Stereo Vision System for the High Speed Vehicle Project

Xuyan Rosalind WANG
SID: 9831649

This thesis has been undertaken as part of the course work required for the degree of Bachelor of Engineering in Mechatronics Engineering

The University of Sydney
Faculty of Engineering
School of Aerospace, Mechanical and Mechatronics Engineering

Abstract

Autonomous vehicles are very useful in applications where it would be difficult or dangerous for human operations. The High Speed Vehicle (HSV) Project is an ongoing successful project within the Australian Centre for Field Robotics since 1997.

For an autonomous vehicle to operate without any or very little input from human it needs to have a fully integrated system involving sensors, controls and actuators. The sensor system is the eyes of the vehicles, it provides the vehicle with details of its environment. A good sensor system would avoid accidents on board, and allow the vehicle to operate safely.

For my thesis work, I have researched into stereo vision for the vehicle. Stereo vision is the method by which most humans see the world around in three dimensions: width, height and depth. In my project, I try to emulate this natural process and allow the HSV to 'see' in 3D.

A rig is built for the stereo system where the cameras can sit, other hardware for the project such as the cameras, lenses and frame grabber were either purchased or are already a part of the HSV system. Various methods of image rendering processes were researched and implemented, these include edge and corner detectors, camera calibration algorithms, and image filtering algorithms. The conjugate pairs of points from the images are matched and thus a three dimensional disparity map of the surrounding environment can be built.

Acknowledgements

- To Associate Professor Eduardo Nebot, for giving me the opportunity to work on the project.
- To Mr. Trevor Fitzgibbons, who patiently helped me throughout the project with all the theories and programming.
- To Mr. Hasso Nibbe and Mr. Brian Scaysbrook from the mechanical workshop for building the test rig for the stereo system.
- To everyone involved in the HSV project, who provided constant support and encouragement. Specifically, Mr. Juan Nieto, Mr. Jose Guivant for your guidance.
- To Mr. Richard Wolf for providing 'slave labour' whenever I ask for help.
- To all the people sitting around the 'undergraduate thesis corner' at the ACFR for support and help, motivation, entertainment, and distraction: Astrid, Ian, Jean, Peter, Richard, Scott, and Sharon.
- To Mand, whose great stories have kept me sane during the crazy periods when various programs and algorithms refused to work. Thanks for your support across the pond, and the 'mother-henning' of "get some sleep".
- Last but definitely not least, to my parents who never questioned what I choose as a course for university study, who know to stay away when I lock myself in my room to work, and who are always there to provide constant support.

Statement of Student Contribution

- I conducted the necessary background research in machine vision, especially Stereo Vision.
- I developed the design for the stereo vision test rig, which was used as an experimental tool for the HSV.
- I researched into various lenses necessary for the existing cameras for the Stereo Vision System.
- I researched into the various image rendering techniques and wrote the respective algorithms for the detection of edges and corners.
- I researched into, wrote and implemented the various algorithms for Fundamental Matrix calculation.
- I planned and ran the experiments for the determination of the accuracies for the stereo vision system.

The above statement represents an accurate summary of the student's contribution

X. Rosalind Wang **A/Prof. Eduardo Nebot – Supervisor**

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The High Speed Vehicle (HSV) Project is an ongoing successful project within the Australian Centre for Field Robotics (ACFR) at the University of Sydney. The aim of the project is to develop an fully autonomous land vehicle to operate in an unknown environment. In 2002, four undergraduate thesis are undertaken for the project. My thesis is to investigate the practicality of Stereo Vision for use in the project.

1.1 HSV Background

The HSV project has been a successful ongoing project conducted by the ACFR since 1997 [16]. The primary objective of past undergraduate students and researchers in the project has been the development of navigation and control algorithms to enable autonomously autonomous operations of land vehicle operating in unknown environments.

The aim of the HSV project is to develop technologies for the automation of land vehicles operating, at 'high speed' (speed up to 90 Km/h) in a variety of 'real' environments. The experimental prototype consists of a Holden S-series Commodore Utility car, known to the group as the 'Ute', that is retrofitted with a large number of sensors, actuators and data logging and control system. The Ute is not the prototype for the HSV, but rather it's a test bed for the various hardware and software that are under

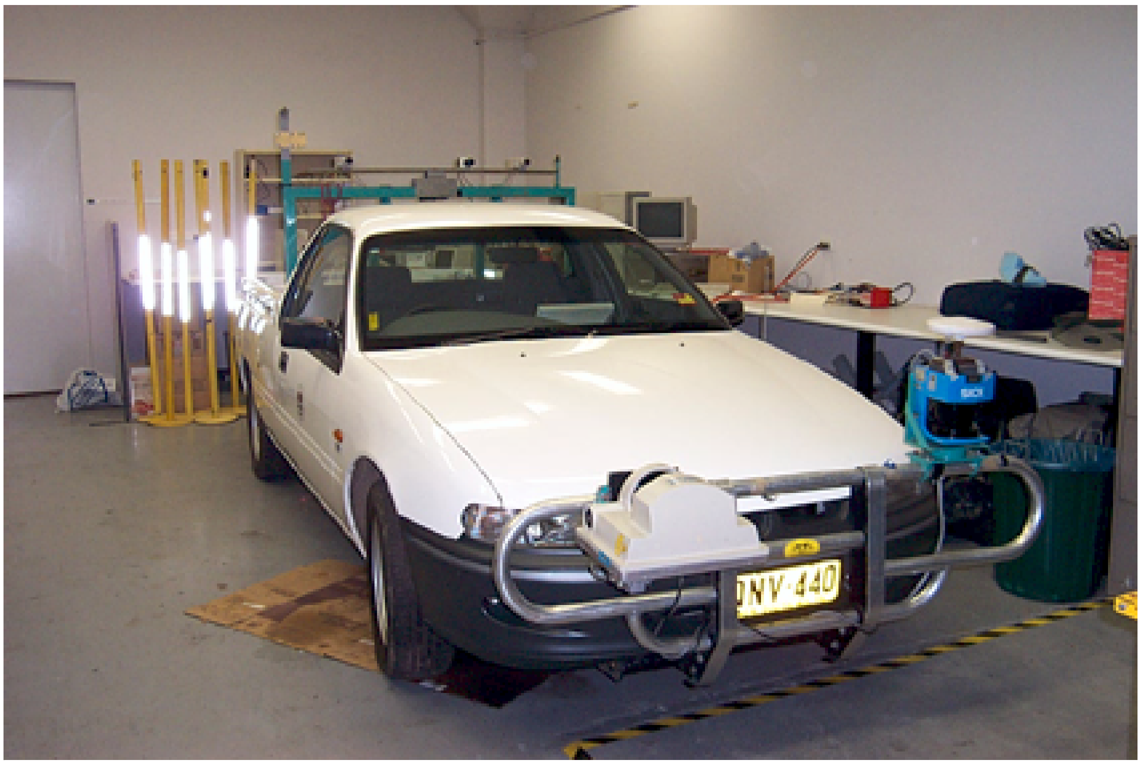


Figure 1.1: The 'Ute' for the HSV project

the development for the project.

1.1.1 Existing Sensors Onboard

Since the purpose of this thesis to develop and evaluate the usefulness of another sensor, it is important to review the existing sensors on board the Ute at the moment. In order for an automobile such as the HSV to autonomously navigate through the external environment without obstruction, it must be provided continually with sensory information advising it of its location, direction of motion, speed and of the current conditions of certain points in the environment during its locomotion.

The Ute is equipped with external sensors such as GPS, and laser giving the vehicle information of the environmental terrain and the location for the vehicle. At the same time internal sensors such as INS, LVDT, and encoders provide for the speed and direction of motion of the vehicle [16]. This entire sensory information, can be used by the HSV in particular, to detect other vehicles, obstacles, and pedestrians with the primary objectives of ensuring that it remains a safe distance away from static or moving objects in its path of motion.

Differential GPS Unit

The *Global Positioning System*, or GPS is a series of 24 satellites orbiting the Earth, in known orbits. When a GPS unit is in communication with four or more satellites, it is able to receive its global position and determine where and what heading it's travelling. The system on the HSV is called a differential GPS, which will return a position accuracy of approximately one meter.

The information gathered from such a system provides a global position in the North, East and Down (vertical elevation), as well as the units velocities in these frames. These are the main six data values used, but other information on the GPS system is also provided, such as the number of satellites transmitting to the unit and the variance of the individual signals.

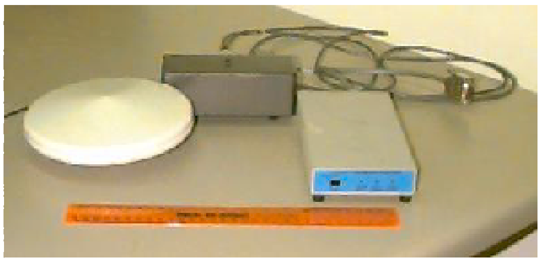


Figure 1.2: The Differential GPS unit for the Ute

SICK Range Bearing Laser

The SICK range bearing *laser* is located at the front of the HSV and is able to provide a picture of the environment around the vehicle. The basis of the bearing laser is that it emits a single infrared laser pulse which is reflected back from any object within its vicinity. The time that the beam takes to return to the unit is measured, and knowing the speed of light, the distance the object is away from the bearing laser can be calculated. An image of the environment is then constructed by rotating the laser and taking samples at known angular intervals.

For this SICK Bearing laser range samples are taken at every 0.5°, for a range of 180° and a distance of up to 80m. The data given for each sweep of the sensor is in the format of a range reading for every sample taken, thus a total of 361 range readings. This model of laser also provides the intensity of the return signal, which is presented in the same format. This feature is useful for identifying markers and reflective objects.

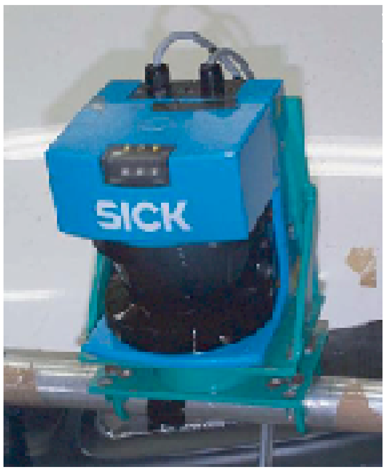


Figure 1.3: The SICK laser unit for the Ute.

Compass

The *compass* is an alternative source of vehicle heading information, measuring the information relative to Earth's magnetic north. The compass used is TCM2, which is an electronic compass with an ability to eliminate mechanical gimbal.

Inertial Navigation System

The *inertial navigation system* (INS) consists of gyros, accelerometers and inclinometers arranged along 3 perpendicular axes. The three types of sensors measures angular velocity, acceleration, and inclination respectively. The INS provides data in all six degrees of freedom, and is used to provide dead-reckoning estimation of the HSVs position, through integration of the acceleration and angular velocity to provide position and heading.

LVDT

An *LVDT* is a measurement device that uses the electromagnetic force that is induced in the movement of a ferrous core through two electromagnetic coils. The LVDT uses this principle to measure linear movement, by attaching the two ends to the linear distance needed to be measured. For the HSV, this has been attached to the steering system of the Ute, and returns a value of the angle the vehicles steering has shifted

Wheel Encoder

The ROD-430 wheel *encoders* is an incremental rotary velocity encoders from Heidenhain. The encoders operate on the principle of photo electrically scanning very fine grating with a line counts between 50 to 5000. Shaft attaching to the wheel encoders can travel up to 12 000 rpm. Output signal for this particular model is a HTL square-wave signal which meant that an output signals are square-wave signals incorporating a circuit that digitizes sinusoidal scanning signals, providing two 90 deg phase-shifted pulse trains and a reference pulse. The encoders is power by the 12 Vdc from the fuse box.

1.1.2 Past Work on Vision

Machine vision is the process of acquiring images using camera, processing the images, recognising the features in the images and thus allow the robots to function in the environment much like what human does. The topic, however, it extensive and challenging in many ways such as the amount of computer power required to interpret the vast amounts of image data, and to perform the tasks quickly and reliably. It is due to these problems that vision is an area that is looked into later on in any such projects. For the HSV, there are a couple of previous thesis projects that investigate the use of vision as a sensor.

Panoramic Vision System

Panoramic vision is a recently established method of image acquisition in which the global scene (360°) surrounding an image acquisition device, such as a camera can be viewed. The imaging system was first investigated in 1999 by Dimo [10] for the HSV project to function as an additional external sensor for the HSV with the ability to attain global information about the surroundings.

The system on board achieves the viewing of the 360 degree environment by using a reflective mirror to transform the image onto the camera lens. The camera which was used in field experiment is an ELMO TSP481 PAL CCD Colour Camera. The camera is a 12 Vdc power input and medium resolution with backlight compensation capabilities ie saturation resistance of the sunlight in an image. Thus the camera can detect high and low intensity images in the outdoors and most importantly natural landmarks in outdoor navigation.

The panoramic imaging system set up on the HSV is able to effectively view the entire scene surrounding the vehicle instantaneously, in a single image frame without distortion, and without the use of moving parts. The combination of these image frames in series makes it possible for the image system to operate in real time. Secondly, the imaging system serves the purpose of being able to appropriately depict and identify all the known landmarks panoramically surrounding the vehicle in each of it's images and identify the bearing to these landmarks in relation to a reference on the vehicle. This facilitates a starting point for the global homing or localisation of the vehicle through vision.

Path Following

In 2001, Mactier [19] investigated the problem of autonomous road following in an urban environment using a single CCD camera. This involves the autonomous localisation of the road, the determination of the relative position between the vehicle and the road, and the determination of a suitable path for the vehicle to follow to remain on the road.

This thesis details the design and implementation of an approach to perform autonomous road following using a single CCD camera. It can be shown, that provided distinct features run parallel to the road scene we can determine the position of the lane boundaries and the curvature of the road ahead of the vehicle.

1.2 Thesis Objective

For the HSV to operate in an autonomous fashion, its sensor system requires a three-dimensional (3D) map of the environment in front of the vehicle to navigate safely and avoid collisions. This map must include position and motion estimates of the relevant traffic participants and potential obstacles. Stereo vision is the application of two video cameras to construct 3D maps through the analysis of the images returned. This application mirrors those of other external sensors, such as the radar and laser, where the system will localise objects within its field of view, build a map of its surrounding and thus control the vehicle to safely navigate through the environment. Unlike the radar and laser systems, stereo vision is a passive system, which means it does not require the emission of signals to 'see' obstacles. This gives the system an advantage over the active sensors in that there is little potential for interference with the environment. Furthermore, it provides a much higher spatial resolution than active sensors [12].

Much research has gone into the application of stereo vision in autonomous vehicles around the world. Whitehorn et al. [34] worked in applying stereo vision for the enhancement of safety and productivity in the operation of a load-haul-dump vehicle in underground mining. Nguyen and Graefe [25] worked on an approach to realise learning calibration-free stereo vision-based robot for manipulating objects. Gavrila et al. [12] uses stereo vision in their approach to build a sensor system for driver assistance in urban vehicles. And Broggi et al. [4] investigated into a calibration method for on-board cameras used on autonomous vehicles.

The objectives of my thesis are:

- Build a stereoscopic system for the HSV,
- Develop algorithms to analyse the image returned,
- Build a 3D map from the analysis, and
- Investigate into its uses and practicality.

1.3 Thesis Structure

The thesis is organised in the following way:

Chapter 1, this chapter is the **Introduction**. Here I have done a quick review of the High Speed Vehicle project's background, its existing systems on board and a review of the past work done on vision for the project. Furthermore, this chapter also introduces the concept of stereo vision and the objective of this thesis.

Chapter 2 is a review of **Stereo Vision**. It includes the definition of stereo vision, and its various implications to the human vision system. This chapter also deals with the theoretical aspect of applying what we know of stereo vision's mechanics to uses in machine vision and how we can use it as a sensor.

Chapter 3 is on **Hardware Consideration**. This chapter will discuss the various hardware used for the projects. These are the cameras and their components, the rig for the placement of the cameras on board, and the hardware to capture the images from the camera for analysis. Lastly, the chapter will discuss the method of acquiring the data and the format of the data gathered.

Chapter 4 deals with the various algorithms and methods for **Image Rendering**. Firstly, the methods for calibration of the camera are discussed. Secondly, any data gathered will have intrinsic noises and I'll discuss the methods for dealing with the noise. The next section is the various algorithms investigated for detecting the edges of objects within the camera. Finally, we'll look into the methods for corner detection.

Chapter 5 is **Stereo Matching**, which tackles the problem of match the corresponding objects from the two images. This is one of the hardest problems in machine vision, to find the two points that are the project of one point from an object. We'll look into how to find the most accurate point and how to decide which are the corresponding points. Furthermore, we'll look into methods of speeding up the correlation process, reducing the search parameters by the use of fundamental matrix and essential matrix.

Chapter 6, **Disparity Map and Error Calculation** is the process and result of finding the 3D map and determining the error involved. The Disparity Map is the three-dimensional map of the surrounding area of the vehicle using the data gathered by the stereo cameras. There are however, errors in the coordinates of all the points found. This chapter also deals with calculating these errors theoretically and experimentally.

Chapter 7 is the **Conclusion**. In this chapter we'll review the processes into finding the disparity map from the pair of images from the cameras. The advantages and disadvantages of Stereo Vision for the HSV project will be discussed, and the practicality of the system in real-time. Lastly, we'll look into any further works that can be done for the system.

Chapter 2

Stereo Vision

2.1 Background

Stereo vision is an adaptation of the human vision system into machine vision. There is much evidence that human vision utilises disparity, which is the differences in images of real world scene captured by the eyes. The human eyes are located side-by-side at the front of the head ensuring that both eyes see a view of the same area from a slightly different angle. Thus even though there are plenty of common images taken from the two eyes, each eye would pick up visual information the other doesn't. The differences are important because they encode information that often allows a reconstruction of the three-dimensional structure of the scene from two-dimensional projections [2], [26].

Bibson [14] discussed the nature of visual perception in a dynamic environment. He argued that the visual stimulus is inherently dynamic and that the patterns of change in the stimulus are important sources for the perception of the spatial environment. Gibson described the pattern of optical flow that occur when the observer moves. He argued that along with other important visual phenomena, such as texture gradient and linear perspective, they interact with the kinesthetic "body senses" in the perception of a stable, upright, three-dimensional world.

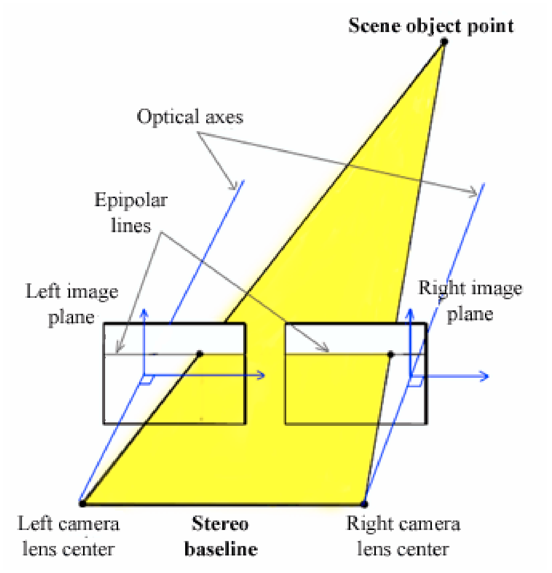
When the two images arrive simultaneously at the brain, they are united into one by matching up the similarities and adding in the small differences. These differences are what allowed us to see the world in 3 spatial dimensions – width, height and depth. It is the added perception of the latter that makes stereo vision so rich and special. It allows us to see where objects are in relation to our own especially whether an object is moving to or away from us; it allows us to see a bit “around” an object without moving the head around; and it allows us to perceive and measure space with our eyes and brain.

Julesz’s experiments with random dot stereograms [18] supported the contention that the human visual system is able to process differences between images. People with normal stereo vision can easily perceive depth in the randomly generated pair of images by measuring the disparity between the shifted dots on the two images.

2.2 Application to Machine Vision

The simplest model of a stereo vision is two identical cameras separated only in the x direction by *a baseline distance* b as shown on figure 2.1, which is what was done for this thesis. The image planes are coplanar in this model. A feature in the scene is viewed by the two cameras at different positions in the image plane. The displacement between the locations of the two features in the image plane is called the *disparity.* The plane passing through the camera centers and the feature point in the scene is called *epipolar plane.* The intersection of the epipolar plane with the image plane defines the *epipolar line.* For the model shown on the figure, every feature in one image will lie on the same row in the second image. In practice, there may be a vertical disparity due to the misregistration of the epipolar lines. Many formulations of binocular stereo algorithms assume zero vertical disparity [17].

In figure 2.1, the scene point $P(x, y, z)$ is observed at points on the two image planes. Without the loss of generality, let us assume that the origin of the coordinate

(a)


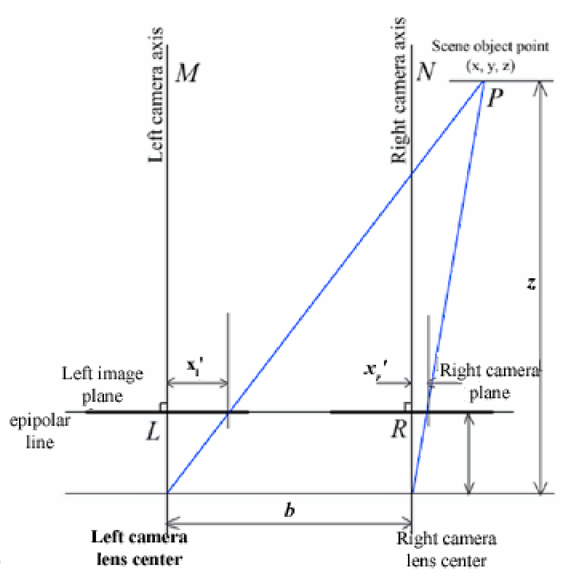
(b)


Figure 2.1: Stereo Matching [17].

system coincides with the left lens centre. It is easily observed that the distances x and x_l', as well as z and f are corresponding sides on similar triangles, thus:

$$\frac{x}{z} = \frac{x_l'}{f} \tag{2.1}$$

Similarly, same thing can be said about the right image, giving:

$$\frac{x-b}{z} = \frac{x_r'}{f} \tag{2.2}$$

where f is the focal length, and combining the equations 2.1 and 2.2 gives:

$$z = \frac{bf}{x_l' - x_r'} \tag{2.3}$$

Thus, the depth at various scene points may be recovered by knowing the disparities of corresponding image points.

Note that due to the discrete nature of the digital images, the disparity values are integers unless special algorithms are used to compute disparities to subpixel accuracy. Thus, for the given set of camera parameters, the accuracy of depth computation for a given scene point is enhanced by the increasing the baseline distance b so that the corresponding disparity is large. Such wide-angle stereopsis methods introduce other problems, however. For instance, when the baseline distance is increased, the fraction of all scene points are seen by both cameras decreases. Furthermore, even those regions that are seen by both cameras are likely to appear different in one image compare to the corresponding regions in the other image due to distortions introduced by perspective projection, making it difficult to identify conjugate pairs.

Many algorithms have been developed to build the disparity map from the stereo imaging technique. The paradigm used by most early algorithms consists of three steps [1]:

1. Detect suitable features in each image.
2. Find corresponding features in the two images.
3. Determine the 3D locations associated with corresponding pairs of image features, and build the map.

Chapter 3

Hardware Consideration

As the project is mainly software based, the hardware requirement is quite simple. Furthermore, most of the visual systems were build and set up a couple of years ago with the panoramic imaging system for the Ute, these includes the cameras, frame grabber and its associated software. The hardware specifically purchased/built for this thesis are the lenses and the stereo rig.

3.1 Cameras

For this project, we are using the Panasonic WV-BP330 series CCD Camera, shown on figure 3.1(a). These are 1/3" Digital Signal Processing B/W CCTV cameras, which delivers a 768×582 pixel picture [27]. The cameras were obtained for a previous thesis for the vision system, here both of them will be used to obtain 3D viewing. See Appendix B.1 for the specifications of the cameras.

Two different lenses were ordered for the previous projects, for the testing of different systems. However, in this case we'll need to use two exact same lenses for the job. The lens on ordering is the Panasonic 2x Variable Focal Lens WV-LZA61/2, shown on figure 3.1(b). The lens has automatic iris, which is a necessity for outdoor use. It also has adjustable focal length and angular field of view [28]. (Appendix B.2 shows the specifications for the lens.) These features are desirable for the project since currently

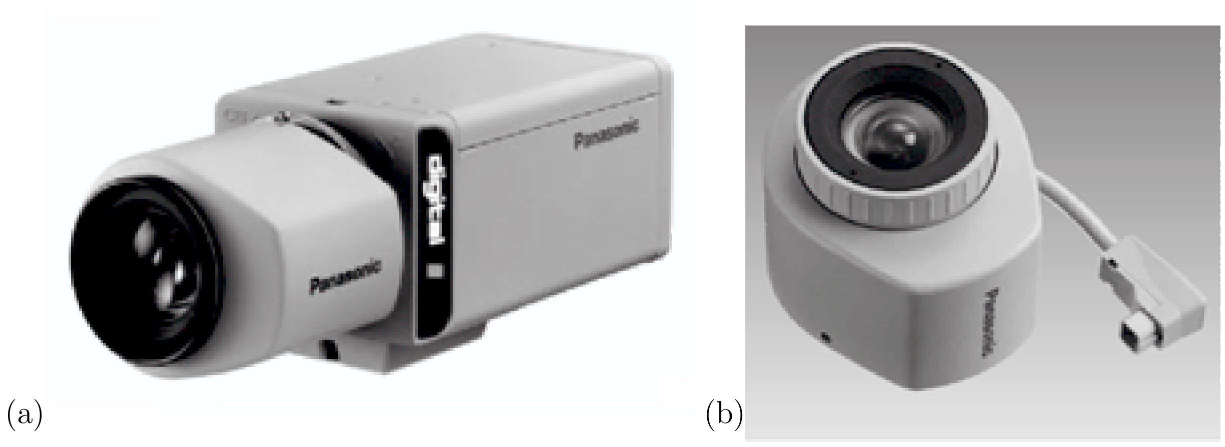


Figure 3.1: The camera [27] and lens [28] used for the project.

we do not know exactly what we need to see. I have chosen to use the same brand as the camera, as it is always recommended in the photography field that the lens and the camera should be from the same manufacture.

