35833;
      clk_div = Clk_Rate*10000/24;
      break;
   case 7:
      //clkdiv = 768;//767; //1200 Baud
      //clk_div = 71667;
      clk_div = Clk_Rate*10000/12;
      break;
   default:
      //clkdiv = 96;//95;   //Default 9600 Baud
      clk_div = 8958;
      break;
}

clk_div = 71667;

do {
   /*data = 0xff;
   par {
      tx_sr = ((unsigned 1) 1)@data@((unsigned 1) 1);
      count = clk_div;
      bits = 10;
   }
   do {
      do { count--; } while (count);
      par {
         count = clk_div;
         bits--;
         tx_sr = 1@(tx_sr\\1);
      }
   } while (bits);*/

   tx_chan ? data;
   par {
      tx_sr = ((unsigned 1) 1)@data@((unsigned 1) 0);
      count = clk_div;
      bits = 10;
   }
   do {
      do { count--; } while (count);
      par {
         count = clk_div;
         bits--;
```

```
            tx_sr = 1@(tx_sr\\1);
          }
        } while (bits);
    } while (1);
}
```

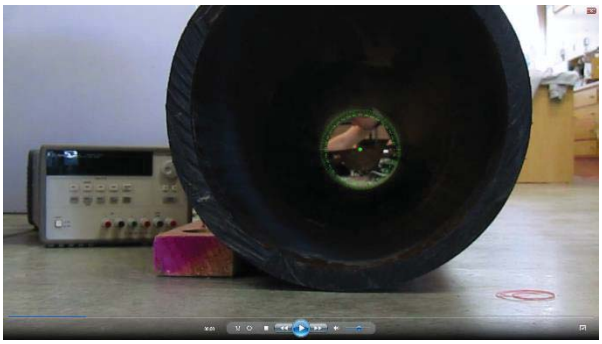
Appendix D - Anti-rolling Test Results

Test results summary table:

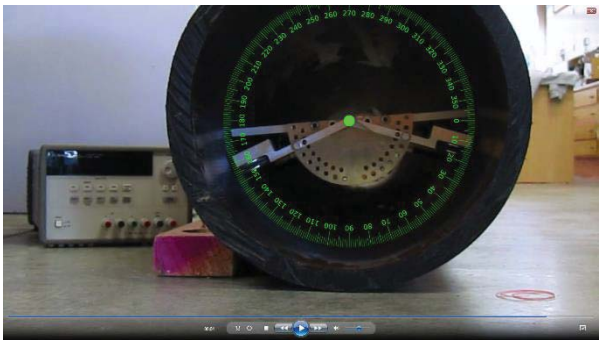
Test	Starting Angle (Degrees)	Finishing Angle (Degrees)	Offset Weight (kg)	Offset Distance (mm)	Rotation (Degrees)
A	356	356	0	0	0
B	3	3	0	0	0
C	1	1	0	0	0
D	13	13	0	0	0
E	28	28	0	0	0
F	56	56	0	0	0
G	27	26	0.5	75	-1
H	27	27	1.2	75	0
I	24	23	2.5	75	-1

Test result images corresponding to above tests A-I:

A:



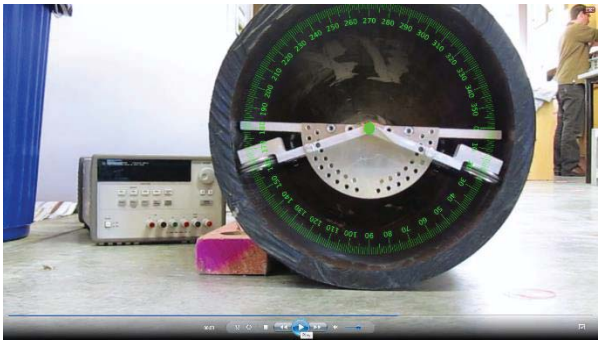
Start Orientation Angle



Finish Orientation Angle

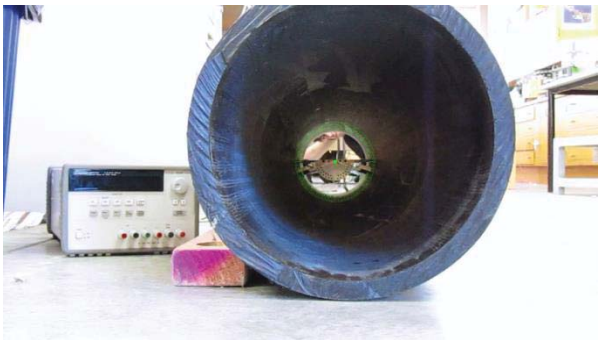
B:

Start Orientation Angle



Finish Orientation Angle

C:

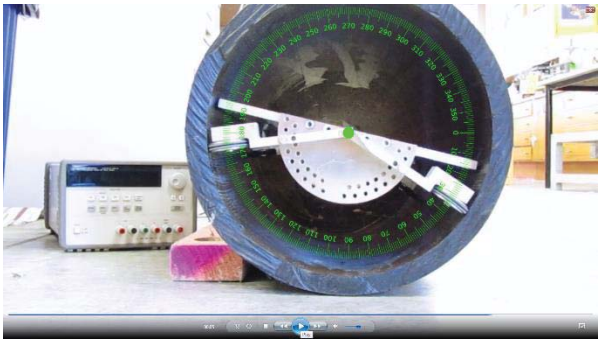


Start Orientation Angle

Finish Orientation Angle

D:

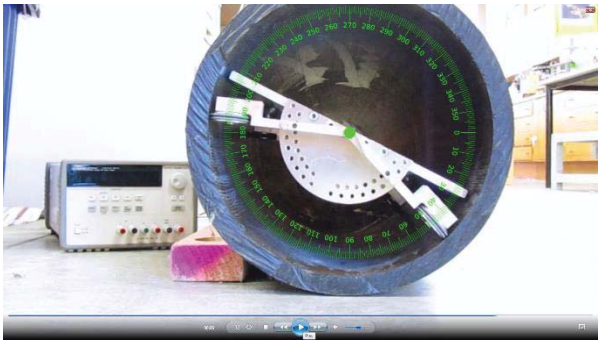
Start Orientation Angle



Finish Orientation Angle

E:

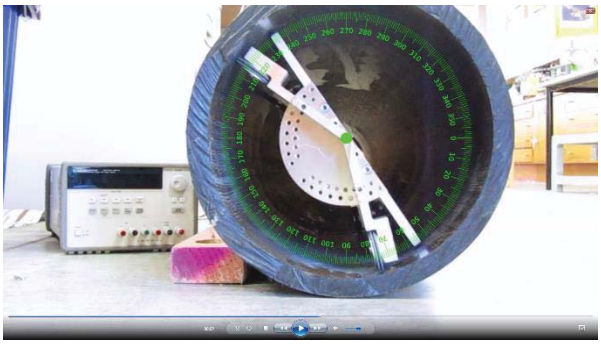
Start Orientation Angle



Finish Orientation Angle

F:

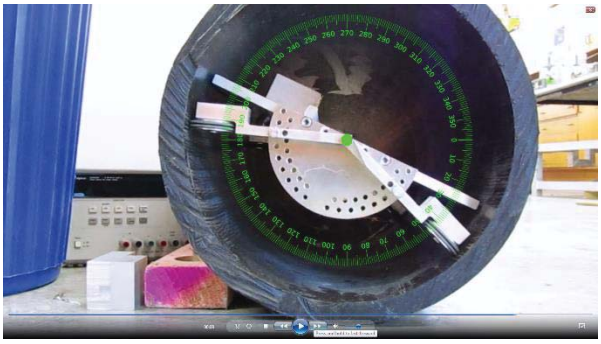
Start Orientation Angle



Finish Orientation Angle

G:

Start Orientation Angle



Finish Orientation Angle

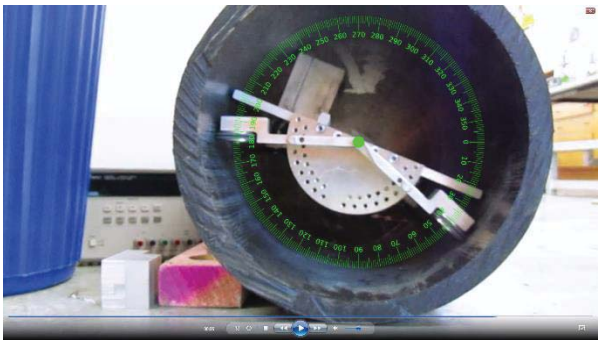
H:

Start Orientation Angle

Finish Orientation Angle

I:

Start Orientation Angle



Finish Orientation Angle

Appendix E – Image System Scale Factors

Scale factors K1 and K2 by image column for the calibrated image system:

Column:	1	2	3	4	5	6	7	8
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.579	3.574	3.570	3.566	3.562	3.557	3.553	3.549

Column:	9	10	11	12	13	14	15	16
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.545	3.541	3.537	3.532	3.528	3.524	3.520	3.516

Column:	17	18	19	20	21	22	23	24
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.512	3.508	3.504	3.500	3.496	3.492	3.489	3.485

Column:	25	26	27	28	29	30	31	32
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.481	3.477	3.473	3.469	3.466	3.462	3.458	3.454

Column:	33	34	35	36	37	38	39	40
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.451	3.447	3.443	3.440	3.436	3.432	3.429	3.425

Column:	41	42	43	44	45	46	47	48
