Vision Based Distance Measurement for In-pipe Robotic Crawler

LEE YANG YANG

UNIVERSITI SAINS MALAYSIA

2016

Vision Based Distance Measurement for In-pipe Robotic Crawler

By

LEE YANG YANG

Thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Mechatronic Engineering)

June 2016

ACKNOWLEDGEMNETS

I am grateful for having the good health and well-being that were necessary to complete this final year project. There are many people that deserve mentions and thankfulness for their help, guidance and support during the duration of this project. It is impossible for me to complete this dissertation without them.

I wish to express my sincere thanks to my final year project supervisor, Prof. Dr. Mohd Rizal Arshad, Dean of School of Electrical and Electronic Engineering USM for providing me with all the necessary and facilities means to complete my project. Prof. Rizal has been providing me with a clear concept on my project which leads to the successfulness of my project. His patience in guiding me to come out with a good thesis is very much appreciated.

Besides, my sincere thanks to Assc. Prof. Dr. Khoo Bee Ee too for her expertise in providing me assistance throughout the project. She had assisted me in solving problems related to machine vision and leads to successful execution of the image processing algorithm. She also provided knowledge of camera and optics that helped me in understanding many concepts regarding machine vision.

Many thanks to my friends, especially Mr. Tan Chee Sheng for helping me in many aspects along the way of this project. Because of his valuable suggestions, results of the project can be obtained successfully.

I am also thankful to my family and relatives for being supportive to my final year study and for understanding me of the importance of this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURE

LIST OF ABBREVIATIONS

ARM	Advanced RISC Machine
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
CSI	Camera Serial Interface
DSP	Digital Signal Processing
FoV	Field of View
GPIO	General Purpose Input Output
GPU	Graphic Processing Unit
HDMI	High Definition Multimedia Interface
HFoV	Horizontal Field of View
IM	Instrumentation and Measurement
I/O	Input Output
IMU	Inertial Measurement Unit
ISO	International Standards Organization (standardized industry scale for measuring sensitivity to light)
IoT	Internet of Things
IP	Internet Protocol
LAN	Local Area Network
LED	Light Emitting Diode

OS	Operating System
PIV	Particle Image Velocimetry
PVC	Polyvinyl Chloride
PWM	Pulse Width Modulation
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
ROI	Region of Interest (of an image)
RPi	Raspberry Pi
SMD	Surface Mounted Device
SoC	System on Chip
USB	Universal Serial Bus
UI	User Interface
VBM	Vision Based Measurement
VFoV	Vertical Field of View

LIST OF SYMBOLS

δ	Delta (difference)
α	Alpha (angle of principal ray relative to ground plane)
β	Beta (angle of principal ray relative to feature point)
a	Height of dummy car relative to surface of culvert
d	Distance of image frame relative to camera
H	Height of camera relative to surface
p	Pixel distance at the verge of FoV relative to principal ray
p'	Pixel distance of feature point relative to principal ray
r	Radius of culvert
y	Distance of feature point relative to camera

LIST OF EQUATIONS

Equation 3.1

$$\textbf{pixel distance} = \frac{\mathbf{2} * \boldsymbol{H} * \tan\left(\frac{\boldsymbol{FOV}}{\mathbf{2}}\right)}{\boldsymbol{image\ size}}$$

Equation 3.2

$$\boldsymbol{\beta} = \tan^{-1}\left[\frac{\boldsymbol{p'}}{\boldsymbol{p}}\tan\left(\frac{\boldsymbol{VFOV}}{\mathbf{2}}\right)\right]$$

Equation 3.3

$$\boldsymbol{y} = \frac{\boldsymbol{H}}{\tan(\boldsymbol{\alpha} + \boldsymbol{\beta})}$$

Equation 3.4

$$\boldsymbol{camera\ height} = \boldsymbol{H} = r + \frac{\sqrt{4r^2 - width^2}}{\mathbf{2}} - \boldsymbol{offset}$$

Pengukuran Jarak Berdasarkan Penglihatan Untuk Robot Jenis Rangkak Dalam Paip

ABSTRAK

Pengukuran jarak berdasarkan penglihatan untuk robot jenis rangkak dalam paip adalah satu sistem pengukuran jarak yang menggunakan alat optik seperti kamera untuk mengukur jarak perjalanan robot jenis rangkak dalam paip tanpa sentuhan fizikal. Demi mengetahui lokasi paip yang rosak serta mengesan posisi robot, pengukuran jarak yang tepat adalah diperlukan. Robot jenis rangkak dalam paip biasanya dilengkapi pengekod roda yang boleh terus mengesan jarak perjalanan. Walaupun begitu, mengukur jarak dengan satu sistem pengukuran adalah tidak cukup baik, dan seringkalinya gelinciran roda meningkatkan ralat pengukuran. Oleh itu, pengukuran jarak berdasarkan penglihatan telah direka dengan harapan untuk meningkatkan kebolehpercayaan robot jenis rangkak dalam paip dengan sistem pengukuran jarak yang lebih jitu. Dua kaedah telah diperkenalkan untuk mengatasi kondisi kamera yang berbeza dan membandingkan keseluruhan prestasinya. Beberapa kondisi ujian telah dijalankan dengan setiap ujian mempunyai menetapan kamera yang berbeza. Keputusan eksperimen telah dibandingkan dengan alat ukur seperti meter gulung untuk mengesahkan kebolehlaksanaan pengukuran jarak berdasarkan penglihatan bagi robot jenis crawl dalam paip. Ia didapati bahawa jika algoritma dilaksanakan dengan betul, kejituan sebanyak 95% atau lebih boleh dicapai. Selain itu, resolusi pengukuran terbaik dalan projek ini adalah apabila kamera bersudut tepat dengan permukaan paip, iaitu 0.3mm/pixel. Dengan menggunakan teknik pemurataan, ketepatan pengukuran yang lebih tinggi boleh dicapai. Secara keseluruhan, pengukuran jarak berdasarkan penglihatan untuk robot jenis rangkak dalam paip adalah satu kejayaan kerana ia boleh menganggarkan jarak yang dilalui robot dengan ralat kumulatif kurang daripada 5%.

Vision Based Distance Measurement for In-pipe Robotic Crawler

ABSTRACT

Vision based distance measurement for in-pipe robotic crawler is a distance measurement system that uses optical devices like camera to measure distance travelled by in-pipe robotic crawler without physical contact. In order to locate the damaged pipe and to track the position of the robot, accurate distance measurement is required. In-pipe robotic crawler is usually equipped with wheel encoder which can directly measure distance travelled. However, distance measurement based on single measuring system is not good enough, and very often, wheel slippage accumulates error. Therefore, vision distance measurement is being designed, hoping to improve the reliability of the in-pipe robotic crawler with more accurate distance measurement system. Two methods were introduced so as to cope with different conditions of camera and to compare their overall performances. Several test conditions were carried out, each with different camera settings. The results of experiment were compared with measuring tools like metre rule to validate the feasibility of the vision based distance measurement for in-pipe robotic crawler. It was found that if the algorithms were implemented accordingly, accuracy of 95% or more can be achieved. Also, the best resolution of measurement in this project is when the camera is perpendicular to the surface of culvert, which is 0.3mm/pixel. By using the averaging technique, higher accuracy measurement can be achieved. Overall, the vision based distance measurement for in-pipe robotic crawler is a success because it can estimate the distance travelled by robot in less than 5% of cumulative error.

CHAPTER 1

INTRODUCTION

1.1 Background

Vision based measurement (VBM) is an approach of Instrumentation and Measurement (IM) that uses visual devices like camera or any electromagnetic sensor to detect and measure physical properties of subject of interest[1]. It can measure basic physical properties of an object such as distance and speed with virtually no physical contact. With more advanced visual devices such as infrared camera and high-speed camera, other physical properties can also be measured such as temperature and tensile strength. Since it is part of machine vision, it also can be integrated into image processing for other purposes such as speed and stopping distance estimation of vehicles by using surveillance camera[2]. Due to its contactless measurement capability, it is ideal for measurement where physical contact is not feasible, like measuring surface flow field of river with Particle Image Velocimetry (PIV) technique[3]. It is also ideal for uses in remote environment in which only visual sensing is reliable. It is proven to be successfully used as position and orientation estimation on planet Mars by NASA Mars Exploration Rover (MER)[4].

Depending on the resolution of the image frames and the Field of View (FoV) of the visual devices, the sensitivity of the visual based measurement varies from microscopic scale to macroscopic scale, depending on visual device used. Nowadays, due to the high availability of camera and computing equipment, in addition to the drastic reduction in cost of computing, VBM is getting more popular than before, replacing various sensors used in robotics and automated industries. Vision based distance

measurement is one of the example used in mobile robot that usually equipped with camera for visual feedback and recording.

1.2 Motivation

Speaking of remote environment, there are places on planet Earth in which it is usually inaccessible by human. Underground piping is one of the good example. Usually a remote controlled robot attached with camera or fibre optic gauge is used for underground pipe inspection such as culvert, a kind of pipe that made out of concrete. Regular inspection, maintenance and repair need to be done due to aging, corrosion and cracking under prolonged use. By using remote in-pipe robot, culvert can be inspected with ease. Nonetheless, localizing the culvert is a problem since there is no direct access for location marking. Hence wheel encoder is often installed to measure and record distance travelled by in-pipe robot. As a good practice, measurement based on single measuring system is not good enough, and very often, wheel slippage accumulates error due to the aging of the surface of culvert.

In this project, vision based distance measurement for in-pipe robotic crawler is to be designed in the hope of improving the distance measurement system and localization of the in-pipe robotic crawler.

1.3 Problem Statements

The vision based distance measurement system must be able to use optical devices like camera to estimate the distance travelled by in-pipe robotic crawler as accurate as possible. Since the interior surface of the pipe is not flat, it must be able to detect features with uneven surface brightness in order to detect distance travelled.

The camera of the in-pipe robotic crawler is expected to be manoeuvrable (can be rotated so that user can inspect the condition of inner pipe). Also, common techniques seemed to have the camera perpendicularly faced to the interior surface of pipe. Hence the vision distance measurement system may need to deal with different camera angle in order to recover the real distance travelled by the robot.

The system will need to have good lighting system in order to perform distance measurement as well as lighting for visual inspection. A suitable lighting system have to be designed.

1.4 Objectives

The objectives of this project are:

i.) To design vision based distance measurement system with monocular vision camera.

ii.) To measure the performance of vision based distance measurement.

iii.) To compare the performance of the system of different methods.

1.5 Scope

This project is focusing mainly on vision based distance measurement. Only monocular vision camera is being used due to limited data bandwidth and computational power of the computer module. Only the culvert of fixed diameter is used since the focus of the camera is fixed. Only culvert pipe is being tested. The results are obtained with post processing instead of real time processing by recording videos in order to utilize the video stabilization function of Raspberry Pi camera. Only single axis of distance is computed, no rolling and turning of motions involved.

1.6 Outlines of Report

This report consists of 5 Chapters. The first chapter is an introduction chapter which discussed the background of vision based distance measurement system for in-pipe robotic crawler, motivation to design a vision distance measurement, problem statement, objectives of this project, scope and outlines of the report.

The second chapter is literature review. For this chapter, outcomes from researching on the scope of the project is covered to give an overview of what is available and the latest technology available in vision based distance measurement. Besides, theories relevant to vision distance measurement are also included.

Third chapter describes the methodology for the development of vision based distance measurement system. All steps involved are recorded and described in details in this chapter. Also, all related theories and algorithms will be discussed. At the end of this chapter, the method used to measure the performance of vision distance measurement will be discussed.

In the fourth chapter, all the experiment results will be presented. The setup of the experiment will be described in details with the help of attached photos. The data will be tabulated and the performance of the vision distance measurement will be analysed and discussed.

The final chapter contains the conclusion regarding the project so as to conclude the achievement of the project's objectives. Suggestions for future research are also included in this chapter so that reader can have better idea of improvement for this project.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

There are some methods found that uses monocular vision camera to estimate the relative distance of the camera to real-world environment. The methods of vision distance measurement differs depending on the application of mobile robot. Some methods are not used to detect distance but may be useful on improving the consistency of distance measurement algorithm.

2.2 Optical Flow distance estimation

Optical flow field estimation remains an open problem, dozens of methods have been proposed and each with its own strengths and weaknesses. However, in a 1995 survey, Barron, Fleet, et al.[5] reported the Lucas Kanade method as being among the most accurate and most reliable of the methods (then available), when tested using a variety of simulated and real data series. An improved and more efficient form of the Lucas Kanade algorithm is available in the open source computer vision library, OpenCV. This library has been employed by feature tracker to obtain the point correspondences (equivalently, the optical flow field vectors) required in subsequent stages of the algorithm. In selecting features to track, a low corner threshold is used and a large minimum distance between features has been specified. This encourages the detection of features even in low-contrast parts of the image and provides relatively uniform coverage across the field of view, at the cost of generating a higher number of tracking errors. As shown in Figure 2.1, the monocular camera being attached to a test robot. Given that the robust methods used to analyse those feature tracks can deal with substantial numbers of outliers, we have found such wider coverage preferable to minimizing tracking errors [6].

Over a short interval, the robot's motion on the ground plane can be decomposed into a change in heading and a displacement. They are recovered separately using different regions of the image.

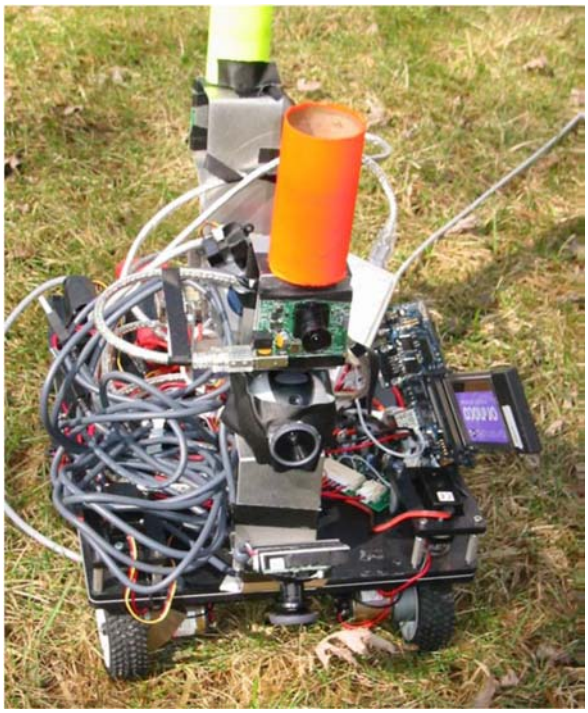

Figure 2.1 Example of robot using monocular visual odometry [6]

Camera rotation causes all observed feature points, both nearby and distant, to move through the same angle. This means a robust estimate for heading change can be derived by observing features far from the camera (since those are relatively insensitive to optical flow induced by translation). Once the effects of any observed heading change are subtracted from the optical flow field, the flow vectors associated with features near the camera can serve as a good basis for estimating translational motion. Vectors below the horizon are good candidates for estimating translation. As in Figure 2.2, a typical image with filtered flow vectors and its decomposition into the three zones. Vectors in a "dead zone" near the horizon are discarded since the observed location of the horizon is affected by brief, transient changes in robot pitch – and this could greatly magnify small errors in translation estimates based on distant points (which lie near the horizon).

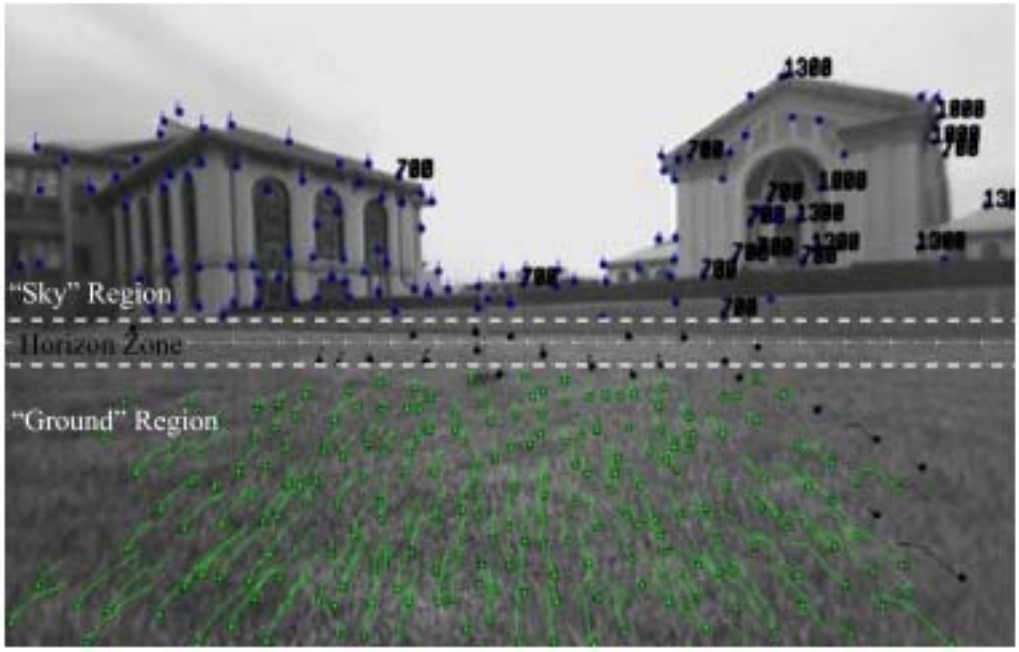

Figure 2.2 Classifying Optical Flow Field Vectors. Vectors above the horizon are used to estimate robot rotation, vectors below the horizon are used to estimate robot translation [6]

2.3 Image Displacement Visual Odometry

There are two main approaches for finding the displacement vector between two images namely feature based and correlation based. Correlation approach has been used since it can work in the environments with low texture. Moreover, the displacement vector between two images is indicated with 3-dimensional vector ($\delta x, \delta y, \delta\theta$) in which two first components shows the amount of translation between two images and the third component specifies the amount of rotation between images. Notice that $\delta\theta$ directly indicates the rotation of the robot (assuming the camera is pointing down perpendicular to floor). Hence, by finding its values the third component of the position vector which is rotation of robot can be found.

The origin of the world coordinate system is located at the beginning and the centre of the cylinder. The z- axis points to above and the y-axis points to the left. Also, the direction of the x-axis is defined using right hand rule. The position of the pipe inspection robot can be represented by a 3- dimensional vector (X_{world} , Y_{world} , θ) in

which Xworld and Yworld are the position of the robot in the world coordinate system and θ indicates the rotation of the robot around z-axis. According to Figure 2.3, the visual odometry algorithm starts with finding the displacement vector between two last images which is represented by 3-dimensional vector (δx, δy, δθ). After obtaining this vector, the displacement of the camera is computed using the calibration parameters[7].