3.2 Test Rig

A rig need to be built for holding the cameras on the Ute. Since we don't know the best distance between the cameras for our situation, the rig is to be the width of the Ute, with holes to screw the camera on at 10cm intervals. Figure 2 shows the drawing of the test rig.

The cameras are mounted on the Ute just behind the cab, on the bar at the front of the tray. Figure 3.3 shows the positioning of the cameras on the test rig on the Ute. They are placed at either side of the Ute imitating nature.

3.3 Frame Grabber

The image capture card used in the system is the same card that was used for the panoramic imaging system. It is a standard Matrox Meteror II board as shown on figure 3.4.

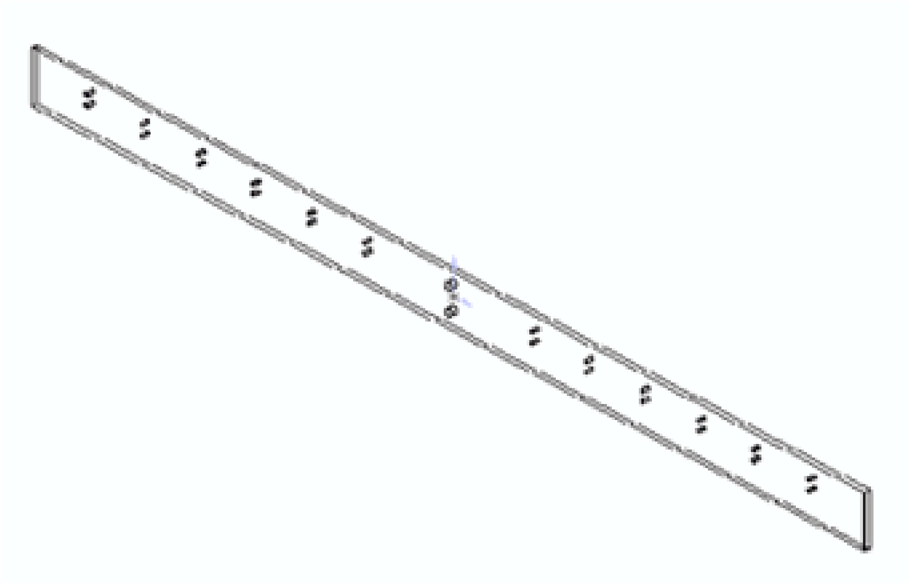


Figure 3.2: The Test Rig

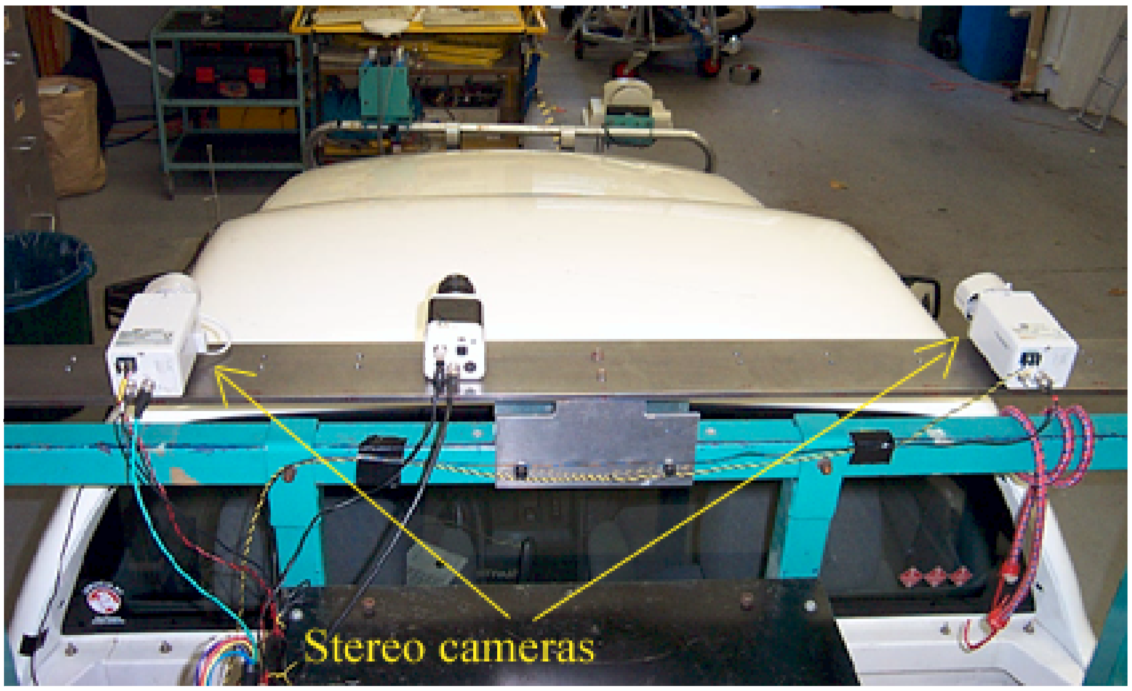


Figure 3.3: The positioning of the cameras for Stereo Vision

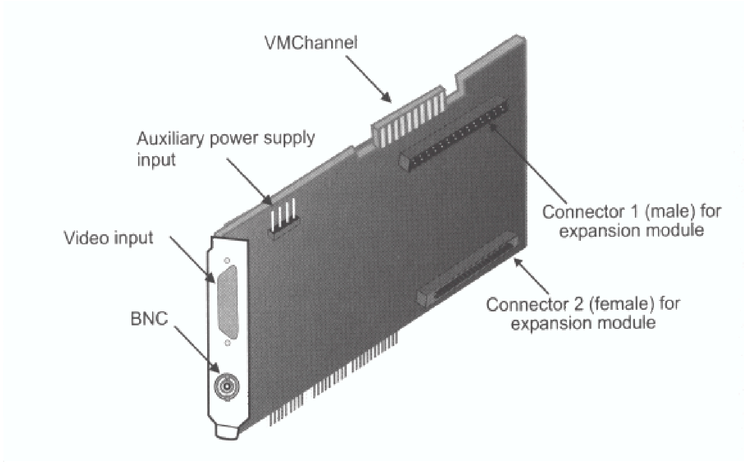

VMChannel
Auxiliary power supply input
Video input
BNC
Connector 1 (male) for expansion module
Connector 2 (female) for expansion module

Figure 3.4: The Matrox Meteor II Frame grabber [20].

The board is a standard monochrome and colour analogue frame grabber [20]. It accepts an external trigger input, and can operate in next valid frame/field mode. The standard board has six connectors, as shown on figure 3.4. The BNC connector is used to receive composite analog video signals from the cameras. Alternatively, Pin 15 (VID_IN1) in the Video Input socket can also be used for the same purpose [10].

As an in-built standard, the board samples or digitises the analogue input signals from the cameras used with a resolution of 768×576 pixels. This means that each of the image frames grabbed by the buffers on the board, are of this default size. However, it was possible to specify in software smaller image frame sizes to be grabbed by the buffers on the board.

The image capture board is located in the PCI bus of the host PC for the system, and allows for the fast transfer of image information between the image buffers located on the board and displayable PC RAM. The layout of the board in figure 3.5 shows that a video multiplexer on the board allows for the analogue video signals of up to four cameras to be connected to the board and sampled individually and synchronously with a high sample rate. This makes the board useful for multiple cameras uses as in the stereo vision system.

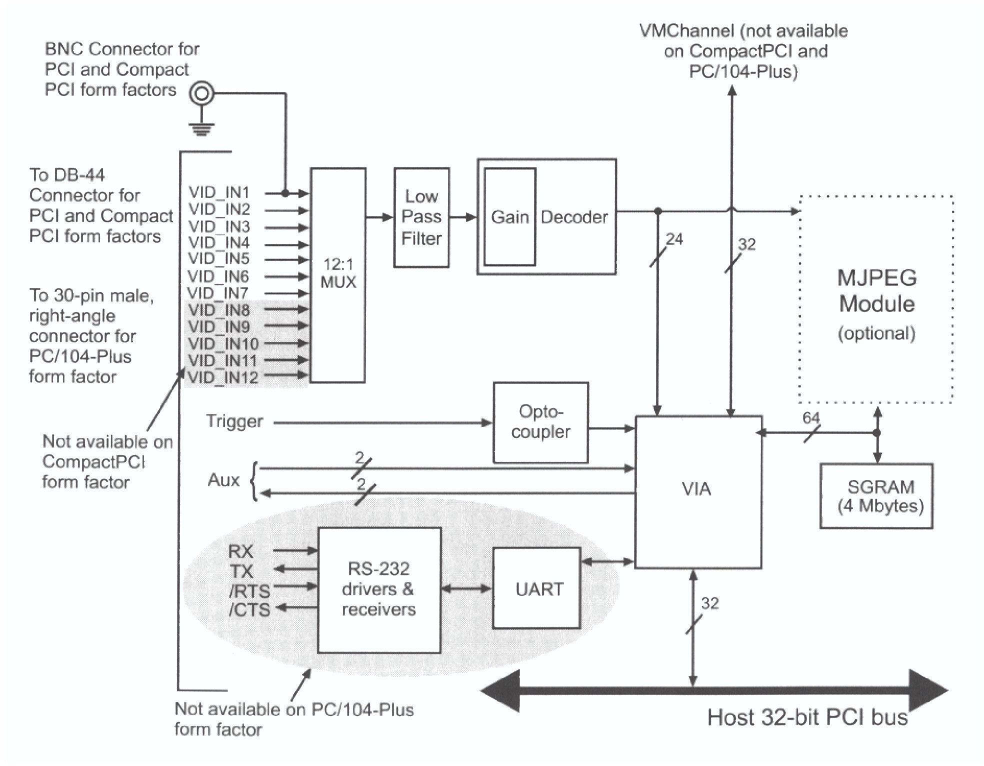

BNC Connector for PCI and Compact PCI form factors
VMChannel (not available on CompactPCI and PC/104-Plus)
To DB-44 Connector for PCI and Compact PCI form factors
To 30-pin male, right-angle connector for PC/104-Plus form factor
VID_IN1
VID_IN2
VID_IN3
VID_IN4
VID_IN5
VID_IN6
VID_IN7
VID_IN8
VID_IN9
VID_IN10
VID_IN11
VID_IN12
12:1 MUX
Low Pass Filter
Gain
Decoder
24
32
MJPEG Module (optional)
Not available on CompactPCI form factor
Trigger
Opto-coupler
64
Aux
2
2
VIA
SGRAM (4 Mbytes)
RX
TX
/RTS
/CTS
RS-232 drivers & receivers
UART
32
Not available on PC/104-Plus form factor
Host 32-bit PCI bus

Figure 3.5: The Matrox Meteor II Frame Grabber board layout [20].

The frame grabber is supported with a library of C functions, which is a part of the Matrox Imaging software products [20]. These are the Matrox Imaging Library (MIL) and its derivatives, one of which is what is used for the system – MIL-Lite. The package allows the capture, process, analyse, transfer, display, and archive of the images. It also contains video signal drivers, which allow for each camera attached to the frame grabber to be configured for image display on the host PC. The frame grabber and the accompanying video signal drivers were particularly chosen as they allow for the displaying of captured images under a real-time operating system such as windows NT on the host PC.

The MIL-Lite package also provides four groups of functions to control the frame grabber, these are shown on figure 3.6. These were used selectively on the host PC in the panoramic imaging system so as to control the following functions:

- The acquisition of image frames from a camera in the system,

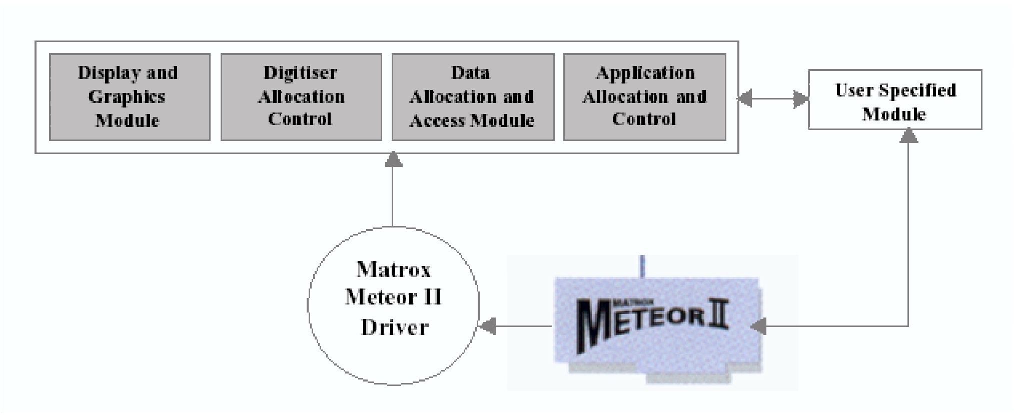

Display and Graphics Module
Digitiser Allocation Control
Data Allocation and Access Module
Application Allocation and Control
User Specified Module
Matrox Meteor II Driver
Matrox Meteor II

Figure 3.6: The MIL-Lite software architecture [10].

- The low level allocation of memory for image buffers on the frame grabber, and
- The display of the contents of the image buffers on an interface on the host PC.

In addition, the software package allowed for the implementation of custom image processing algorithms in C on the host PC, which were able to directly access the image buffers on the board and use the four groups of functions the package provided.

3.4 Data Acquired

The cameras used for the project are black and white for several reasons. Firstly, the data size from a B/W image is only one third the size of colour image. Secondly, as in a B/W image, every pixel is described with one value (in our case a integer between 0 and 255), the two images from the cameras can be combined into one colour image, thus eliminate the need to match data.

Figure 3.8 is a stereo image captured by the camera of the same scene as that of figure 3.7. As seen on the stereo image, the red and green channel of the colour image represents two pictures taken by the cameras. When processing the data, we can separate these images out by just viewing one of the channels as shown on figure 3.9.

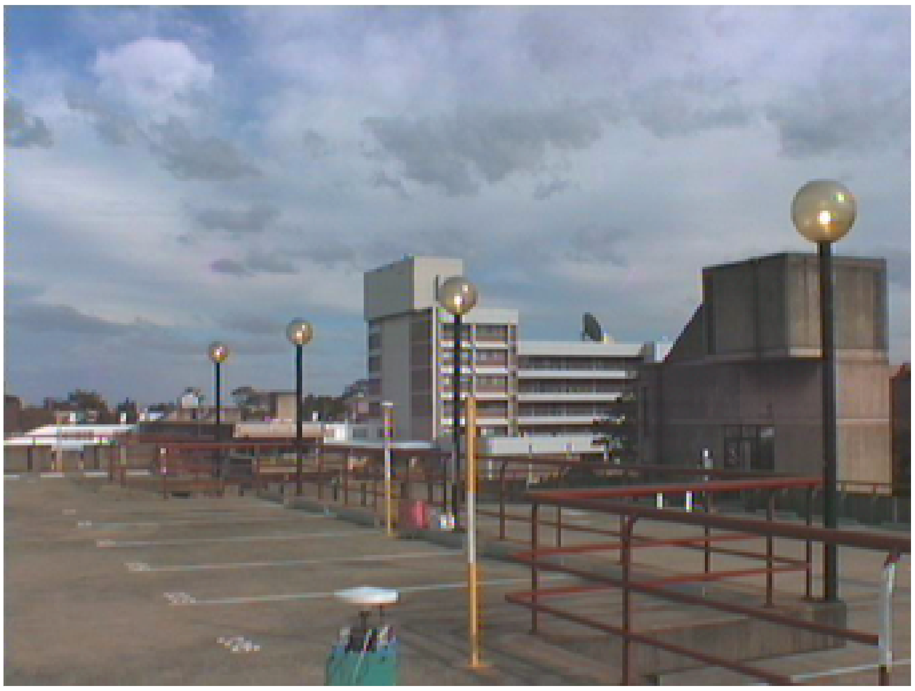


Figure 3.7: A picture taken by the colour camera on board the HSV

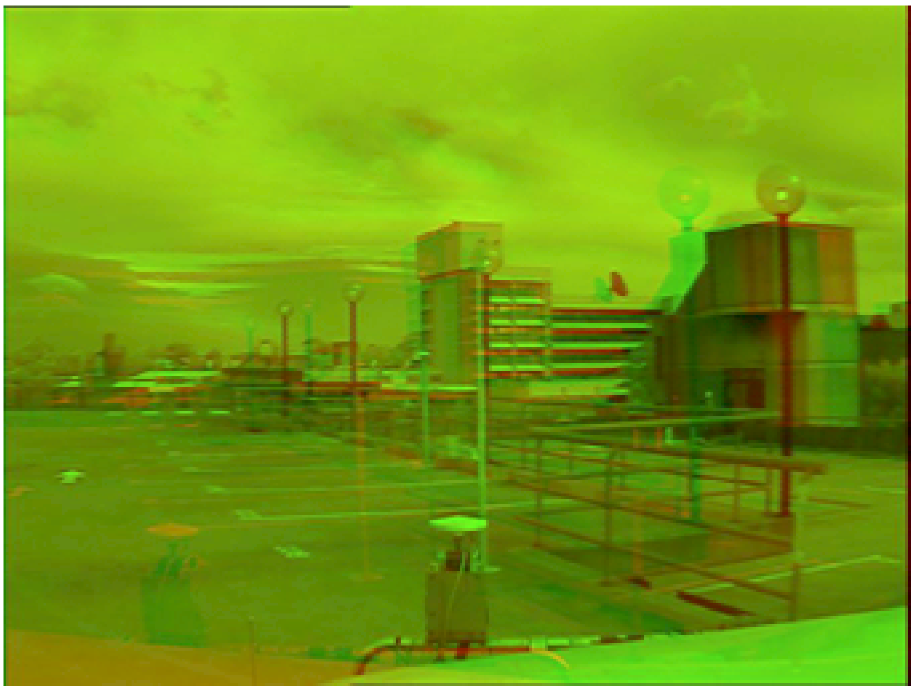
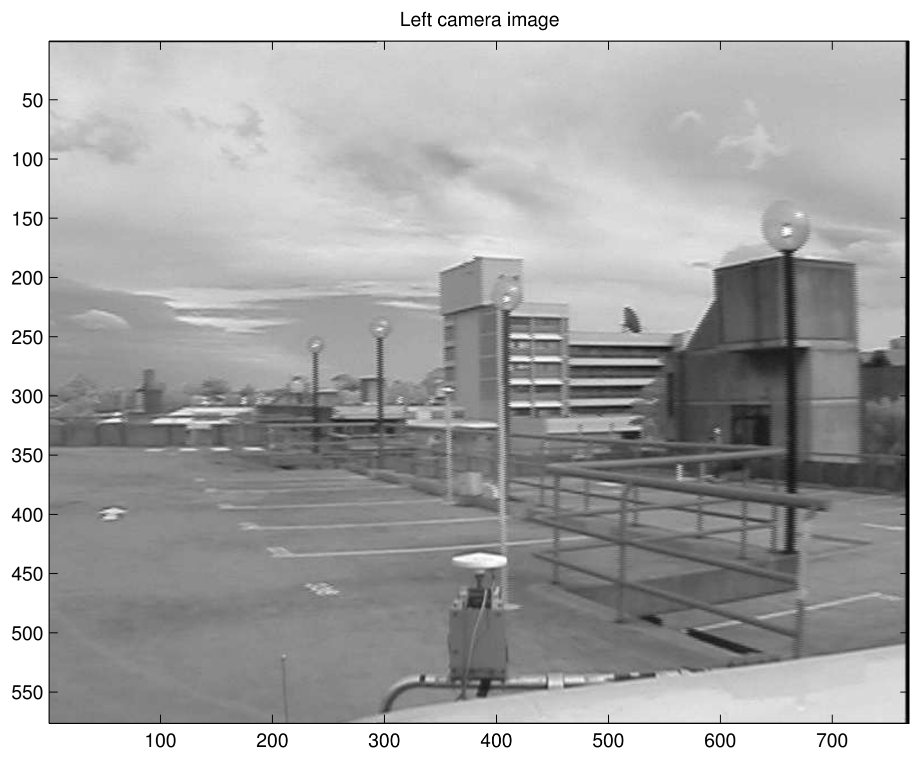


Figure 3.8: The data received by the stereo cameras of the same view as that shown in figure 3.7



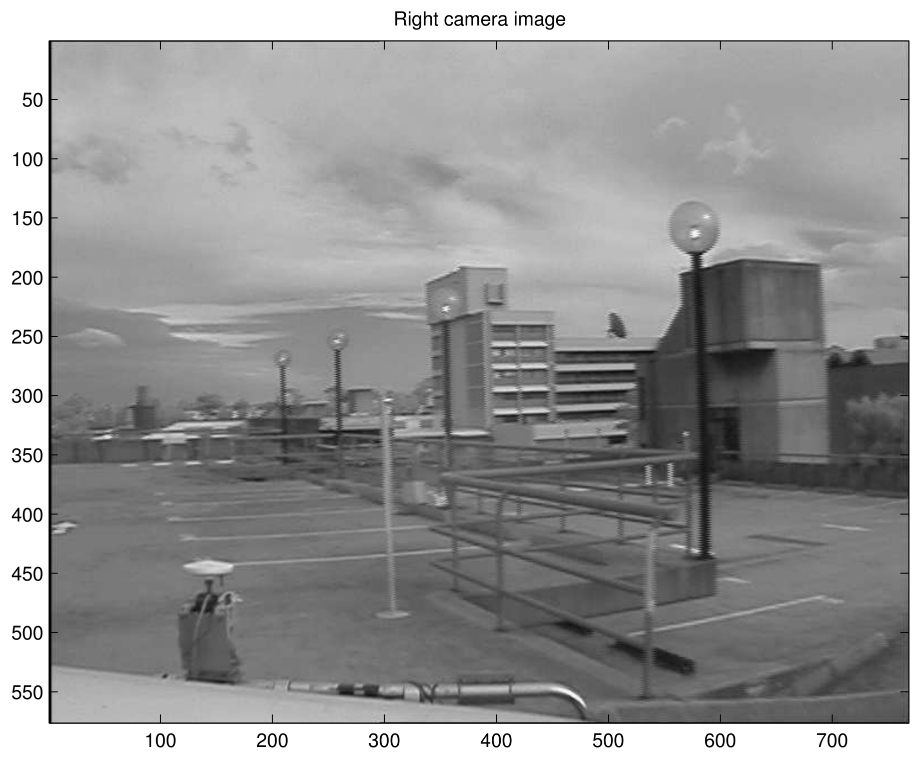


Figure 3.9: The left and right camera images of the stereo image show in 3.8

Chapter 4

Image Rendering

To find the distance to an object, we need to match the object from the left image to that from the right image as discussed in section 2.2. The two points that belong to the some object is called a *conjugate pair*. However, it is virtually impossible to match point by point since the same value of a pixel can be repeated many times even on a single row. Thus we need to find distinguished features on the image, these features would be things like edges and corners.

4.1 Calibration

From figure 2.1, and equations 2.1, 2.2 and 2.3 we can see that there are several variables that we need to know before we can calculate the (x, y, z) coordinates of the objects. These variables are the focal lengths and the image centers of each camera, both of these values are in the x and the y directions.

These calibration values are found using the `calibration` toolbox for MATLAB written by Jean-Yves Bouguet [3]. The package is used with the calibration board, which is a check board like with 12×12 alternating squares of black and white. Each square is $75cm$ in width and height. Figure 4.1 shows the calibration board.

For calibration, sets of images are taken by the cameras (for the best results, the

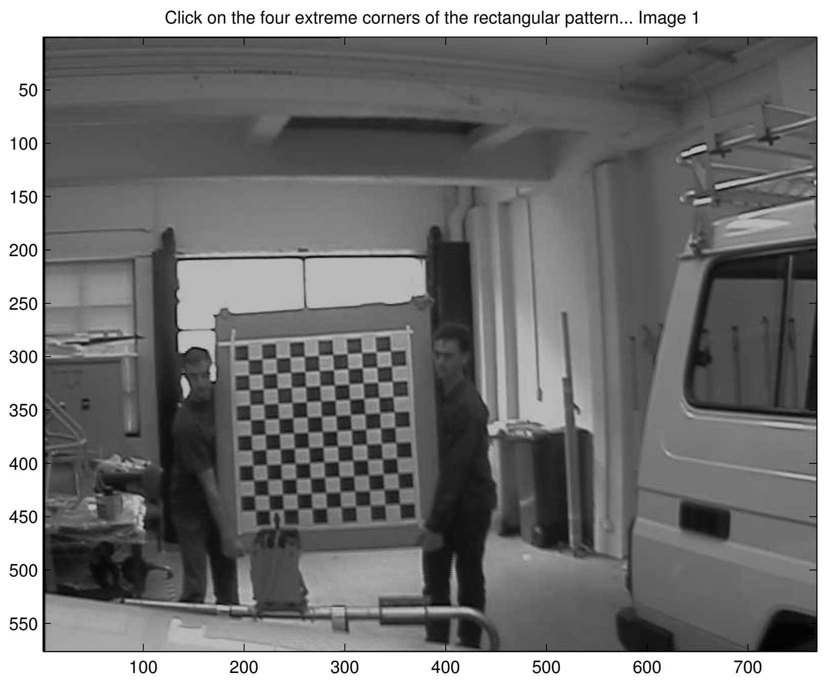

Click on the four extreme corners of the rectangular pattern... Image 1
50
100
150
200
250
300
350
400
450
500
550
100
200
300
400
500
600
700

Figure 4.1: Calibration step 1, original image from one of the cameras, waiting for calibration.