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.421	3.418	3.414	3.411	3.407	3.404	3.400	3.397

Column:	49	50	51	52	53	54	55	56
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.393	3.390	3.387	3.383	3.380	3.377	3.373	3.370

Column:	57	58	59	60	61	62	63	64
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.367	3.363	3.360	3.357	3.354	3.350	3.347	3.344

Column:	65	66	67	68	69	70	71	72
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.341	3.338	3.334	3.331	3.328	3.325	3.322	3.319

Column:	73	74	75	76	77	78	79	80
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.316	3.313	3.310	3.307	3.304	3.301	3.298	3.295

Column:	81	82	83	84	85	86	87	88
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.292	3.289	3.286	3.283	3.280	3.278	3.275	3.272

Column:	89	90	91	92	93	94	95	96
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.269	3.266	3.263	3.261	3.258	3.255	3.252	3.250

Column:	97	98	99	100	101	102	103	104
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.247	3.244	3.242	3.239	3.236	3.234	3.231	3.229

Column:	105	106	107	108	109	110	111	112
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.226	3.223	3.221	3.218	3.216	3.213	3.211	3.208

Column:	113	114	115	116	117	118	119	120
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.206	3.203	3.201	3.198	3.196	3.194	3.191	3.189

Column:	121	122	123	124	125	126	127	128
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.186	3.184	3.182	3.179	3.177	3.175	3.173	3.170

Column:	129	130	131	132	133	134	135	136
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.168	3.166	3.164	3.161	3.159	3.157	3.155	3.153

Column:	137	138	139	140	141	142	143	144
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.150	3.148	3.146	3.144	3.142	3.140	3.138	3.136

Column:	145	146	147	148	149	150	151	152
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.134	3.132	3.130	3.128	3.126	3.124	3.122	3.120

Column:	153	154	155	156	157	158	159	160
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.118	3.116	3.114	3.112	3.110	3.108	3.106	3.104

Column:	161	162	163	164	165	166	167	168
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.102	3.101	3.099	3.097	3.095	3.093	3.092	3.090

Column:	169	170	171	172	173	174	175	176
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.088	3.086	3.085	3.083	3.081	3.079	3.078	3.076

Column:	177	178	179	180	181	182	183	184

K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.074	3.073	3.071	3.070	3.068	3.066	3.065	3.063
Column:	185	186	187	188	189	190	191	192
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.062	3.060	3.058	3.057	3.055	3.054	3.052	3.051
Column:	193	194	195	196	197	198	199	200
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.049	3.048	3.047	3.045	3.044	3.042	3.041	3.039
Column:	201	202	203	204	205	206	207	208
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.038	3.037	3.035	3.034	3.033	3.031	3.030	3.029
Column:	209	210	211	212	213	214	215	216
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.027	3.026	3.025	3.024	3.022	3.021	3.020	3.019
Column:	217	218	219	220	221	222	223	224