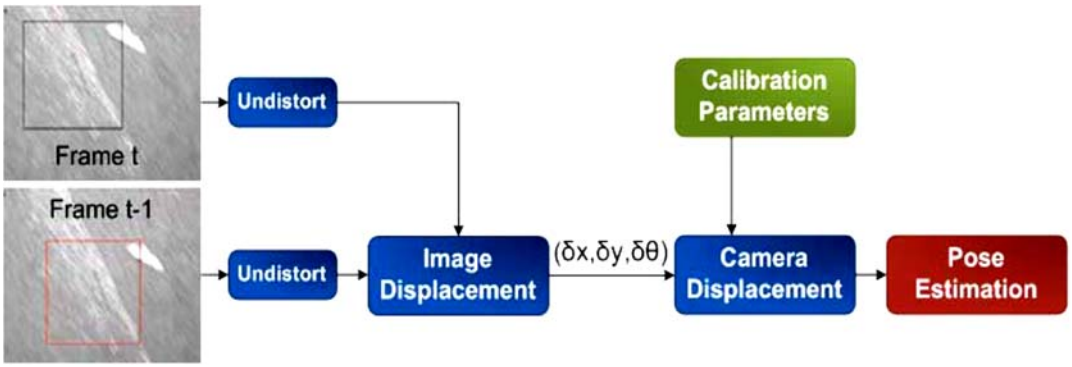

Figure 2.3 Proposed monocular visual odometry algorithm [7]

2.4 Feature tracking with small FoV

Unlike flat surface, normal piping has curved surface. As a result, the pixel displacement is different across the field of view of the camera video, which means the depth is different at every area in the camera frame.

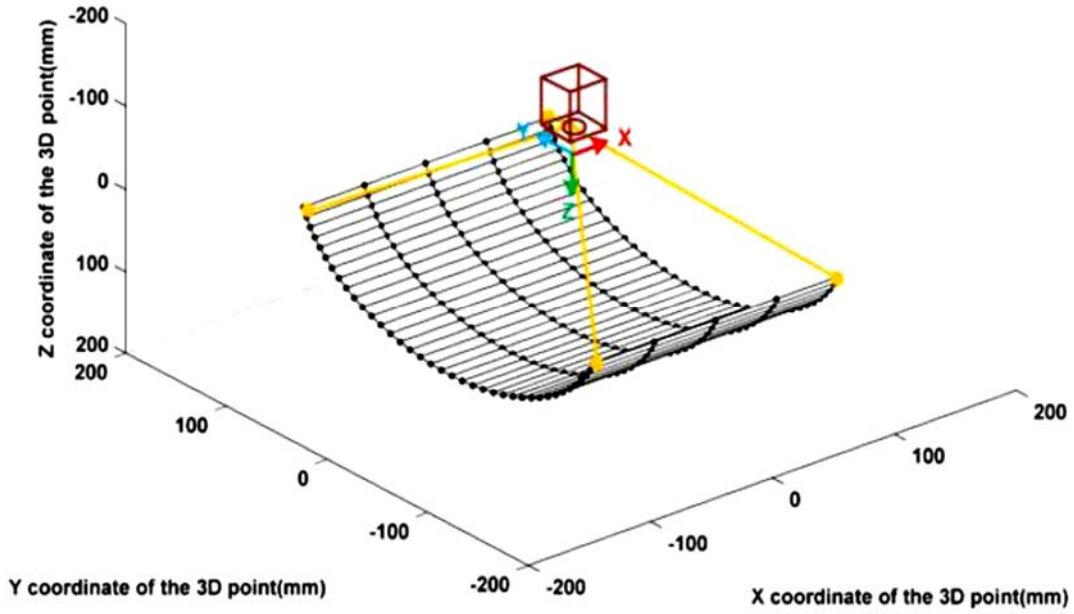

Figure 2.4 Camera attachment [8]

Considering the condition in Figure 2.4, assuming the robot moved 15 mm and 5 mm along its x-axis and y-axis respectively. This means that $\delta X = 15$, $\delta Y = 5$ and $\delta Z = 0$. To obtain the pixel displacement vector of each 3D point, their total derivatives were calculated. In Figure 2.5, 3D points as well as their pixel displacement are indicated. The field of view of the camera is 150° in this example. As in Figure 2.6, the pixel displacement in centre of the image is smaller than the displacement in the sides of the image. It should be noted that there is a significant difference of displacement vector between the centre of camera and the edge of camera. Therefore, the direct method of monocular visual odometry algorithm can be used to calculate the motion of the robot. However, if the field of view of the camera is reduced to 30 and its height is increased to 230 mm, the total derivative of 3D points changed to the chart in Figure 2.7.

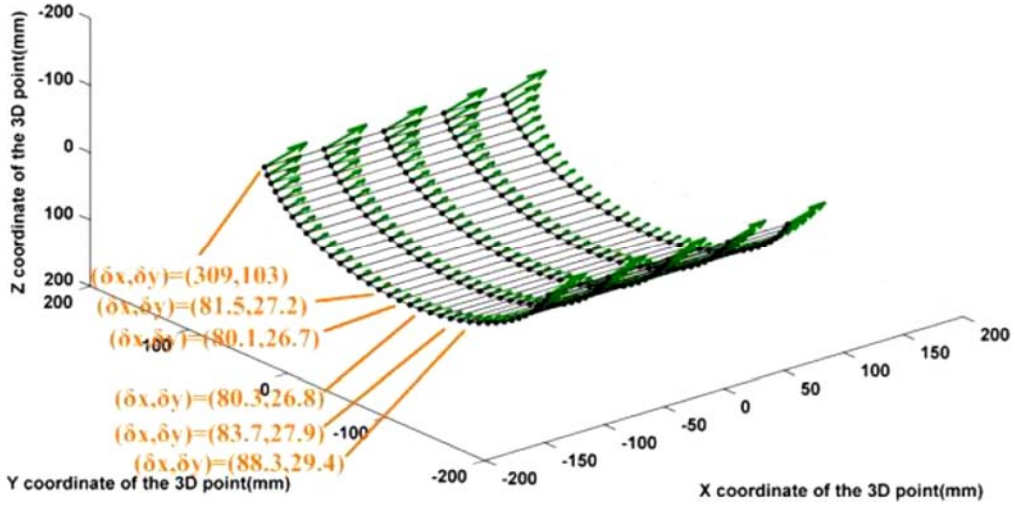

Figure 2.5 Pixel Displacement Vector [8]

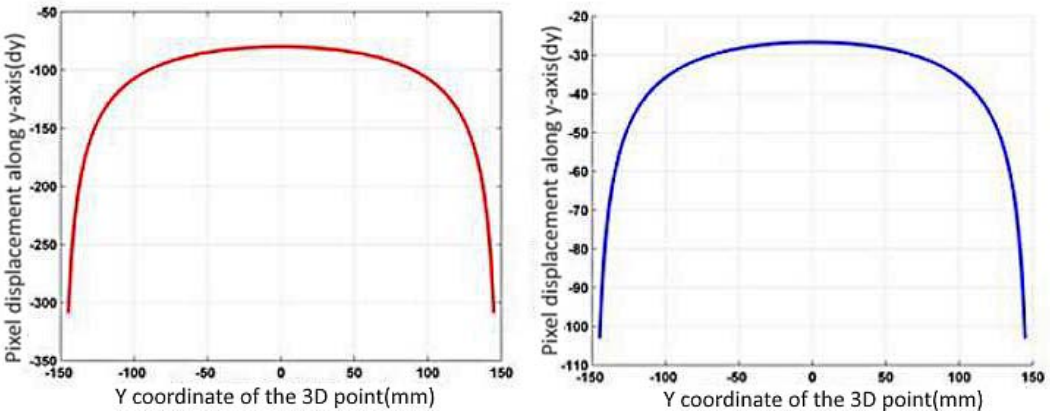

Figure 2.6 Total derivative of 3D points for field of view of the camera [8]

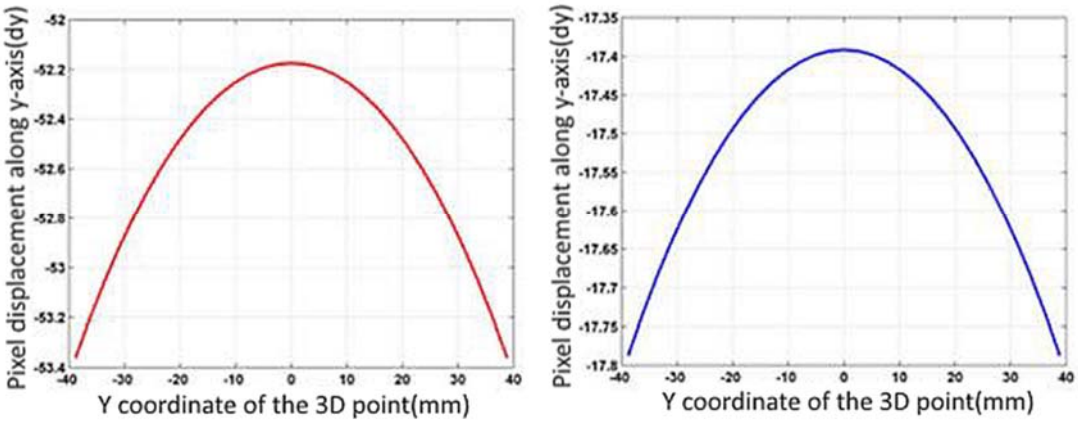

Figure 2.7 Total derivative of 3D points for 30 degree field of view of the camera [8]

As the results, by reducing the field of view the pixel difference between displacement vectors can be ignored since the difference in pixel displacement of each point is insignificant along x-axis and y-axis. Hence direct method of monocular visual odometry algorithms can be used to calculate the robot motion[8].

2.5 Depth measurement with monocular vision

There are several drawbacks in using the stereo vision system such as it requires more power consumption, more space for mounting the camera and more memory stack space for computational process especially in embedded system. Therefore monocular vision camera is more preferable in remote robotics, especially those with embedded system. However, depth estimation using monocular vision is a challenging problem. For example, although features such as the texture or colour of an object can give the information about its depth, the method fails to accurately determine the object's absolute depth. To overcome these shortcomings, some researchers integrated the monocular vision system with a laser system so as to obtain accurate depth estimation by using triangulation. Known fix fixtures can also be used as special fixed features from known object as a reference for visual depth estimation[9].

Figure 2.8 Object being image processed for depth detection[9]

Known fix feature technique uses special fixed features from known object as a global reference for visual depth estimation. To calibrate the system, Ibrar et al. [10]used circles of a known size as a reference pattern on the horizontal plane as shown in Figure 2.8. The camera field of view (FoV) should be perpendicular to the horizontal plane in the experiment. Then the geometrical relations of the mounted camera on a robot can be used to calculate the world coordinate frame and the image coordinate frame.

2.6 Gain-mask correction

High intensity LEDs provide the only light source in each pipe. To account for their non-uniform lighting distribution over the surface of curvy pipe, each image is pre-processed by dividing it with a gain mask [11].

Figure 2.9 Brightness captured (left) and gain corrected version (right)[11]

The gain mask is the mean of 50 images previously taken by the camera observing a white piece of card affixed to the interior surface of the pipe with the LEDs operating at their specified power. In Figure 2.9, the gain mask for pipe 1 and the gain corrected version images are shown.

2.7 Odometry with Commodity Sensor

Conventional forward-facing visual odometry as used on the Mars Exploration Rovers (MERs) was considered but deemed impractical because lighting the surrounding area would require more power than the rover can provide and induce complex shadows. Instead, downward-facing visual odometry is used as shown in Figure 2.10. This provides tractable lighting requirements, however these forward-facing techniques cannot be directly applied because the situation is ill-posed to compute the 3-D incremental pose differences and the terrain beneath the rover is likely to be too homogeneous for point-based feature tracking to work effectively. Rather, we chose to rely on optical flow to provide an estimate of vehicle speed, which could then be integrated to estimate incremental distance travelled as part of a broader odometry framework.

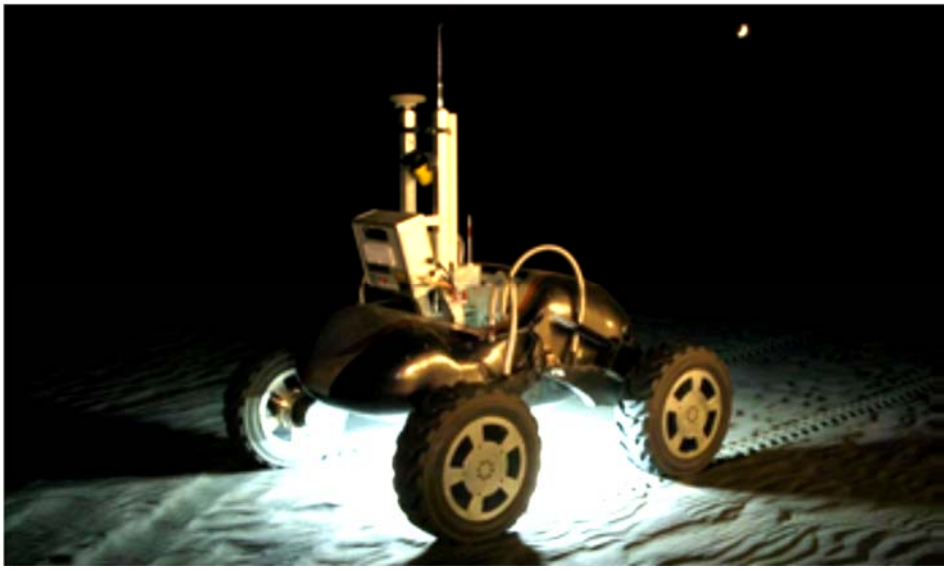

Figure 2.10 The Scarab lunar rover. The glow beneath is from LED lighting for the optical flow sensor[12]

Various methods have long existed for computing so-called dense optical flow over regions between images. In the odometry realm, downward-facing cameras have been successfully used for positioning in pre-explored environments by correlating the visible area against an existing database, however such methods are inapplicable to planetary rovers observing most patches for the first time. Although an optical flow implementation using a typical camera would have been straightforward, an intriguing alternative presented itself in the form of a custom-built optical flow sensor used for stabilization on its quad-rotor helicopters. The device shown in Figure 2.11 contains four commodity optical mouse sensors each attached to a lens of a different focal length and reports 2-D velocity (in m/s) along the planar surface at which it is pointed and an estimate of distance to this plane. The precise details of its operations are proprietary to the manufacturer, but given that a scalar "tracking quality" value is known to be returned by mouse sensors, the researcher conjectured that height is derived by interpolating across the quality values returned by each of the four sensors, with speed similarly interpolated across the four focal-distance-compensated speed values. The sensor nominally reports velocity with a resolution of 0.3mm/s within a range of ±10m/s and height up to 2.5m with a resolution of 1cm.

Figure 2.11 The optical sensor of Scarab, in a custom ruggedized enclosure[12]

The use of optical mouse sensors for measuring ground velocity has a number of benefits leveraging over a decade of commercial refinement. Key among these are remarkably robust operation on a wide variety of surfaces, significant lighting insensitivity, and extremely low cost (several US dollars) due to the volume in which they are produced. These typically contain a 15 to 100 pixels-square camera sensor and are believed to implement a fast hardware version reporting sub-pixel flow rates at on the order of 1kHz. Designs based on laser interferometry are becoming prevalent, however they are not as adaptable as they do not use simple lenses. Due to their low cost and simplicity, there exists wide interest in the hobbyist robotics community in these sensors. Several examples of their use in limited indoor scenarios exist in the robotics literature but they have yet to see use in a field robot application[12].

2.8 Texture addition on optical flow performance in images with poor texture

No matter how sophisticated the algorithm is, performance could be undesirable if original frames have poor texture. Exploring the outcomes of modifying original images is encouraged, rather than modifying the computing algorithms. The regions in the background are the main reason for propagation of foreground flow to neighbouring pixels, object shape distortion, and even object merging. Furthermore, adding a static texture to these regions can be performed with sufficient accuracy. However, to generate a moving texture for the foreground region, pixels' correspondence need to be known, which is not possible without calculating the optical flow. If optical flow need to be calculated, even small inaccuracies in the optical flow vector field leads to addition of the texture to wrong pixels and hence induction of erroneous flow. Moreover, any interpolation using feature matching is prone to mismatching errors and it requires

knowledge about the type of object motion (rigid or flexible as they need different types of interpolations), thus it cannot be used to generate an accurate moving texture.

Therefore only poorly-textured background regions are treated and leave the modification of foreground regions to future investigations. It is important to mention that the camera is assumed to be stationary in this paper. Therefore, only the background pixels are add with texture. To localize and add texture to poorly-textured regions in the background, a texture segmentation is performed using Laws' masks and generate a texture map. Next, using a binary frame difference, poorly-textured regions is constrained to those with negligible motion. Finally, the optical flow for the modified images with added texture is calculated as shown in Figure 2.12. Optical flow calculations suffer from poor texture, and image differencing cannot provide motion information, being significantly sensitive to illumination changes and the binarizing threshold; however, computing optical flow while using image differencing and texture addition as just described provides improved accuracy and robustness in foreground detection[13].

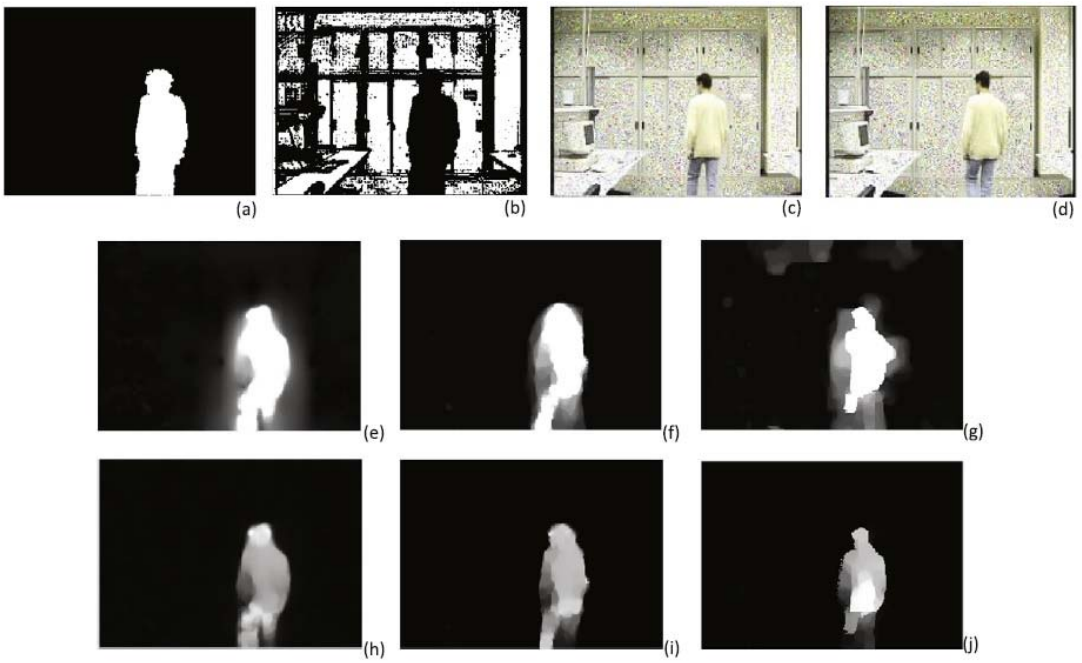

Figure 2.12 First row: Binary frame difference (a); background regions with poor texture rendered in solid white (b); modified first frame (c); modified second frame (d); second row: Magnitude of optical flow for original frames; third row: Magnitude of optical flow for modified frames[13]

2.8 Comparison of Image Processing Methods

Table 2.1 shows the comparison of optical flow and image displacement methods in various aspect. All of the methods rely on camera for image processing to measure distance. However optical flow method allow the uses of commodity sensors such as optical mouse sensor which has lower image resolution but offers much higher frame rate as compared to camera. Newer mouse sensor is also integrated with hardware accelerated DSP circuit which eliminates the uses of microprocessor for image processing. Optical flow method also allows for alignment of different camera angles. Image displacement methods however has its own advantage by being able to adapt to image with low texture which cannot be done by optical flow method.