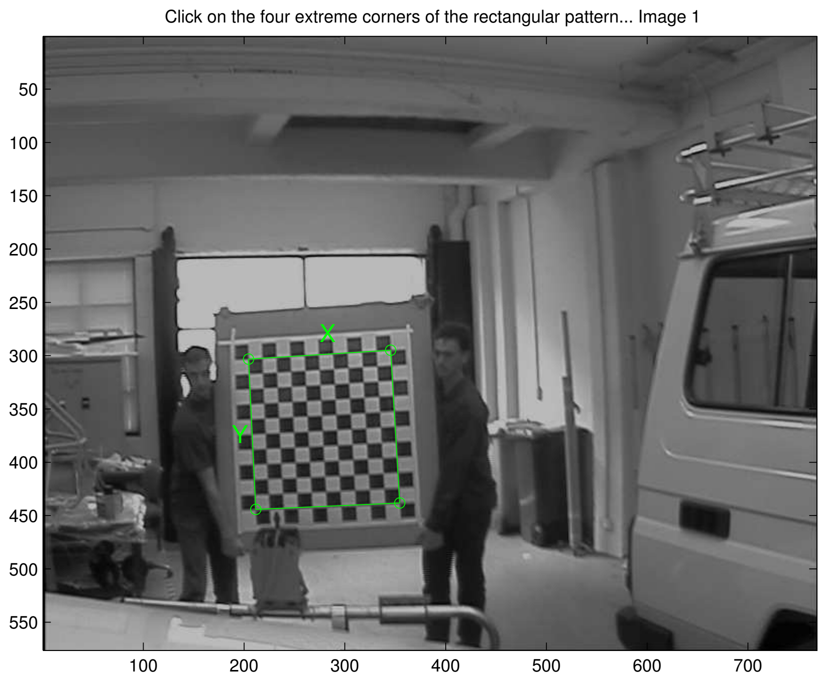

Click on the four extreme corners of the rectangular pattern... Image 1
X
Y
50
100
150
200
250
300
350
400
450
500
550
100
200
300
400
500
600
700

Figure 4.2: Calibration step 2, the four extreme corners identified using the mouse.

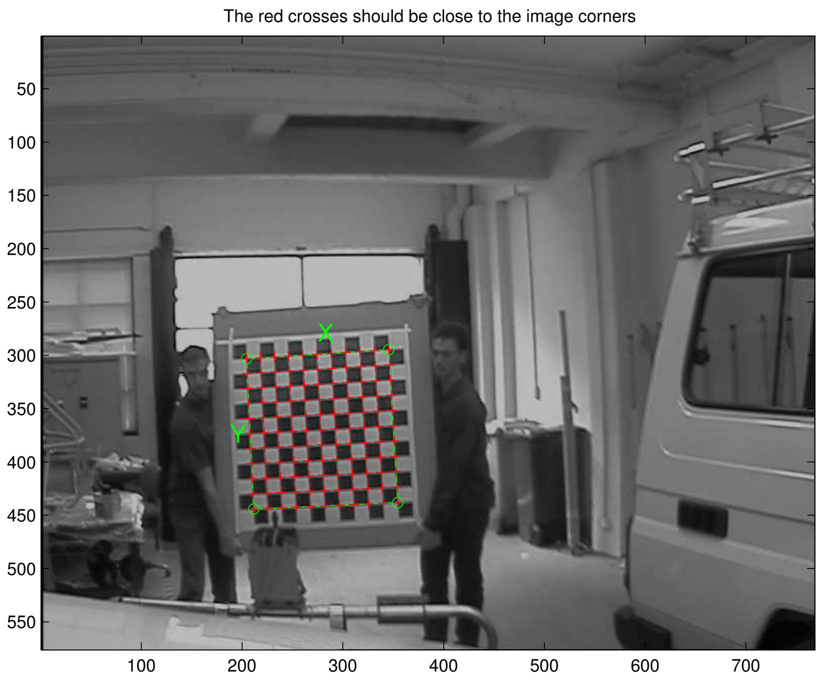


Figure 4.3: Calibration step 3, all corners within the bound predicted, shown as red crosses.

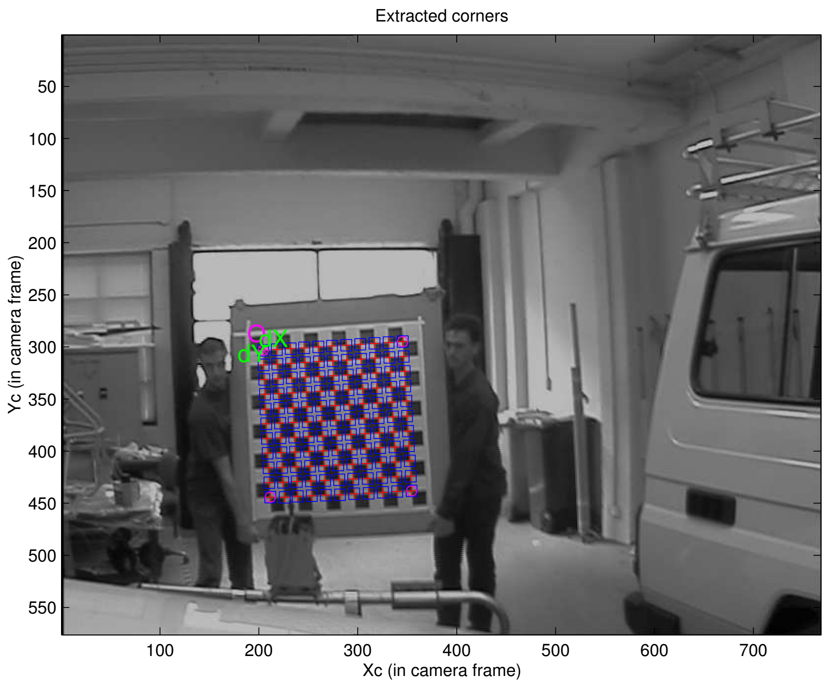


Figure 4.4: Calibration step 4, all corners are extracted, the blue squares shows the limits of the corner finder.

calibration board should be moved throughout the image with close-up and far-away shots). The calibration program finds the various values using statistical methods, thus a number of images need to be used. The minimum number required for the program is 9, however, it is best to use more than 9 sets of images.

During calibrations, the program first identifies all the images for the individual camera to be calibrated in the working path. For each image, the user is asked to identify the four extreme corners, which are shown on figure 4.2. Straight lines are drawn between the corners to identify the boundary of the calibration grid, the code also checks if there are ten (or the specified number of) squares on each side. The size of each square is entered by the user.

Next, all the corners within the boundary are predicted as shown on figure 4.3. The user is then asked if an initial distortion value is to be guessed, based on how close the corners predicted are to the real image corners. The corners are automatically extracted using these positions as initial guess. Figure 4.4 shows the final stage for each image where the corners are extracted, the blue squares around the corner points show the limits of the corner finder window. The corners are extracted to an accuracy of about 0.1 pixel.

After corner extraction of all the images, the code proceeds to run the main camera calibration procedure. This is done in two steps: first initialisation, and then nonlinear optimisation.

The initialisation step computes a closed-form solution for the calibration parameters based not including any lens distortion. The non-linear optimisation step minimises the total re-projection error (in the least squares sense) over all the calibration parameters. The optimisation is done by iterative gradient descent with an explicit (closed-form) computation of the Jacobian matrix. The resulting parameters of the calibration process are:

- `fc`: the focal length in x and y direction,
- `cc`: the principle points, i.e. the central pixel of the image in x and y direction,
- `kc`: the distortions of the image, this will be discussed in the later sections,
- `alpha_c`: the skew of the image in angle, and
- `err`: the errors associated with the values.

For a complete description of the calibration parameters and how they are related to the images see Appendix C.

4.2 Criteria for Image Rendering

Before any image rendering process is discussed, it is necessary to take a look at the various criteria that should be set that defines the "best" algorithm for our project. Canny [5], and Smith and Brady [33] all explained what sort of criteria is necessary for image processing. Note that most of the criteria applies more to the edge and corner detector than the filtering process, which is to be expected as that's ultimately the desired result. These criteria can be summarised as following:

Firstly and most importantly, the algorithm has to have a low error rate. It is important that edges and corners that are present in the image and thus the real world surround of the cameras are not missed and that there are no spurious responses. This criteria, according to Canny means that the filter should maximise the signal to noise ratios. This makes sense in the case of linear filters, which has features thresholded at a later stage than the enhancement. However, in the case of non-linear filters, Smith and Brady framed the criteria as it "should be a minimum number of false negatives and false positives".

The second criterion for the image rendering is that the features need to be well localised. In other words, the location of the features found by the algorithms need to be as close to be actual location as possible. This is particularly important for stereo vision as a small disparity in measurements between the left and the right images would affect the actual coordinate of the objet in 3-D space.

Thirdly, detectors should be able to fix the multiple responses problem. This is important as the algorithms will almost always turning up several detected 'edges' or 'corners' at one place. A good algorithm should be able to give just one response at every feature.

The purpose of this thesis is to investigate the use of stereo vision systems for real time applications, thus the fourth criterion as pointed out by Smith and Brady is speed. The algorithm should be fast enough to be usable in the image processing system so that the resulting data can be used by the vehicle in its control systems. Smith and Brady also pointed out that this criterion should not be allowed to dominate the development of the feature extraction algorithms to be used to an extent that the quality of the results is adversely affected. It is a fine balancing act to decide the use of a fast performing program verses one with very good detection ability.

4.3 Filtering

The first step in any vision based analysis is filtering of the image. As shown on figure 4.5, the objects close to the edges of the image are distorted such that none of the supposed straight edges are straight. This curved nature of the images are due to the projective geometry of the camera [22]. As light enters a camera, it has to pass through a complex lens system. However, the lenses are designed to mimic point-like elements (pinholes). Practical lens systems are nonlinear and can easily introduce significant distortions in the *intrinsic perspective* mapping from external optical rays to internal pixel coordinates. This distortion can be corrected by a nonlinear deformation

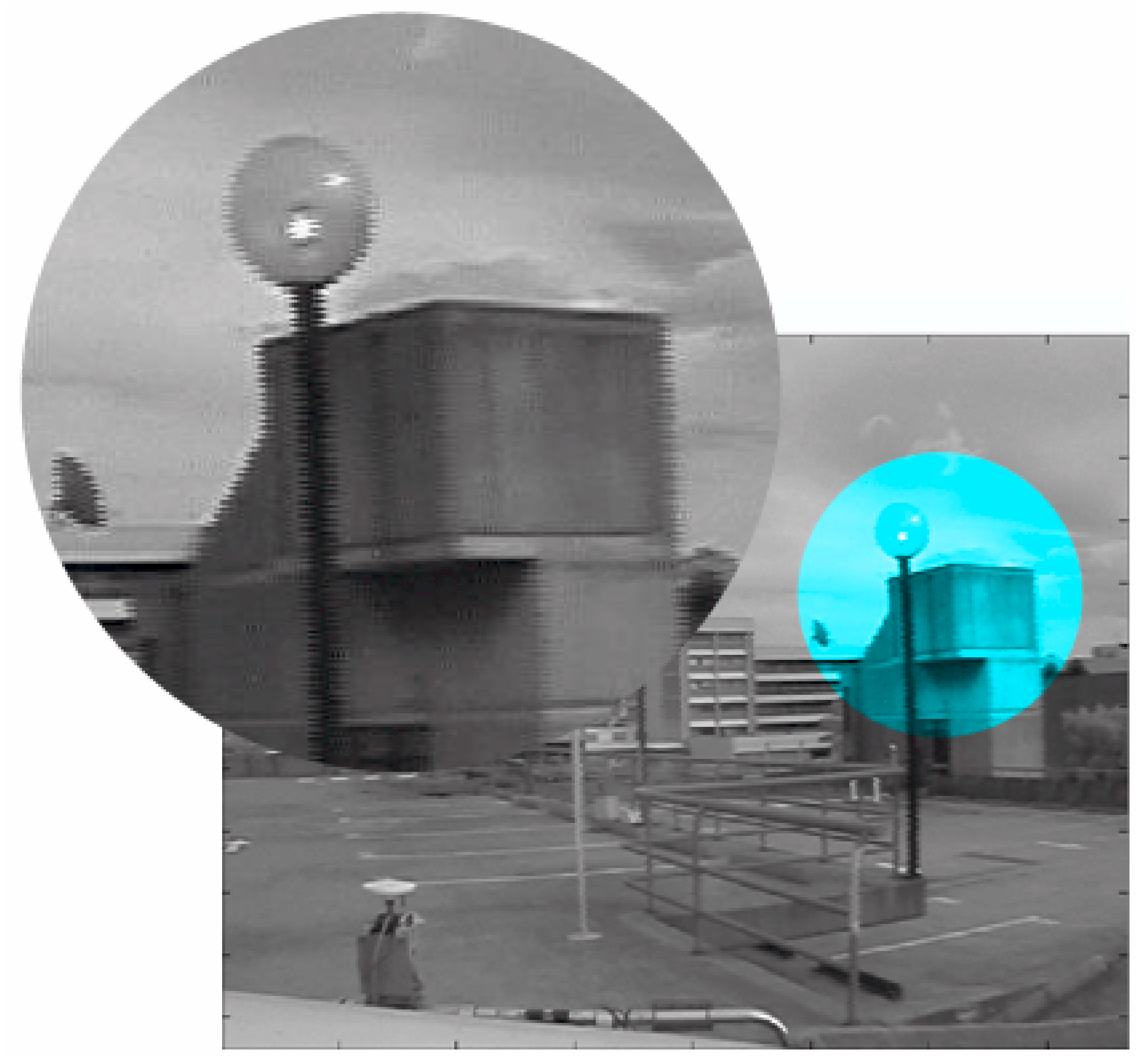


Figure 4.5: Unfiltered image, note the curves of the lamp posts and from the enlarged section, the roughness of the edges

of the image-plane coordinates.

Furthermore, the enlarged section on figure 4.5 shows that all the edges are very rough. This situation is caused by the way the camera captures an image. When an image is recorded, the camera first records the odd rows and then fill in the even rows of pixels on the image. Thus, if the vehicle is moving then the rows of the pixels will misalign from each other and result in the unevenness of the edges. This problem can be solved simply by realigning the rows of the images.

Figure 4.6 shows the result of the filtering process. As shown on the image, the

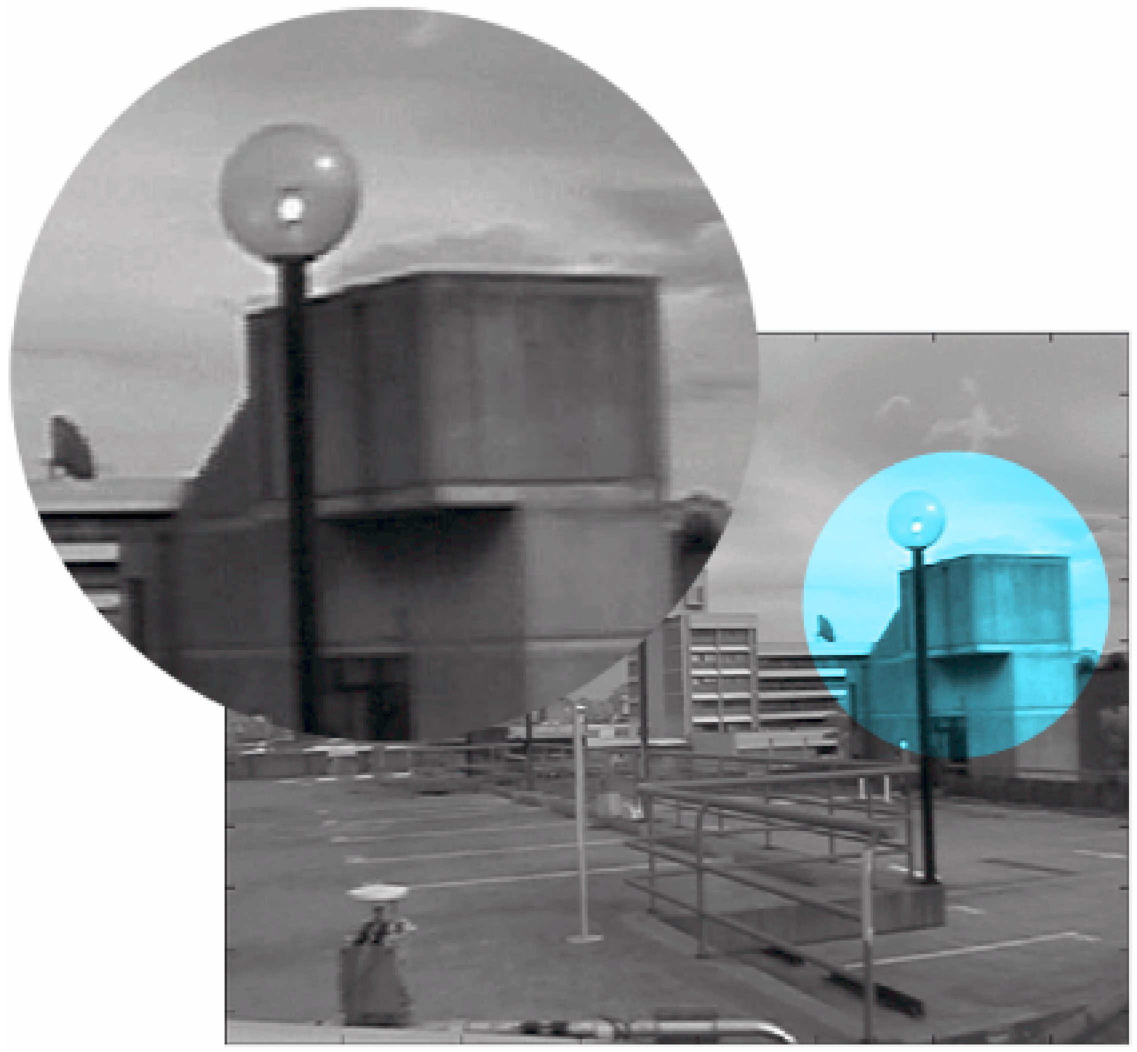


Figure 4.6: Filtered image, note that now the lamp posts are straight and from the enlarged section, the edges in the images are smooth

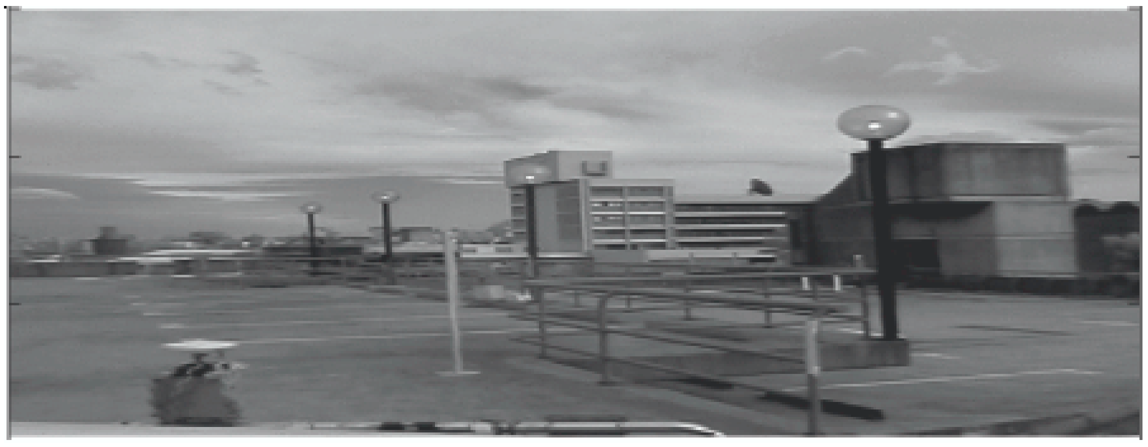


Figure 4.7: Filtered image where every second row of the image is deleted

lamp post near the right edge is now clearly straight from the undistortion process. Also, from the enlarged section of the building and the lamp, we can see that the edges are now smooth as the rows are now aligned. However, the second process depends on the speed of the vehicle as to how many pixels the rows have to be shifted by. The easiest method in this case would be to ignore every second row of the image. This will not affect the process of detection edges and corners, although slight adjustment will need to be made in depth determination. Figure 4.7 shows the image from the right camera where every second row of pixels is deleted.

4.4 Edge Detection

Distinct features from the images need to be extract in order for stereo matching to take place. Edges are often the first stage of information extraction from images as they are significant local changes in the image. Edges typically occur on the boundary between two different regions in an image. Algorithms for edge detection contain three steps [17]:

Filtering: Since gradient computation based on intensity values of only two points are susceptible to noise and other vagaries in discrete computations, filtering is commonly used to improve the performance of an edge detector with respect to noise. However, there is a trade-off between edge strength and noise reduction. More filtering to reduce noise results in a loss of edge strength.

Enhancement: In order to facilitate the detection of edges, it is essential to determine changes in intensity in the neighbourhood of a point. Enhancement emphasises pixel where there is a significant change in local intensity values and is usually performed by computing the gradient magnitude.

Detection: We only want points with strong edge content. However, many points in an image have a nonzero value for the gradient, and not all of these points are edges for a particular application. Therefore, some method should be used to determine which points are edge points. Frequently, thresholding provides the criterion used for detection.

It is important to note that detection merely indicates that an edge is present near pixel in an image, but does not necessarily provide an accurate estimate of edge location or orientation. The errors in edge detection are errors of misclassification: false edges and missing edges. The errors edge estimations are modelled by probability distributions for the location and orientation estimates.

4.4.1 Gradient

As mentioned previously, edges are significant local changes in an image. Mathematically, this is the first derivative of the intensity, i.e. the gradient. The gradient is a measure of change in a function, and an image can be considered to be an array of samples of some continuous function of image intensity. By analogy, significant changes in the gray values in an image can be detected by using a discrete approximation to the gradient. The gradient is the two-dimensional equivalent of the first derivative and is defined as the *vector*:

$$\mathbf{G}[f(x,y)] = \begin{bmatrix} G_x \\ \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \\ \frac{\partial f}{\partial y} \end{bmatrix} \tag{4.1}$$

There are two important properties associated with the gradient: (1) the vector $\mathbf{G}[f(x,y)]$ points in the direction of the maximum rate of increase of the function f(x,y), and (2) the magnitude of the gradient, given by

$$G[f(x,y)] = \sqrt{G_x^2 + G_y^2} \tag{4.2}$$

equals the maximum rate of increase of $f(x,y)$ per unit distance in the direction G. It is common practice, however, to approximate the gradient magnitude by absolute values:

$$G[f(x,y)] \approx |G_x| + |G_y| \tag{4.3}$$

or

$$G[f(x,y)] \approx \max(|G_x|, |G_y|) \tag{4.4}$$

From vector analysis, the *direction* of the gradient is defined as:

$$\alpha(x,y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{4.5}$$

where the angle α is measured with respect to the x axis.

Many edge detectors have been developed in the last two decades. In the following sections are the various methods that were investigated for the thesis.

4.4.2 Simple Gradient Operator

For digital images, the derivative in equation 4.1 are approximated by differences. The simplest gradient approximation is:

$$\begin{aligned} G_x &\cong f[i,j+1] - f[i,j] \\ G_y &\cong f[i,j] - f[i+1,j] \end{aligned} \tag{4.6}$$

Remember that j corresponds to the x direction and i to the negative y direction. These can be implemented with simple convolution masks as shown below:

$$G_x = \begin{array}{|c|c|}\hline -1 & 1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|}\hline 1 \\ \hline -1 \\ \hline \end{array} \tag{4.7}$$

When computing an approximation to the gradient, it is critical that the x and y partial derivatives be computed at exactly the same position in space. However, using the above approximations, G_x is actually the approximation to the gradient at the interpolated point $[i, j+\frac{1}{2}]$ and G_y at $[i+\frac{1}{2}, j]$. For this reason, 2×2 first differences, rather than 2×1 and 1×2 masks, are often used for the x and y partial derivatives:

$$G_x = \begin{array}{|c|c|}\hline -1 & 1 \\ \hline -1 & 1 \\ \hline \end{array} \qquad G_y = \begin{array}{|c|c|}\hline 1 & 1 \\ \hline -1 & -1 \\ \hline \end{array} \tag{4.8}$$

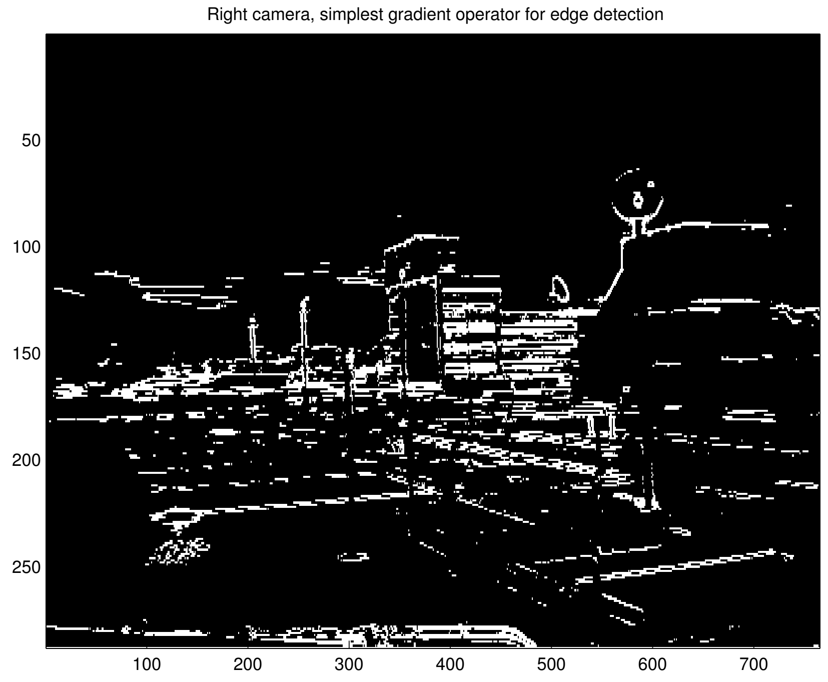


Figure 4.8: Results from the simplest gradient edge detector on the right camera image, using a threshold $\sigma = 40$.

Figure 4.8 shows the result of applying the masks in equation 4.8 on the right camera image. Each resulting pixel is the magnitude of the gradient as shown in equation 4.3.