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.018	3.016	3.015	3.014	3.013	3.012	3.011	3.009
Column:	225	226	227	228	229	230	231	232
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.008	3.007	3.006	3.005	3.004	3.003	3.002	3.001
Column:	233	234	235	236	237	238	239	240
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.000	2.999	2.998	2.997	2.996	2.995	2.994	2.993
Column:	241	242	243	244	245	246	247	248
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.992	2.991	2.990	2.989	2.989	2.988	2.987	2.986
Column:	249	250	251	252	253	254	255	256
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.985	2.984	2.983	2.983	2.982	2.981	2.980	2.979
Column:	257	258	259	260	261	262	263	264
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.979	2.978	2.977	2.976	2.976	2.975	2.974	2.974
Column:	265	266	267	268	269	270	271	272
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.973	2.972	2.972	2.971	2.970	2.970	2.969	2.969

Column:	273	274	275	276	277	278	279	280
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.968	2.967	2.967	2.966	2.966	2.965	2.965	2.964

Column:	281	282	283	284	285	286	287	288
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.964	2.963	2.963	2.962	2.962	2.961	2.961	2.961

Column:	289	290	291	292	293	294	295	296
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.960	2.960	2.959	2.959	2.959	2.958	2.958	2.957

Column:	297	298	299	300	301	302	303	304
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.957	2.957	2.957	2.956	2.956	2.956	2.955	2.955

Column:	305	306	307	308	309	310	311	312
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.955	2.955	2.954	2.954	2.954	2.954	2.954	2.953

Column:	313	314	315	316	317	318	319	320
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.953	2.953	2.953	2.953	2.953	2.953	2.953	2.952

Column:	321	322	323	324	325	326	327	328
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.952	2.952	2.952	2.952	2.952	2.952	2.952	2.952

Column:	329	330	331	332	333	334	335	336
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.952	2.952	2.952	2.952	2.952	2.952	2.952	2.952

Column:	337	338	339	340	341	342	343	344
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.952	2.953	2.953	2.953	2.953	2.953	2.953	2.953

Column:	345	346	347	348	349	350	351	352
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.954	2.954	2.954	2.954	2.954	2.955	2.955	2.955

Column:	353	354	355	356	357	358	359	360
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.955	2.956	2.956	2.956	2.956	2.957	2.957	2.957

Column:	361	362	363	364	365	366	367	368

K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.958	2.958	2.958	2.959	2.959	2.960	2.960	2.960
Column:	369	370	371	372	373	374	375	376
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.961	2.961	2.962	2.962	2.963	2.963	2.964	2.964
Column:	377	378	379	380	381	382	383	384
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.965	2.965	2.966	2.966	2.967	2.967	2.968	2.969
Column:	385	386	387	388	389	390	391	392