Table 2.1 Comparison of Optical Flow and Image Displacement methods.

Method	Optical flow feature tracking	Image Displacement
Sensor	Camera, mouse sensor	Camera
Texture requirement	High	Low
Frame rate	- High on mouse sensor. - Varies on camera	- Varies on camera
Camera angle	Able to tolerate different angles	Must be perpendicular

2.9 Summary

So far, there are few methods of visual distance measurement being implemented by other researchers. However, only culvert pipe which has high amount of surface texture by itself is to be used. Also, since camera is usually equipped in the robotic crawler itself, it can be utilized to be a VBM sensor. Since pipe diameter is a standard parameter, depth measurement is unnecessary. Gain mask correction is not required since no pipe mapping is implemented. Furthermore, feature tracking requires angle lightning for sharper features and images, gain correction is useless and will only burden CPU loads. The most useful technique in pipe distance measurement is small FoV measurement. Much simpler algorithm can be implemented thanks to this techniques.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, methods used to visually estimate the distance travelled by in-pipe robotic crawler were explained in details. Section 3.2 described the flow of the methods used in this project. The project flow was visualized with flowchart. Whilst section 3.3 discussed about the important assumptions made in the project in order for the distance measurement algorithm to work. Section 3.4 was about method selections, in which 2 methods were selected for this project. Section 3.5 and section 3.6 discussed about hardware and software selected for this project respectively. Next, section 3.7 emphasised about important parameters need to be considered in this project in order for the vision distance measurement to work correctly. Lastly, section 3.8 was about summarizing this the methodology that has been discussed.

3.2 Overview

Flow chart shown in Figure 3.1 is the research methodology. By considering the assumption made, suitable method of image processing was selected. Then by considering the method, corresponding hardware and software were chosen to make the method works.

Hardware design and software development were performed in parallel so as to familiarize with the software and hardware specification as well as to speed up the progress. Then several calibrations such as camera FoV and focus were required before the modified program can be tested. Then few sample videos were recorded and tested on computer so that the initial performance of hardware setup can be evaluated. If there was

huge disturbance visible in the optical flow output or any undesired input was found and cannot be minimized, hardware part need to be redesigned to reduce input disturbance as much as possible. If necessary, methods of distance measurement had to be changed.

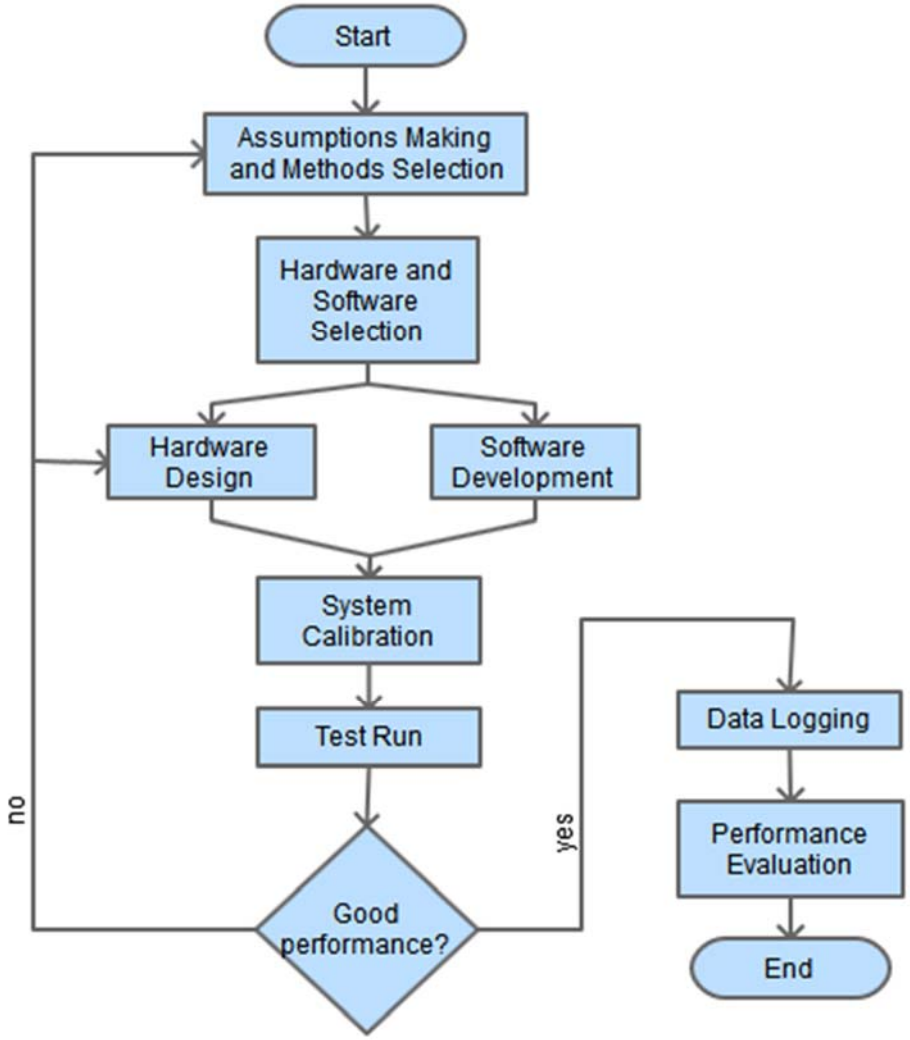

Figure 3.1 Research methodology flow chart

After the initial performance testing showed green light, data can be collected and the performance of the method can be evaluated. Two methods were proposed in this project, but since the hardware was identical and the differences in software coding for both methods were small, data logging for both methods can be done in the same time by merging both coding into one program execution cycle, reducing experiment time.

3.3 Assumptions Made

Those methods worked under few assumptions:

- The surface is considered flat by utilizing small ROI of an image.
- The intrinsic references made are constant (in real case they did vary due to various reasons such as suspensions of dummy base)
- If Python program works on PC, it works on Raspberry Pi too. (In real case there is difference in computational performance, affecting the performance of the program as well as the results)
- The pipe is straight. (Any curve or bending of the pipe will not be considered, only one-dimensional axis will be recorded)

3.4 Method Selection

Two methods has been selected for comparison in this project. First method is about pixel distance[11] and second method is about real distance retrieval with monocular vision camera[6]. Each of the method will be explained in details together with the theories behind them.

3.4.1 Method 1 - Optical Flow – Pixel Distance Measurement

By knowing the working distance and the FoV of the camera as shown in Figure 3.1, pixel distance can be found as proven by Equation 3.1:

$$\tan\left(\frac{FOV}{2}\right) = \frac{Real\ distance/2}{H}$$

$$pixel\ distance = \frac{Real\ distance}{image\ size}$$

$$\boldsymbol{pixel\ distance = \frac{2 * H * \tan\left(\frac{FOV}{2}\right)}{image\ size}} \quad (3.1)$$

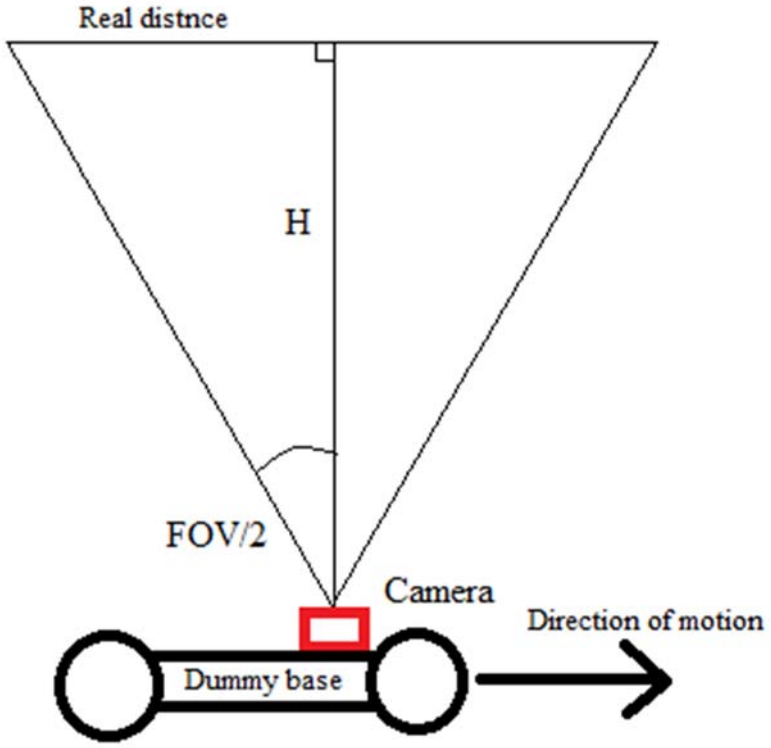

Figure 3.2 Trigonometry of pixel distance measurement

Where, $\boldsymbol{pixel\ distance}$ is the real distance represented by a pixel of an image. By integrating the total pixel of an image of single axis, true distance can be recovered. In order to use this method, the camera must be perpendicular to the surface of the culvert. Executing this method with other angles of camera placement is expected to have wrong output. Also, the surface must be flat for accurate estimation, but since only small ROI is used, the effect of differences of pixel displacement vector can be neglected.

In Figure 3.3, the flow of the program for Method 1 is shown. The ROI was first extracted and being converted to grayscale. Then Lucas-Kanade Optical Flow will be performed on that cropped image. The algorithm would return a list of data that shows the x-y coordinates of good features and their provious trajectories in pixel unit for up to 10 frames length (length of history can be manually set). The list must be converted into Numpy array format so as to perform mathematical calculation by using Numpy library.

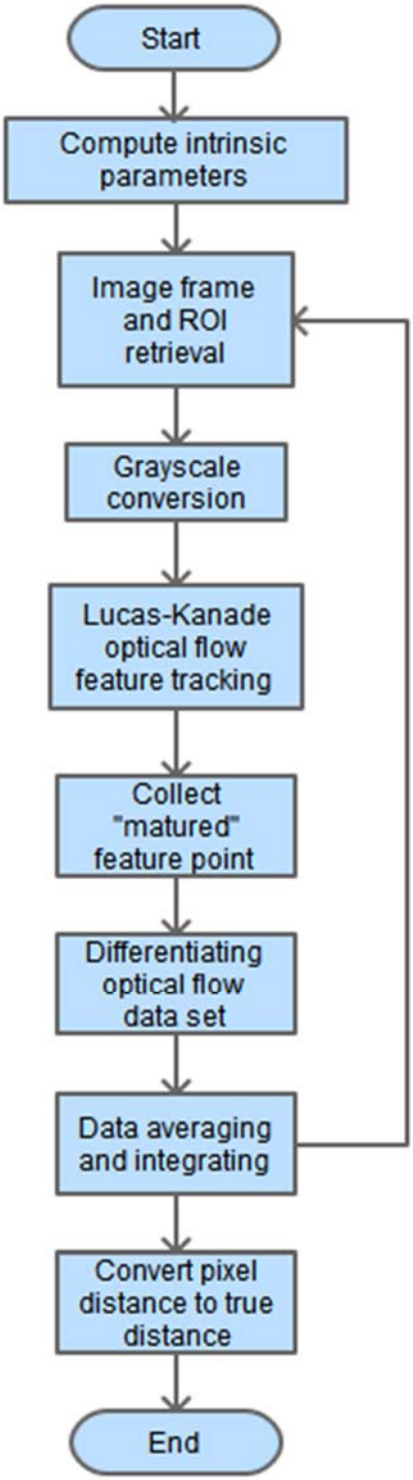

Figure 3.3 Flow chart of Method 1

3.4.2 Method 2 –Optical Flow–Retrieving real distance of every feature point.

By using known references such as working distance and angle of camera, the real distance of feature points can be retrieved by using trigonometric function. As illustrated in Figure 3.4, the formula for real distance retrieval can be derived as shown below.

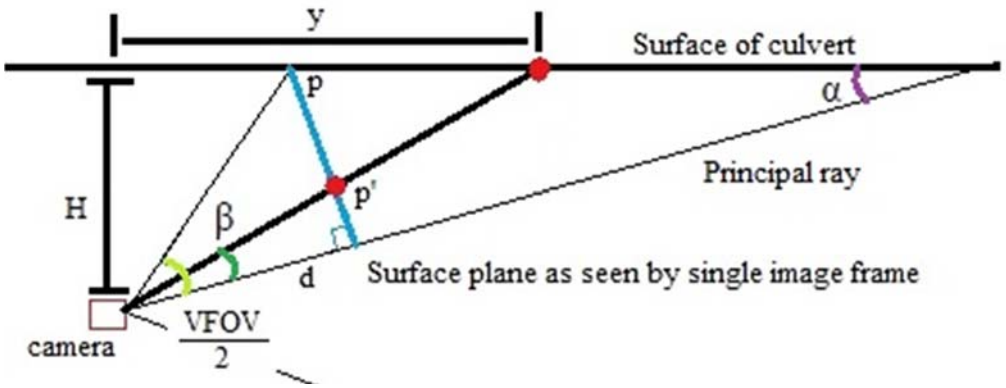

Figure 3.4 Visualization of condition of angled camera relative to flat surface.

From side view of camera, it can be deduced that:

$$\tan\beta = \frac{p'}{d}\,, \tan\frac{VFOV}{2} = \frac{p}{d} \quad \rightarrow \; d = \frac{p}{\tan\left(\frac{VFOV}{2}\right)}$$

By solving d, β can be found as Equation 3.2

$$\tan\beta = \frac{p'}{p}\tan\left(\frac{VFOV}{2}\right)$$

$$\boldsymbol{\beta = \tan^{-1}\left[\frac{p'}{p}\tan\left(\frac{VFOV}{2}\right)\right]} \tag{3.2}$$

With known angle of camera α as in Figure 3.4, the real distance y can be calculated as shown by Equation 3.3:

$$\tan(\alpha+\beta) = \frac{H}{y}$$

$$\boldsymbol{y = \frac{H}{\tan(\alpha+\beta)}} \tag{3.3}$$

This method can deal with rotating camera, providing more robust operation. It works only if the camera does not rotate for too much. For example, if the real distance reaches infinity, or the camera is facing parallel to the culvert, floating points overflow will occur and the program will fail. The angle of tilting is also limited by the focus of the camera. The image will become blur for features that are further away from the camera, causing inaccurate feature tracking since one of the assumption made in optical flow is that the pixel intensities of an object do not change between consecutive frames [14]. However, there should have no problem if the ROI below the principal ray remained in focus since only that small ROI below principal ray is required for feature tacking. The difference of the program flow that compared to Method 1 was that the values did not pass any unit conversion process. The program flow for Method 2 is shown in Figure 3.5.

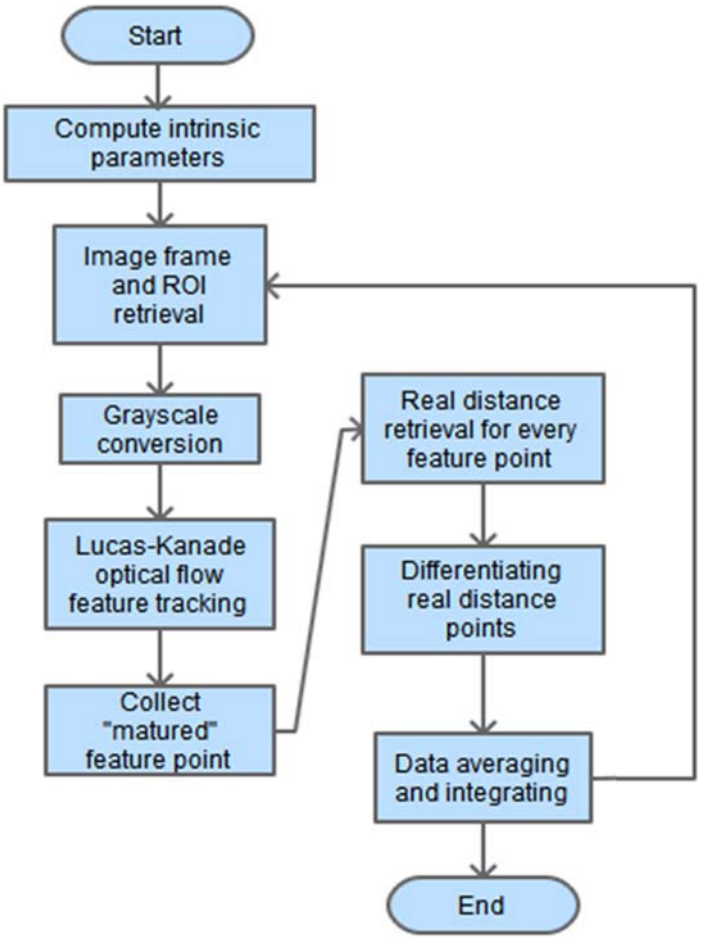

Figure 3.5 Flow chart of Method 2

3.5 Hardware

The hardware selected for this project will be described in this section. The respective reasons of hardware selection will be explained in details.

3.5.1 Raspberry Pi 2 Model B

Raspberry Pi (RPi) was selected since image processing and computer interface were required. RPi (precede of RPi 2), which was released in 2012 received widespread of uses among hobbyist, educators and even engineers with positive receptions. With the new beast of its own, RPi 2 like one in Figure 3.6 is packed with more computational power much equivalent to old-day Intel Pentium IV processor[15]. Since it consumes only a little power due to its ARM cored CPU which uses RISC architectures similar to most Android smartphone, a 5V @ 1A power supply such as portable power bank will be adequate for normal operation. It also offered many other features such as 100Mbps Ethernet port, 4 USB ports, 40 GPIO pins and camera interface (CSI)[16]. They fulfil the requirement for remote controlling and video recording. If necessary, it can be used to control basic I/O and low level serial communication. It also came with HDMI port, enabling this credit card sized computer module to be directly connected to monitor display to quicken program development and troubleshooting processes.