4.4.3 Roberts Operator

The Roberts cross operator provides a simple approximation to the gradient magnitude:

$$G[f[i,j]] = |f[i,j] - f[i+1,j+1]| + |f[i+1,j] - f[i,j+1]| \tag{4.9}$$

Using convolution masks, this becomes

$$G[f[i,j]] = |G_x| + |G_y| \tag{4.10}$$

where G_x and G_y are calculated using the following masks:

$$G_x = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline\end{array} \qquad G_y = \begin{array}{|c|c|}\hline 0 & -1 \\ \hline 1 & 0 \\ \hline\end{array} \tag{4.11}$$

The result of Robert edge detection is shown in figure 4.9, this is the same image as that of figure 4.8.

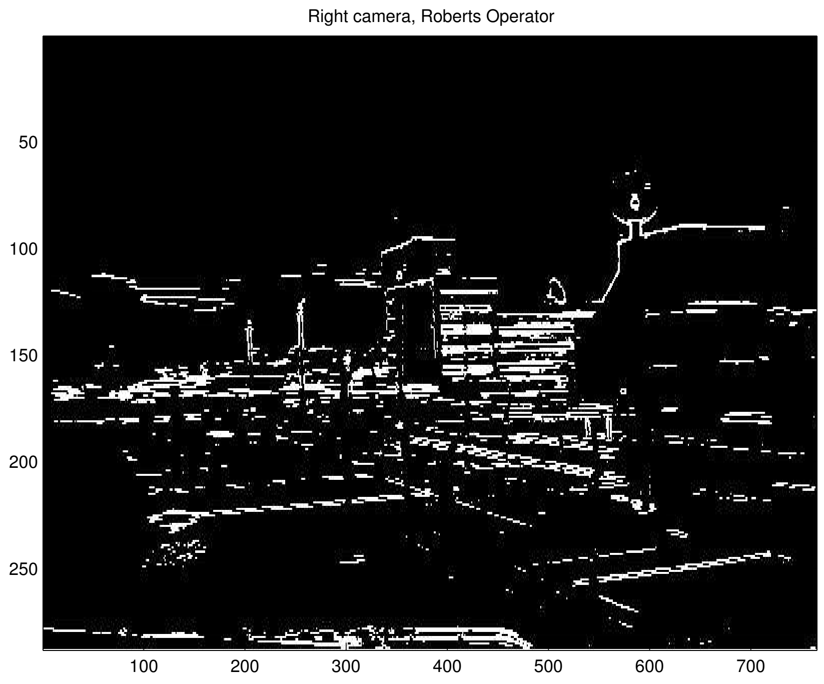


Figure 4.9: Results from the Roberts edge detector on the right camera image, $\sigma = 40$.

a_0	a_1	a_2
a_7	$[i, j]$	a_3
a_6	a_5	a_4

Figure 4.10: The labelling of neighbourhood pixels for the 3×3 masks

4.4.4 Prewitt Operator

The previous two detectors are uses 2×2 masks for the derivatives, the positions about which the gradients in the x and y directions are calculated are the same. This point lies between all four pixels in the 2×2 neighbourhood at the interpolated point $[i + \frac{1}{2}, j + \frac{1}{2}]$. This fact may lead to some confusion. Therefore, an alternative approach is to use a 3×3 neighbourhood and calculate the gradient about the centre pixel.

Consider the arrangement of pixels about the pixel $[i, j]$ shown in figure 4.10. The Prewitt operator is the magnitude of the gradient computed by

$$M = \sqrt{s_x^2 + s_y^2} \tag{4.12}$$

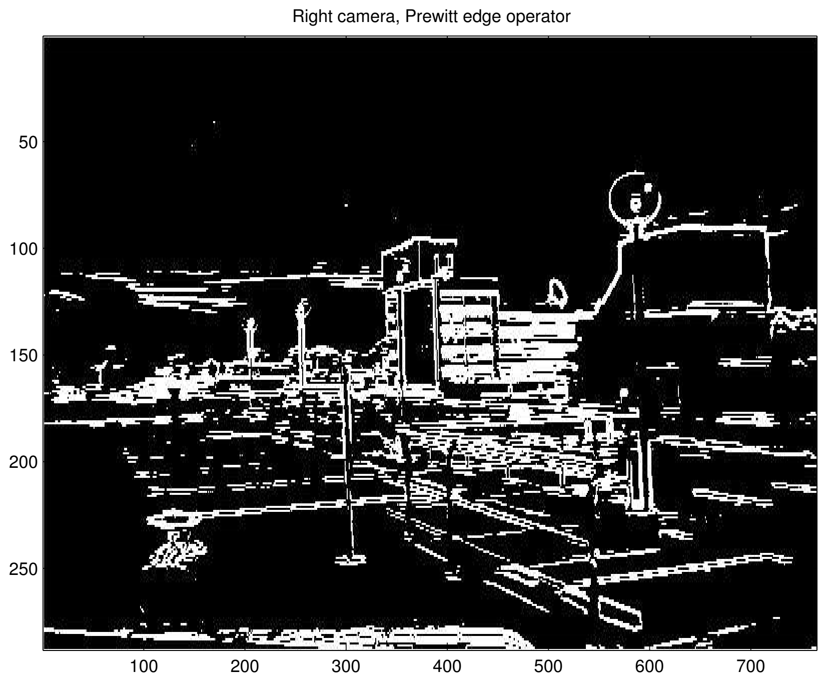


Figure 4.11: Results from the Prewitt edge detector on the right camera image, $\sigma = 50$.

where the partial derivatives are computed by

$$\begin{aligned} s_x &= (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \\ s_y &= (a_0 + ca_1 + a_2) - (a_6 + ca_5 + a_4) \end{aligned} \tag{4.13}$$

with the constant $c = 1$.

The gradient operators in equation 4.13 can be written as the convolution masks:

$$s_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad s_y = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array} \tag{4.14}$$

Figure 4.11 shows the performance of the Prewitt operator on the image from the right camera of the view shown in figure 4.7.

4.4.5 Sobel Operator

The Sobel operator uses the same equations as the Prewitt operator, except tat the constant $c = 2$. Therefore, the convolution masks from equation 4.13 is:

$$s_x = \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad s_y = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array} \tag{4.15}$$

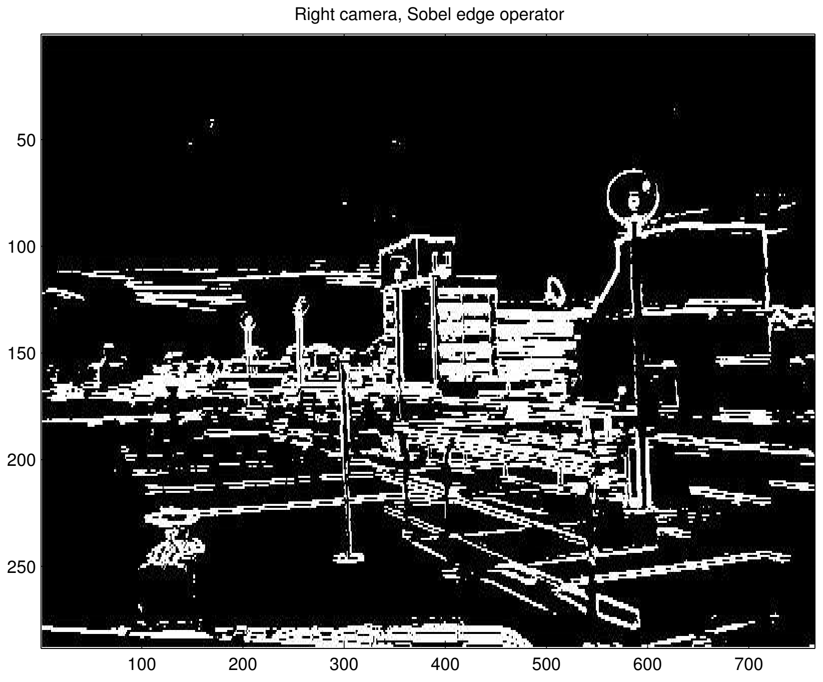


Figure 4.12: Results from the Sobel edge detector on the right camera image, $\sigma = 60$.

Unlike the Prewitt operator, these masks places an emphasis on pixels that are closer to the centre of the mask. This makes the Sobel operator one of the most commonly used edge detectors, as it would detect a corner at the same time of edge detection. The figure 4.12 shows the performance of the Sobel detector, again on the same image.

4.4.6 Laplacian Operator

The edge detectors discussed earlier computes the first derivative and, if it was above a threshold, the presence of an edge point was assumed. this results in detection of too many edge points. (Notice the thick lines especially in figures 4.11 and 4.12.)

A better approach would be to find only the points that have local maxima in gradient values and consider them edge points, in other words, an edge has a zero value in the second derivative. The Laplacian operator is one in two dimensions that correspond to the second derivative. The formula for the Laplacian of a function $f(x, y)$

is

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{4.16}$$

The second derivatives along the x and y directions are approximated using difference equations, giving the following equations [17]:

$$\begin{aligned} \frac{\partial^2 f}{\partial x} &= f[i,j+1] - 2f[i,j] + f[i,j-1] \\ \frac{\partial^2 f}{\partial y} &= f[i+1,j] - 2f[i,j] + f[i-1,j] \end{aligned} \tag{4.17}$$

By combining these two equations into a single operator, the following mask can be used to approximate the Laplacian:

$$\nabla^2 \approx \begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \tag{4.18}$$

Figure 4.13 shows the result of applying the above mask to the right camera image from figure 4.7. The Laplacian operator signals the presence of an edge when the output of the operator makes a transition through zero. Trivial zeros (uniform zero regions) are ignored.

4.4.7 Canny Operator

The *Canny Edge Detector* was first discussed by John Canny [5] and the result is shown on figure 4.14 for the image captured by the right camera.

Canny operator is optimised to the three criteria as shown in section 4.2. The detection problem is formulated as following: Begin with an edge with white Gaussian noise as in figure 4.15(a). Convolve the edge with a filter as shown on figure 4.15(b) or (d). The outputs of these are as shown in figure 4.15(c) or (e).

Detection and Localisation Criteria

Firstly, we deal with the signal-to-noise ratio for the problem. Let the impulse response of the filter be $f(x)$ and denote the edge itself by $G(x)$. We will assume that the edge

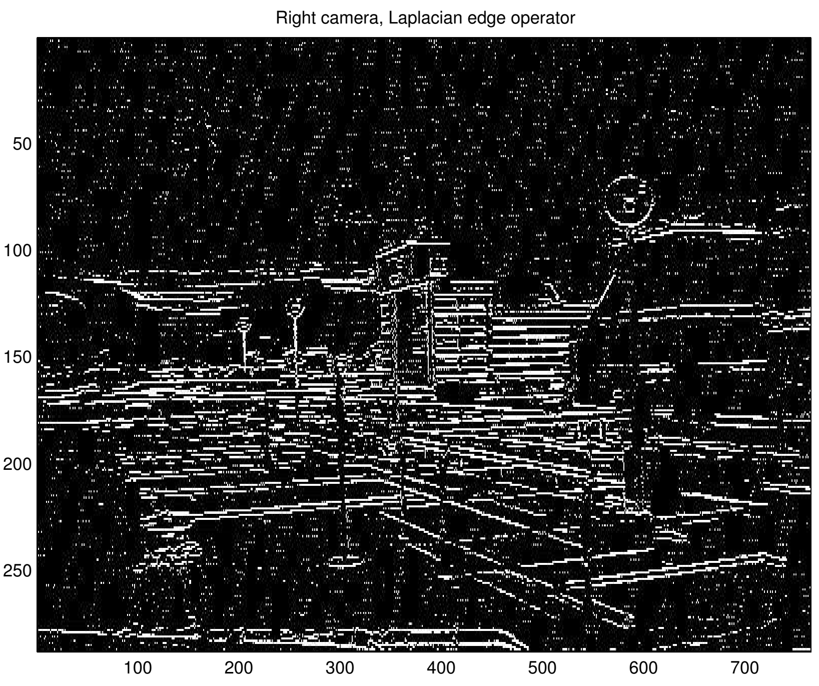


Figure 4.13: Results from the second order Laplacian edge detector of equation 4.18 on the right camera image, $\sigma = 60$.

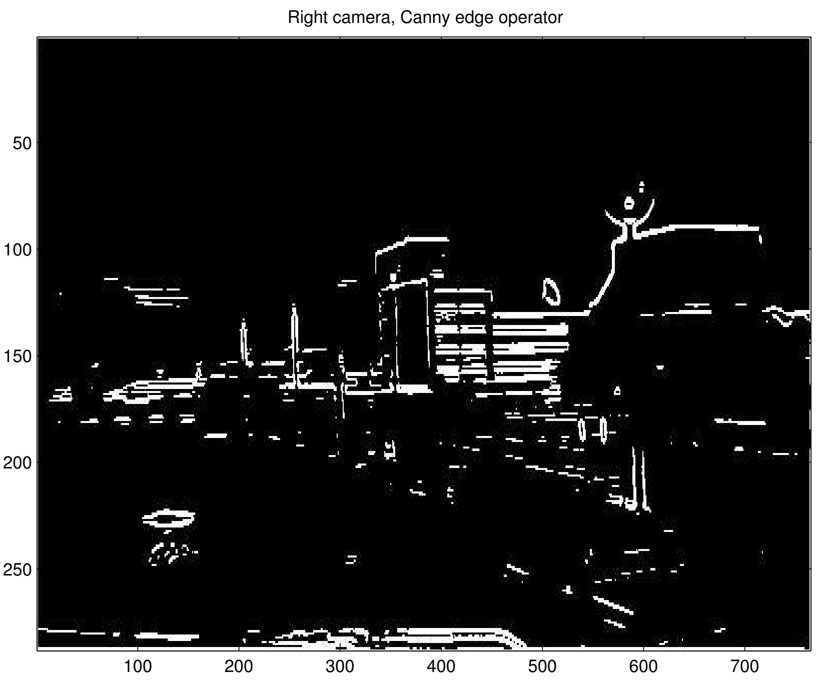


Figure 4.14: Results from the Canny edge detector on the right camera image.

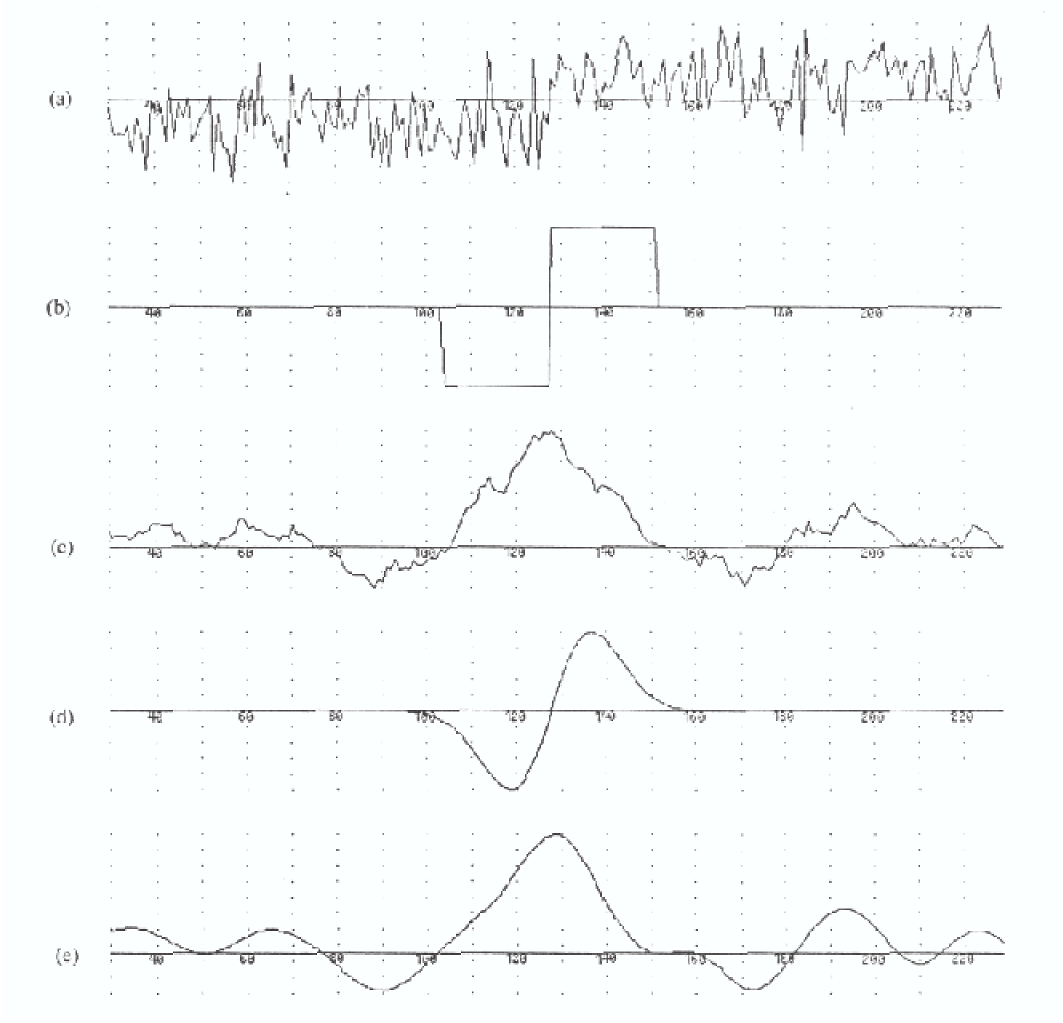


Figure 4.15: Optimisation of edge detection as demonstrated by Canny: (a) A noisy step edge. (b) Difference of boxes operator. (c) Difference of boxes operator applied to the edge. (d) First derivative of Gaussian operator. (e) First derivative of Gaussian applied to the edge. [5]

is centred at $x = 0$. Then the response of the filter to this edge at its centre H_G is given by a convolution integral:

$$H_G = \int_{-W}^{+W} G(-x)f(x)dx$$

assuming the filter has a finite impulse response bounded by $[-W, W]$. The root-mean-squared response to the noise $n(x)$ only, will be

$$H_n = n_0 \left[\int_{-W}^{+W} f^2(x)dx \right]^{1/2}$$

where n_0^2 is the mean-squared noise amplitude per unit length. The output SNR is thus defined as:

$$\text{SNR} = \frac{\left| \int_{-W}^{+W} G(-x)f(x)dx \right|}{n_0 \sqrt{\int_{-W}^{+W} f^2(x)dx}} \tag{4.19}$$

For localisation, we have defined the edge as the local maximum value in response to the filter $f(x)$, i.e. the first derivative of the response will be zero. Furthermore, the reciprocal of the root-mean-squared distance of the marked edge from the centre of the true edge is used as a measurement of the localisation criterion.

Let $H_n(x)$ be the response of the filter to noise only, and $H_G(x)$ be its response to the edge, and suppose there is a local maximum in the total response at the point $x = x_0$. The we have

$$H'_n(x_0) + H'_G(x_0) = 0$$

The Taylor expansion of $H'_G(x_0)$ about the origin gives

$$H'_G(x_0) = H'_G(0) + H''_G(0)x_0 + O(x_0^2)$$

Canny showed that the response $H_G(x)$ is always symmetric, and that its derivatives of odd orders are zero at the origin, thus giving:

$$H''_G(0)x_0 \approx -H'_n(x_0) \tag{4.20}$$

Now $H_n'(x_0)$ is a Gaussian random quantity whose variance is the mean-squared value of $H_n'(x_0)$, and is given by

$$E[H_n'(x_0)^2] = n_0^2 \int_{-W}^{+W} f'^2(x)dx$$

where $E[y]$ is the expectation value of y. Combining this result with equation 4.20 and substituting for $H_G''(0)$ gives

$$E[x_0^2] \approx \frac{n_0^2 \int_{-W}^{+W} f'^2(x)dx}{\left[\int_{-W}^{+W} G'(-x)f'(x)dx\right]^2} = \delta x_0^2 \tag{4.21}$$

where δx_0 is an approximation to the standard deviation of x_0. The localisation is defined as the reciprocal of δx_0.

$$\text{Localisation} = \frac{\left|\int_{-W}^{+W} G'(-x)f'(x)dx\right|}{n_0\sqrt{\int_{-W}^{+W} f'^2(x)dx}} \tag{4.22}$$

Equations 4.19 and 4.22 are mathematical forms for the first two criteria, and design problem reduces to the maximisation of both of these simultaneously. Thus, we seek the maximum of the product of the criteria for arbitrary edges:

$$\frac{\left|\int_{-W}^{+W} G(-x)f(x)dx\right|}{n_0\sqrt{\int_{-W}^{+W} f^2(x)dx}} \frac{\left|\int_{-W}^{+W} G'(-x)f'(x)dx\right|}{n_0\sqrt{\int_{-W}^{+W} f'^2(x)dx}} \tag{4.23}$$

Eliminating Multiple Responses

The first two criteria can be maximised as following: From the Schwarz inequality for integrals, the SNR in equation 4.19 is bounded by

$$n_0^{-1}\sqrt{\int_{-W}^{+W} G^2(x)dx}$$

and localisation in equation 4.22 by

$$n_0^{-1}\sqrt{\int_{-W}^{+W} G'^2(x)dx}$$

Both bounds are attained, and the product of SNR and localisation is maximised when $f(x) = G(-x)$ in $[-W, W]$. Thus, the optimal detector according to the first to criteria is a truncated step, and does not take into account the behaviour of the filter *nearby* the edge centre.

The maxima are close together that it is not possible to select one as the response to the step while identifying the others as noise. We need to add to our criteria the requirement that the function f will not have "too many" responses to a single step edge in the vicinity of the step. In order to have a low probability of declaring more than one edge, we'll need to limit the number of peaks in the response. Ideally, we would like to make the distance between peaks in the noise response approximate the width of the response of the operator to a single step. This width will be some fraction of the operator width W.

In order to express this as a functional constraint on f, we need to obtain an expression for the distance between adjacent noise peaks. Firstly, the mean distance between adjacent maxima in the output is twice the distance between adjacent zero-crossing in the derivative of the operator output. The mean distance between zero-crossings of f' is:

$$x_{zc}(f) = \pi \left(\frac{\int_{-\infty}^{+\infty} f'^2(x)dx}{\int_{-\infty}^{+\infty} f''^2(x)dx} \right)^{1/2} \tag{4.24}$$

The distance between adjacent maxima in the noise response of f, denoted $x_{\max}$, will be twice x_{zc}. We set this distance to be some fraction k of the operator width.

$$x_{\max}(f) = 2x_{zc}(f) = kW \tag{4.25}$$

This is a natural form for the constraint because the response of the filter will be concentrated in a region of width $2W$, and the expected number of noise maxima in this region is N_n where

$$N_n = \frac{2W}{x_{\max}} = \frac{2}{k}$$

Fixing k fixes the number of noise maxima that could lead to a false response.

4.5 Corner Detection

Another method in selecting features from the images is to detect the interesting points or *feature points*. These points are what we consider as corners, they are better in stereo matching problems than edges as these points are a lot more unique than edges. Corner detection should satisfy a number of important criteria [24]:

- All the true corners should be detected;
- No false corners should be detected;
- Corner points should be well localised;
- Corner detector should be robust with respect to noise;
- Corner detector should be efficient.

As with the edge detectors, there are a lot of proposed algorithms for the corner detector, here I will discuss the various methods that was used during the thesis.

4.5.1 Sobel Operator

In addition to its ability to detect the edges, the Sobel operator also has the added advantage of finding the corners due to the extra weighting used on the centre pixels (ref. equation 4.15). Figure 4.16 is the un-threshold version of figure 4.12 and is in colour instead of black and white. This give us the idea of the coordinates of the highest values, which should be where the corners are. However, we can see that in practice it does not distinguish the corners too well. Thus, other algorithms need to be employed.

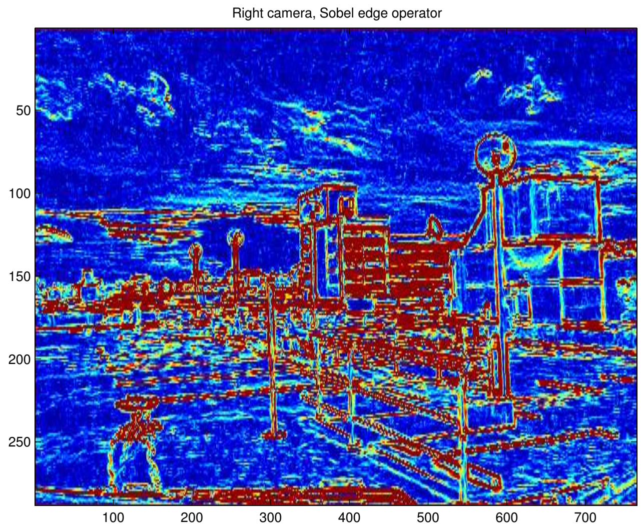


Figure 4.16: Results from the Sobel edge detector on the right camera image using the `jet` colour map and without applying thresholding.

4.5.2 Plessey Operator

The *Plessey Operator* was devised by Charnley et al. for the Plessey Company [6]. Corke applied the Plessey detector in his work for stereo vision used on autonomous helicopter [9]. The algorithm is the following:

The first gradients are approximated by

$$\begin{aligned} X &= I \otimes \begin{pmatrix} -1, & 0, & 1 \end{pmatrix} \approx \partial I/\partial x \\ Y &= I \otimes \begin{pmatrix} -1, & 0, & 1 \end{pmatrix}^T \approx \partial I/\partial y \end{aligned} \tag{4.26}$$

where $I(x, y)$ is the grey level intensity of each pixel on the image. Let

$$A = X^2 \otimes w$$

$$B = Y^2 \otimes w$$

$$C = (XY) \otimes w$$

where w is the Gaussian smoothing function. And let the matrix $\mathbf{M}$ be:

$$\mathbf{M} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}$$

The Plessey corner function is given by:

$$R = \text{trace}\mathbf{M} / \det \mathbf{M} \tag{4.27}$$

However, I have found the operator is not very effective in that it misidentifies a lot of the corners that do not exist in the image. This could be due to $\det \mathbf{M}$ can be equals to zero sometimes and from equation 4.27 we can see that it would give infinity as an answer.