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.969	2.970	2.970	2.971	2.972	2.972	2.973	2.974
Column:	393	394	395	396	397	398	399	400
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.974	2.975	2.976	2.976	2.977	2.978	2.979	2.979
Column:	401	402	403	404	405	406	407	408
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.980	2.981	2.982	2.982	2.983	2.984	2.985	2.986
Column:	409	410	411	412	413	414	415	416
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.987	2.987	2.988	2.989	2.990	2.991	2.992	2.993
Column:	417	418	419	420	421	422	423	424
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	2.994	2.995	2.996	2.997	2.998	2.999	3.000	3.001
Column:	425	426	427	428	429	430	431	432
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.002	3.003	3.004	3.005	3.006	3.007	3.008	3.009
Column:	433	434	435	436	437	438	439	440
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.010	3.011	3.013	3.014	3.015	3.016	3.017	3.018
Column:	441	442	443	444	445	446	447	448
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.020	3.021	3.022	3.023	3.025	3.026	3.027	3.028
Column:	449	450	451	452	453	454	455	456
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.030	3.031	3.032	3.034	3.035	3.036	3.038	3.039

Column:	457	458	459	460	461	462	463	464
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.040	3.042	3.043	3.045	3.046	3.048	3.049	3.050

Column:	465	466	467	468	469	470	471	472
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.052	3.053	3.055	3.056	3.058	3.060	3.061	3.063

Column:	473	474	475	476	477	478	479	480
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.064	3.066	3.067	3.069	3.071	3.072	3.074	3.076

Column:	481	482	483	484	485	486	487	488
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.077	3.079	3.081	3.082	3.084	3.086	3.087	3.089

Column:	489	490	491	492	493	494	495	496
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.091	3.093	3.094	3.096	3.098	3.100	3.102	3.104

Column:	497	498	499	500	501	502	503	504
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.105	3.107	3.109	3.111	3.113	3.115	3.117	3.119

Column:	505	506	507	508	509	510	511	512
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.121	3.123	3.125	3.127	3.129	3.131	3.133	3.135

Column:	513	514	515	516	517	518	519	520
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.137	3.139	3.141	3.143	3.145	3.147	3.149	3.151

Column:	521	522	523	524	525	526	527	528
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.154	3.156	3.158	3.160	3.162	3.165	3.167	3.169

Column:	529	530	531	532	533	534	535	536
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.171	3.174	3.176	3.178	3.180	3.183	3.185	3.187

Column:	537	538	539	540	541	542	543	544
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.190	3.192	3.195	3.197	3.199	3.202	3.204	3.207

Column:	545	546	547	548	549	550	551	552

K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.209	3.212	3.214	3.217	3.219	3.222	3.224	3.227
Column:	553	554	555	556	557	558	559	560
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.229	3.232	3.235	3.237	3.240	3.243	3.245	3.248