Figure 3.6 Raspberry Pi 2 [16]

3.5.2 Raspberry Pi Camera Module (PiCamera)

There were a lot of camera devices that are compatible with RPi. RPi camera module as shown in Figure 3.7 was chosen instead due to several reasons:

- Its data transfer speed is considerably higher than USB Webcam
- It works with built-in library in the Linux Raspbian OS.
- It has fully customizable settings, such as exposure rate and frame rate.
- I can be fit easily to any angle of placement thanks to its modular design.

However the camera focus is fixed. Even the focus is, technically speaking, manually adjustable, it is not recommended to alter it since the lens is secured with hardened epoxy glue and has standard focus range of 1m to infinity as factory default [17]. However in this project, the focus must be set to 30cm range, hence it had to be re-focused by turning the lens mounting with care.

Figure 3.7 PiCamera module

3.5.3 High power LED and LED Driver

Visual odometry requires adequate amount of lighting to perform surface features tracking. It is not advised to power up high power LED by just connecting it to power source. High power LED will heat up under long operating period, causing its junction voltage to drop due to properties of semiconductor, drawing more current from power supply. Without proper current limiting, the LED will meltdown eventually or thermal runaway will happen[18].

Figure 3.8 3W LED driver module

Hence LED driver like one in Figure 3.8 is necessary as a bridge between power source and LED. It will limit the current draw to 700mA regardless of voltage output and protect the LED from overcurrent. The LED driver also support PWM input for dimming function but it was not being used in this project. The LEDs were also being attached to small heat sink with thermal grease as shown in Figure 3.9. This can improve the heat dissipation of the LED by covering microscopic air cavities formed due to the imperfection between two heat conductive surfaces[19]. Improved heat dissipation will also help extending the lifespan of the LEDs.

Figure 3.9 Two high power LEDs are attached to heat sink and connected to LED drivers

3.5.4 Dummy Base

An old toy car was modified to hold everything such as RPi and USB portable power bank. There were 2 versions of it, the 1st version was a failed attempt due to huge amount of disturbance collected and cannot cope with uneven interior surface of culvert. As in Figure 3.11, the first version of modified model car was lacking of suspension and had small wheel size. 2nd version is a current version in use and is a successful version. It is able to cope with uneven surface due to its larger wheels, as well as reducing vibration on the camera to reduce disturbance of uneven surface of the interior pipe thanks to the full suspension design.

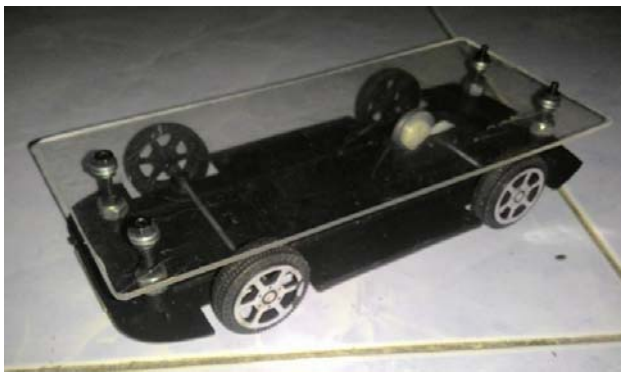

Figure 3.10 First version dummy base

3.6 Software

Software selection will be discussed in this section. The respective reasons of hardware selection will be explained in details.

3.6.1 Python Language

Python is a high level programming language very much similar to C++ language, with advantages of code readability, self-assigning variable types and cross platform. It does not require very specific annotations in programming and very much English-liked expression makes the code easier to be understand[20]. One of its advantage is that it can support wide range of open source libraries available, including OpenCV library.

Especially for RPi, a lot of dedicated libraries such as GPIO library and PWM library are being developed for Python language so that beginner as well as expert users can make use of it to control external devices which is ideal for IoT project.

Figure 3.11 Logo of Python Language[20]

3.6.1.1 Numpy Library

Numpy is a built-in math-oriented library in Python. It is used for various mathematical and scientific operations in many applications such as statistical analysis and image processing. Due to its Matlab-liked expression[21], this enables Matlab user to quickly adapt to Python language and optimize the program by using faster Matlab annotations instead of OpenCV library, especially in manipulating multidimensional matrix array in image processing.

Figure 3.12 Logo of Numpy library [21]

3.6.1.2 PiCamera Library

PiCamera library is a library specially being developed to control CSI port camera like RPi Camera module. It has various manual control on the camera module such as ISO and contrast. It also capable of alter data path of the retrieved image such as LAN streaming and streaming into RAM.

3.6.2 OpenCV Optical Flow

OpenCV is a library specialized for image processing with source code of optical flow included. Optical flow is one of the techniques of feature tracking in image processing. It tracks the motion of features in a set of image frames like one as shown in Figure 3.13. Though there are variations of optical flow algorithms, Lucas-Kanade optical flow that uses Shi-Tomasi algorithm [14] (enhanced version of Harris corner detection algorithm) is used in this project. Each frame will be compared with the computed feature points. Those points will be drawn with OpenCV polylines function to show the flow of the feature points.

Figure 3.13 Example output from Lucas-Kanade algorithm [14]

3.6.3 Remote Desktop Connection

Remote Desktop Connection is a Windows default program built-in in almost all versions of Windows OS. It can connect and remote control any computer over LAN as long as the host computer has Remote Desktop Connection enabled or with compatible applications installed in computer of different OS. Since RPi run on Linux based OS, it has few applications for that purpose, such as “XRDP” and “VNC”. Hence, as long as the RPi is connected to the same network as the main computer, remote controlling the RPi

is possible by knowing the exact IP address of the RPi.

3.6.4 Video Stabilization

There are 2 modes of operation to capture image frames in RPi camera, which are continuous capture mode and video recording mode. However, video stabilization function which will help reducing linear disturbance is only available in video recording mode.[22] Video stabilization is performed within the RPi hardware encoder inside GPU, hence no special image processing library is needed. It also cause no extra load in CPU, hence it is ideal for live image processing, However, since it is enabled only in video recording mode, this built-in feature cannot be utilized in real time distance measurement. Hence post processing will be carried out instead.

3.7 Important parameters

There were few intrinsic parameters of the camera need to be understood in order for the visual distance measurement to work properly. This section will discuss about those parameters in details, including the theory as well as the parameters of the selected hardware.

3.7.1 Focus (working distance)

In both methods, working distance of a camera must be known. There are formulas of calculating the values of the working distance. The value will be different depends on the diameter of the culvert and the size of the robotic crawler (in this case is a dummy base) as illustrated in Figure 3.14. Since the diameter of the culvert and the size of the dummy car were known, the working distance (in this case is camera height) can be derived and Equation 3.4 is obtained which can be used in the program. The derivation of formula will be shown at next page.

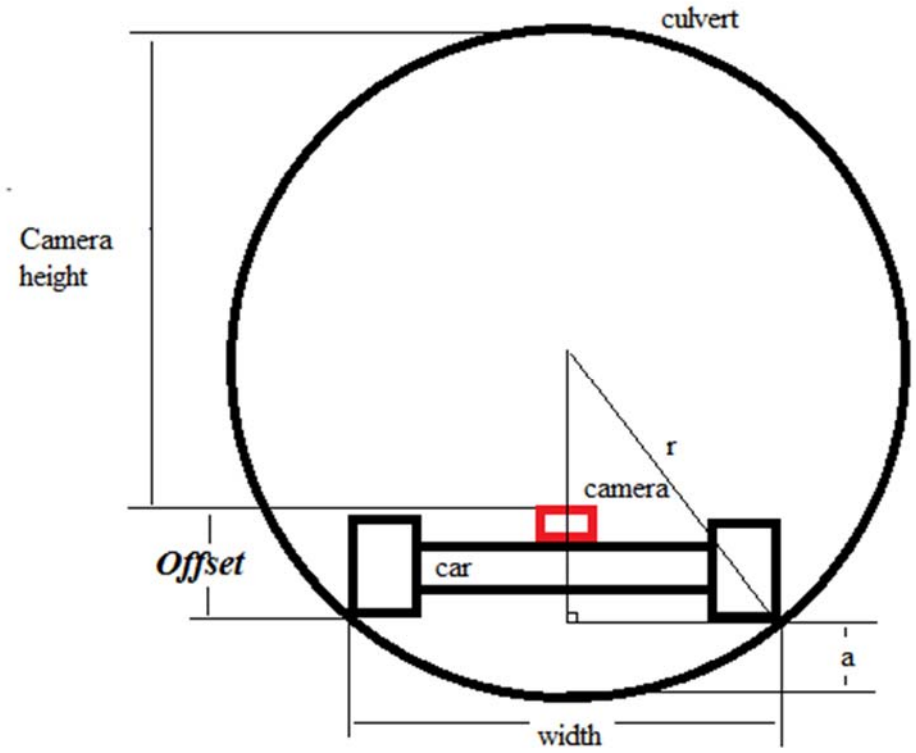

Figure 3.14 Finding the height of camera relative to the surface of culvert

According to Pythagoras Theorem:

$$r^2 = (r - a)^2 + \left(\frac{width}{2}\right)^2$$

By forming a quadratic equation with a as subject:

$$a^2 - 2ar + \frac{width^2}{4} = 0$$

Parameters a can be found by solving its roots:

$$a = r \pm \frac{\sqrt{4r^2 - width^2}}{2}, width < 2r$$

Meanwhile:

$$camera\ height = diameter - a - offset = 2r - a - offset$$

Where, $offset$ is the distance between the camera and the contact surface of the wheel.

Then:

$$camera\ height = 2\mathrm{r} - \left(r \pm \frac{\sqrt{4r^2 - width^2}}{2}\right) - offset$$

$$= r \mp \frac{\sqrt{4r^2 - width^2}}{2} - offset$$

Since the required *a* is minimum value, then:

$$\boldsymbol{camera\ height = H = r + \frac{\sqrt{4r^2 - width^2}}{2} - offset} \quad (3.4)$$

3.7.2 Field of View (FoV)

FoV can be obtained from product datasheet or from simple calibration. It can be measured as shown in Figure 3.15. Both methods require this parameter and were being used in different ways. The FoV is different depends on the aspect ratio of the image as explained in the next section. With known distance of camera relative to the image and the real height and width of image, FoV can be determined with simple trigonometry.

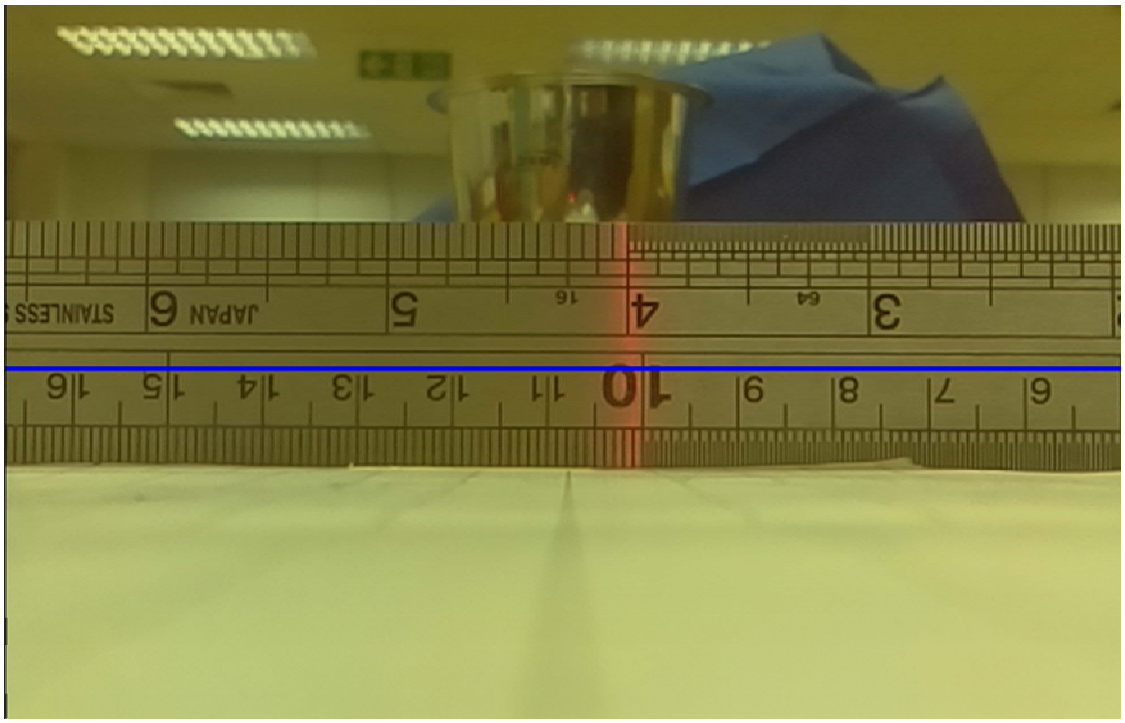

Figure 3.15 Measuring VFoV (rotated image)

3.7.3 Sensor Size

Sensor size can be obtained from datasheet. It is important especially in setting video frame size and orientation. The sensor resolution is 2592 x 1944 pixels. Despite of its fixed sensor size, different resolution images can be achieved by using specific amount of binning of pixels in the sensor [23].

The sensor itself is in 4:3 ratio. Hence in order to capture an image that make uses of the FoV of the camera, the captured image or video frame must also be in 4:3 ratio as visualised in Figure 3.16. Full FoV will be used regardless of any retrieved image size, such as 640 x 480 pixel or 960 x 720 pixel. Any image with different ratios, such as 1280 x 720 pixels which is 16:9 ratio will not work since only partial FoV is used [24]. The RPi's camera also has a discrete set of input modes which limit the flexibility of the camera. Those input modes are shown in Table 3.1.

Figure 3.16 FoV coverage [24]

Table 3.1 Hardware default of Pi Camera [24]

#	Resolution	Aspect Ratio	Video	Image	FoV	Binning
1	1920x1080	16:9	x		Partial	None
2	2592x1944	4:3	x	x	Full	None
3	1296x972	4:3	x		Full	2x2
4	1296x730	16:9	x		Partial	2x2
5	640x480	4:3	x		Full	4x4

The image frame can also be software rotated without physically rotating the PiCamera module. That means if the image frame is software rotated by 90° or 270°, the apparent HFoV is actually the true VFoV of the camera as illustrated in Figure 3.17. Only the HFoV of the image is true in software rotated image since it is equivalent to true VFoV of the CMOS sensor. VFoV of the image is just the partial HFoV of the CMOS sensor hence it is not true. If that case happened, the value of VFoV inserted in the program is actually representing the HFoV of the image frame which will confuse the programmer.

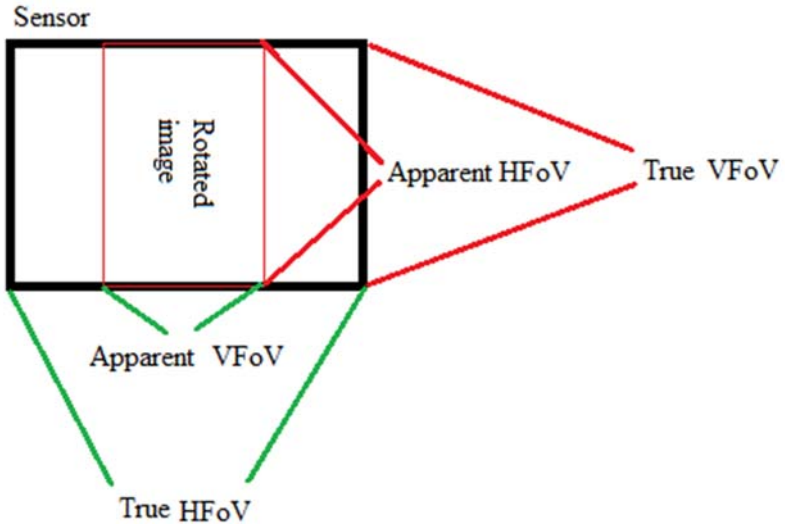

Figure 3.17 FoV of rotated image

Therefore, image rotation was avoided by rotating the PiCamera module to the desired orientation instead. Figure 3.18 shows the top view of FoV calibration setup with PiCamera module being attached to RPi and hold in place with retort stand. Figure 3.19 shows the comparison of normal image frame of 4:3 ratio (shaded area) and software rotated image frame of 16:9 ratio (bright area, rotated image overlay).

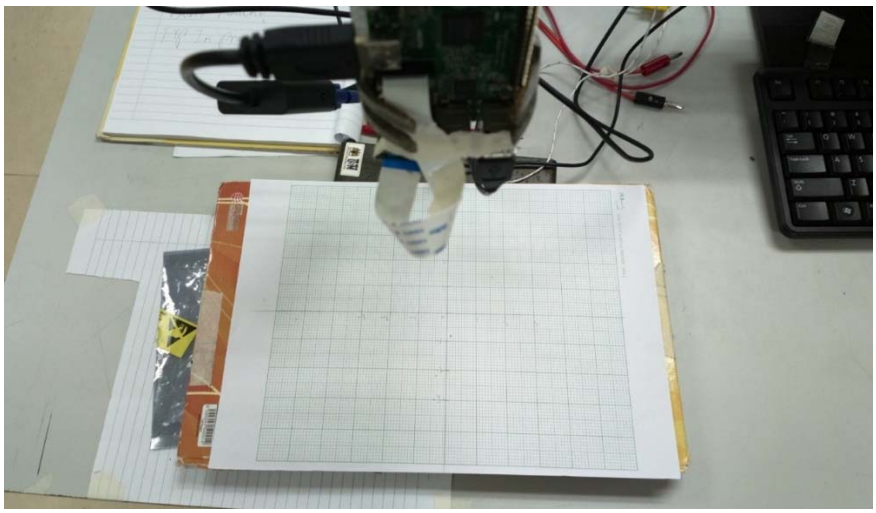

Figure 3.18 Top view of calibration setup.