4.5.3 Harris Operator

Some papers claims that the *Harris Corner Detector* is another name for the Plessey detector [33]. However, in their paper, Harris and Stephens showed that it is slightly different [15].

The corner response function is given by:

$$R = \det \mathbf{M} - k(\text{trace}\mathbf{M})^2 \tag{4.28}$$

where k is a parameter set to 0.04 [30].

The Harris operator implemented here is slightly different from what is stated here. In finding $\partial I/\partial x$ and $\partial I/\partial y$, instead of the masks used in equation 4.26, I'm using the first derivatives are those used in the Sobel detector as shown in equation 4.15 due to the central weight. If at a certain point of the matrix $\mathbf{M}$ are large, then a small motion in any direction will cause an important change of grey level. This indicates that the point is a corner.

Figure 4.17 shows the result from running the Harris corner detector on the right camera image as shown in figure 4.7. Here, the highest 100 corners are picked out. We can see from the figure that the detector is a fairly good algorithm, since most of the corners are detected. However, some of the corners are not present due to various reason. These include when the edges almost run like a straight line thus the corner is been 'hidden', or when the contrast between the two regions are not large enough.

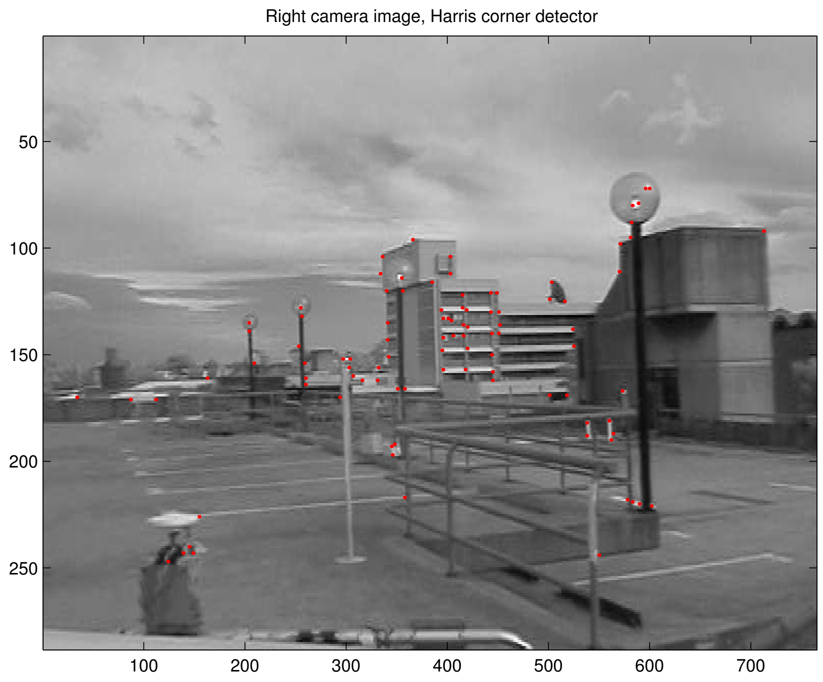


Figure 4.17: Results from the Harris corner detector on the right camera image

4.5.4 SUSAN operator

The SUSAN operator was introduced by Smith and Brady as a new approach to image processing [33]. The acronym SUSAN stands for "Smallest Univalue Segment Assimilating Nucleus" and is explained as follows:

Consider figure 4.18, which shows a gray rectangular shape on top of a white background. At various positions around the figure, we see that there are five circular masks (with a centre pixel called the "nucleus" by the authors) covering different amount of the rectangle. If the brightness of each pixel within a mask is compared with the brightness of that mask's nucleus then an area of the mask can be defined with having the same (or similar) brightness as the nucleus. This area of the mask shall be known as the "USAN". In figure 4.19 each mask from figure 4.18 is depicted with its USAN shown in white.

This concept of each image point having associated with it a local area of similar brightness is the basis for the SUSAN principle. The local area or USAN contains much

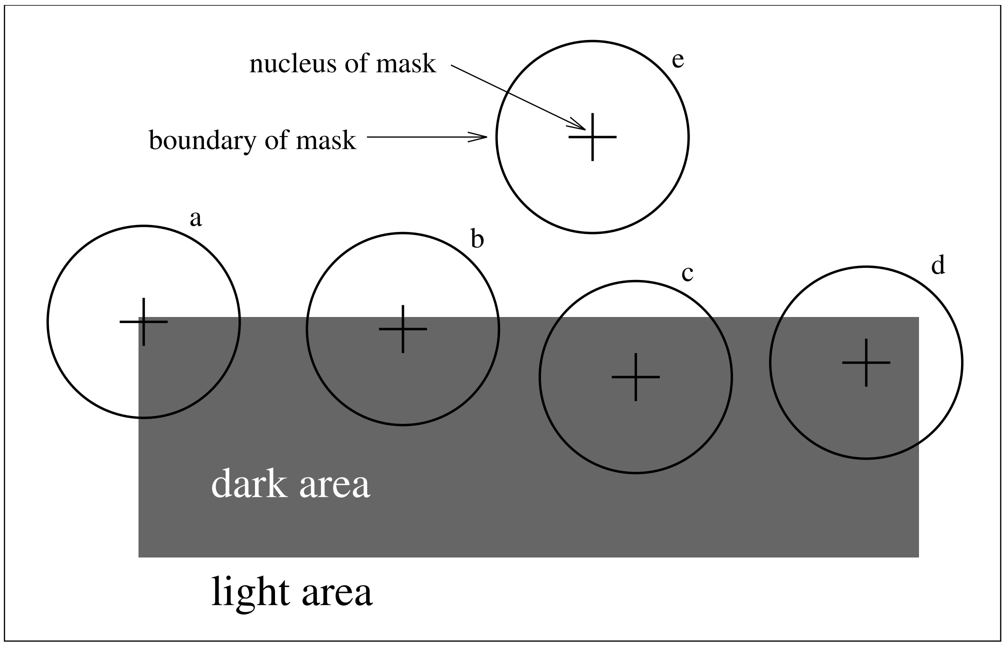


Figure 4.18: Four circular masks at different places on a simple image [33].

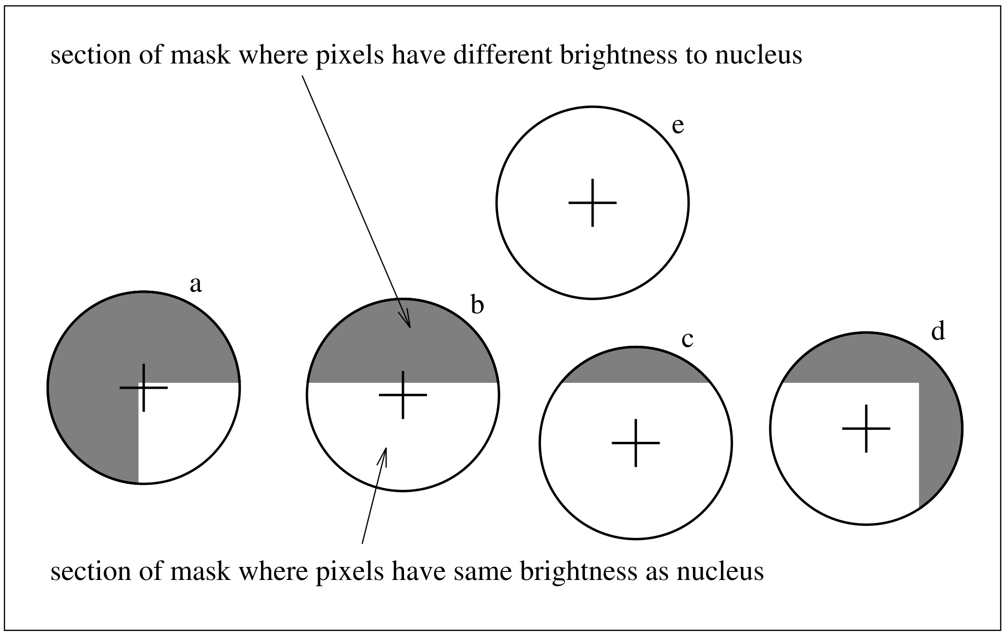

section of mask where pixels have different brightness to nucleus
e
a
b
c
d
section of mask where pixels have same brightness as nucleus

Figure 4.19: Four circular masks with similarity colouring; USANs are shown as the white parts of the masks [33].

information about the structure of the image. It is effectively region finding on a small scale. From the size, centroid and second moments of the USAN two dimensional features and edges can be detected. This approach to feature detection has many difference to the Harris operator, for example, the most obvious being that no image derivatives are used and that no noise reduction is needed.

The area of an USAN conveys the most important information about the structure of the image in the region around any point in question. As can be seen from figures 4.18 and 4.19, the USAN area is at a maximum when the nucleus lies in a flat region of the image surface, it falls to half of this maximum very near a straight edge, and falls even further when inside a corner. It is this property of the USAN's area which is used as the main determinant of the presence of edges and two dimensional features. The SUSAN principle is:

> "An image processed to give as output inverted USAN area has edges and two dimensional features strongly enhanced, with the two dimensional features more strongly enhance than edges."

SUSAN operator uses no image derivatives, the integrating effect of the principle, together with its non-linear response, give strong noise rejection. This can be understood simply if an input signal with identically independently distributed Gaussian noise is considered. As long as the noise is small enough for the USAN function to contain each "similar" value, the noise is ignored. The integration of individual values in the calculation of areas further reduces the effect of noise. Another strength of the SUSAN edge detector is that the use of controlling parameters is much simpler and less arbitrary (and therefore easier to automate) than other algorithms.

The SUSAN corner algorithm will now be discussed in detail. Firstly, all the pixels within a circular mask are compared with the nucleus, using the following equation:

$$c(\overrightarrow{r}, \overrightarrow{r_0}) = \exp\left(-(\frac{I(\overrightarrow{r}) - I(\overrightarrow{r_0})}{t})^6\right) \tag{4.29}$$

where $\overrightarrow{r}$ is any pixel in the mask, $\overrightarrow{r_0}$ is the nucleus of the mask, and t is the brightness difference. Although it might take some processing power to run equation 4.29 as it is, in practise, we can create a look up table with the variables and results and thus speed up the process. The sum of these comparisons is calculated using

$$n(\overrightarrow{r_0}) = \sum_{\overrightarrow{r}} c(\overrightarrow{r}, \overrightarrow{r_0}) \tag{4.30}$$

within the mask to determine the minimum contrast of features. Note that performance is not dependent on any 'fine-tuning' of the value t.

Next, n is compared with the geometric threshold g. For a "corner" to be present, n must be less than half of its maximum possible value, n_{max}. This is the same as saying that if the nucleus of the mask lies on top of a corner, then the USAN area will be less than half of the mask area, and will be a local minimum. It is safe to set g to be exactly half of n_{max} because of the nature of a quantised straight edge.

The geometric threshold g affects the "quality" of the output, in other words, it affects the shape of the "corners" detected. For example, if it were reduced, the allowed "corners" would be sharper. Thus this threshold can be fixed and will need no further tuning. This is a good feature of the SUSAN detector, since an algorithm where the threshold need to be adjusted for single uses are ultimately limited in its use. No weakness, however, is introduced into the algorithm by the use of the geometric threshold.

The brightness threshold t, on the other hand affects the "quantity" of the output, i.e. it affects the number of "corners" detected. This is due to the fact that t determines the allowed variation in brightness within the USAN, a reduction in this threshold picks up more subtle variations in the image and gives a correspondingly greater number of reported "corners". Although it might seem to be a negative point about the detector, it should be noted that in practise the threshold can be determined by the contrast of the images, where a low number be assigned to low contrast images and vice versa.

To create an initial response image, the following equation is used with $g = n_{\max}/2$:

$$R(\overrightarrow{r_0}) = \begin{cases} g - n(\overrightarrow{r_0}), & \text{if } n(\overrightarrow{r_0}) < g, \\ 0, & \text{otherwise} \end{cases} \tag{4.31}$$

where $R(\overrightarrow{r_0})$ is the initial response.

The final stage in the SUSAN corner detector is to search the initial response over 5×5 pixel regions for local maxima (above zero). However, before this non-maximum suppression is carried out, two small procedures are run to reduce false positive responses from edges and from noise.

SUSAN will give false detection when blurring of boundaries between regions occurs. This problem is eliminated by the following method: The centre of gravity of the USAN is found. Then the distance of the centre of gravity from the nucleus is found. A proper corner occurs when the centre of gravity if not near the nucleus.

Lastly, if there is a lot of noise or fine complicated structures, it will also cause false corners to be reported. It is noted that all of the pixels within the mask, lying in a straight line pointing outwards from the nucleus in the direction of the centre of gravity of the USAN must be part of the USAN, for a corner to be detected. This addition to the algorithm forces it to have a degree of uniformity.

In summary, the SUSAN corner detection algorithm performs the following steps at each image pixel:

1. Place a (circular[1]) mask around the pixel in question (the nucleus);

[1]In their paper, Smith and Brady experimented with different sized masks: 3×3 square, and circular mask with radius of 3.5 pixels making the mask of 25 pixels in total. It was felt however, a mask of this size would slow the algorithm down and was not investigated.

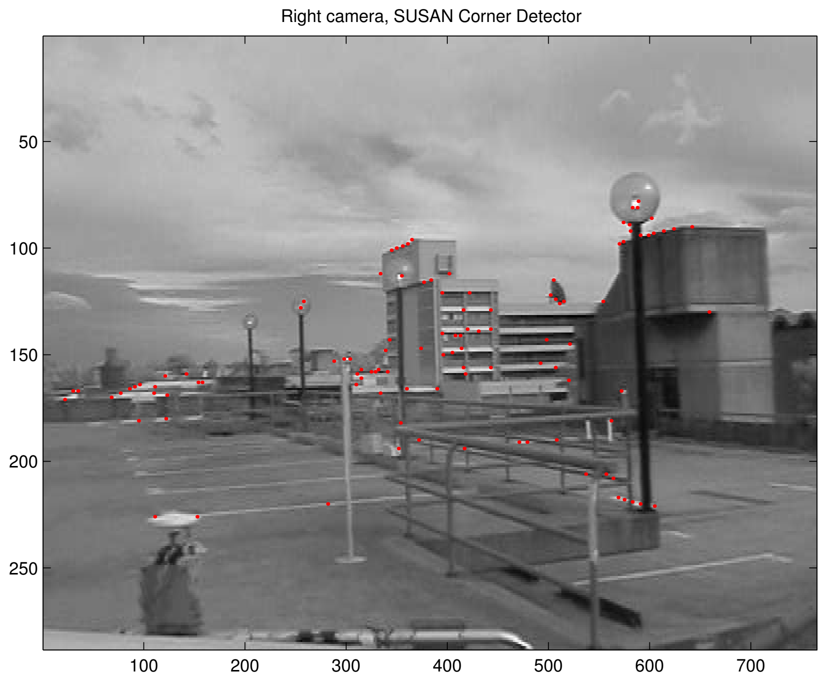


Figure 4.20: Result from the SUSAN corner detector on the right camera image using the same sorting criteria as the Harris detector as shown on figure 4.17.

2. Use equation 4.29 to calculate the number of pixels within the circular mask that have similar brightness to the nucleus (these are the USAN pixels);
3. Use equation 4.31 to subtract the USAN size from the geometric threshold to produce a corner strength image;
4. Test for false positives by finding the USAN's centroid and its contiguity;
5. Use non-maximum suppression to find corners.

Figure 4.20 shows the result from the SUSAN corner detector on the right camera image. It shows the highest 100 corners from the detector. Note that most of corners are detected by the algorithm, however some corners are also missed due to bad contrast.

4.6 Discussion

As discussed in section 4.2 of this chapter, there are four important criteria in image rendering algorithms:

1. Low error rate;
2. Features well localised;
3. Fix multiple response problem;
4. Fast enough for use in real time image processing system.

In this section, I will discuss about how the various algorithms shown in this chapter measures up to the criteria as stated above.

4.6.1 Filtering

The second and third criteria in the list above does not apply to the filtering process of the image rendering phase of the stereo system. This is because the filtering process is simply there to reduce the noise in the received data and possible error in future distance calculation.

In section 4.3, it was shown that there are two processes in the filtering stage: un-distortion and un-interlacing. Un-distortion restores the image to a more ‘natural’ look instead of the ‘fish-eyed’ look caused by the pin-camera effect, and un-interlacing removes every second row in the image to smooth out the edges.

The first criteria discussed above is low error rate. For the two filtering process, un-distortion utilises the intrinsic parameters of the camera found using the calibration toolbox, thus errors in the process is basically caused by the quality of the calibration procedure. The accuracy of the algorithm can be checked by finding the equation of the lines of the edges in the image and check if each and every pixel belonging to the edge lies on the line.

As for the un-interlacing process of filtering, there are no immediate errors associated with the procedure, since it is a simple process of removing half of the rows and no other data are introduced nor algorithms performed. It however, does affect later algorithms, such as edge/corner detection and fundamental matrix calculation. In case of the former, the positions of vertical edges would not be affected, but the position of any horizontal edges will have an error of $\pm$ 1 pixels. The same situation applies to the corner detector, the positives from the algorithm will have a $\pm$ 1 pixel in the y direction. The fundamental matrix will be discussed in detail in Chapter 5, but the data for it's calculation are from the matched pixels in the two images, thus un-interlacing will add an extra $\pm$ 1 pixel in the y direction to the input data.

The fourth criterion for image rendering is speed of the algorithms. This is a criterion that need to be kept in mind all the time, in real-time image processing, it is always a trade-off between speed and accuracy. In the filtering process, it should be noted that both procedures need to be performed on the left and the right camera images (unlike in edge/corner detection where only one image is processed, to be discussed later).

To find the speed of the algorithms, we use the MATLAB functions `tic` and `toc` to measure the processing time. It was found that using a 64Mb RAM, 600MHz processor, the un-distortion process took nearly $430s$ and the un-interlacing took $14.2s$ for *each* image. The time spent un-distorting the images is obvious far too long, as both images will take almost $15min$ to process.

Therefore, we need to think about whether it is "worth it" to un-distort the images. Firstly, it should be kept in mind that this is only the first process in the stereo vision algorithm in building the 3D map. The most process intensive part of the program is in finding the conjugate pairs of the same point in 3D space and thus finding its coordinates. Secondly, the distortion of the images varies radially outwards. In other words, the further out a pixel is, the more distorted it will be, this is seen from the before and after images of figures 4.5 and 4.6. Therefore, an object at the edge of the image could be 'out' by 10, 20 pixels, while at the centre of the image, it could only have an error of 1 or 2 pixels. Thirdly, we should remember that at $30km/h$ the Ute would've travelled approximately $7.5km$, and at the desired top speed for the HSV of $90km/h$ it would have travelled $25km$. This is highly unacceptable, since any map found through stereo vision will be useless by the end of the program execution.

From the points here, it can be concluded that the un-distortion process should not be included in the stereo vision program, unless in future days when it can be done within seconds. And the errors resulted from this exclusion might have to be a shortcoming that has to be included to give "real-time" processing ability to the Ute.

On the other hand, the un-interlacing problem occurs due to the moving vehicle and the method by which the camera records the data. We know that the faster the vehicle moves, the more misaligned the rows in the images will be. Unlike the un-distortion process, which affects the resulting coordinates of the objects identified, the misalignment of the rows will affect the edge/corner detectors in a way that false positives will be identifies and real features missed. Furthermore, although the timing of the process is about half a minute for the pair of images, the algorithm is written in MATLAB which is particularly slow in its execution of `for` loops. For the program to run in real-time it needs to be written in C or C++ both of which would give much faster run time with the same algorithm. Therefore, the un-interlacing procedure is one that has to be included in the whole stereo vision program.

4.6.2 Edge Detector

Every one of the criterion shown on the list on page 52 need to be considered when deciding the best detector for edge finding in this case. Each one of them will be discussed in detail below.

The first criterion for the detector is to have a low error rate. This is not much of a problem for all of the detectors except the second derivative Laplacian. As shown on figure 4.13 the result of the aforementioned detector is very noisy with speckles of white dots over the entire image. All of the first derivative detectors however, do not have the same problem

The second criterion is that the detector need to be able to localised the features well. As shown by the low threshold value for the result of Simplest Gradient detector and the Roberts detector on figure 4.8 and 4.9 respectively, these two algorithms are not very efficient in detecting most of the features. Comparing with figure 4.12 of the Sobel detector, where the threshold is set 50% higher, these two detectors still have less features identified. The localisation and thresholding of the Canny detector as shown on figure 4.14 however, can not be used for comparison as this algorithm compares the maximum and minimum pixel values of the result and finds the threshold (which is equals to $0.1 \times (I_{\max} - I_{\min}) + I_{\min}$).

The third criterion for the detector is avoiding multiple responses. The best result from the algorithms under investigation is the Laplacian detector, this is reasonable as in second derivative an edge has a value of zero, unlike in first derivative where it is simply the local maxima. This gives Laplacian operator the advantage of finely defined edges as results.

The last criterion for the algorithms is the speed of the process. Table 4.1 shows the various run time speed of the algorithms.

Algorithm	**MATLAB run time**
Gradient	$0.94s$
Roberts	$0.55s$
Prewitt	$0.77s$
Sobel	$0.72s$
Laplacian	$0.39s$
Canny	$1.59s$

Table 4.1: Comparison of the speed of the edge detection algorithms runtime per image

We can see from the table that there are not a lot of differences in the run time of the various algorithms. It should be noted however, that the Canny detector is run without the 'thinning' process which would reduce the multi-response problem but at the same time very process intensive. Even with this omission, Canny detector is still the slowest algorithm and its result is not superior than the others. The Laplacian detector is the quickest as it only utilises one mask while the rest uses two to find the derivative in x and y direction separately.

In conclusion, Laplacian detector can be categorised as the 'best' here as it has superior results in 3 of the 4 criteria. The problem, with this detector is the noise generated at the same time. A different mask of similar nature might give a better result than the mask used in this thesis. However, finding the best edge detector is slightly out of the scope of this thesis as shown in later chapters.

4.6.3 Corner Detector

The same criteria discussed in the last section also applies to finding the best corner detector. In other words, we need firstly an algorithm with low error rate of detecting the corners. In this case, both the Sobel Operator and the Plessey Operator are not suitable for the purpose, as neither have a good detection rate. The same can be said of the two detectors about the second criterion, which is that the features has to be well localised.

Algorithm	**MATLAB run time**
Harris	$3.9s$
SUSAN	$109.9s$

Table 4.2: Comparison of the speed of the corner detection algorithms runtime per image.

The Harris and SUSAN operator on the other hand presents a different point to the criteria. Both operator have very little error as shown on figure 4.17 and 4.20 and both have good response to the actual corners. However, SUSAN give a lot more corners than the Harris operator. Although it is hard to say which operator give the best response, both have misses of various corners in the image. The criteria for eliminating multiple response is hard to judge from the figures as another algorithm was used to "clean up" the corners detected to eliminate possible multiple responses.

Therefore, the major comparison between the two algorithms is the last criterion: time. Table 4.2 shows the run time between two the corner detection algorithms. From the table we can see that while SUSAN detector might theoretically be a better algorithm in corner detection than the Harris operator, the latter only requires a fraction of the run time of the former. And in the case here, since both algorithms give very similar corner responses, then the Harris operator is a superior detector.

Chapter 5

Stereo Matching

Once unique features in the images are found, then it is necessary to identify the *conjugate pairs* of the images for use in the stereo imaging formulae (equation 2.3). A conjugate pair is two points in different images that are the projections of the same point in the scene [17] (refer to figure 2.1 for the geometry).

Detection of the conjugate pair in stereo images, however, has been an extremely challenging research problem known as the *correspondence problem.* It can be stated as follows: *for each point in the left image, find the corresponding point in the right image* [17].

5.1 Edge Matching

Edge matching involves comparing the edges' orientations and strength on the epipolar lines. The edges are first matched at coarse resolution to avoid false matches as there are fewer edges at coarse scale. To simplify the matching procedure, the search for a match for each feature takes place along the corresponding epipolar line in the second image for a limited distance centred around its expected position [17].

There are however, many problems associated with edge matching. Firstly, as the edges are matched along the epipolar line, the horizontal edges can not be matched.

Secondly, the matching process must begin at a coarse scale to determine an approximate value for the disparity and then redone at a finer scale. This is done to avoid false matches, however, it is an extra process for computing thus not for real time operation. Most importantly, the value of the computed depth is not meaningful along occluding edges where the depth is not well defined. Along these edges, the depth is anything between the foreground object and the background object. Therefore, edge matching is not an viable option in stereo matching.

5.2 Region Correlation

The *region correlation* method is one of the many methods developed for finding potential features for correspondence using the corners detected. Using the corners have many advantages over the edges: Firstly, as they are only points, we don't need to worry about the problem of not picking up horizontal edges. Secondly, corners are a lot more unique than edges, as the area surrounding the corners have high variance.

A simple technique in matching feature points is to compute the correlation between a small window of pixels centred around every potential matching feature in the second image. The feature with the highest correlation is considered as the match [17], [30]. One approach to this is to look at the local intensity value, however, many points will have similar intensity value and that this measure is very sensitive to noise and changes in illumination.

A better approach is to take the information of more pixels into account by comparing local neighbourhoods of the corners. A small window of $(2N+1) \times (2N+1)$ pixels centred around the points for comparison is used. A measurement of the similarity between the windows is given by the *correlation coefficient* C defined as [17] [30]:

$$C = \frac{\sum_{i=-N}^{N}\sum_{j=-N}^{N}(I(x-i,y-j)-\overline{I})(I'(x'-i,y'-j)-\overline{I'})}{\left\{\sum_{i=-N}^{N}\sum_{j=-N}^{N}\left[(I(x-i,y-j)-\overline{I})\right]^2\sum_{i=-N}^{N}\sum_{j=-N}^{N}\left[(I'(x'-i,y'-j)-\overline{I'})\right]^2\right\}^{1/2}} \tag{5.1}$$

with I and I' the intensity/pixel values at a certain point, and $\overline{I}$ and $\overline{I'}$ are the mean intensity values of the window. Also the points (x, y) and (x', y') are the points of interest. The value of N should be chosen with care, as although a larger window would produce a more accurate result, it will slow the computation time. Typically, $N = 2$ or 3, here I used the latter giving window size of 7×7.