Column:	561	562	563	564	565	566	567	568
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.251	3.253	3.256	3.259	3.261	3.264	3.267	3.270
Column:	569	570	571	572	573	574	575	576
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.273	3.275	3.278	3.281	3.284	3.287	3.290	3.293
Column:	577	578	579	580	581	582	583	584
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.296	3.298	3.301	3.304	3.307	3.310	3.313	3.316
Column:	585	586	587	588	589	590	591	592
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.319	3.322	3.326	3.329	3.332	3.335	3.338	3.341
Column:	593	594	595	596	597	598	599	600
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.344	3.347	3.351	3.354	3.357	3.360	3.364	3.367
Column:	601	602	603	604	605	606	607	608
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.370	3.373	3.377	3.380	3.383	3.387	3.390	3.394
Column:	609	610	611	612	613	614	615	616
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.397	3.400	3.404	3.407	3.411	3.414	3.418	3.421
Column:	617	618	619	620	621	622	623	624
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.425	3.428	3.432	3.436	3.439	3.443	3.447	3.450
Column:	625	626	627	628	629	630	631	632
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.454	3.458	3.461	3.465	3.469	3.473	3.476	3.480
Column:	633	634	635	636	637	638	639	640
K1:	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002	-0.002
K2:	3.484	3.488	3.492	3.496	3.500	3.503	3.507	3.511

Appendix F – Battery Calculation Data

General battery and motor calculations:

Parameters				
Symbol	**Value**	**Equation**	**Units**	**Desciption**
m	10	10	kg	Mass
phi	70	70	degrees	Angle of wheels from vertical
theta	5	5	degrees	Max angle of incline
u	0.05	0.05		Rolling resistance coefficient (generally 0.010->0.015)
Cd	1	1		Drag Coefficient
A	10625	10625	mm^2	Cross sectional area
v	0.75	0.75	m*s^-1	Desired velocity
a	0.5	0.5	m*s^-2	Desired acceleration
D	45	45	mm	Diameter of drive wheel
N	2	2	#	Number of drive wheels
e	65	65	%	Efficiency
V	12	12	V	Supply voltage
t	2	2	hr	Required running time

Constants				
Symbol	**Value**	**Value**	**Units**	**Desciption**
g	9.81	9.81	m*s^-2	Gravity
p	1.2	1.2	kg*m^-3	Density of Air

Calculations				
Symbol	**Value**	**Value**	**Units**	**Desciption**
rpm	318.3099	=(B9*60000)/(PI()*B11)	rpm	Required RPM
f_hill	8.549978	=C3*C19*SIN(C5*PI()/180)	N	Force to overcome not to roll back down incline
f_roll	14.34126	=(C6*C3*C19)/COS(C4*PI()/180)	N	Force to overcome for steady forward motion
f_air	0.003586	=0.5*C7*C8/1000000*C20*C9*C9	N	Force due to air resistance
F	22.89482	=C25+C26+C27	N	Total drag
P_v	26.41711	=C28*C9*100/C13	W	Power for constant forward motion
P_a	2.884615	=0.5*C3*C9*C10*100/C13	W	Power for acceleration
P	29.30172	=C29+C30	W	Total power reqquired
P_m	14.65086	=C31/C12	W	Power per motor
T	0.439526	=C32/(C24/60*2*PI())	Nm	Motor Torque (per motor)
I	2.44181	=C31/C14	Amps	Maximum Current (all motors)
C	4.88362	=C34*C15	Ah	Battery capacity (all motors)

Motor sizing for different diameter wheels:

Symbol	Units	Value						
m	kg	13	13	13	13	13	13	13
phi	degrees	70	70	70	70	70	70	70
theta	degrees	5	5	5	5	5	5	5
u		0.015	0.015	0.015	0.015	0.015	0.015	0.015
Cd		1	1	1	1	1	1	1
A	mm^2	10625	10625	10625	10625	10625	10625	10625
v	m*s^-1	3	3	3	3	3	3	3
a	m*s^-2	1.5	1.5	1.5	1.5	1.5	1.5	1.5
D	mm	45	50	55	60	65	70	75
N	#	2	2	2	2	2	2	2
e	%	75	75	75	75	75	75	75
V	V	12	12	12	12	12	12	12
t	hr	1	1	1	1	1	1	1

Symbol	Units	Value						
g	m*s^-2	9.81	9.81	9.81	9.81	9.81	9.81	9.81
p	kg*m^-3	1.2	1.2	1.2	1.2	1.2	1.2	1.2

Symbol	Units	Value						
rpm	rpm	**1273.24**	**1145.92**	**1041.74**	**954.93**	**881.47**	**818.51**	**763.94**
f_hill	N	11.11	11.11	11.11	11.11	11.11	11.11	11.11
f_roll	N	5.59	5.59	5.59	5.59	5.59	5.59	5.59
f_air	N	0.06	0.06	0.06	0.06	0.06	0.06	0.06
F	N	16.77	16.77	16.77	16.77	16.77	16.77	16.77
P_v	W	67.06	67.06	67.06	67.06	67.06	67.06	67.06
P_a	W	39.00	39.00	39.00	39.00	39.00	39.00	39.00
P	W	106.06	106.06	106.06	106.06	106.06	106.06	106.06
P_m	W	53.03	53.03	53.03	53.03	53.03	53.03	53.03
T	Nm	**0.40**	**0.44**	**0.49**	**0.53**	**0.57**	**0.62**	**0.66**
I	Amps	8.84	8.84	8.84	8.84	8.84	8.84	8.84
C	Ah	8.84	8.84	8.84	8.84	8.84	8.84	8.84

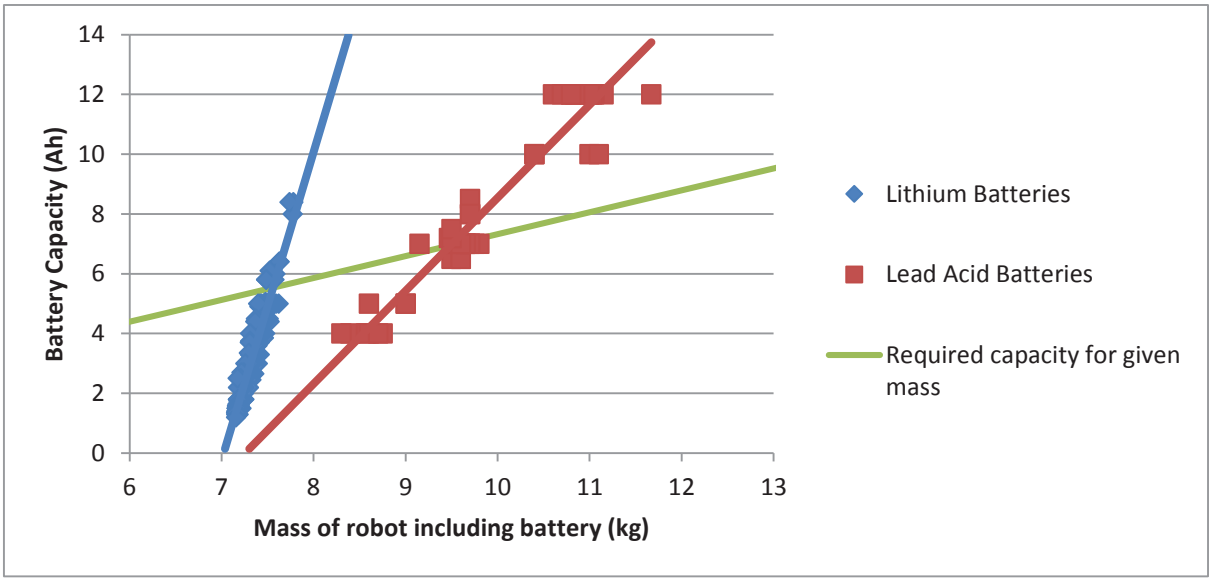
Battery capacity for different weight robots:

Symbol	Units	Value						
m	kg	6	6.5	7	7.5	8	8.5	9
phi	degrees	70	70	70	70	70	70	70
theta	degrees	5	5	5	5	5	5	5
u		0.05	0.05	0.05	0.05	0.05	0.05	0.05
Cd		1	1	1	1	1	1	1
A	mm^2	10625	10625	10625	10625	10625	10625	10625
v	m*s^-1	0.75	0.75	0.75	0.75	0.75	0.75	0.75
a	m*s^-2	0.5	0.5	0.5	0.5	0.5	0.5	0.5
D	mm	45	45	45	45	45	45	45
N	#	2	2	2	2	2	2	2
e	%	65	65	65	65	65	65	65
V	V	12	12	12	12	12	12	12
t	hr	3	3	3	3	3	3	3

Symbol	Units	Value						
g	m*s^-2	9.81	9.81	9.81	9.81	9.81	9.81	9.81
p	kg*m^-3	1.2	1.2	1.2	1.2	1.2	1.2	1.2

Symbol	Units	Value						
rpm	rpm	318.31	318.31	318.31	318.31	318.31	318.31	318.31
f_hill	N	5.13	5.56	5.98	6.41	6.84	7.27	7.69
f_roll	N	8.60	9.32	10.04	10.76	11.47	12.19	12.91
f_air	N	0.00	0.00	0.00	0.00	0.00	0.00	0.00
F	N	13.74	14.88	16.03	17.17	18.32	19.46	20.61
P_v	W	15.85	17.17	18.49	19.81	21.13	22.46	23.78
P_a	W	1.73	1.88	2.02	2.16	2.31	2.45	2.60
P	W	17.58	19.05	20.51	21.98	23.44	24.91	26.37
P_m	W	8.79	9.52	10.26	10.99	11.72	12.45	13.19
T	Nm	0.26	0.29	0.31	0.33	0.35	0.37	0.40
I	Amps	1.47	1.59	1.71	1.83	1.95	2.08	2.20
C	Ah	**4.40**	**4.76**	**5.13**	**5.49**	**5.86**	**6.23**	**6.59**

Battery/weight trade-off for 7kg robot:



Lithium					Lead Acid			
Source [64]	Capacity (mAh)	Capacity (Ah)	weight	total weight	Source [65, 66]	Capacity (Ah)	Weight	Total weight
HobbyKing	8400	8.4	780	7.78	RS	4	1.75	8.75
HobbyKing	8000	8	772	7.772	RS	4	1.4	8.4
HobbyKing	8400	8.4	737	7.737	RS	5	1.6	8.6
HobbyKing	6400	6.4	630	7.63	RS	5	2	9
HobbyKing	6400	6.4	625	7.625	RS	5	2	9
HobbyKing	5000	5	620	7.62	RS	6.5	2.5	9.5
HobbyKing	5000	5	590	7.59	RS	6.5	2.6	9.6
HobbyKing	6000	6	583	7.583	RS	7	2.6	9.6
HobbyKing	5800	5.8	573	7.573	RS	7	2.75	9.75
HobbyKing	5000	5	571	7.571	RS	7	2.7	9.7
HobbyKing	5000	5	570	7.57	RS	7	2.65	9.65