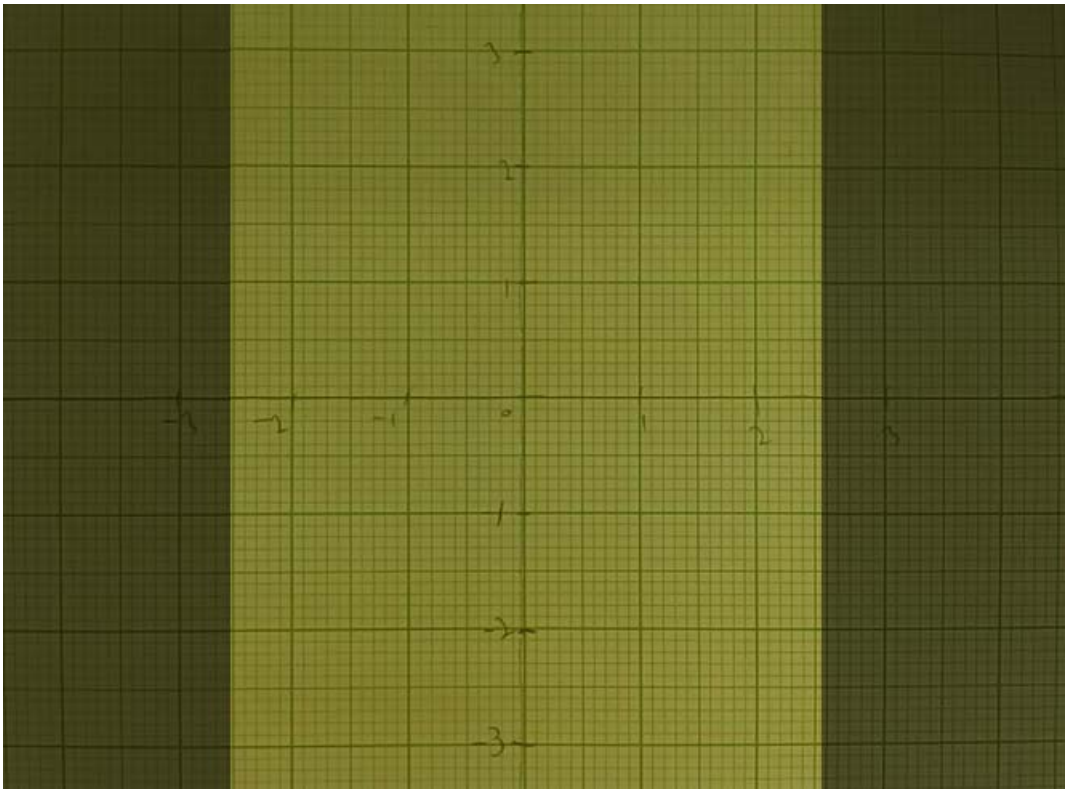

Figure 3.19 Normal orientation (shaded area) vs software rotated (bright area, rotated 90 degree for comparison)

3.8 Summary

So far, the methods of vision distance measurement has been determined. By considering the dimensions of the dummy base and culvert, intrinsic parameters for the optical flow algorithm were able to be computed and to be used in distance measurement. After the integration of all hardware such as LED drivers, Raspberry Pi and power bank, calibration of hardware was carried out and system testing can be executed. Once the output of software showed no signs of abnormality, series of 20 seconds length of video recordings will be taken and post-processing will be carried out to collect results. Those results will be discussed at next chapter.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Introduction

In this chapter, results of the testing are going to be presented. There were 3 experiments for both methods, each with 10 trials. Those experiments were named as:

- Case 1, camera is aligned perpendicularly to the surface of culvert.
- Case 2, camera is tilted approximately 24° to the surface of culvert.
- Case 3, camera is tilted approximately 40° to the surface of culvert.

Data was tabulated and visualized by bar graphs, average and standard deviation of the error were also been tabulated to show their performances for each case. Images below show the experimental setup for 3 case studies. As in Figure 4.1, the power bank attached to the dummy base as power source for both RPi and high power LEDs.

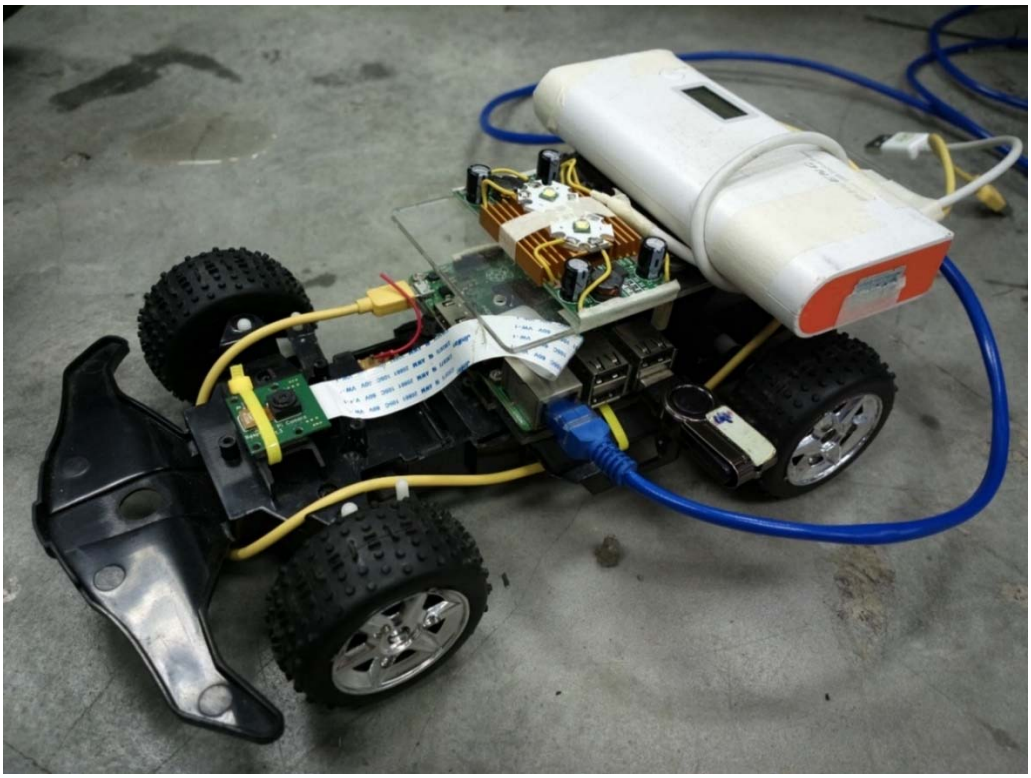

Figure 4.1 Overall setup. USB power bank as remote power source for high power LEDs and RPi.

The side view of the overall setup was as shown in Figure 4.2. The dummy car was equipped with full suspension (spring absorbers attached to all shaft drive) as shown in Figure 4.3 to reduce bumping effect and to help reducing disturbance. Its big sized wheels also helped reducing bump caused by miniature hole on the interior surface of the culvert.

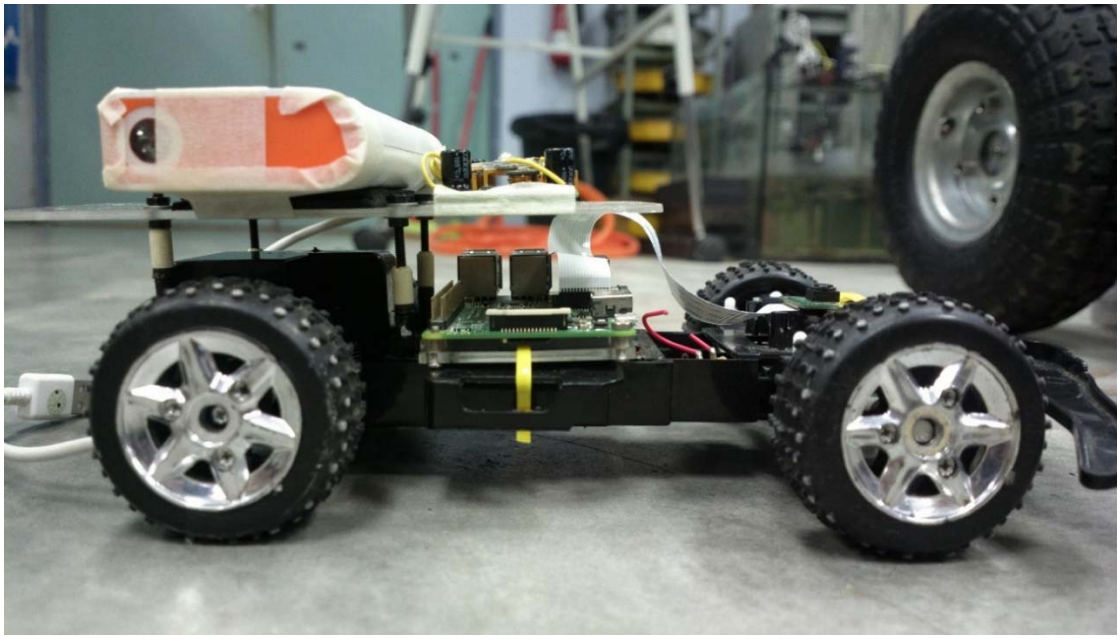

Figure 4.2 Side view of model car

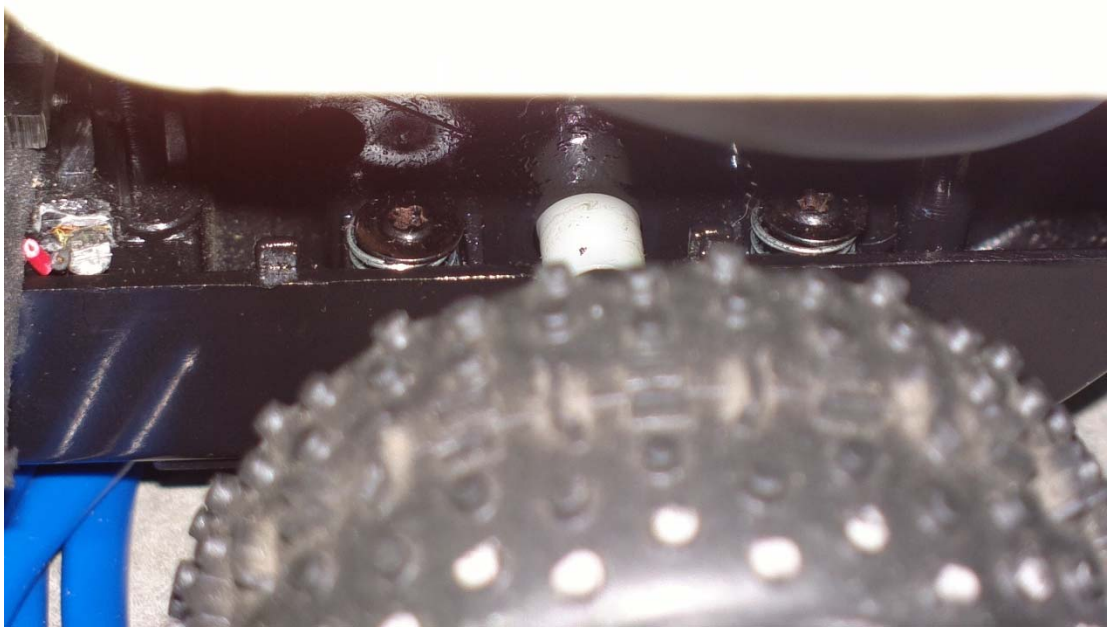

Figure 4.3 Rear wheel suspensions

LAN cable shown in Figure 4.4 was used for remote control as well as pulling cable as shown in Figure 4.5 to pull the dummy base during video recording.

Figure 4.4 LAN cable connected to laptop for remote control and as pulling wire. USB remote storage connected for video recording.

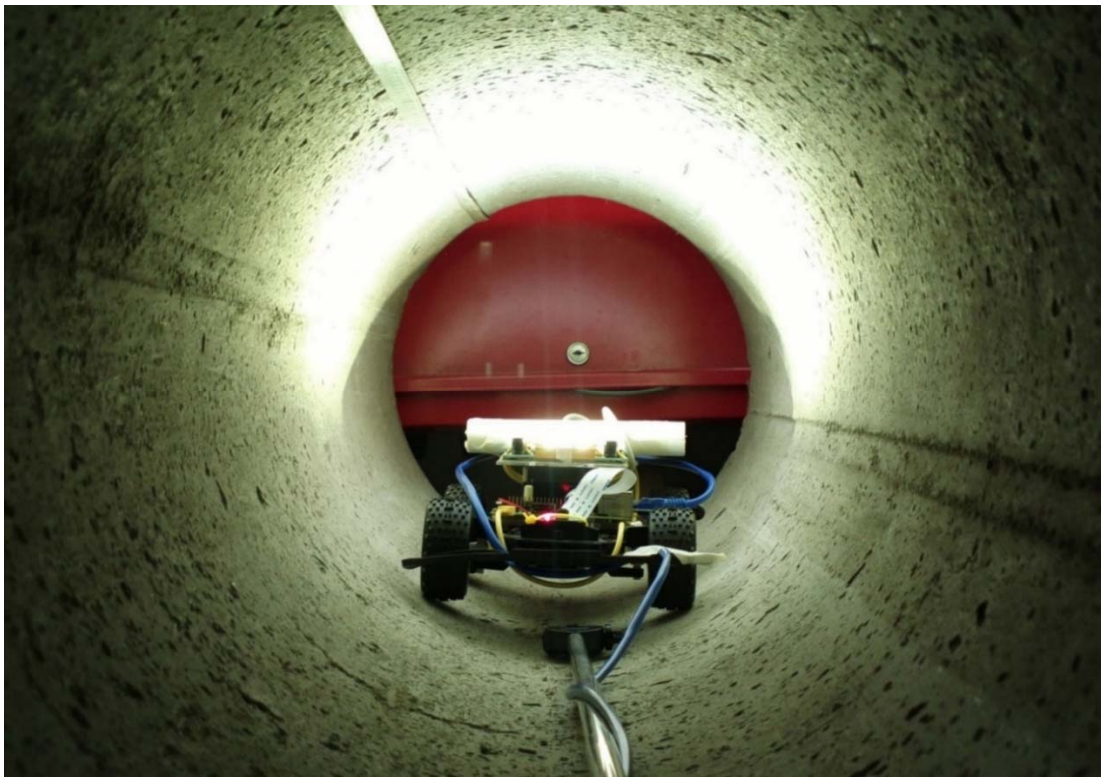

Figure 4.5 Experiments were carried out inside the culvert. Above is attached with metre rule for actual distance readout.

Figure 4.6 shows the area of image processing under the principal ray (centre line drawn across the image frame). As seen from Figure 4.7, “Distance y” represented the output from Method 1 and “Distance yy” represented the output from Method 2. Since the video recording was a standard 20 second length throughout the experiment, average speed in m/s unit can be attained by dividing the total distance travelled by 20 seconds, then being converted to m/s unit.

The consistency of the algorithm in 10 trials can be observed by using standard deviation of the errors. The lower the value, the more consistent the output of the algorithm. The overall accuracy of the system can be seen by taking average of all the errors accumulated in 10 trials.

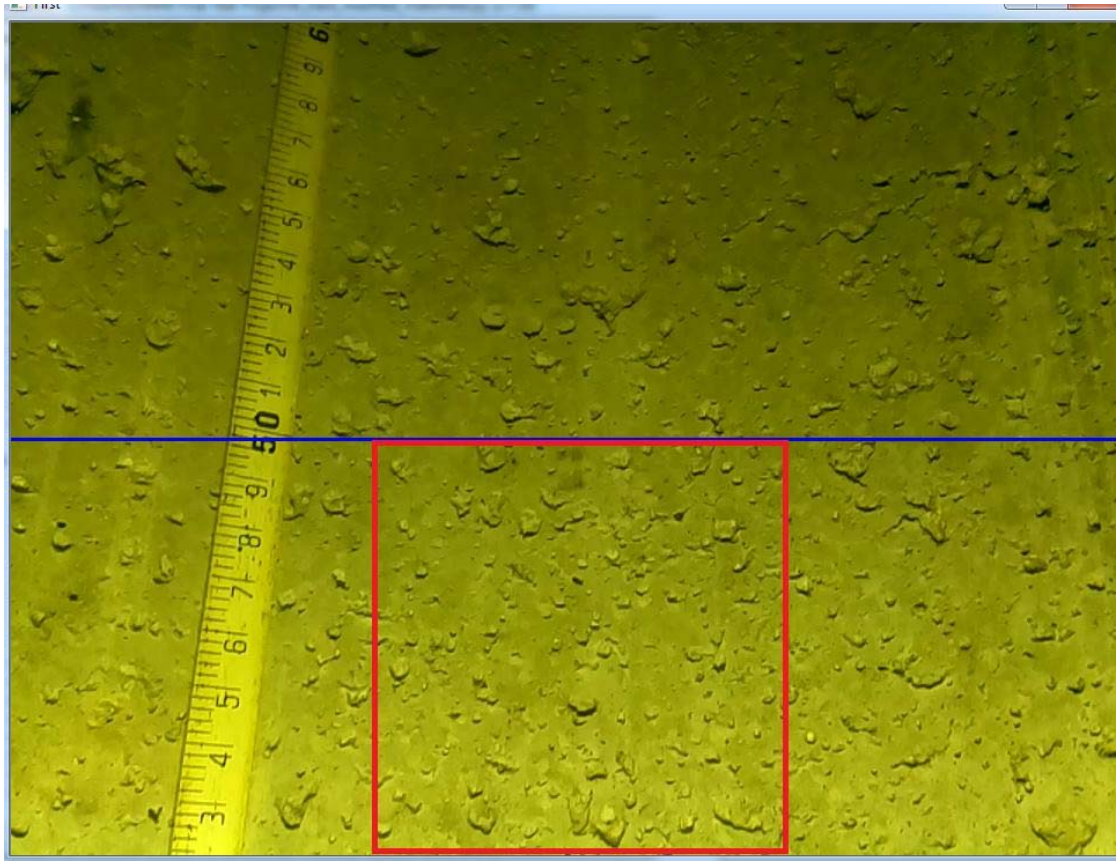

Figure 4.6 Blue line (principal ray), Red box (area to be image processed)

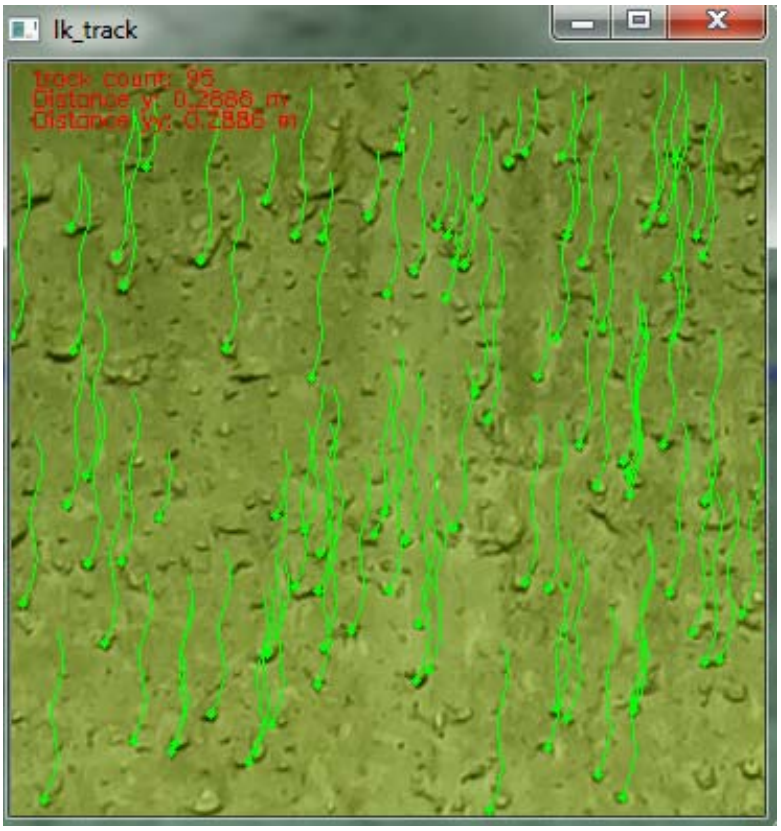

Figure 4.7 Optical flow algorithm has been executed.