Figure 5.1 shows the left and right camera images (after removing half of the rows) with the corners detected. To illustrate the result of the correlation test from equation 5.1, we will take corner 1 labelled on the figure and run the correlation test over the corresponding corner on the right image.

Figure 5.2 shows the aforementioned corners, the figure on the left centres on the corner detected by a corner detector, and the window is 7×7 pixels in accordance to the size of the window used in the the correlation test. The figure on the right is centred on the corresponding corner detected by the same corner detector. The window outlined by yellow is the search window in which each pixel will be compared with the central pixel on the left. The entire window is 3 pixels larger on each side to show the area to be used for equation 5.1.

Table 5.1 shows the result of applying the correlation test to compare the central pixel from the left image on figure 5.2 to the window bounded by the yellow box from the right image. The value in the centre of the table is the comparison between the two identified corners, which shows a correlation of 55.78%. However, we can also see that on the same row there are three pixels with correlation coefficients of higher than 90%, with the second pixel having a value of nearly 98%. Therefore, we can say that

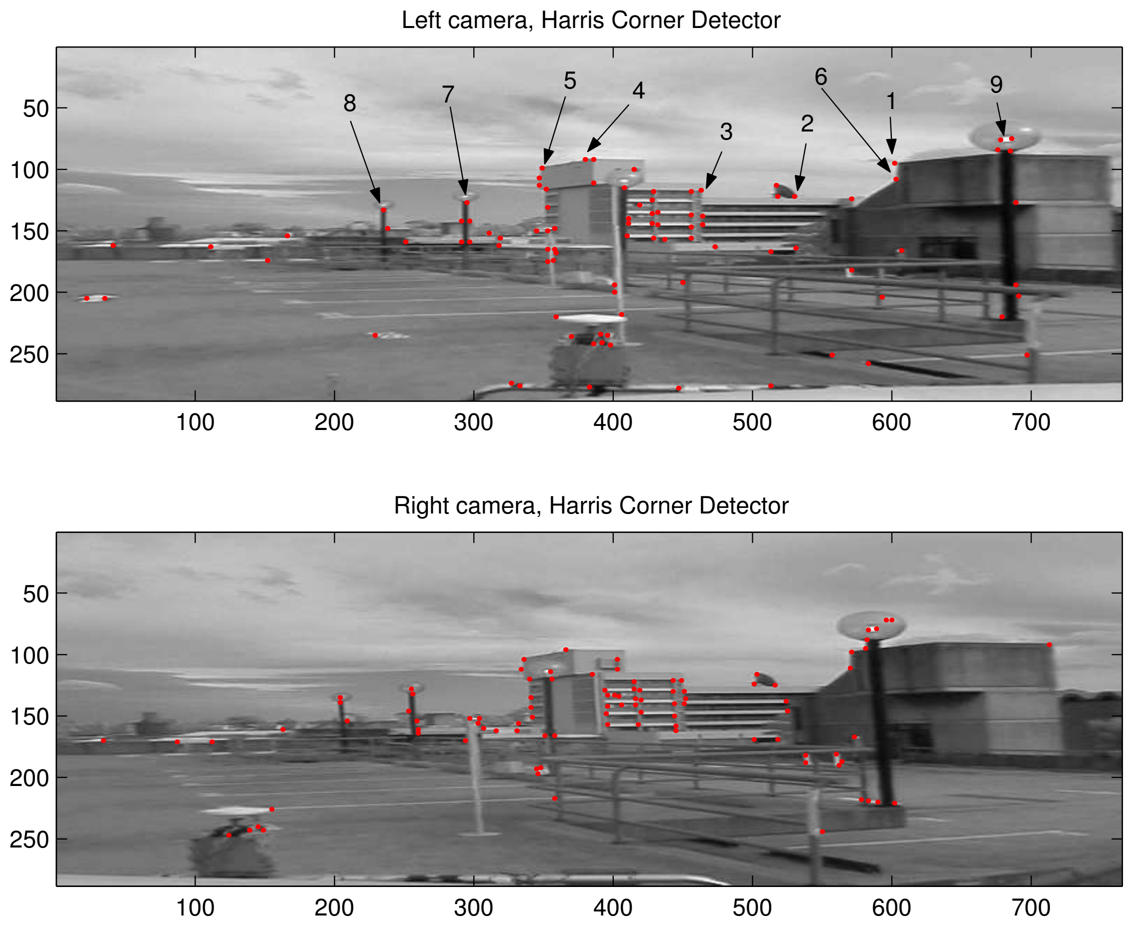


Figure 5.1: The left and right images with the corners detected

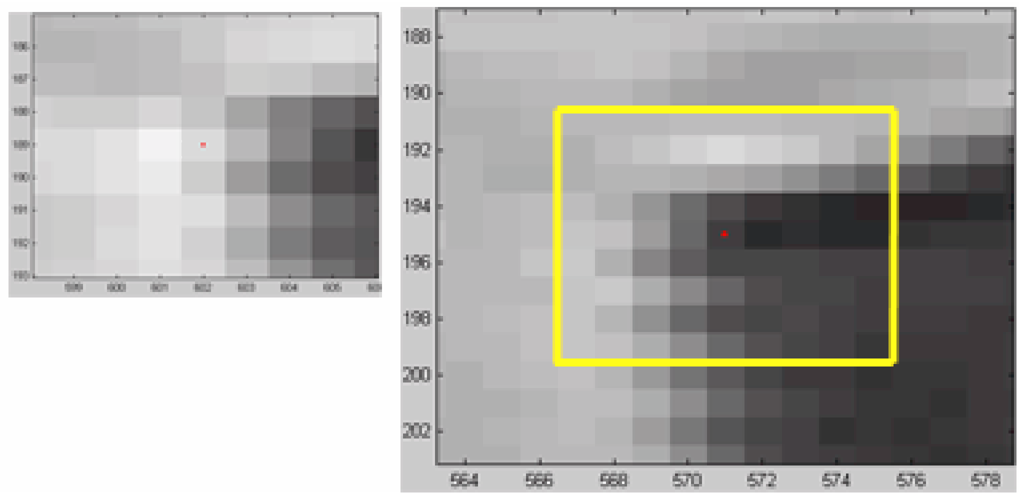


Figure 5.2: The pixels used for the correlation test

0.2176	0.2707	0.2929	0.2831	0.2699	0.2695	0.2731	0.2779	0.2742
0.3246	0.3828	0.3878	0.3488	0.3092	0.2856	0.2738	0.2752	0.2816
0.5700	0.6131	0.5723	0.4809	0.3970	0.3372	0.2983	0.2847	0.2866
0.7514	0.8276	0.7827	0.6589	0.5190	0.3969	0.2960	0.2361	0.2080
$\boxed{0.9164}$	$\boxed{0.9797}$	$\boxed{0.9036}$	0.7378	**0.5578**	0.4033	0.2694	0.1788	0.1274
0.7252	0.8150	0.7773	0.6598	0.5199	0.4017	0.2707	0.1493	0.0684
0.6579	0.7067	0.6578	0.5517	0.4316	0.3478	0.2515	0.1287	0.0045
0.5083	0.5910	0.5910	0.5332	0.4541	0.4096	0.3661	0.3211	0.2060
0.5908	0.6823	0.6822	0.6173	0.5411	0.4992	0.4770	0.4604	0.3177

Table 5.1: Result of the correlation test on figure 5.2

this is the pixel that forms the conjugate pair with the identified corner from the left image.

5.3 Epipolar Constraints and Fundamental Matrices

In the previous section, we used the correlation test to find the similarities between the points on the images. It should be noted that in the situation presented previously, we found the corresponding pixel of the conjugate pair by **visually** looking for its approximate position and then use the correlation test of equation 5.1 to find the exact pixel position. In actual correlation tests, however, we need to search the entire second image for the corresponding pixel of the conjugate pair, since we do not know it's approximate position.

With the size of a normal camera image (768×582 pixels in this case, or 768×291 for the un-interlaced images), it would be impossible to check every point on the second image to find the conjugate pair, as it would be quite process intensive. Thus, it is necessary to reduce this process down to just several dozens of pixels near the expected position of the conjugate pixel. This is possible by looking for the point along the corresponding epipolar line (see figure 2.1 for a reminder of the stereo geometry).

The geometry shown in figure 2.1 assumes that the epipolar lines are on the same

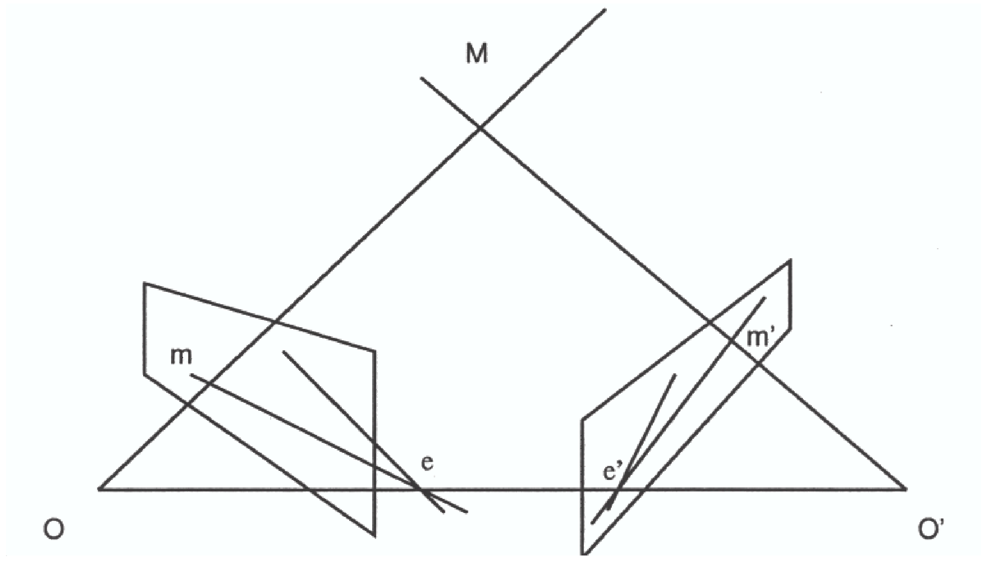


Figure 5.3: Epipolar Geometry [22]

horizontal line, in other words, the cameras are on a level plane and are at the exact height. However, in practice, it is impossible to place the cameras at an exact height with an accuracy of one pixel, which is usually a few micrometres. Furthermore, as the vehicle moves, the cameras will inevitably shift slightly and cause the corresponding epipolar lines to move. Therefore, it is necessary to find the epipolar lines for any potential conjugate pairs before the correlation tests. This is done using the epipolar constraints and fundamental matrix.

5.3.1 Basic Theory

The epipolar line-line and line-plane correspondences are projective and it is all there is to know about a stereo rig. Using the epipolar geometry, we can reduce the second image from a 2D plane to a 1D epipolar line that corresponds with the first image's point. The following steps shows the method of finding the epipolar lines [22] [30].

Let $m = (x, y, t)^t$ be the homogeneous coordinates of a point in the first image and $e = (u, v, w)$ be the coordinates of the epipole of the second camera in the first image. The epipolar line through m and e is represented by the vector $l = (a, b, c)^t = m \times e$.

The mapping $m \to l$ is linear and can be represented by a 3×3 rank 2 matrix C:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} yw - zv \\ zu - xw \\ xv - yu \end{bmatrix} = \begin{bmatrix} 0 & w & -z \\ -w & 0 & u \\ z & -u & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{5.2}$$

The mapping of epipolar lines l from image one to the corresponding epipolar lines l' in image two is a collineation defined on the 1D pencil of lines through e in image one. It can be represented (non-uniquely) as a collineation on the entire dual space of lines in $\mathbb{P}^2$. Let A be one such collineation: $l' = Al$. Let the matrix $F = AC$ the equation becomes:

$$l' = ACm = Fm \tag{5.3}$$

F is called the *fundamental matrix*. As C has rank 2 and A has rank 3 [22], F has rank 2. The right kernel of C – and hence F – is obviously the epipole e. The fact that all epipolar lines in the second image pass through e' (i.e. $e'^t.l' = 0$ for all transferred l') shows that the left kernel of F is e': $e'^t F = 0$.

F defines a bilinear constraint between the coordinates of corresponding image points. If m' is the point in the second image corresponding to m, it must lie on the epipolar line $l' = Fm$, and hence $m'^t.l' = 0$. The **epipolar constraint** can therefore be written:

$$m'^t F m = 0 \tag{5.4}$$

5.3.2 Estimating the Fundamental Matrix

To obtain the epipolar geometry, we need to estimate F. This can be done using some initial point correspondences between the images found using the correlation tests.

Assume that we have already found some matches $m \leftrightarrow m'$ between the images. Each correspondence provides a linear constraint on the coefficients of F: $m'^t F m = 0$. Expanding, we get:

$$xx' f_{11} + xy' f_{12} + x f_{13} + yx' f_{21} + yy' f_{22} + y f_{23} + x' f_{31} + y' f_{32} + f_{33} = 0$$

where the coordinates of m and m' are respectively $(x, y, 1)^t$ and $(x', y', 1)^t$ and f_{ij} are the elements of the matrix $\mathbf{F}$ in equation 5.4. Combining the equations obtained for each match gives a linear system that can be written as:

$$\mathbf{Af} = 0 \tag{5.5}$$

where $\mathbf{f} = \begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{bmatrix}$ is a vector containing the 9 coefficients of F, and each row of $\mathbf{A}$ is built from the coordinates m and m' of a single match. Since there are nine unknowns in the fundamental matrix (nine elements in the 3×3 matrix), we need 8 equations ((A) has eight rows) to obtain an unique solution [22].

This can be done by taking the Singular Value Decomposition (SVD) of F and setting the smallest singular value to zero. Basically, SVD decomposes F in the form

$$F = \mathbf{USV}^T$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, and $\mathbf{S}$ is diagonal matrix containing the singular values. These singular values σ_i are positive and in decreasing order. Therefore in our case σ_9 is guaranteed to be identically zero (8 equations for 9 unknowns) and thus the last column of $\mathbf{V}$ is the correct solution (at least as long as the eight equations are linearly independent, which is equivalent to all other singular values being non-zero). A complete mathematical proof of the SVD algorithm is shown in Appendix A.1. It is trivial to reconstruct the fundamental matrix $\mathbf{F}$ from the solution vector $\mathbf{f}$.

5.3.3 Normalising the Fundamental Matrix

The above method is standard, but if applied naïvely it is quite unstable. A typical image coordinate in a 512×512 image might be ~ 200. Some of the entries in a typical row of A are $xx' \sim 200^2$, others are $x \sim 200$, and the last entry is $1 \sim 1$, so there is a variation in size of $\sim 200^2$ among the entries of A, and hence of $200^4 \sim 2 \cdot 10^9$ among the entries of A^tA. This means that numerically, A^tA is extremely ill-conditioned: the solution contains an implicit least squares trade off, but it is nothing like the trade off we would actually like for maximum stability.

It has been pointed out that in practic it is very important to normalise the equations. This is for example achieved by transforming the image to the interval $[-1, 1] \times [-1, 1]$ so that all elements of the matrix A are of the same order of magnitude.

A simple solution to this is to normalise the pixel coordinates from $[0, 512]$ to $[-1, 1]$ before proceeding. This provides a well-balanced matrix A and much more stable and accurate results for F. In a practical implementation, a considerable effort must also be spent on rejecting false correspondences in the input data, as well as choosing data with high correlation values.

5.3.4 More Points

In the presence of noise, the fundamental matrix can easily be affected that the resulting matrix will give a huge error when applied in transformation. It is important, therefore, to choose the initial m and m' pairs to have a very high correlation value. When more points are available the redundancy should be used to minimise the effect of the noise [30]. In other words, we can add as many pairs as possible into the matrix $\mathbf{A}$ of equation 5.5.

Even then the error that is minimised is an algebraic error which has no real "physical" meaning. It is always better to minimise a geometrically meaningful criterion. In Fundamental Matrix, the physical meaning of the error is the difference between the epipolar line calculated from the fundamental matrix for the corresponding pixel and the actual pixel in the second image. Assuming that the noise on every feature point is independent zero-mean Gaussian with the same sigma for all points, the minimisation of the following criterion yields a maximum likelihood solution:

$$\mathcal{C}(F)\sum(D(m',\mathbf{F}m)^2+D(m,\mathbf{F}^Tm')^2) \tag{5.6}$$

with $D(m,l)$ the orthogonal distance between the point m and the line l. This criterion can be minimised through a Levenberg-Marquardt algorithm [31], which is discussed in detail in Appendix A.2. The results obtained through linear least-squares can be used for initialisation [30].

5.3.5 Cost Functions

The cost function method is discussed extensively by Chojnacki, Brooks and van den Hengel in their report [7]. The promise of using this method is to greatly reduce the effect of the noise on the value of the fundamental matrix. The method can be summarised as a weighted least squares (WLS) cost function is obtained by considering an optimal maximum likelihood formulation, and applying certain necessary approximations to solve the problem numerically. Various iterative schemes are tried for the problem.

Let's rewrite the epipolar constraint as shown in equation 5.4 as:

$$\theta^T\mathbf{u}(\mathbf{x})=0 \tag{5.7}$$

where

$$\theta=\begin{bmatrix}f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33}\end{bmatrix}^T$$

is the vector of parameters,

$$\mathbf{x}=\begin{bmatrix}x, & y, & x', & y'\end{bmatrix}^T$$

is the vector of variables, and

$$\mathbf{u}(\mathbf{x})=\begin{bmatrix}xx', & yx', & x', & xy', & yy', & y', & x, & y, & 1\end{bmatrix}^T$$

is the vector of carriers.

For simplicity, setting aside the ancillary constraint, given a cost function $J = J(\theta; x_1, \ldots, x_n)$, a corresponding estimate $\hat{\theta}$ is defined by

$$J(\hat{\theta}) = \min_{\theta \neq 0} J(\theta; x_1, \ldots, x_n)$$

A straightforward cost function is:

$$J_a(\theta; x_1, \ldots, x_n) = \frac{\sum_{i=1}^{n} \theta^T u(x_i) u(x_i)^T \theta}{(\theta_1 + \cdots + \theta_l)} \tag{5.8}$$

It leads to an *ordinary least squares estimate* $\hat{\theta}_{OLS}$ from the previous sections (i.e. the fundamental matrix found using the SVD method).

When some information about the measurement error is available, more involved cost functions can be devised. If the uncertainty of each data point x_i can be described by a $k \times k$ *covariance matrix* Σ_x, then an appropriate cost function is:

$$\begin{aligned} J_b(\theta; x_1, \ldots, x_n) = \min\{\sum_{i=1}^{n} (x_i - \overline{x}_i)^T \Sigma_{x_i}^{-1} (x_i - \overline{x_i}) : \theta \text{ and } \overline{x}_1, \ldots, \overline{x}_n \\ \text{such that } \theta^T u(\overline{x}_i) = 0 \text{ for each } i = 1, \ldots, n\} \end{aligned} \tag{5.9}$$

This function is justified via the principle of maximum likelihood and represents the minimum sum of the squared *Mahalanobis distances* between the data x_i and he 'corrected data' $\overline{x}_i$, each $\overline{x}_i$ lying on a hypersurface of the form $\{x : \theta^T u(x) = 0\}$. The minimiser for this function of J is called the *maximum likelihood estimate* and denoted $\hat{\theta}_{\mathrm{MLE}}$, which is the minimum adjustment required in the data in the sense of the Mahalanobis distance. The first order approximation of J_b is:

$$J_c(\theta; x_1, \ldots, x_n) = \sum_{i=1}^{n} \frac{\theta^T u(x_i) u(x_i)^T \theta}{\theta^T \partial_x u(x_i) \Sigma_{x_i} \partial_x u(x_i)^T \theta} \tag{5.10}$$

where, for any k vector $\mathbf{y}$, $\partial_x u(y)$ denotes the $l \times k$ matrix of the partial derivatives of $x \rightarrow u(x)$ evaluated at $\mathbf{y}$ and can be written as:

$$\partial_x u(y) = \begin{bmatrix} \frac{\partial u_1}{\partial x_1}(y) & \cdots & \frac{\partial u_1}{\partial x_k}(y) \\ \cdots & \cdots & \cdots \\ \frac{\partial u_l}{\partial x_1}(y) & \cdots & \frac{\partial u_l}{\partial x_k}(y) \end{bmatrix}$$

The various equations are for *continues* data, however, in real images the data are always *discrete*. Therefore, we'll need numerical schemes to find the WLS estimate in practise.

Fundamental Numerical Scheme

The *fundamental numerical scheme* can be regarded as a variant of the Newton-Raphson method. In the algorithm a guess of θ_k is made, then an improved guess can be obtained by picking a vector θ_{k+1} from that eigenspace of X_{θ_k}, which most closely approximates the null space of X_θ. The matrix X_θ satisfies $X_\theta \theta = 0$.

The value of X_θ is found using the following procedure: Let's first introduce a $l \times l$ matrix

$$A(x) = u(x)u(x)^T \tag{5.11}$$

For each $i = 1, \ldots, n$, let

$$A_i = A(x_i) \tag{5.12}$$

A straightforward estimator is derived from the cost function to be:

$$J_1(\theta; x_1, \ldots, x_n) = \frac{\theta^T M \theta}{(\theta_1 + \cdots + \theta_l)} \tag{5.13}$$

where

$$M = \sum_{i=1}^{n} A_i \tag{5.14}$$

Writing $J_1(\theta)$ as a short for $J_1(\theta; x_1, \ldots, x_n)$, the value of X_θ is found by:

$$X_\theta = \frac{M - J_1(\theta) I}{(\theta_1 + \cdots + \theta_l)} \tag{5.15}$$

An vector θ would satisfy the *variational equation*

$$X_\theta \theta = 0 \tag{5.16}$$

only if it falls into the null space of the matrix X_θ.

The steps for the fundamental numerical scheme is as the following:

1. Set $\theta_0 = \hat{\theta}_{\text{OLS}}$;
2. Assuming that θ_{k-1} is known, compute the matrix $X_{\theta_{k-1}}$;
3. Compute a normalised eigenvector of $X_{\theta_{k-1}}$ corresponding to the eigenvalue closest to zero and take this eigenvector for θ_k;
4. If θ_k is sufficiently close to θ_{k-1}, then terminate the procedure, otherwise increment k and return to step 2.

Iteratively Reweighted Least Squares Scheme

In this scheme, the noise in the data is taken into account by using the covariance matrix Σ. The value for the matrix A is still the same as that of equations 5.11 and 5.12.

Let the matrix B take into the account any $k \times k$ matrix Σ:

$$B(y, \Sigma) = \partial_x u(y) \Sigma \partial_x u(y)^T \tag{5.17}$$

and for each $i = 1, \ldots, n$, let

$$B_i = B(x_i, \Sigma_{x_i}) \tag{5.18}$$

Now the matrix M as shown in equation 5.14 is written as:

$$M_\theta = \sum_{i=1}^{n} \frac{A_i}{\theta^T B_i \theta} \tag{5.19}$$

The iteratively reweighted least squares scheme has the procedure:

1. Set $\theta_0 = \hat{\theta}_{\text{OLS}}$;
2. Assuming that θ_{k-1} is known, compute the matrix $M_{\theta_{k-1}}$;
3. Compute a normalised eigenvector of $M_{\theta_{k-1}}$ corresponding to the smallest (non-negative) eigenvalue and take this eigenvector for θ_k;

4. If θ_k is sufficiently close to θ_{k-1}, then terminate the procedure, otherwise increment k and return to step 2.

First Order Renormalisation Scheme, version I

Renormalisation is a different type of modification to M_θ based on statistical consideration. The requirement is that the modified or *renormalised* M_θ be *unbiased* in some sense. Using the renormalised M_θ an equation an be formulated to define an estimate of θ. The first order renormalisation will be applicable under the condition that noise in an appropriate statistical model is small.

The unbiased M_θ can be written as:

$$Y_\theta = \sum_{i=1}^{n} \frac{A_i - J_6(\theta)B_i}{\theta^T B_i \theta} = M_\theta - J_6(\theta)N_\theta \tag{5.20}$$

where M_θ is given in equation 5.19 and N_θ is defined by:

$$N_\theta = \sum_{i=1}^{n} \frac{1}{\theta^T B_i \theta} B_i \tag{5.21}$$

and the cost function $J_6(\theta)$ is:

$$J_6(\theta; x_1, \ldots, x_n) = \frac{1}{n}\sum_{i=1}^{n} \frac{\theta^T A_i \theta}{\theta^T B_i \theta} = \frac{\theta^T M_\theta \theta}{\theta^T N_\theta \theta} \tag{5.22}$$

The renormalisation equation is:

$$Y_\theta \theta = 0 \tag{5.23}$$

which is analogous to the variation equation is non-linear and unlikely to admit solutions in closed form.

The first order renormalisation scheme, version I (FORI) has the following procedure:

1. Set $\theta_0 = \hat{\theta}_{\mathrm{OLS}}$;

2. Assuming that θ_{k-1} is known, compute the matrix $Y_{\theta_{k-1}}$ using equation 5.20;
3. Compute a normalised eigenvector of $Y_{\theta_{k-1}}$ corresponding to the eigenvalue closest to zero and take this eigenvector for θ_k;
4. If θ_k is sufficiently close to θ_{k-1}, then terminate the procedure, otherwise increment k and return to step 2.