HobbyKing	5000	5	569	7.569	RS	7	2.15	9.15
HobbyKing	5000	5	561	7.561	RS	7	2.65	9.65
HobbyKing	5000	5	556	7.556	RS	7	2.5	9.5
HobbyKing	5000	5	550	7.55	RS	7.2	2.47	9.47
HobbyKing	5000	5	531	7.531	RS	8	2.7	9.7
HobbyKing	4400	4.4	522	7.522	RS	8.5	2.7	9.7
HobbyKing	6100	6.1	521	7.521	RS	10	3.4	10.4
HobbyKing	5000	5	511	7.511	RS	10	4	11
HobbyKing	4500	4.5	510	7.51	RS	12	3.8	10.8
HobbyKing	5000	5	501	7.501	RS	12	3.6	10.6
HobbyKing	5000	5	499	7.499	RS	12	3.8	10.8
HobbyKing	5000	5	497	7.497	RS	12	4.15	11.15
HobbyKing	5000	5	490	7.49	RS	12	4.05	11.05
HobbyKing	5000	5	489	7.489	RS	12	4.03	11.03

HobbyKing	4500	4.5	485	7.485	E14	4	1.57	8.57
HobbyKing	5000	5	485	7.485	E14	4	1.6	8.6
HobbyKing	5800	5.8	485	7.485	E14	4	1.7	8.7
HobbyKing	5800	5.8	483	7.483	E14	4	1.3	8.3
HobbyKing	5000	5	476	7.476	E14	6.5	2.6	9.6
HobbyKing	5050	5.05	475	7.475	E14	7	2.8	9.8
HobbyKing	5000	5	475	7.475	E14	7	2.7	9.7
HobbyKing	5000	5	474	7.474	E14	7	2.65	9.65
HobbyKing	4000	4	473	7.473	E14	7	2.6	9.6
HobbyKing	5000	5	472	7.472	E14	7.2	2.5	9.5
HobbyKing	4500	4.5	465	7.465	E14	7.5	2.5	9.5
HobbyKing	5000	5	460	7.46	E14	8	2.7	9.7
HobbyKing	3850	3.85	459	7.459	E14	10	3.4	10.4
HobbyKing	4500	4.5	459	7.459	E14	10	4.1	11.1
HobbyKing	4000	4	457	7.457	E14	12	4	11
HobbyKing	4900	4.9	455	7.455	E14	12	3.7	10.7
HobbyKing	4500	4.5	443	7.443	E14	12	4.67	11.67
HobbyKing	4500	4.5	443	7.443	E14	12	3.85	10.85
HobbyKing	4000	4	441	7.441	E14	12	4	11
HobbyKing	4000	4	431	7.431	E14	12	3.84	10.84
HobbyKing	5000	5	430	7.43	E14	12	3.8	10.8
HobbyKing	4000	4	429	7.429				
HobbyKing	4000	4	425	7.425				
HobbyKing	4000	4	425	7.425				
HobbyKing	4000	4	423	7.423				
HobbyKing	5000	5	419	7.419				
HobbyKing	4000	4	419	7.419				
HobbyKing	4500	4.5	415	7.415				
HobbyKing	3700	3.7	413	7.413				
HobbyKing	3300	3.3	413	7.413				
HobbyKing	4900	4.9	412	7.412				
HobbyKing	4000	4	411	7.411				
HobbyKing	4000	4	409	7.409				
HobbyKing	3300	3.3	407	7.407				
HobbyKing	4000	4	405	7.405				
HobbyKing	4400	4.4	405	7.405				
HobbyKing	4000	4	402	7.402				
HobbyKing	5000	5	397	7.397				
HobbyKing	3000	3	394	7.394				
HobbyKing	3700	3.7	390	7.39				
HobbyKing	3600	3.6	390	7.39				
HobbyKing	4000	4	389	7.389				
HobbyKing	3300	3.3	383	7.383				
HobbyKing	3700	3.7	380	7.38				

HobbyKing	4500	4.5	371	7.371
HobbyKing	4000	4	370	7.37
HobbyKing	4000	4	369	7.369
HobbyKing	3300	3.3	376	7.376
HobbyKing	3000	3	367	7.367
HobbyKing	4400	4.4	365	7.365
HobbyKing	3000	3	361	7.361
HobbyKing	3700	3.7	357	7.357
HobbyKing	3600	3.6	356	7.356
HobbyKing	3000	3	355	7.355
HobbyKing	2650	2.65	354	7.354
HobbyKing	2900	2.9	354	7.354
HobbyKing	3300	3.3	351	7.351
HobbyKing	3700	3.7	341	7.341
HobbyKing	3300	3.3	338	7.338
HobbyKing	2650	2.65	337	7.337
HobbyKing	2700	2.7	336	7.336
HobbyKing	3300	3.3	335	7.335
HobbyKing	2650	2.65	332	7.332
HobbyKing	3000	3	331	7.331
HobbyKing	2450	2.45	323	7.323
HobbyKing	2650	2.65	317	7.317
HobbyKing	3000	3	313	7.313
HobbyKing	3300	3.3	311	7.311
HobbyKing	2650	2.65	311	7.311
HobbyKing	4000	4	309	7.309
HobbyKing	2650	2.65	309	7.309
HobbyKing	3750	3.75	309	7.309