4.2 Case 1 – Camera is aligned perpendicularly to the surface of culvert.

Parameters:

- Camera VFoV = 37.16°
- Camera angle = 0°
- Camera offset = 5.4cm
- Diameter of culvert = 30cm
- Width of dummy car = 15.1cm
- Camera height relative to the surface (calculated)= 22.56cm
- Video frame size = 960 x 720 pixel
- Crop area = 180 x 180 pixel

Those parameters are illustrated as in Figure 4.8 for reference. Video frame size is the pixel height and width of the recorded video whilst crop area is the size of area for feature tracking. The configuration of the camera in Case 1 study was shown in Figure 4.9. A small PVC foam was attached in between the camera module and the dummy base to cover up the uneven surface at the back of the camera module that's full of SMD electronics.

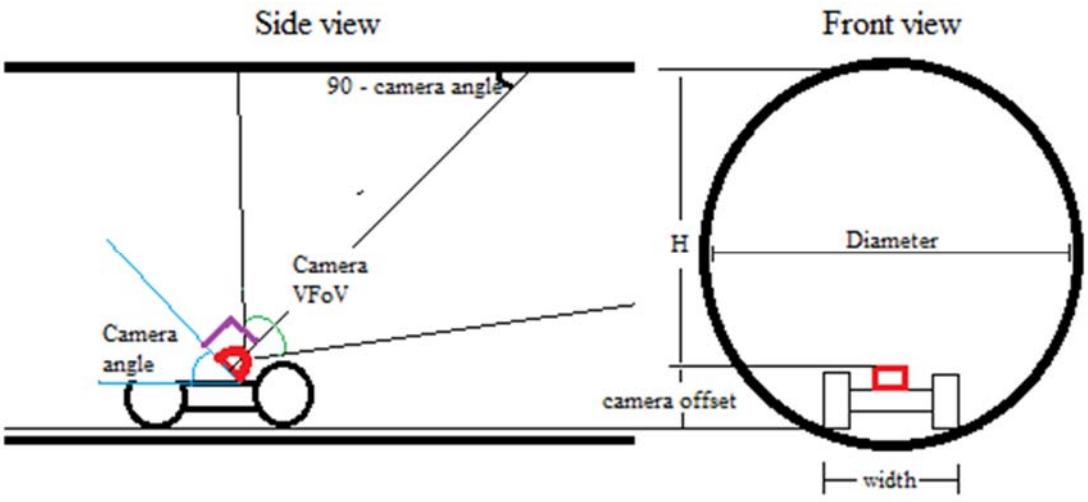

Figure 4. 8 Illustration of the parameters

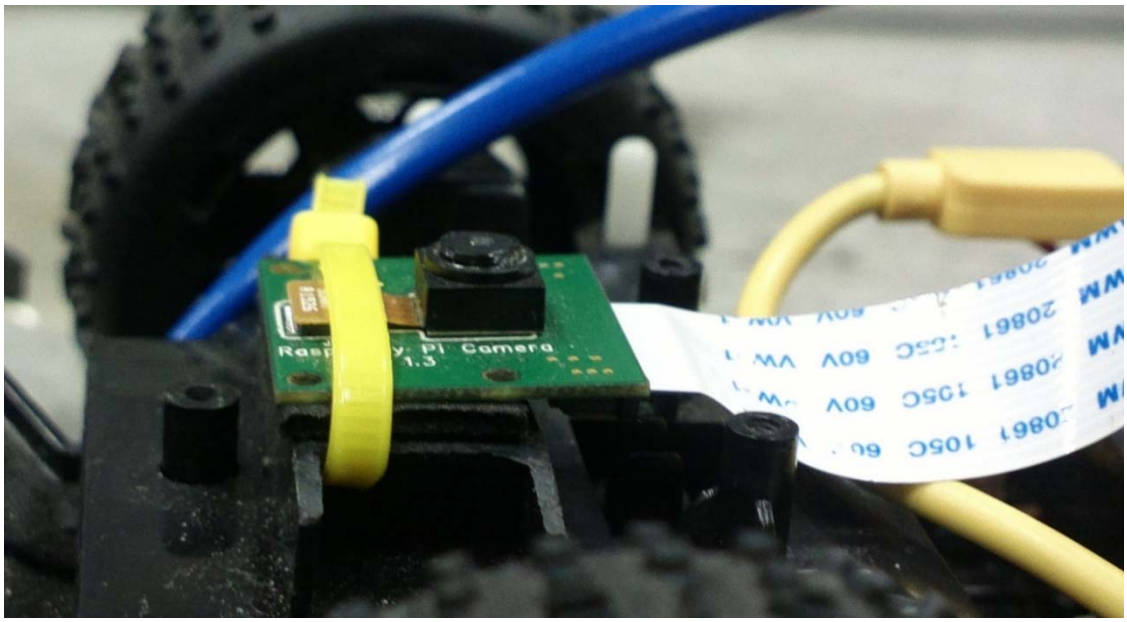

Figure 4.9 Side view of camera for Case 1

Cable tie was used to ensure the camera module was fixed to the dummy base. The camera might still not perfectly perpendicular to the surface of culvert since there were suspensions at the front wheel, but just assume that the difference is miniscule. The effect was hardly visible as shown in Figure 4.10.

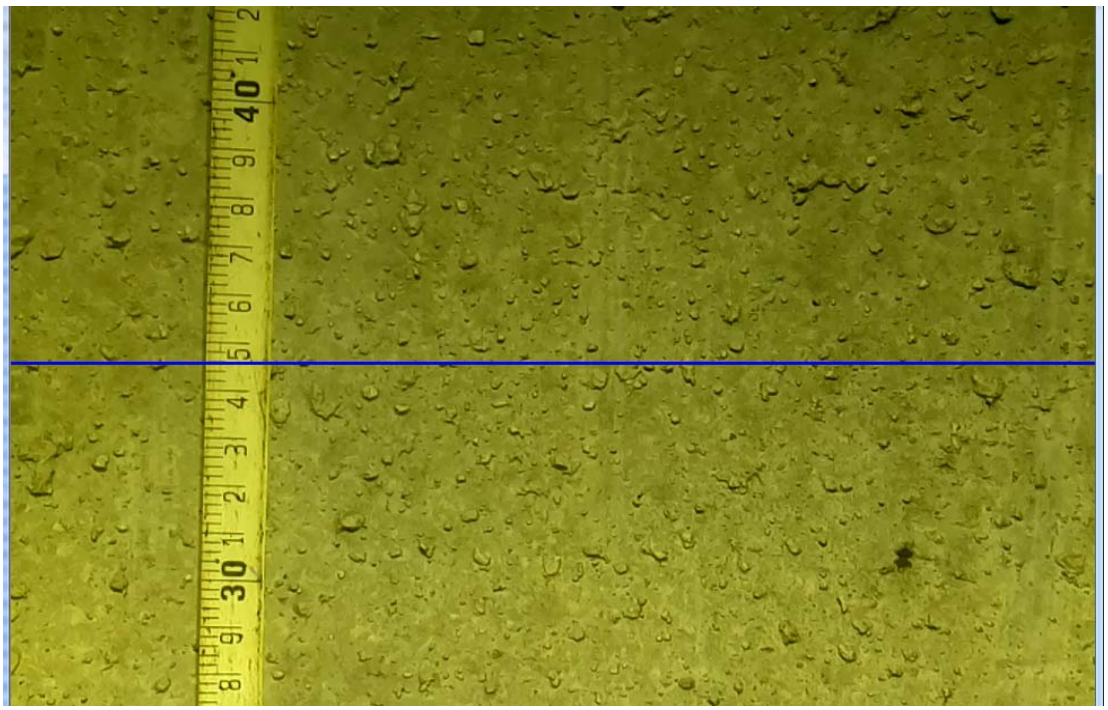

Figure 4.10 Image frame as seen from camera for Case 1

As seen from image obtained in Figure 4.10, the metre rule was a bit leaned to the right of the image frame, which suggested that either camera module was not perfectly perpendicular or the dummy base was not parallel to the culvert. Nonetheless, ten trials were taken and Table 4.1 shows both collected and calculated data for Case 1 study.

Table 4.1 Data of both methods of Case 1 for 10 trials.

Trial	Start distance (cm)	Stop distance (cm)	Actual distance travelled (cm)	Average speed (m/s)	Method	Estimate distance (cm)	Error (%)
1	40.3	115.2	74.9	0.037	1st	71.20	4.940
					2nd	71.20	4.940
2	40.0	120.4	80.4	0.040	1st	76.58	4.751
					2nd	76.58	4.751
3	34.6	133.4	98.8	0.049	1st	93.88	4.980
					2nd	93.88	4.980
4	36.6	100.7	64.1	0.032	1st	61.89	3.448
					2nd	61.89	3.448
5	29.9	118.2	88.3	0.044	1st	84.61	4.179
					2nd	84.61	4.179
6	44.2	113.3	69.1	0.035	1st	67.62	2.142
					2nd	67.62	2.142
7	28.8	124.4	95.6	0.048	1st	94.13	1.538
					2nd	94.13	1.538
8	36.4	115.2	78.8	0.039	1st	77.09	2.170
					2nd	77.09	2.170
9	33.3	107.3	74.0	0.037	1st	72.48	2.054
					2nd	72.48	2.054
10	36.2	122.9	86.7	0.043	1st	84.16	2.930
					2nd	84.16	2.930

The average error and the standard deviation of the error is tabulated in Table 4.2, whilst the graph of error and average speed of both methods against the trial number of Case 1 study were shown in Figure 4.11.

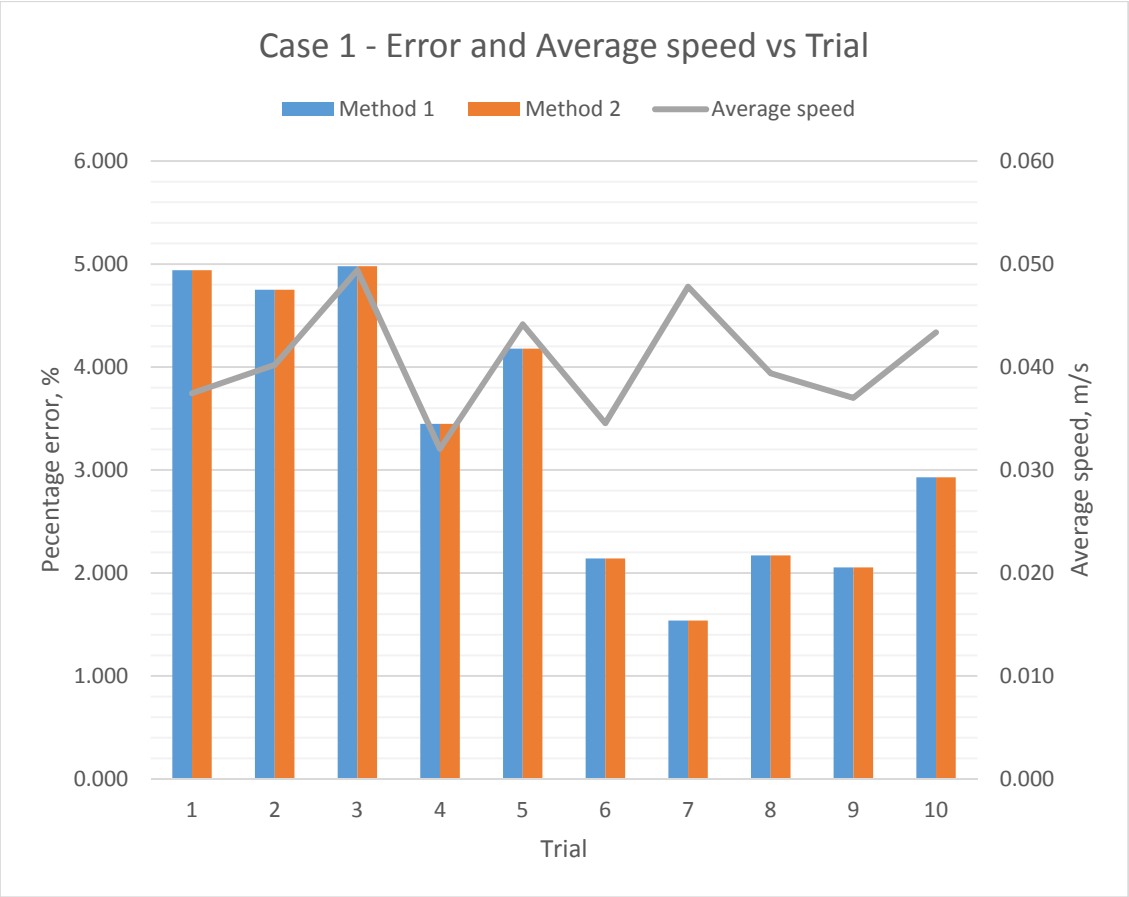

Figure 4.11 Graph of Error and Average Speed vs. Trial for both methods in Case 1 experiment

Table 4.2 Average and Standard deviation of error for both methods in Case 1

Case 1	Method 1	Method 2
Average	3.313	3.313
Standard deviation	1.258	1.258

4.3 Case 2 – Camera is tilted (approximately 24 degrees)

Parameters:

- Camera offset = 5.8cm
- Camera VFoV = 37.16°
- Camera angle = 24°
- Diameter of culvert = 30cm
- Camera height relative to the surface= 22.56cm
- Video frame size = 960 x 720 pixel
- Crop area = 180 x 180 pixel

Both Figure 4.12 and Figure 4.13 show the configuration of the camera in Case 2 study. A folded paper block was used to tilt the camera. Masking tape was used to fix its position.

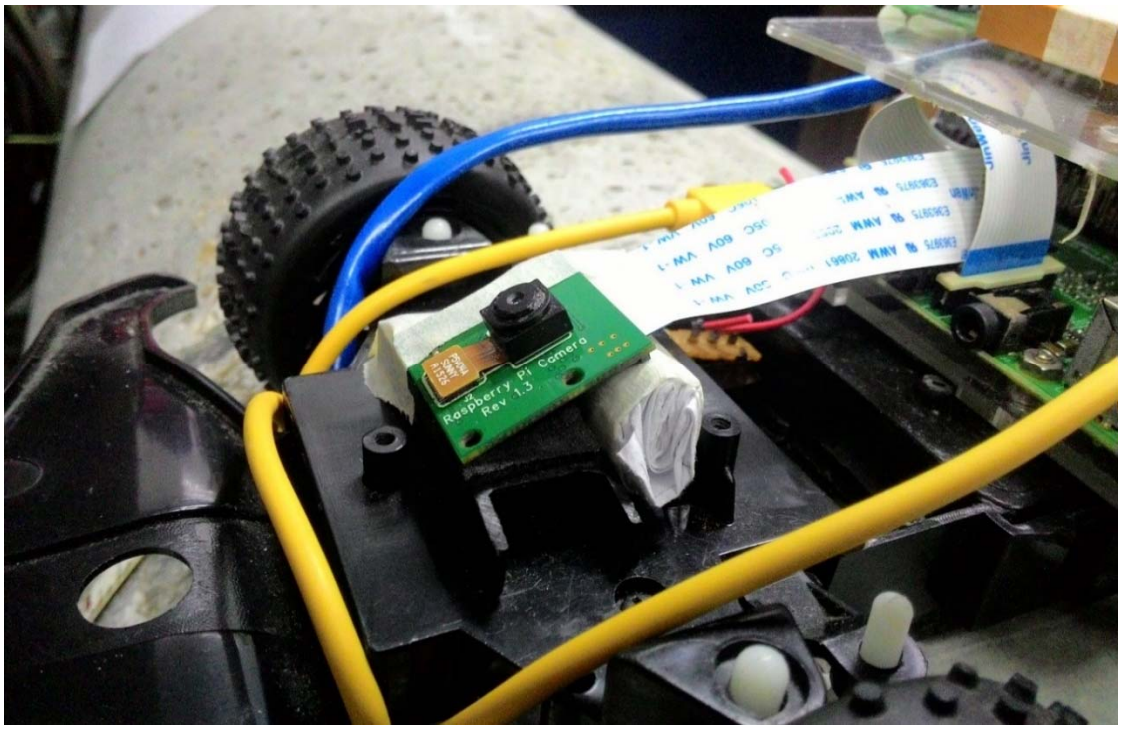

Figure 4.12 Camera module is tilted approximately 24 degrees.

The camera tilt angle was found to be approximately 24° by using protractor. The camera offset was also being altered and was measured.