First Order Renormalisation Scheme, Version II

The FORI scheme can be slightly modified to give the first order renormalisation scheme, version II. The cost function in this scheme is given by:

$$J_7(\theta, \xi; x_1, \dots, x_n) = \frac{\theta^T M_\xi \theta}{\theta^T N_\xi \theta} \tag{5.24}$$

Let,

$$Y_{\theta,\xi} = \sum_{i=1}^{n} \frac{A_i - J_7(\theta, \xi) B_i}{\xi^T B_i \xi} = M_\xi - J_7(\theta, \xi) N_\xi \tag{5.25}$$

where $J_7(\theta, \xi)$ is an abbreviation of $J_7(\theta, \xi; x_1, \dots, x_n)$. It can be shown that this satisfies the renormalisation equation of 5.23, where Y_θ is $Y_{\theta,\xi}$ for each θ and each $\xi \neq 0$. Also,

$$J_7(\theta, \theta) = J_6(\theta) \tag{5.26}$$

and

$$Y_{\theta,\theta} = Y_\theta \tag{5.27}$$

giving

$$Y_{\theta_{k-1}} = Y_{\theta_{k-1},\theta_{k-1}} = M_{\theta_{k-1}} - c_{k-1} N_{\theta_{k-1}} \tag{5.28}$$

where

$$c_{k-1} = J_7(\theta_{k-1}, \theta_{k-1}) \tag{5.29}$$

Under certain approximation, c_k can be updated directly from c_{k-1} rather than using the above equation, and it is given by:

$$c_k = c_{k-1} + \frac{\lambda_k}{\theta_k^T N_{\theta_{k-1}} \theta_k} \tag{5.30}$$

Therefore, version II of the first order renormalisation scheme has the following procedure:

1. Set $\theta_0 = \hat{\theta}_{\mathrm{OLS}}$ and $c_0 = 0$;
2. Assuming that θ_{k-1} and c_{k-1} are known, compute the matrix $M_{\theta_{k-1}} - c_{k-1}N_{\theta_{k-1}}$;
3. Compute a normalised eigenvector of $M_{\theta_{k-1}} - c_{k-1}N_{\theta_{k-1}}$ corresponding to the smallest eigenvalue λ_k and take this eigenvector for θ_k, then define c_k by equation 5.30;
4. If θ_k is sufficiently close to θ_{k-1}, then terminate the procedure, otherwise increment k and return to step 2.

Algebraic Least Squares Estimator

In another paper, Chojnacki et al. proposed a fast algebraic least square estimator based on the same principle [8]. The matrix A in this scheme is the same as those shown in equations 5.11 and 5.12. The matrix B however, is slightly different from those shown in equations 5.17 and 5.18 and is written as:

$$B_i = \partial_x u(x_i)\partial_x u(x_i)^T \tag{5.31}$$

The cost function for the ALS estimator is the same as those of equation 5.15, and the θ for which J_{ALS} is minimal is denoted $\hat{\theta}_{\mathrm{ALS}}$. Adopting a maximum likelihood approach, the cost function for approximated maximum likelihood (AML) estimator is:

$$J_{\mathrm{AML}}(\theta; x_1, \ldots, x_n) = \sum_{i=1}^{n} \frac{\theta^T A_i \theta}{\theta^T B_i \theta} \tag{5.32}$$

and the associated θ is denoted $\hat{\theta}_{\mathrm{AML}}$.

The variational equation is phrased as:

$$X'\theta = 0 \tag{5.33}$$

where

$$X' = \sum_{i=1}^{n} \frac{A_i}{\theta^T B_i \theta} - \sum_{i=1}^{n} \frac{\theta^T A_i \theta}{(\theta^T B_i \theta)^2} B_i \tag{5.34}$$

In practise $\hat{\theta}_{\text{AML}}$ has to be found numerically, and assume $\hat{\theta}_{\text{AML}}$ lies close to $\hat{\theta}_{\text{ALS}}$. This assumption is to increase the chances that any candidate minimiser obtained via a numerical method seeded with $\hat{\theta}_{\text{ALS}}$ coincides with $\hat{\theta}_{\text{AML}}$.

The fundamental numerical scheme for this estimator is:

1. Set $\theta_0 = \hat{\theta}_{\text{ALS}}$;
2. Assuming that θ_{k-1} is known, compute the matrix $X'_{\theta_{k-1}}$;
3. Compute a normalised eigenvector of X'_{k-1} corresponding to the eigenvalue closest to zero and take this eigenvector for θ_k;
4. If θ_k is sufficiently close to θ_{k-1}, then terminate the procedure, otherwise increment k and return to step 2.

5.3.6 Summary and Discussion

In summary, the procedure for estimating the epipolar geometry is:

- Extract points from the images
- Using correlation test to find appropriate conjugate pairs (typically use pairs with correlation coefficients of over 97%)
- Build the matrix A using the conjugate pairs, we'll need at least 8 pairs but use as many pairs as possible to give less errors
- Normalise all the values of the pixels to [-1,1]
- Use the SVD algorithm to estimate the first guess for the fundamental matrix

- Minimise the errors using Levenberg-Marquard algorithm
- Optimise the result using the Cost Function algorithms.

Theoretically, the algorithms would return a matrix that satisfies the epipolar constraints, which says:

$$\mathbf{m_L^T F m_R} = 0$$

where $\mathbf{m_L} = [x_L, y_L, 1]^T$ and $\mathbf{m_R} = [x_R, y_R, 1]^T$ are any conjugate pairs for the stereo images. However, upon closer inspection, it was found that fundamental matrix is very easily affected by noise in the data. Using over 20 conjugate pairs each with correlation of 97% and above, the resulting matrix would still give the value in the order of 10^3 when substituted into the equation above, using conjugate pairs from the input data. The result improves slightly when the input data have correlation of 98% and above, but would still give the result of $\mathbf{m_L^T F m_R}$ in the order of several hundreds.

Furthermore, although it was claimed that the cost function methods will give a satisfactory result for the fundamental matrix, in practice it does not vary the resulting matrix much.

5.4 MATLAB Calibration Toolbox

In the latest release (9th October 2002) of the MATLAB Calibration Toolbox [3], an undocumented feature of Stereo Calibration is added. This toolbox calibrates the intrinsic and extrinsic parameters of stereo cameras.

The toolbox assume that the two cameras have first been calibrated with the same view of a 3D location. Furthermore, the same points on the images are selected during calibration. Of course, the same number of images are selected for use in the calibration of both cameras.

These criteria are easily met in my case, as the left and right images at a given time is save onto a single file. Therefore, in calibration of each camera, care need to be taken not only of choosing a 'good' image, but also one which is used in both calibration processes.

The inputs to the stereo calibration functions are the calibration results from the left and the right cameras, generated by the standard calibration toolbox on the two cameras individually. The main result from the stereo calibration are the relative rotation matrix `R`, and the translation vector `T` of the right camera with respect to the left camera.

Consider a point P of coordinates $\mathbf{X_L}$ and $\mathbf{X_R}$ in the left and right camera reference frames respectively. From the discussion on epipolar geometry in section 5.3, we know that these two coordinates are related in a certain way. In fact, using the output of the calibration toolbox, we can write the following rigid motion transformation for $\mathbf{X_L}$ and $\mathbf{X_R}$:

$$\mathbf{X_R} = \mathbf{R}\mathbf{X_L} + T \tag{5.35}$$

Therefore, $\mathbf{R}$ and T fully describe the relative displacement of the two cameras.

Now, introducing the *Essential Matrix* $\mathbf{E}$ [11] [21]:

$$\begin{aligned} \mathbf{E} &= T \times \mathbf{R} \\ &= [\mathbf{T}]_\times \mathbf{R} \end{aligned} \tag{5.36}$$

where $[\mathbf{T}]_\times$ is the "cross product matrix" of T and defined as:

$$T = \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix} \qquad [\mathbf{T}]_\times = \begin{bmatrix} 0 & -T_Z & T_Y \\ T_Z & 0 & -T_X \\ -T_Y & T_X & 0 \end{bmatrix}$$

The transformation of $\mathbf{X_L}$ and $\mathbf{X_R}$ in equation 5.35 can also be written using the above relationship as:

$$\mathbf{X_L'}\mathbf{E}\mathbf{X_R} = 0 \tag{5.37}$$

The vectors $\mathbf{X_L}$ and $\mathbf{X_R}$ are the real world coordinates of the point P in the left camera and right camera reference frame respectively. They are related to the camera coordinates $m = [x, y, 1]'$ as:

$$\begin{aligned} m_i &= \mathbf{K}_i\mathbf{X}_i \\ \mathbf{X}_i &= \mathbf{K}_i^{-1}m_i \end{aligned} \tag{5.38}$$

where the matrix $\mathbf{K}$ contains the intrinsic parameters of a camera in the following arrangement:

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

where f is the focal length and C is the centre of image.

Substituting the relationships in equation 5.38 into essential matrix equation 5.37, we have the following:

$$\begin{aligned} \mathbf{X'_R E X_L} &= 0 \\ (\mathbf{K}_R^{-1}m_R)^T\mathbf{E}(\mathbf{K}_L^{-1}m_L) &= 0 \\ m_R^T(\mathbf{K}_R^{-1})^T\mathbf{E}\mathbf{K}_L^{-1}m_L &= 0 \\ m_R^T\mathbf{F}m_L &= 0 \end{aligned} \tag{5.39}$$

Equation 5.39 is simplest the epipolar constraints as shown in equation 5.4. Therefore the fundamental matrix $\mathbf{F}$ and the essential matrix $\mathbf{E}$ are related using the following:

$$\mathbf{F} = (\mathbf{K}_R^{-1})^T\mathbf{E}\mathbf{K}_L^{-1} \tag{5.40}$$

Therefore, using the intrinsic and extrinsic parameters from the stereo calibration process we can find the fundamental matrix. We found that resulting value of $m_R^T\mathbf{F}m_L$ using the fundamental matrix calculated from the calibration parameters are within an order of magnitude. The result, while not satisfactory is still a great improvement from the fundamental matrix calculation in the last section.

Chapter 6

Disparity Map and Error Calculation

The final procedure in Stereo Vision algorithm is the building of the *disparity map*. This is the process by which the distances to the objects identifies in the previous steps are found and stored for use by the vehicle. In this chapter, we'll discuss the shortcomings of building the disparity map, and the errors involved in the 3D coordinates of the objects from the many variables in the calculation.

6.1 Disparity Map

Disparity is the distance between points of a conjugate pair when the two images are superimposed [17]. To build the disparity map we need to first find all the conjugate pairs of known corners in the image.

When the corners from the left image are found by the corner detector, the corresponding corners from the right image can be find using the fundamental matrix **F**:

$$l = \mathbf{F} m_L \tag{6.1}$$

where $m_L = [x_L, y_L, 1]'$ is the image coordinates of the object, and $l = [a, b, c]'$ are the coefficients of the line where m_R lies. The equation of the line is:

$$y = -\frac{a}{b}x - \frac{c}{b} \tag{6.2}$$

Using the correlation equation shown on page 60 and search along the line, we can find m_R assuming it is the one pixel that gives the highest correlation with m_L. A decision also need to be made of the thresholding correlation value at which we can judge that m_R is related to m_L. This is because not every point on the left image will have a corresponding pixel in the right image. Furthermore, we know that the x coordinates of m_R is to the left of the the x coordinates of m_L, however as the distance to the object increases, it will shift towards the right and eventually move to the right of m_L. The search pattern for m_R will ignore those pixels on the right hand side of m_L (see section 6.3 for a discussion on the reason for this decision), thus even if an object appears in both images, it's coordinates in the right image might not be part of the search pattern.

When the conjugate pairs of points are detected then we can use the equation for stereo imaging on page 12 and find the points' x, y and z co-ordinates in space. However, it should be noted that as the focal length of the cameras are inevitably slightly different, equation 2.3 need to be altered slightly as a result to give:

$$z = \frac{b}{\frac{x_l - x_{cl}}{f_{xl}} - \frac{x_r - x_{cr}}{f_{xr}}} \tag{6.3}$$

And thus able to generate a 3D map such as that shown on figure 6.1.

A two dimensional representation for the three dimensional space is always slightly misleading especially when there are no connections between the points. Therefore, figure 6.2 shows the views from the top and front of the same map, giving a better understanding of the data. Note that the origin of the x axis is the central axis of the Ute, the origin of the y and z axis are at the position of the cameras (approximately $2m$ and $3.2m$ above ground and from the very front of the Ute respectively).

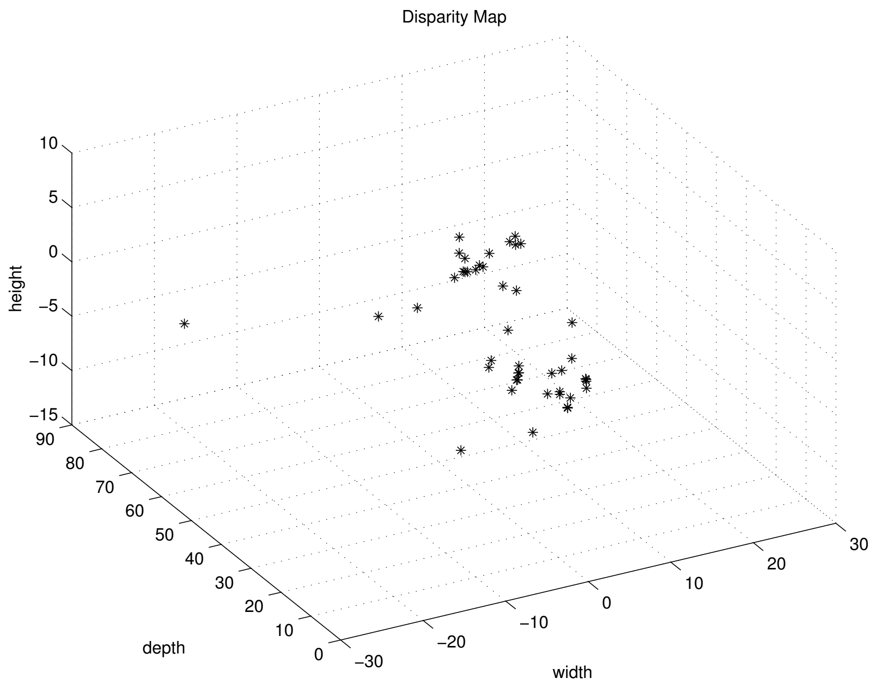
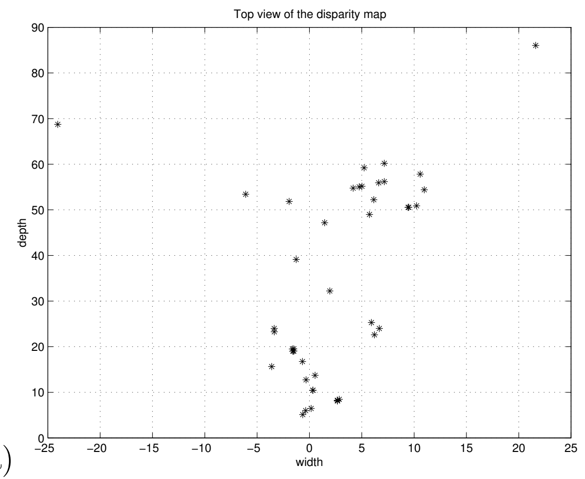


Figure 6.1: The disparity map of the images shown through this thesis.

(a)


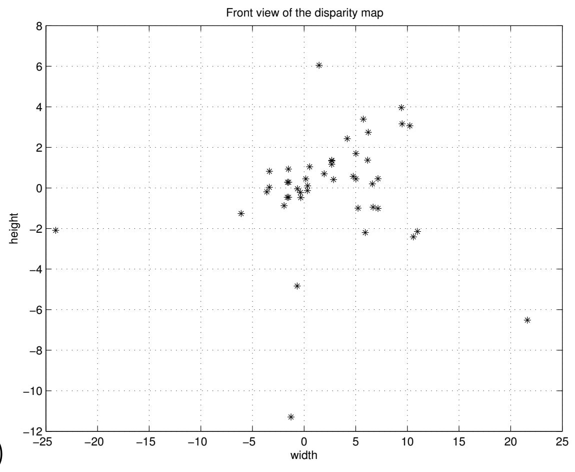
(b)


Figure 6.2: Different views of the disparity map shown on figure 6.1: (a) top view, and (b) view from the Ute

From the figures, we can see several shortcomings of the algorithm in finding the disparity map. The most obvious error is the Electrical Engineering building (refer to figure 3.7 on page 20 for the colour photo of the image, the Electrical Engineering building is the white building at the background with the satellite dish on the roof), which we know is at least over $500m$ from the site. However from the patterns on the figures, it seems that the program find it is to be around 50 to $60m$ in the distance.

This is because the corners of the 'windows' on the buildings are very similar in its individual pixel intensity and the surrounding patterns. Therefore in the correlation process, the corners can easily be mismatched and thus leading to the wrong distance calculation.

The most important issue with vision is the speed of the algorithm. We require the algorithm to run in real time to give the vehicle an idea of its surrounding. The map on figure 6.1 is generated using 50 corners from the left image, the search pattern is all the x values to the left of each m_L at a width of $\pm$ 10 pixels from the expected y value. The algorithm for the disparity map for these corners took slightly less than $600s$ to run in MATLAB.

Furthermore, any real life equipments will inevitably have errors associated with the result, in this case, it is the error in the calculated (x, y, z) position of objects from its real position. The estimation of the percentage error of the result is very important for the vehicle to navigate safely.

For the rest of this chapter, we will look into the errors resulted from the distance calculations both theoretically and experimentally. The experimental validation of any on board system requires that they go through various testing procedures and verify the data received from the sensor from a reliable control source.

The error calculation will also enable us to select the best camera parameters for use in the stereo system. There are three variables associated with the system: baseline distance, focal length and camera field of view.

6.2 Experiment Setup

In finding the error of the resulting distance, we need to know the position of the object exactly to accurately check the result from the stereo system. The control sensor used in the experiment is the SICK laser, which has been tested vigorously in the past therefore it is an excellent system to use for the verification of the results. On figure 1.1 we can see two lasers mounted at the front of the Ute, one facing forwards and the other facing upwards. Since the stereo system is used mainly for finding the depth to a distant object, only the forward facing laser will need to be used.

For the testing process, poles are set up on an open area and the distance to these poles will be measured by the laser. The poles used can be seen at the back of figure 1.1. The laser system will provide the x and z position of the poles (i.e. the horizontal distance from the central axis of the Ute and the depth of the object). It will not give the height of any object, this however is not a very big problem as the poles are of known height and thus can be checked easily.

Furthermore, there are inevitable angle bias between the laser and the stereo system. In other words, the horizontal axis as seen by the laser (from its $0°$ to $180°$) is not exactly parallel to the horizontal axis of the stereo system. Also, we do not know the exact distance between the origin of the stereo system and the laser. Therefore, it is best to compare the absolute distance measured by the sensors, that is the distance between the poles.

It should also be noted that the laser system on board has a system error of roughly $\pm 5cm$ [32]. Also it is possible for the laser to measure one side of the pole while the stereo camera to detect the other side, thus it will add a further $10cm$ to the errors.

6.3 Baseline

From figure 3.2 we can see that the cameras can be set at various baselines from 0.1m to 1.1m at 0.1m increment. However, it is best to keep the cameras at either side of the central axis of the Ute. Therefore, only three different baselines were tested, they are 0.7m, 0.9m and 1.1m respectively.

Although the focus of the camera has not changed during the tests for each baseline, the various calibration values of the camera would inevitably vary slightly. Therefore, the camera calibration procedure discussed in section 4.1 is used for finding these values and are shown on table 6.1.

Calibration variables	$b = 0.7m$		$b = 0.9m$		$b = 1.1m$	
	left	right	left	right	left	right
x focal length, f_x	857.23	866.71	864.01	865.32	903.62	867.71
f_x error	22.85	11.73	11.84	12.04	41.17	12.41
y focal length, f_y	857.62	866.10	863.43	865.22	901.93	868.03
f_y error	22.18	11.80	11.99	12.16	39.46	12.54
x centre, c_x	359.20	363.14	362.76	360.85	366.83	359.65
c_x error	7.04	2.48	4.24	2.88	11.26	3.77
y centre, f_y	265.81	273.90	261.87	266.04	239.51	268.48
c_y error	9.10	6.51	6.23	8.03	16.19	6.95

Table 6.1: The focal length and centre of view for the two cameras at various baseline values.

Calibration variables	$b = 0.7m$		$b = 0.9m$		$b = 1.1m$	
	left	right	left	right	left	right
x focal length, f_x	864.08	858.74	871.00	865.36	867.18	861.42
f_x error	6.76	6.69	7.76	7.64	9.70	9.78
y focal length, f_y	863.46	857.68	870.11	864.04	866.91	860.86
f_y error	6.69	6.67	7.76	7.51	9.62	9.70
x centre, c_x	365.58	364.51	364.11	356.18	366.72	362.13
c_x error	5.18	1.71	2.64	6.15	6.84	2.46
y centre, f_y	269.06	278.94	270.55	277.84	265.45	274.80
c_y error	3.78	3.20	2.73	4.57	3.63	3.69

Table 6.2: The focal length and centre of view for the two cameras at various baseline values after re-calibration using the Stereo Calibration Toolbox on the values in table 6.1.

It should be noted that some of the error values from the calibration are quite large, even though they are equals to 3 standard deviations. However, these can be reduced using the stereo calibration toolbox discussed in section 5.4. As the toolbox takes into account that the cameras are related and the images from the left and right cameras are of the same three dimensional space view, the error for the intrinsic values for each camera after stereo calibration would reduce greatly. Table 6.2 shows the optimised calibration values, note how all the error haves have reduced to less than 10 pixels.

6.3.1 Theoretical Calculations

The theoretical calculations are made to check for the following:

- Maximum distance;
- Error due to corner detection;
- Distance error due to the calibrated intrinsic parameters.

Maximum distance

From figure 2.1 we assume that the horizontal position of an object in the right image will always be to the left of the same object in the left image. However, as the distance to the object increases, the difference between the pixel coordinates of that object in the two images decreases. Eventually, the pixel coordinates in the images will 'switch side' and the horizontal pixel position of the object on the right image will be larger than the corresponding one on the left image.

From equation 6.3, we can see that due to the difference in the focal length and centre of image between the two cameras, the distance to an object depends not only on the value of $x_r - x_l$ but both x coordinates of the pixel on the images.

Tables 6.3, 6.4 and 6.5 show the calculated z for a stereo baseline of $0.7m$, $0.9m$

Pixel difference	**z value** (m)			
	$x_l = 50$	$x_l = 250$	$x_l = 400$	$x_l = 650$
-3	155.0932	227.7320	351.0412	3598.3
-1	320.4497	939.8459	-2090.1	-327.9532
0	686.3168	-1667.9	-466.8503	-212.1891
1	-4842.4	-441.8683	-262.7724	-156.8298
2	-534.7396	-254.6681	-182.8442	-124.3797
3	-282.9951	-178.8832	-140.1994	-103.0560
4	-192.4116	-137.8588	-113.6847	-87.9738
5	-145.7566	-112.1408	-95.6039	-76.7425

Table 6.3: The calculated value of z for $x_r \geq x_l$, for baseline $b = 0.7m$ at various positions of x_l.

Pixel difference	**z value** (m)			
	$x_l = 50$	$x_l = 250$	$x_l = 400$	$x_l = 650$
-10	190.0476	277.8263	425.0761	3643.9
-1	-158.8811	-125.6837	-108.6563	-88.6413
0	-131.9610	-108.2196	-95.3532	-79.5836
1	-112.8416	-95.0168	-84.9524	-72.2053
2	-98.5613	-84.6852	-76.5973	-66.0791
3	-87.4894	-76.3801	-69.7385	-60.9111
4	-78.6539	-69.5584	-64.0071	-56.4929
5	-71.4392	-63.8554	-59.1463	-52.6723

Table 6.4: The calculated value of z for $x_r \geq x_l$, for baseline $b = 0.9m$ at various positions of x_l.

Pixel difference	**z value** (m)			
	$x_l = 50$	$x_l = 250$	$x_l = 400$	$x_l = 650$
-8	171.7694	226.2422	296.8457	618.5793
-1	-638.7286	-337.0036	-248.8420	-173.2874
0	-381.5412	-248.5914	-197.0850	-146.4965
1	-272.0136	-196.9278	-163.1510	-126.8804
2	-211.3439	-163.0432	-139.1860	-111.8972
3	-172.8022	-139.1076	-121.3597	-100.0789
4	-146.1496	-121.3001	-107.5812	-90.5186
5	-126.6201	-107.5343	-96.6123	-82.6256

Table 6.5: The calculated value of z for $x_r \geq x_l$, for baseline $b = 1.1m$ at various positions of x_l.

and 1.1m for the situations of when the image of an object from the right camera will have a greater horizontal position than that of the corresponding image from the left camera.

From the table, we can see that for $x_r > x_l$ (even when $x_r < x_l$ and the two values are very close) the calculated depth is unreliable since it is impossible to have an object with negative depth and still in the images. From the tables, we can also see that the maximum "reliable" distance calculated varies as the position of x_l moves. It is safe to say, however, that any conjugate pair where $x_r > x_l - 10$ should be discarded. Therefore, giving the system a maximum "reliable" depth perception of over 150m.

This distance is reasonable is the feature found is a building or other large objects where its size on the images will have enough pixels length and width wise for the detectors to identify it. It is at the same time not possible for objects such as the poles to be detected at these distance as they would be reduced to approximate 1 or 2 pixels in width, which is not enough for the detectors.

Corner Detection

From the figures 4.17 and 4.20 of the results from Harris and SUSAN detector, we know that the corners detected by the algorithm do not always lie directly on top of a said corner. This is mostly due to two reasons: Firstly, any feature recorded by the camera (an edge or a corner) upon close inspection would not have very sharp change of intensity, the colour will inevitably blur into the next few pixels. Secondly, the detector is not perfect and thus misses the actual pixel where the corner is.

Thus, we need to have an idea of the differences to the final depth value when the detected corner is several pixels away from the 'real' corner. Again, we will calculate these theoretically using the parameters shown on table 6.2.