HobbyKing	3300	3.3	308	7.308
HobbyKing	2450	2.45	307	7.307
HobbyKing	2650	2.65	306	7.306
HobbyKing	3000	3	306	7.306
HobbyKing	3700	3.7	305	7.305
HobbyKing	2650	2.65	300	7.3
HobbyKing	2650	2.65	297	7.297
HobbyKing	3000	3	297	7.297
HobbyKing	3350	3.35	295	7.295
HobbyKing	2450	2.45	285	7.285
HobbyKing	2200	2.2	297	7.297
HobbyKing	2250	2.25	274	7.274
HobbyKing	2650	2.65	272	7.272
HobbyKing	2200	2.2	267	7.267
HobbyKing	2200	2.2	266	7.266
HobbyKing	2200	2.2	262	7.262

HobbyKing	2800	2.8	261	7.261
HobbyKing	2550	2.55	261	7.261
HobbyKing	3000	3	256	7.256
HobbyKing	2200	2.2	255	7.255
HobbyKing	2350	2.35	254	7.254
HobbyKing	2200	2.2	253	7.253
HobbyKing	2200	2.2	249	7.249
HobbyKing	2200	2.2	248	7.248
HobbyKing	1800	1.8	247	7.247
HobbyKing	2200	2.2	247	7.247
HobbyKing	2100	2.1	245	7.245
HobbyKing	2600	2.6	245	7.245
HobbyKing	1800	1.8	244	7.244
HobbyKing	2250	2.25	243	7.243
HobbyKing	2000	2	242	7.242
HobbyKing	1800	1.8	241	7.241
HobbyKing	2700	2.7	241	7.241
HobbyKing	2550	2.55	239	7.239
HobbyKing	2200	2.2	293	7.293
HobbyKing	2200	2.2	237	7.237
HobbyKing	2200	2.2	237	7.237
HobbyKing	2000	2	235	7.235
HobbyKing	2350	2.35	235	7.235
HobbyKing	2200	2.2	234	7.234
HobbyKing	2200	2.2	233	7.233
HobbyKing	1800	1.8	232	7.232
HobbyKing	2200	2.2	231	7.231
HobbyKing	1800	1.8	230	7.23
HobbyKing	2200	2.2	229	7.229
HobbyKing	2350	2.35	229	7.229
HobbyKing	2000	2	228	7.228
HobbyKing	2250	2.25	228	7.228
HobbyKing	2200	2.2	227	7.227
HobbyKing	2500	2.5	224	7.224
HobbyKing	2200	2.2	224	7.224
HobbyKing	2500	2.5	224	7.224
HobbyKing	1800	1.8	223	7.223
HobbyKing	2450	2.45	220	7.22
HobbyKing	2250	2.25	220	7.22
HobbyKing	1800	1.8	216	7.216
HobbyKing	1500	1.5	215	7.215
HobbyKing	2200	2.2	215	7.215
HobbyKing	2150	2.15	215	7.215
HobbyKing	2700	2.7	215	7.215

HobbyKing	2150	2.15	215	7.215
HobbyKing	1800	1.8	215	7.215
HobbyKing	1800	1.8	210	7.21
HobbyKing	2200	2.2	210	7.21
HobbyKing	2150	2.15	209	7.209
HobbyKing	1800	1.8	208	7.208
HobbyKing	1850	1.85	207	7.207
HobbyKing	1800	1.8	205	7.205
HobbyKing	2500	2.5	205	7.205
HobbyKing	1800	1.8	203	7.203
HobbyKing	2200	2.2	195	7.195
HobbyKing	1600	1.6	195	7.195
HobbyKing	2200	2.2	192	7.192
HobbyKing	1600	1.6	191	7.191
HobbyKing	1620	1.62	190	7.19
HobbyKing	1750	1.75	188	7.188
HobbyKing	1500	1.5	186	7.186
HobbyKing	1600	1.6	185	7.185
HobbyKing	1300	1.3	185	7.185
HobbyKing	1750	1.75	185	7.185
HobbyKing	1800	1.8	179	7.179
HobbyKing	2200	2.2	179	7.179
HobbyKing	1800	1.8	178	7.178
HobbyKing	1300	1.3	176	7.176
HobbyKing	1500	1.5	175	7.175
HobbyKing	2500	2.5	175	7.175
HobbyKing	1650	1.65	175	7.175
HobbyKing	1400	1.4	173	7.173
HobbyKing	1300	1.3	171	7.171
HobbyKing	1550	1.55	170	7.17
HobbyKing	1550	1.55	170	7.17
HobbyKing	1600	1.6	167	7.167
HobbyKing	1500	1.5	167	7.167
HobbyKing	1550	1.55	167	7.167
HobbyKing	1300	1.3	165	7.165
HobbyKing	1300	1.3	165	7.165
HobbyKing	1500	1.5	163	7.163
HobbyKing	1400	1.4	162	7.162
HobbyKing	1500	1.5	162	7.162
HobbyKing	1300	1.3	161	7.161
HobbyKing	1400	1.4	160	7.16
HobbyKing	1350	1.35	156	7.156
HobbyKing	1200	1.2	156	7.156