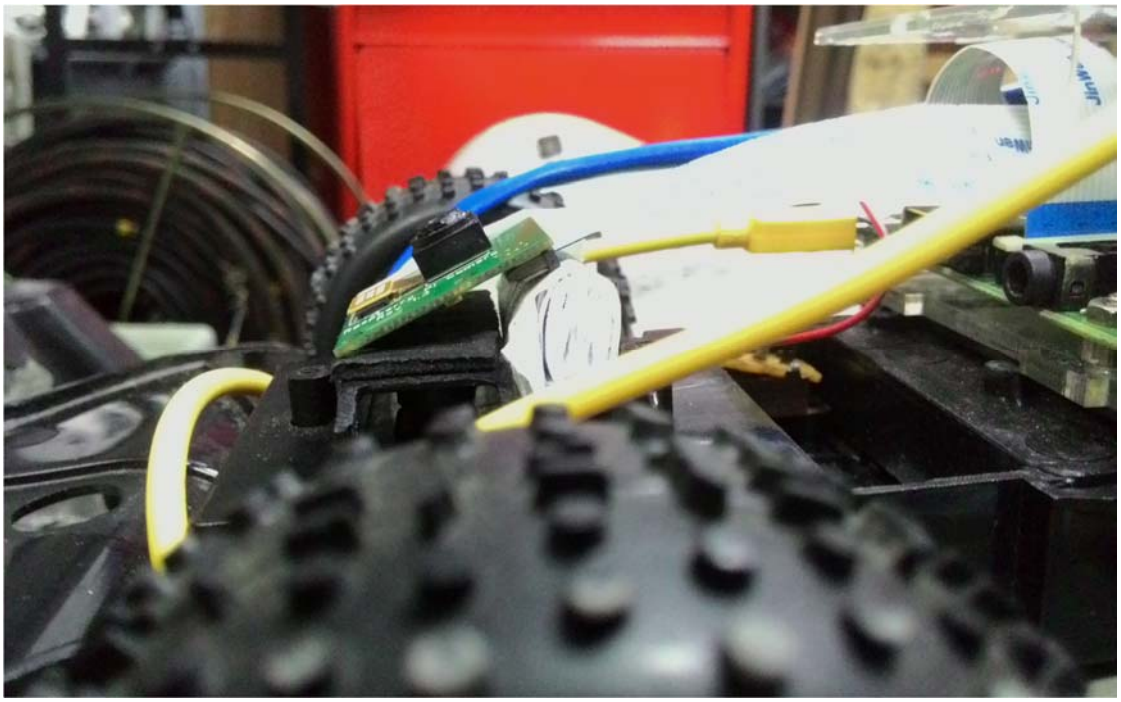

Figure 4.13 Side view of camera configuration for Case 2

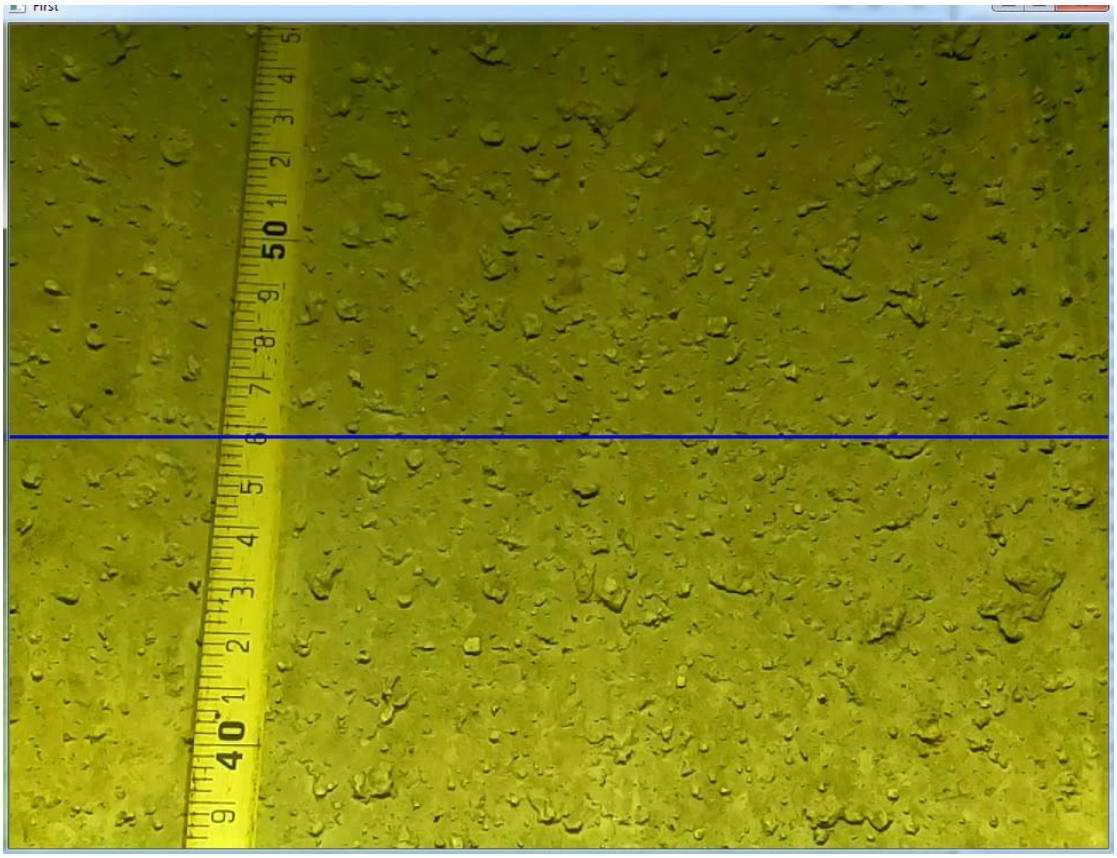

Figure 4.14 Image frame as seen from camera for Case 2

As seen from image frames obtained in Figure 4.14, the metre rule was obviously tilted. All area in the image seemed to be in focus. Ten trials were taken and Table 4.3 shows both collected and calculated data for Case 2 study.

Table 4.3 Data of both methods of Case2 for 10 trials.

Trial	Start distance (cm)	Stop distance (cm)	Actual distance travelled (cm)	Average speed (m/s)	Method	Estimate distance (cm)	Error (%)
1	38.6	129.0	90.4	0.045	1st	88.52	2.080
					2nd	88.26	2.367
2	41.4	111.3	69.9	0.035	1st	67.18	3.891
					2nd	66.90	4.292
3	49.7	130.7	81.0	0.041	1st	79.34	2.049
					2nd	79.08	2.370
4	42.1	109.6	67.5	0.034	1st	66.09	2.089
					2nd	65.86	2.430
5	40.3	122.9	82.6	0.041	1st	81.88	0.872
					2nd	81.44	1.404
6	44.5	129.2	84.7	0.042	1st	82.96	2.054
					2nd	82.72	2.338
7	39.9	121.6	81.7	0.041	1st	80.39	1.603
					2nd	80.23	1.799
8	39.4	126.2	86.8	0.043	1st	84.50	2.650
					2nd	84.01	3.214
9	38.6	129.0	90.4	0.045	1st	88.52	2.080
					2nd	88.26	2.367
10	34.6	134.2	99.6	0.050	1st	96.20	3.414
					2nd	96.04	3.574

The average error and the standard deviation of the error were tabulated in Table 4.4, whilst the graph of error and average speed of both methods against the trial number of Case 2 study were shown in Figure 4.15.

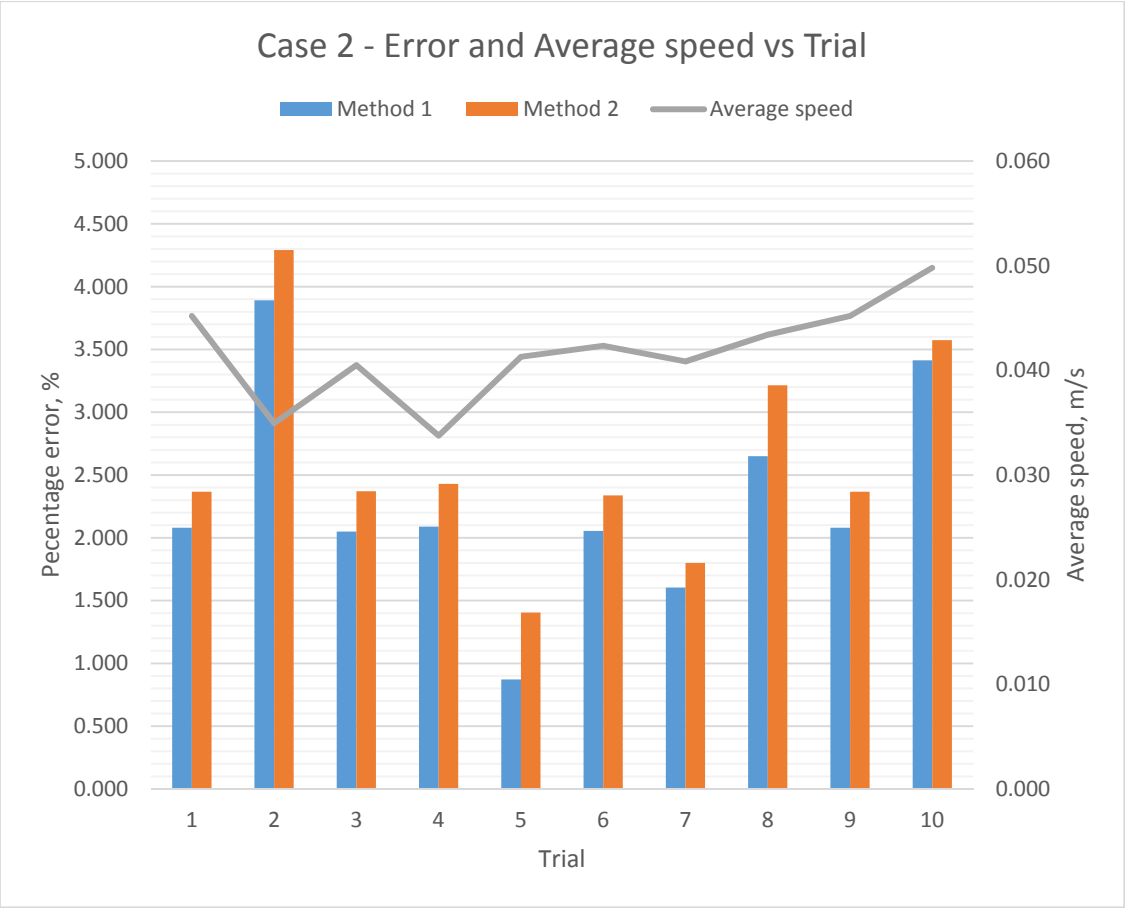

Figure 4.15 Graph of Error and Average Speed vs. Trial for both methods in Case 2 experiment

Table 4.4 Average and Standard deviation of error for both methods in Case 2

Case 2	Method 1	Method 2
Average	2.278	2.616
Standard deviation	0.817	0.807

4.4 Case 3 – Camera is tilted (approximately 40 degrees)

Parameters:

- Camera offset = 6cm
- Camera VFOV = 37.16°
- Camera angle = 40°
- Diameter of culvert = 30cm
- Camera height relative to the surface = 22.56cm
- Video frame size = 960 x 720 pixel
- Crop area = 180 x 180 pixel

The configuration of the camera in Case 3 study was shown in both Figure 4.16 and Figure 4.17. Another thicker paper block was used to tilt the camera more oblique. Masking tape was used to fix its position.

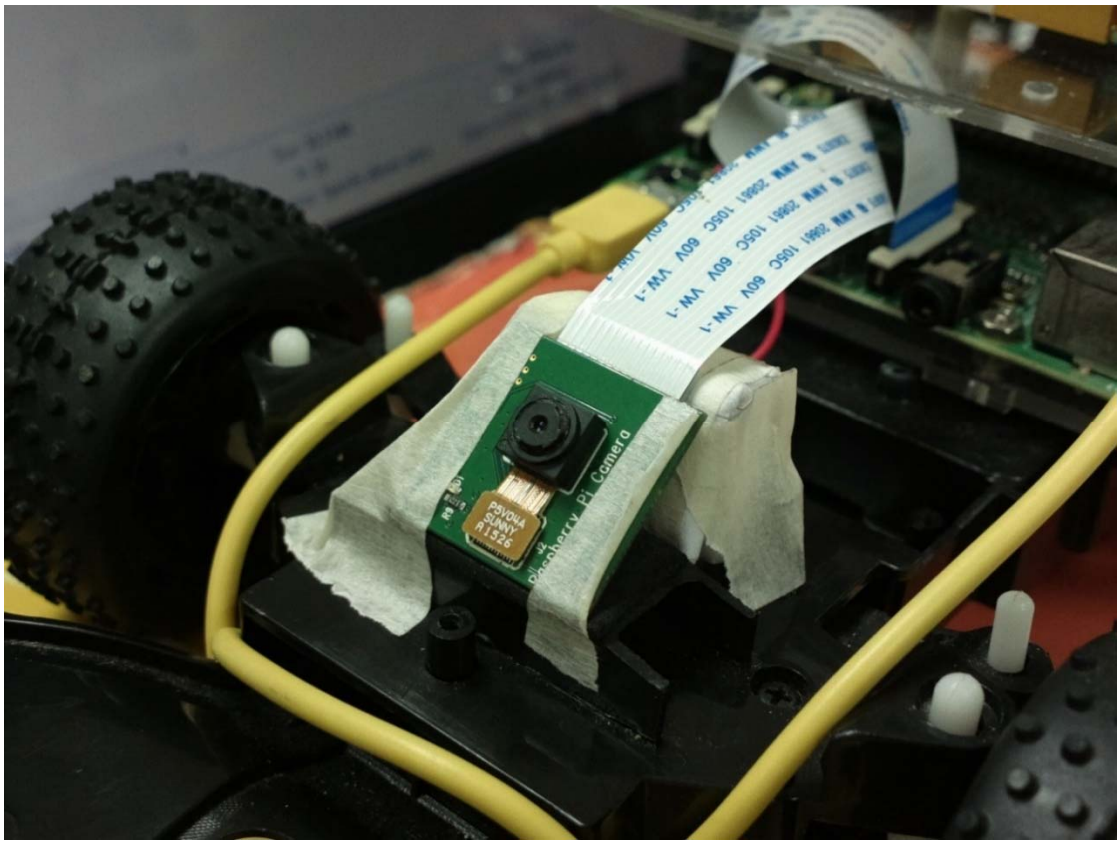

Figure 4.16 Camera module is tilted approximately 40 degrees.

The camera tilt angle was found to be approximately 40° by using protractor. The camera offset was also being altered and was measured.

Figure 4.17 Side view of camera configuration for Case 3

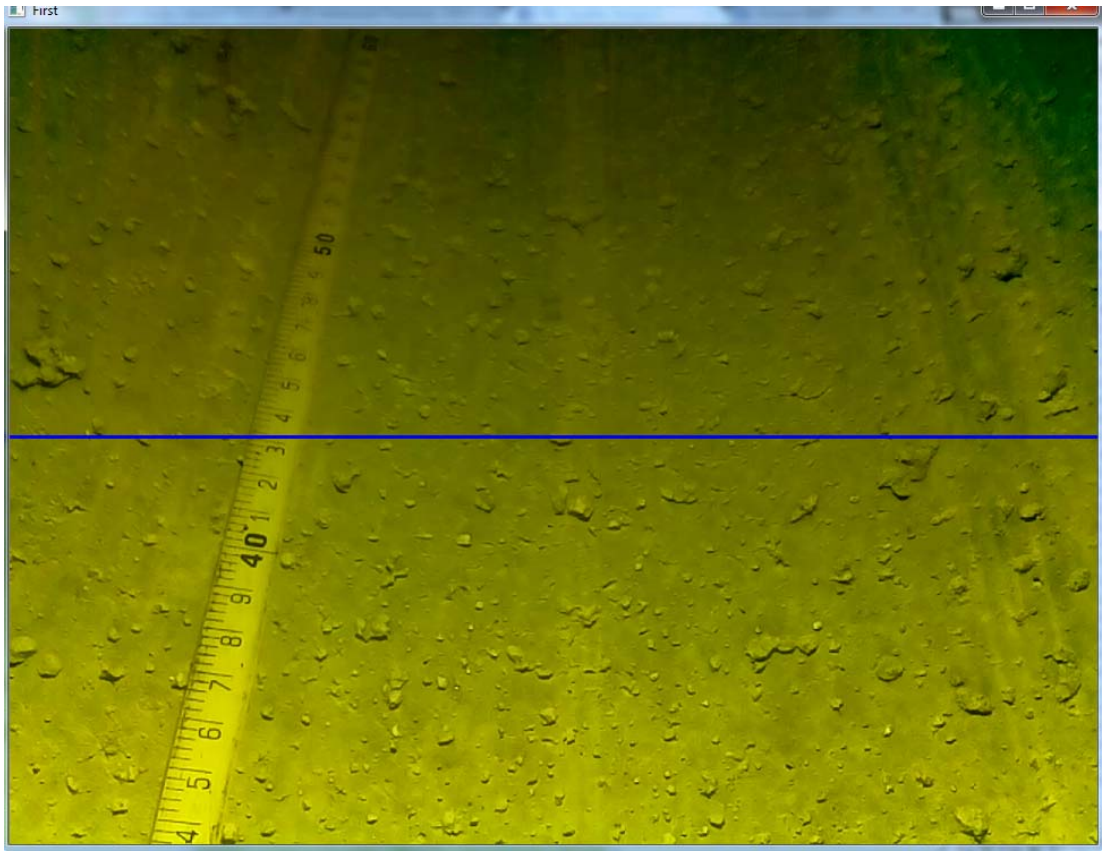

Figure 4.18 Image frame as seen from camera for Case 3

As seen from image frames obtained in Figure 4.18, the metre rule was tilted more than that of Case 2. However, not all image area in the image is in focus. The features at the upper side (further features) seemed to be blurred. Since only the small area under the principal ray was used, it should cause no problem to the optical flow. Ten trials were taken and both collected and calculated data for Case 3 study were shown in Table 4.5.

Table 4.5 Data of both methods of Case3 for 10 trials.

Trial	Start distance (cm)	Stop distance (cm)	Actual distance travelled (cm)	Average speed (m/s)	Method	Estimate distance (cm)	Error (%)
1	36.8	118.8	82.0	0.041	1st	64.80	20.976
					2nd	79.48	3.073
2	50.4	127.6	77.2	0.039	1st	61.23	20.687
					2nd	74.94	2.927
3	50.4	138.0	87.6	0.044	1st	70.13	19.943
					2nd	85.79	2.066
4	42.6	137.5	94.9	0.047	1st	74.64	21.349
					2nd	91.92	3.140
5	48.2	121.6	73.4	0.037	1st	58.48	20.327
					2nd	71.33	2.820
6	36.3	110.6	74.3	0.037	1st	59.76	19.569
					2nd	72.84	1.965
7	43.0	116.6	73.6	0.037	1st	58.89	19.986
					2nd	71.67	2.622
8	39.4	128.1	88.7	0.044	1st	70.67	20.327
					2nd	86.62	2.345
9	41.0	130.4	89.4	0.045	1st	71.86	19.620
					2nd	86.91	2.785
10	36.4	129.8	93.4	0.047	1st	73.69	21.103
					2nd	90.03	3.608

The average error and the standard deviation of the error were tabulated as in Table 4.6, whilst the graph of error and average speed of both methods against the trial number of Case 3 study were shown in Figure 4.19.

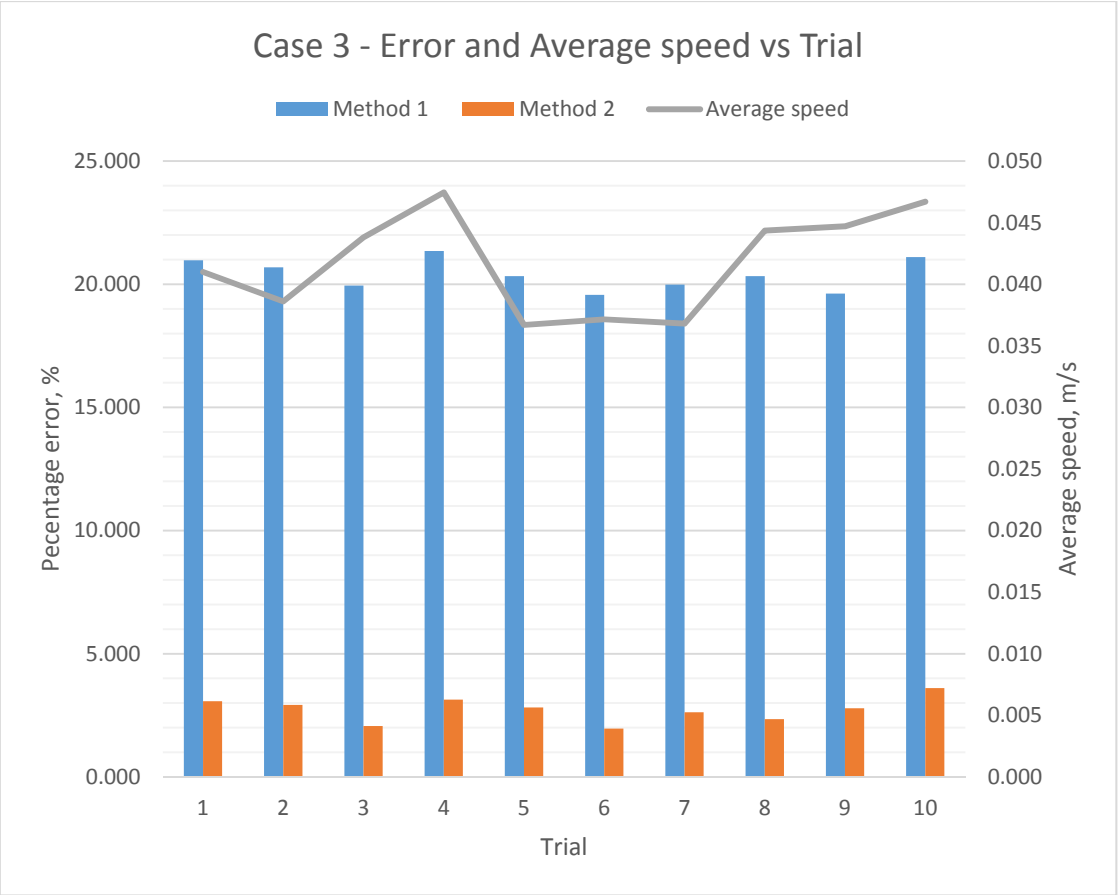

Figure 4.19 Graph of Error and Average Speed vs. Trial for both methods in Case 3 experiment

Table 4.6 Average and Standard deviation of error for both methods in Case 3

Case 3	Method 1	Method 2
Average	20.389	2.735
Standard Deviation	0.591	0.479

4.5 Discussion

Data for 3 cases has been collected. Those results revealed some interesting facts.

4.5.1 Case 1

For Case 1, the camera was perpendicular to the surface of culvert. The Method 1 was designed to estimate the distance with this configuration. Nonetheless, Method 2 was also used so that the results can be compared. Method 1 worked within 5% of error. In the meantime, Method 2 showed the same results as of Method 1. Since the accuracy of Method 1 was considered acceptable, the output of Method 2 also was considered true despite of using different algorithm.

4.5.2 Case 2

For Case 2, the camera was approximately 24° to the surface of culvert (or 66° relative to the principal ray of the camera). The Method 1 was not being designed to encounter with such configuration while Method 2 was, hence Method 1 was expected to fail in this experiment. However both methods were applied so that the outputs can be compared. It was found that the output from Method 1 has less error (more accurate) than Method 2 which is theoretically not supposed to happen.

This might due to the unusually small FOV of the RPi camera. In Case 2, the camera was inclined at 24°. Even so, the pixel distances of further feature points did not differ much from that of closer feature points within the frames. As shown in Figure 4.20, the pixel distance output from Method 1 along y-axis did not differ much between physically further feature points and closer feature points. Even the data averaging process was insignificant to the output. As a result it seemed to be correct but actually it is theoretically not relevant. Also, it was believed that the speed of the dummy base is one of the factor. If the dummy base is pulled at higher speed, the difference of pixel

distance between further feature points and closer feature points should be higher and the effect should be more obvious.

Figure 4.20 Pixel distance output of 2 frames of images (2 big [ ] brackets) of Method 1, clearly show that the difference of pixel distance didn't differ much from each other even the camera is tilted by 24 degree.

Case 2 cannot show whether Method 2 was working properly or not since the output of Method 2 was being compared with another set of false result, although it should work theoretically. Nonetheless, Method 2 still provide the output less than 5% of error as of Case 1.

4.5.3 Case 3

For Case 3, the camera was placed approximately 40° to the surface of culvert. Errors from Method 1 escalated to about 20% whilst Method 2 maintained the error level of about 3%. This showed that although the camera is angled, providing the reference values were correct, Method 2 was still able to retrieve the real-world distance. It can be concluded that Method 2 worked correctly since Method 1 failed to measure distance.

Both methods worked as expected in terms of theories. Hence Case 3 was a successful experiment.

4.6 Summary

The vision based distance measurement for in-pipe robotic crawler was a success based on Case 1 and Case 3 studies. It can measure the single axis distance within 95% or higher percentage of accuracy. It was believed that the errors can be further reduced by using more advanced statistical approaches such as normality check. Method 2 in Case 2 study can be considered as a successful one since the output error was less than 5% which was quite similar to the rest of output of case studies, although the output of Method 1 was considered as false negative output.

The best resolution of measurement achievable in this project is when the camera was perpendicular to the surface of culvert, which is 0.3mm/pixel. However, due to the averaging of data set, more accurate result can be achieved, resulting average cumulative error as low as 2.735% was achieved as presented in Case 3 study.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

This project has three objectives. The main objective is to design vision based distance measurement system with monocular vision camera. A program was developed to measure distance travelled by in-pipe robot in culvert. It was capable of working with culvert with different diameter as well as of different camera angle (Method 2 only). The program is cross-platform which can be run on wide range of OS like Linux and Windows.

The second objective is to measure the performance of vision based distance measurement. Three experiments has been carried out, each with ten trials. The outputs were compared with true measurement with metre rule attached to the inner surface of culvert and been recorded in the video recordings. When correct method was used, it can measure the distance travelled with less than 5% of error. In most cases, the percentage error will increase with increasing average speed of the robot.

The last objective is to compare the performance of the system of different methods. Method 1 only work well in perpendicular camera. Whilst performance in 24 degree camera tilt, it was believed to be an error since Method 1 is theoretically not supposed to be functional. In 40 degree camera tilt, the Method 1 gave huge distance estimation error which did match the expectation. Conversely, Method 2 did well in all experiments as expected.

5.2 Future works

Due to the current hardware limitation of the camera, real time distance measurement is still feasible but is not recommended due to extremely low frame rate. The image frames must be of high resolution in order to capture details and to perform feature tracking, however not all area of image is being used for distance measurement, causing unnecessary low image frame rate due to high data bandwidth, affecting the accuracy of real time measurement. In order to achieve high frame rate and high resolution image that are needed to do real time measurement, only a region of sensor is required instead of full sensor size which cannot be done by PiCamera. Better camera hardware and embedded platform are suggested so that the camera is of higher customizability and realize the real time distance measurement system.

Nowadays, since the performance of camera and computational power of consumer smartphone are exploding, especially smartphone, it can be tested to run on consumer smartphone. If it can work on smartphone, it will further reduce the initial cost of the project. Since each smartphone's camera has different specification, more time is required to develop the application that can fit with different camera specifications. Since most smartphone nowadays come with IMU like accelerometer, the camera angle can be measured more precisely.

REFERENCES

[1] S. Shirmohammadi and A. Ferrero, "Camera as the instrument: the rising trend of vision based measurement," *IEEE Instrum. Meas. Mag.*, vol. 17, no. 3, pp. 41–47, 2014.

[2] R. Sun, X. Zhuang, C. Wu, G. Zhao, and K. Zhang, "The estimation of vehicle speed and stopping distance by pedestrians crossing streets in a naturalistic traffic environment," *Transp. Res. Part F Traffic Psychol. Behav.*, vol. 30, pp. 97–106, Apr. 2015.

[3] E. Engineering, H. Province, and H. Province, "Study on Particle Image Velocimetry Technique in the Surface Flow Field of River Model," pp. 968–972, 2013.

[4] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the Mars Exploration Rovers," *J. F. Robot.*, vol. 24, no. 3, pp. 169–186, 2007.

[5] S. S. Beauchemin and J. L. Barron, "The computation of optical flow," *ACM Comput. Surv.*, vol. 27, no. 3, pp. 433–466, 1995.

[6] J. Campbell, R. Sukthankar, I. Nourbakhsh, and A. Pahwa, "A robust visual odometry and precipice detection system using consumer-grade monocular vision," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2005, pp. 3421–3427, 2005.

[7] H. H. Aghdam, H. A. Kadir, and M. R. Arshad, "Localizing Pipe Inspection Robot using Visual Odometry," in *2014 IEEE International Conference on Control System, Computing and Engineering*, 2014, no. November, pp. 28–30.

[8] H. A. Kadir, M. R. Arshad, H. H. Aghdam, and M. Zaman, "Monocular Visual Odometry for In-pipe Inspection Robot," *J. Teknol.*, vol. 9, pp. 31–39, 2015.

[9] Z. Said, K. Sundaraj, and M. N. A. Wahab, "Depth Estimation for a Mobile Platform Using Monocular Vision," vol. 41, no. Iris, pp. 945–950, 2012.

[10] I. U. Jan and N. Iqbal, "A new technique for geometry based visual depth estimation for uncalibrated camera," *2009 Int. Conf. Emerg. Technol. ICET 2009*, pp. 213–218, 2009.

[11] P. Hansen, H. Alismail, P. Rander, and B. Browning, "Monocular visual odometry for robot localization in LNG pipes," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 3111–3116, 2011.

[12] M. Dille, B. Grocholsky, and S. Singh, "Outdoor Downward-facing Optical Flow Odometry with Commodity Sensors," *F. Serv. Robot.*, no. July, pp. 1–10, 2009.

[13] M. Andalibi, L. L. Hoberock, and H. Mohamadipanah, "Effects of texture addition on optical flow performance in images with poor texture," *Image Vis.*

Comput., vol. 40, pp. 1–15, 2015.

[14] "OpenCV Optical Flow," 2015. [Online]. Available: http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html.

[15] T. Holyoake, "Benchmarking the Raspberry Pi 2," 2015. [Online]. Available: http://www.tenpencepiece.net/blog/2015/02/07/benchmarking-the-raspberry-pi-2/.

[16] "Raspberry Pi 2 Model B," 2008. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[17] "Camera Module Hardware Specification," *Raspberry Pi Foundation*, 2016. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/README.md.

[18] M. Berman and T.-L. Americas, "How to safely power LEDs," 2011. [Online]. Available: https://www.newark.com/wcsstore/ExtendedSitesCatalogAssetStore/cms/asset/images/americas/common/storefront/tdk_lambda/Safely-Power-LEDs.pdf.

[19] djbbenn, "Thermal Paste ad How To Use It," 2006. [Online]. Available: https://www.techpowerup.com/articles/overclocking/134.

[20] "PythonOverview," 2016. [Online]. Available: http://www.tutorialspoint.com/python/python_overview.htm.

[21] "Numpy for Matlab Users," 2015. [Online]. Available: https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html.

[22] D. Hughes, "PiCamera API Reference," 2013. [Online]. Available: http://picamera.readthedocs.io/en/release-0.8/api.html#classes.

[23] "CCD Binning," 2015. [Online]. Available: http://www.andor.com/learning-academy/ccd-binning-what-does-binning-mean.

[24] "Camera Hardware," 2013. [Online]. Available: http://picamera.readthedocs.io/en/release-1.10/fov.html.

APPENDIX

Python Source Code

```
import numpy as np
import math
import cv2
import time
from matplotlib import pyplot as plt
#from matplotlib import pyplot as plt
#from picamera.array import PiRGBArray
#from picamera import PiCamera

lk_params = dict( winSize  = (15, 15),
                  maxLevel = 2,
                  criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))

feature_params = dict( maxCorners = 20,
                       qualityLevel = 0.3,
                       minDistance =7,
                       blockSize = 7 )

class App:
    def __init__(self):
        self.track_len = 10
        self.detect_interval = 5
        self.tracks = []
        self.cam = cv2.VideoCapture("C:/Year4/EEM449 Final Year Project/Videos/experiment5c.h264") #PiCamera()
        #self.cap = cv2.VideoCapture(0)
        #self.cam.set(3,1280)
        #self.cam.set(4,720)
        #self.cam.resolution = (640, 480)
        #self.cam.framerate = 32
        #self.rawCapture = PiRGBArray(self.cam, size=(640, 480))
        self.frame_idx = 0
        # allow the camera to warmup
        time.sleep(0.1)
        # initialize variables
        self.frame_skip=1
        self.first_pic=0
        self.frame_count=0
        self.absolute_x=0
        self.absolute_y=0
        self.absolute_xx=0
        self.absolute_yy=0
        self.carWid=0.151
        self.pipeDia=0.3
        self.camOffset=0.06#54#6
        self.VFOV=math.radians(37.16)
        self.HFOV=math.radians(53.50)
        self.inclination=math.radians(90-40)
        #Calcluations of constants for 1st method
        self.WorkDis=self.pipeDia/2 + math.sqrt(4*(self.pipeDia/2)**2-self.carWid**2)/2 - self.camOffset
        self.resolution=(2*self.WorkDis*math.tan(self.VFOV/2))/720

    #Function for 2nd method (if video rotated for 90 or 270 degree)
    '''def convert_x(self,point,width,cropsize):
        beta=math.atan(2*((point+width/2-cropsize)/width)*math.tan(self.VFOV/2))
        x=self.WorkDis/math.tan(self.inclination+beta)
        return x'''
    #Function for 2nd method (if video rotated for 0 or 180 degree)
    def convert_y(self,point,height):
        beta=math.atan(2*(point/height)*math.tan(self.VFOV/2))
        y=self.WorkDis/math.tan(self.inclination+beta)
        return y

    def run(self):
        ############################## Feature tracking starts here ########################
        while 1:
        #for frame in self.cam.capture_continuous(self.rawCapture, format="bgr", use_video_port=True):
                # grab the raw NumPy array representing the image, then initialize the timestamp
                # and occupied/unoccupied text
            global skip_count
            (grabbed, image) = self.cam.read()
            if not grabbed:
                break
            #image = frame.array
```

```
skip_count = (skip_count+1)%self.frame_skip
self.frame_count+=self.frame_count
if skip_count!=0:
   continue
height, width,channel = image.shape
size=180
frame_crop = image[(height/2):(height/2+size*2),(width/2-size):(width/2+size)]
if self.first_pic==0:
   #cv2.line(frame_crop,(size,0),(size,size*2),(255,0,0),2)
   cv2.line(image,(0,height/2),(width,height/2),(255,0,0),2)
   #cv2.imshow('First', frame_crop)
   print self.WorkDis
   cv2.imshow('First', image)
   self.first_pic=1
frame_gray = cv2.cvtColor(frame_crop, cv2.COLOR_BGR2GRAY)
vis = frame_crop.copy()

if len(self.tracks) > 0:
   img0, img1 = self.prev_gray, frame_gray
   p0 = np.float32([tr[-1] for tr in self.tracks]).reshape(-1, 1, 2)
   p1, st, err = cv2.calcOpticalFlowPyrLK(img0, img1, p0, None, **lk_params)
   p0r, st, err = cv2.calcOpticalFlowPyrLK(img1, img0, p1, None, **lk_params)
   d = abs(p0-p0r).reshape(-1, 2).max(-1)
   good = d < 1
   new_tracks = []
   for tr, (x, y), good_flag in zip(self.tracks, p1.reshape(-1, 2), good):
      if not good_flag:
         continue
      tr.append((x, y))
      if len(tr) > self.track_len:
         del tr[0]
      new_tracks.append(tr)
      cv2.circle(vis, (x, y), 2, (0, 255, 0), -1)
   self.tracks = new_tracks
   cv2.polylines(vis, [np.int32(tr) for tr in self.tracks], False, (0, 255, 0))
   cv2.putText(vis, 'track count: %d' % len(self.tracks),(10, 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)

if self.frame_idx % self.detect_interval == 0:
   mask = np.zeros_like(frame_gray)
   mask[:] = 255
   for x, y in [np.int32(tr[-1]) for tr in self.tracks]:
      cv2.circle(mask, (x, y), 5, 0, -1)
   p = cv2.goodFeaturesToTrack(frame_gray, mask = mask, **feature_params)
   if p is not None:
      for x, y in np.float32(p).reshape(-1, 2):
         self.tracks.append([(x, y)])


self.frame_idx += 1
self.prev_gray = frame_gray
#cv2.imshow('lk_track', vis)

###################### measurement of angle and distance starts here ###################

deg=np.zeros(1)
delta_x=np.zeros(1)
delta_y=np.zeros(1)
#converted_x=np.zeros(1)
converted_y=np.zeros(1)
for tracks in range (0,(len(self.tracks)-1)):
   test=np.squeeze(np.asarray(self.tracks[tracks:tracks+1]))  #make one of the track point as an array, and reduce its dimension
   if len(test)<=2:                                 #check to prevent error,check not to use new track point
      continue
   #if(test[-1,0]-test[0,0])== 0:
   #   continue

   delta_x=np.append(delta_x,(test[-1,0]-test[-2,0]))
   delta_y=np.append(delta_y,(test[-1,1]-test[-2,1]))

   ########## Method 2 by retriving the true distance of the point ##################
   #x1=self.convert_x(test[-1,0],width,size)
   #x2=self.convert_x(test[-2,0],width,size)
   #converted_x=np.append(converted_x,x2-x1)

   y1=self.convert_y(test[-1,1],height)
   y2=self.convert_y(test[-2,1],height)
```

```
            converted_y=np.append(converted_y,y2-y1)

        #self.absolute_x+=np.average(delta_x)*self.resolution
        self.absolute_y+=np.average(delta_y)*self.resolution
        #self.absolute_xx+=np.average(converted_x)
        self.absolute_yy+=np.average(converted_y)
        cv2.putText(vis, 'Distance y: %.4f m' % self.absolute_y,(10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)
        #cv2.putText(vis, 'Distance: %.4f m' % self.absolute_xx,(10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)
        cv2.putText(vis, 'Distance yy: %.4f m' % self.absolute_yy,(10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)

        #plt.plot(self.frame_count,self.absolute_y)

        print self.absolute_y, self.absolute_yy
        #print delta_y
        #print converted_y
        cv2.imshow('lk_track', vis)
        cv2.line(image,(0,height/2),(width,height/2),(255,0,0),2)
        cv2.imshow('Now', image)

        # clear the stream in preparation for the next frame
        #self.rawCapture.truncate(0)"'

        ch = 0xFF & cv2.waitKey(1)
        if ch == 27 or ch ==ord("q"):
            break
        if ch ==ord("r"):
            self.absolute_x=self.absolute_y=self.absolute_xx=0

def main():
    #import sys
    #try: video_src = sys.argv[1]
    #except: video_src = 0

    #print __doc__
    #App().distance()
    global skip_count
    skip_count=0
    App().run()
    cv2.destroyAllWindows()

if __name__ == '__main__':
    main()
```