Table 6.6 shows the difference in the depth value if the detected corner is not

Distance (m)	**Pixel difference**	**Difference in z value** (m)		
		$b = 0.7m$	$b = 0.9m$	$b = 1.1m$
10	1	0.0010	0.0008	0.0007
	2	0.0021	0.0017	0.0014
	4	0.0042	0.0033	0.0028
	8	0.0083	0.0067	0.0056
	16	0.0167	0.0133	0.0111
20	1	0.0042	0.0033	0.0028
	2	0.0084	0.0067	0.0056
	4	0.0167	0.0133	0.0113
	8	0.0334	0.0266	0.0226
	16	0.0670	0.0534	0.0452
30	1	0.0094	0.0075	0.0064
	2	0.0188	0.0149	0.0128
	4	0.0377	0.0299	0.0257
	8	0.0755	0.0599	0.0514
	16	0.1514	0.1200	0.1030

Table 6.6: The difference in depth when the detected corner is not exactly on top of the actual feature, for various distances of objects at the different baselines. Assuming the object is directly in front of the central axis of the vehicle.

exactly at where the actual corner is. From the table, we can see that the error caused by varying the position of the detected corner is very small. Even at $30m$, using a baseline value of $0.7m$ and the detected corner is 16 pixels from the actual corner on the image, the resulting error is still only $15cm$.

Recalling from the discussion in Chapter 4 on the distortion of the image due to the properties of the camera, and whether it is a worth while concept to un-distort the image. It was decided that un-distortion is an unnecessary process because the algorithm takes approximately $15min$ to run for both images, thus making it unrealistic for this application. From the error calculation done here, it is safe to say that the omission of un-distortion process in image rendering is sound, as the resulting error in the depth of an object is very insignificant.

	$b = 0.7m$	$b = 0.9m$	$b = 1.1m$
$\Delta z/z$ (%)	3.447	4.225	4.798

Table 6.7: The percentage error of the depth calculation for each baseline from the errors of camera intrinsic parameters.

Intrinsic Parameters

During camera calibration, errors will result in finding the various parameters. These errors were also given in the result and they are shown in table 6.2 for the set up for this thesis. These errors will accumulate into the resulting depth value of the system. To calculate the percentage error as a result, let's write the stereo vision equation as shown on page 79 here with the errors for the intrinsic parameters:

$$z \pm \Delta z = \frac{b}{\frac{x_l - (x_{cl} \pm \Delta x_{cl})}{f_{xl} \pm \Delta f_{xl}} - \frac{x_r - (x_{cr} \pm \Delta x_{cr})}{f_{xr} \pm \Delta f_{xr}}} \tag{6.4}$$

The percentage error of the result is approximately the sum of the percentage errors in the components [29]. Thus:

$$\frac{\Delta z}{z} = \frac{\Delta x_{cl}}{x_{cl}} + \frac{\Delta x_{cr}}{x_{cr}} + \frac{\Delta f_{xl}}{f_{xl}} + \frac{\Delta f_{xr}}{f_{xr}} \tag{6.5}$$

Using the values shown on table 6.2, the percentage error for $\Delta z/z$ for the various baselines are shown on table 6.7. The percentage errors are less than 5% for all the baseline, this equates to less than $0.5m$ at $z = 10m$. This value looks much larger than the other errors associated with the depth calculation, however, it should be remembered that the errors given by the calibration toolbox are of 3 standard deviations (90% of the samples), thus more than half of the times the percentage error is less than 2% (i.e. less than $0.2m$ at $z = 10m$).

6.3.2 Experimental Calculations

Experimentally verifying the accuracy of the stereo system is very important in determining its practicality of the sensor. This is because experimental results are often very different from theoretical calculations. Experimentally checking the theoretical error calculations using both the stereo system as well as an independent system would further validate the sensor.

As discussed in section 4.3 the camera records its images one row at a time, and it interlaces its recorded images. Therefore when the vehicle is moving, the recorded image will not have smooth edges due to the interlacing, furthermore, the bottom rows of the image will be shifted away from the top rows. Thus to avoid complications, all the data recorded for use in the experiments are those where the vehicle is kept at stationary.

The second independent control sensor used by the experiment is the SICK laser, which has an error of $\pm 5cm$ as discussed previously. The laser sweeps over an area of $180°$, and the data is taken every $0.5°$. Figure 6.3 shows a map of the surrounding area of the Ute generated from the data collected by the laser. The corresponding image gathered by the right camera is shown on figure 6.4 as a comparison. Note how the laser has missed some of the poles in the image, this is most likely be caused by the angular displacement of the particular pole is less than $0.5°$.

Many sets of data were taken for the experiment, these includes sets of all three baselines, for each baseline value, the data are taken with the Ute close up and far away from the poles to check for the depth of the stereo system for finding objects with various z values. It should also be noted that although the laser has a maximum range of $80m$ it is reliable up to $20m$. Therefore, any coordinates found that are more than $20m$ from the vehicle can not be double checked.

As discussed earlier in this chapter, there is a slight angle bias between the laser and the stereo system. Furthermore, the exact relationship between the position of the laser and the stereo vision rig is not known. Therefore it is useless to compare the relative position of the poles as measured by the laser and the stereo vision algorithms.

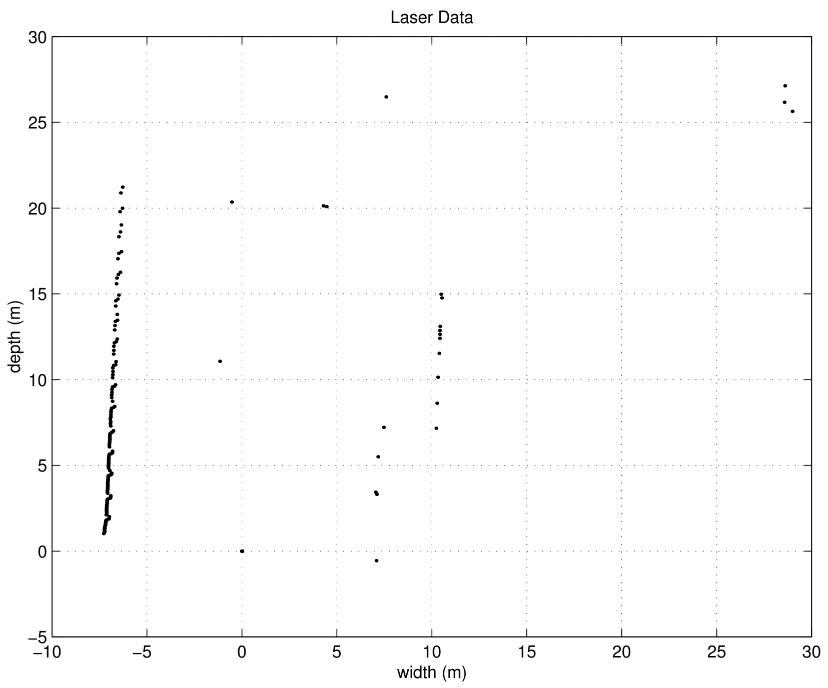


Figure 6.3: A typical map of the surrounding area generated by the laser on board the Ute.

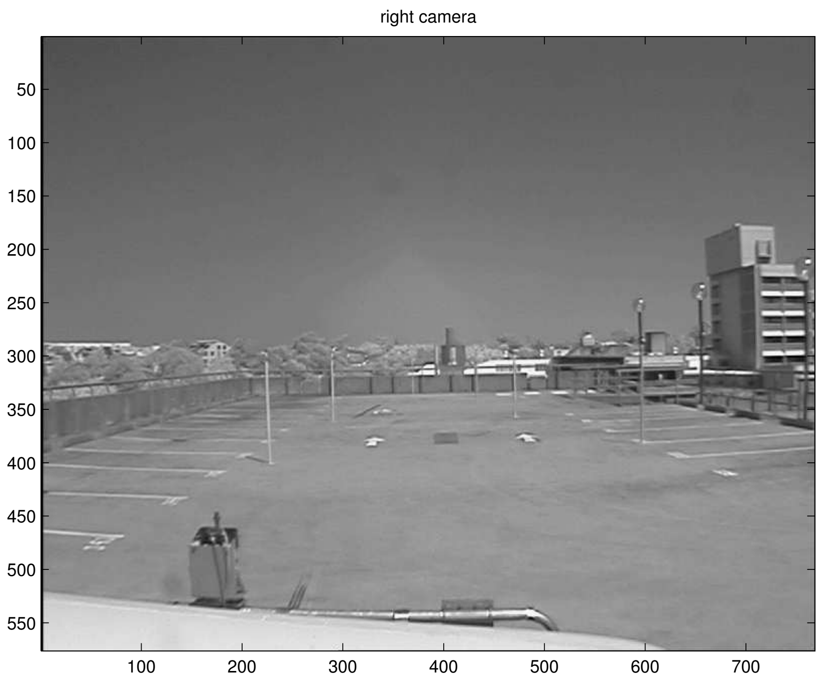


Figure 6.4: The view from the right camera of the same scene shown on the map generated by the laser data on figure 6.3.

The solution is to compare the absolute distances between the poles, since this will not involve any local frame of reference. Using this method, the various distances between the poles were calculated and compared. It was found that even with some poles at $20m$, the difference between the laser and stereo absolute distances are around $20cm$. Keeping in mind that the laser also has its own associated errors, this result shows that the stereo vision system would give reliable depth of objects with very little error.

6.4 Focal Length and Angular Field of View

From the stereo vision equation 2.3, we can see that the change in the focal length will affect the depth of an object given the same image coordinates. Specifically, increase in the focal length will increase the resulting depth. Therefore, the variation in the focal length of the system will change the maximum perceived "reliable" depth of the stereo vision system.

Error-wise, from section 6.3 the most significant error in the final result is in the calibrations of the camera system, the differences in pixel coordinates contributes only a fraction of the total error. Therefore, the variable of focal length in the system is not very important in terms of reducing the errors in the final 3D coordinates.

The last variable, angular field of view, only affects how much data the camera can take in and is something that need to be considered when evaluating the mission objective of the vehicle. In other words, is it necessary to know whether there are objects besides the vehicle. However, recall that the camera records data with a distortion due to the pin-hole effect, a large angular field of view might not be very desirable. The situation can be better solved if the cameras are in a pan-and-tilt unit. Therefore, the angular field of view was not be adjusted and kept at the constant value of approximately $45°$.

Chapter 7

Conclusion

7.1 Stereo Vision Procedure

The procedure used for Stereo Vision algorithm as used in this thesis is:

1. Open image;
2. Split the data into its left and right camera components;
3. Un-interlace the image to remove every second row;
4. Use the Harris corner detector to find the features on the left image;
5. For each corner find in the left image, use the fundamental matrix to find the corresponding epipolar line on the right image and use the correlation test to find the conjugate pair;
6. For every conjugate pair identified, find the associated coordinates of the object in three dimensional space;
7. Thus building the disparity map for the surrounding of the vehicle.

7.2 Conclusion

The aim of this thesis is to develop a stereo vision system, as well as to investigate the practicality of such a system for the High Speed Vehicle project. The section above shows the procedure that was developed to implement the stereo vision system to build a map of the three dimensional space surrounding the vehicle. I have showed that by using the various algorithms discussed, we can employ two camera to generate a disparity map.

The various feature detection algorithms presented in this thesis do give very good results in which most features are identified. Although fundamental matrix calculation is one of the most difficult task of the project, a method of finding the optimal fundamental matrix is ultimately utilised. Furthermore, it was found that the errors involved in the resulting object 3D coordinates are small compared to their distances from the vehicle. And the largest error in the system is from the camera calibration process, which can be reduced greatly when the images are chosen with care.

There are however many shortcomings in the stereo system for use as a real-time sensor in an unknown environment. The biggest and most important of these is the speed at which a map can be generated by the system. Currently, the entire process from loading the images, to filtering, to corner detection, to finding the conjugate pairs, to calculating coordinates and thus generating the map will take approximately $15mins$. This is obviously far too long for any practical use as a real-time sensor, where the turn-over period should be in milliseconds to provide any useful information.

Also due to missed corners from the detectors, not all of the object seen in the images are presented in the final disparity map. And objects that are very similar in make up and are close together will confuse the program and placing obstacles in areas where it is empty. Therefore, although the stereo vision system works, it is not a practical sensor system for real-time use on the High Speed Vehicle.

7.3 Further Work

There are a lot of further work to be done on this project, the following is a short list of ideas that might help to bring stereo vision into practise as a reliable sensor.

- Faster algorithm;
- Better corner detectors;
- Better fundamental matrix calculation;
- Pan-tilt system.
- Dynamic vision;
- Object recognition;

7.3.1 Faster Algorithm

The most important shortfall of the current stereo vision system is the speed of the algorithms. This criterion has been discussed constantly throughout this thesis, as the speed of the program determines its practicality. All of the algorithms written for the thesis has been in MATLAB, because the program is a lot easy to use for image viewing and matrix manipulation. However, MATLAB is particularly slow in its execution of `for` loop, as well as been a slow language for programming in general. A better language for use is C/C++, thus it should be the first thing in follow up work – transfer all the MATLAB codes into C/C++ language.

7.3.2 Corner Detector

Another of the shortfall of the system is in the accurate detection of all the corners in the image. The two algorithms that was investigated in detail are the most popular detectors. A third detector was also looked into, that is the Curvature Scale Space corner detector (for a brief overview of the algorithm see Appendix A.3). It is claimed to be a better algorithm in finding the corners as well as low error rate of detecting false corners. Furthermore, the authors claims that 85% of the run time for the detector is in the corner detection algorithms, which is the Canny Operator. From the tables 4.1 and 4.2, we can see that it would be faster than the Harris algorithm. The algorithm, however, was not developed fully to verify these claims.

7.3.3 Fundamental Matrix

Much work was concentrated on finding the fundamental matrix. However, as was discussed in Chapter 5, the final result for the matrix still have an error of slightly less than 10. This is not a good enough answer as it increases the search pattern for the conjugate pairs and thus slow down the algorithm for disparity map. More work need to be done on finding better methods and algorithms for calculating the fundamental matrix.

7.3.4 Pan and Tilt System

As the cameras has a angular field of view of approximately 45°, it has a very limited view. A pan and tilt unit for the system would allow the cameras to move in vertical and most importantly horizontal direction, giving the system more lateral viewing perspectives. A lot of adjustments will need to be made in the current version of algorithm to allow for the extra feature.

7.3.5 Dynamic Vision

Most biological vision systems have evolved to cope with the changing world. In stereo vision, we need to remember that not only the camera is moving with the vehicle, objects within the scene is also moving at the same time. A dynamic vision system would be able to cope with moving and changing objects, changing in illumination, and changing in viewpoints. A dynamic vision system will also be important for the stereo vision system to build up the disparity map as the vehicle moves around, and able to react to moving objects in the environment.

7.3.6 Object Recognition

From the disparity map shown in figure 6.1 on page 80, we can see that the data presented does not represent a good idea of the three dimensional space the vehicle occupies in. We only see the position of a point on a feature in the surrounding space, there are no other data concerning the rest of the object. This might be enough if the objective is to avoid obstacles in the path of the vehicle. However, to operate the vehicle in an unknown environment we need to local objects.

An object recognition system finds objects in the real world from an image of the world. The task is surprisingly difficult, when considering that humans perform object recognition effortlessly and instantaneously [17]. In the algorithms an object must be recognised from its background scenery. And in real world cases, a scene would contain multiple entities. Discussions on the complexity of object recognition algorithms is beyond this thesis, but for a truly dynamic stereo vision system, it is an essential part of the whole.

Appendix A

Matrix Computations and Other Algorithms

A.1 Singular Value Decomposition

Singular Value Decomposition (SVD) is a popular algorithm used to solve matrix equations such as those in equation 5.5. Golub and van Loan [13] explained the theory behind the algorithm:

If $A \in \mathbb{R}^{m \times n}$ then there exist orthogonal matrices

$$U = [u_1, \ldots, u_m] \in \mathbb{R}^{m \times n} \tag{A.1}$$

and

$$V = [v_1, \ldots, v_n] \in \mathbb{R}^{m \times n} \tag{A.2}$$

such that

$$U^T AV = \text{diag}(\sigma_1, \ldots, \sigma_p) \tag{A.3}$$

where

$$p = min\{m, n\} \qquad \text{and} \qquad \sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$$

The σ_i are the *singular values* of A and the vectors u_i and v_i are respectively, the i-th *left singular vector* and i-th *right singular vector*. So we can write the matrix A

as:

$$A = U\Sigma V^T \tag{A.4}$$

where Σ is the matrix of singular values, as defined in equation A.3. the singular values of a matrix A are precisely the lengths of the semi-axes of the hyperb ellipsoid E defined by

$$E = \{y | y = Ax, \|x\|_2 = 1\} \tag{A.5}$$

Golub & van Loan [13] also described an algorithm for SVD. Given $A \in \mathbb{R}^{m\times n}(m \geq n)$ and ϵ, a small multiple of the unit roundoff, the following algorithm overwrites A with $U^T AV = D + E$, where $U \in \mathbb{R}^{m\times n}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal, $D \in \mathbb{R}^{m\times n}$ is diagonal, and E satisfies $\|E\|_2 \cong \mathbf{u}\|A\|_2$:

1. Use the QR with Column Pivoting algorithm on page 99 to compute the bidiagonalisation

$$A := (U_1 \dots U_n)^T A(V_1 \dots V_{n-2})$$

2. Set $a_{i,i+1}$ to zero if $|a_{i,i+1}| \leq \epsilon(|a_{ii}| + |a_{i+1,j+1}|)$ for any $i = 1, \dots, n-1$

3. Find the largest q and the smallest p such that if

$$A = \begin{array}{cc} \begin{bmatrix} A_{11} & 0 & 0 \\ 0 & A_{22} & 0 \\ 0 & 0 & A_{33} \\ 0 & 0 & 0 \end{bmatrix} & \begin{array}{c} p \\ n-p-q \\ q \\ m-n \end{array} \end{array}$$

   then A_{33} is diagonal and A_{22} has a nonzero superdiagonal.

4. If $q = n$ *then* quit

5. If any diagonal entry in A_{22} is zero, then zero the superdiagonal entry in the same row and go to 2

6. Apply the *Golub-Kahan SVD step* on page 99 to A_{22}

$$A := \text{diag}(I_p, \bar{U}, I_{q+m-n})^T A\text{diag}(I_p, \bar{V}, I_q)$$

7. Go to 2

In the *QR with Column Pivoting* algorithm given $A \in \mathbb{R}^{m\times n}$, the following algorithm computes the factorisation $A\Pi = QR$.

```
1. $c_j := j \qquad (j = 1, \ldots, n)$
2. $\gamma_j := \sum_{i=-1}^{m} a_{ij}^2 \qquad (j = 1, \ldots, n)$
3. For $k = 1, \ldots, n$
    • Determine $p(k \le p \le n)$ so $\gamma_p = \max_{k \le j \le n} \gamma_j$
    • If $\gamma_p = 0$, then *quit*
    • *else*
        (a) Interchange $c_k$ and $c_p$, $\gamma_k$ and $\gamma_p$, and $a_{ik}$ and $a_{ip}$ for $i = 1, \ldots, m$
        (b) Determine a Householder $\tilde{Q}_k$ such that
            $$\tilde{Q}_k \begin{bmatrix} a_{kk} \\ \vdots \\ a_{mk} \end{bmatrix} = \begin{bmatrix} * \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$
        (c) $A := \mathrm{diag}(I_{k-1}, \tilde{Q}_k)A$
        (d) $\gamma_j := \gamma_j - a_{kj}^2 \qquad (j = k+1, \ldots, n)$
```

In the *Golub-Kahan SVD step* given a bidiagonal matrix $B \in \mathbb{R}^{n\times n}$ having no zeros on its diagonal or superdiagonal, the following algorithm overwrites B with the bidiagonal matrix $\bar{B} = U^T B\bar{V}$ where $\bar{U}$ and $\bar{V}$ are orthogonal.

```
1. Let $\mu$ be the eigenvalue of the trailing 2-by-2 submatrix of $T = B^T B$ that is
   closer to $t_{nn}$
2. $y := t_{11} - \mu$
```

```
3. $z := t_{12}$

4. For $k = 1, \ldots, n-1$

    • Determine $c = \cos(\theta)$ and $s = \sin(\theta)$ such that

      $$\begin{bmatrix} y & z \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} * & 0 \end{bmatrix}$$

    • $B := BJ(k, k+1, \theta)$

    • $y := b_{kk}$

    • $z := b_{k+1,k}$

    • Determine $c = \cos(\theta)$ and $s = \sin(\theta)$ such that

      $$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$

    • $B := J(k, k+1, \theta)^T B$

    • If $k < n-1$ *then*

        (a) $y := b_{k,k+1}$

        (b) $z := b_{k,k+2}$
```

A.2 Levenberg-Marquardt Algorithm

The *Levenberg-Marguardt* has become the standard of nonlinear least-squares routines, in their book, Press et al. described the procedure as following [31]:

Consider fitting when the model depends *nonlinearly* on the set of M unknown parameters $a_k, k = 0, 1, \ldots, M-1$. We define a χ^2 merit function and determine best-fit parameters by its minimisation. With nonlinear dependence, however, the minimisation must proceed iteratively. Given trial values for the parameters, we develop a procedure that improves the trial solution. The procedure is then repeated until χ^2 stops decreasing.

Sufficiently close to the minimum, we expect the χ^2 function to be well approximated by a quadratic form, which is written as:

$$\chi^2(\mathbf{a}) \approx \gamma - \mathbf{d} \cdot \mathbf{a} + \frac{1}{2}\mathbf{a} \cdot \mathbf{D} \cdot \mathbf{a} \tag{A.6}$$

where $\mathbf{d}$ is an M-vector and $\mathbf{D}$ is an $M \times M$ matrix. If the approximation is a good one, we know how to jump from the current trial parameters $\mathbf{a}_{\text{cur}}$ to the minimising ones $\mathbf{a}_{\text{min}}$ in a single leap, namely

$$\mathbf{a}_{\text{min}} = \mathbf{a}_{\text{cur}} + \mathbf{D}^{-1} \cdot [-\nabla\chi^2(\mathbf{a}_{\text{cur}})] \tag{A.7}$$

On the other hand, equation A.6 might be a poor local approximation to the shape of the function that we are trying to minimise at $\mathbf{a}_{\text{cur}}$. In that case, we'll take a step down the gradient using the steepest descent method:

$$\mathbf{a}_{\text{next}} = \mathbf{a}_{\text{cur}} - \text{constant} \times \nabla\chi^2(\mathbf{a}_{\text{cur}}) \tag{A.8}$$

where the constant is small enough not to exhaust the downhill direction.

To use equation A.7 and A.8, we must be able to compute the gradient of the χ^2 function at any set of parameters $\mathbf{a}$. To use equation A.7 we also need the matrix D, which is the second derivative matrix (Hessian matrix) of the χ^2 merit function, at any $\mathbf{a}$.

Let the model to be fitted to be

$$y = y(x; \mathbf{a}) \tag{A.9}$$

and the χ^2 merit function is

$$\chi^2(\mathbf{a} = \sum_{i=0}^{N-1} \left[\frac{y_i - y(x_i; \mathbf{a})}{\sigma_i}\right]^2 \tag{A.10}$$

The gradient of χ^2 with respect to the parameters $\mathbf{a}$, which will be zero at the χ^2 minimum, has components

$$\frac{\partial\chi^2}{\partial a_k} = -2\sum_{i=0}^{N-1} \frac{[y_i - y(x_i; \mathbf{a})]}{\sigma_i^2}\frac{\partial y(x_i; \mathbf{a})}{\partial a_k} \qquad k = 0, 1, \ldots, M-1 \tag{A.11}$$

Taking an additional partial derivative gives

$$\frac{\partial^2 \chi^2}{\partial a_k \partial a_l} = 2\sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i;\mathbf{a})}{\partial a_k} \frac{\partial y(x_i;\mathbf{a})}{\partial a_l} - [y_i - y(x_i;\mathbf{a})] \frac{\partial^2 y(x_i;\mathbf{a})}{\partial a_k \partial a_l} \right] \tag{A.12}$$

Removing the factor of 2 by defining:

$$\beta_k \equiv -\frac{1}{2}\frac{\partial \chi^2}{\partial a_k} \qquad \alpha_{kl} \equiv \frac{1}{2}\frac{\partial^2 \chi^2}{\partial a_k \partial a_l} \tag{A.13}$$

making $[\alpha] = \partial 12\mathbf{D}$ in equation A.7, in terms of which that equation can be rewritten as the set of linear equations

$$\sum_{l=0}^{M} \alpha_{kl} \delta a_l = \beta_k \tag{A.14}$$

This set is solved for the increments δa_l that, added to the current approximation, give the next approximation. In the context of least-squares, the matrix $[\alpha]$, equal to one-half times the Hessian matrix, is usually called the *curvature matrix.* Furthermore, the equation A.8, the steepest descent formula, translates to

$$\delta a_l = \text{constant} \times \beta_l \tag{A.15}$$

The Levenberg-Marquardt method is based on two elementary insights. Consider the "constant" in equation A.15, there is no information about what value (even what order of magnitude) it should be in the gradient. The first insight is that the components of the Hessian matrix, even if they are not usable in any precise fashion, give *some* information about the order-of-magnitude scale of the problem.

The quantity χ^2 is non-dimensional, i.e., is a pure number, evident from its definition in equation A.10. On the other hand, β_k has the dimensions of $1/a_k$, which may well be dimensional, i.e., have units like cm^{-1}, or kilowatt-hours. The constant of proportionality between β_k and δa_k must therefore have the dimensions of a_k^2. Scan the components of $[\alpha]$ and you see that there is only one obvious quantity with these dimensions, and that is $1/a_{kk}$, the reciprocal of the diagonal element. So that must set the scale of the constant. But that scale might itself be too big. So divide the constant by some fudge factor λ, withe the possibility of setting $\lambda \gg 1$ to cut down the step. In other words, replace equation A.15 by

$$\delta a_l = \frac{1}{\lambda \alpha_{ll}} \beta_l \qquad \text{or} \qquad \lambda \alpha_{ll} \delta a_l = \beta_l \tag{A.16}$$

It is necessary that α_{ll} be positive, but this is guaranteed by:

$$\alpha_{kl} = \sum_{i=0}^{N-1} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i; \mathbf{a})}{\partial a_k} \frac{\partial y(x_i; \mathbf{a})}{\partial a_l} \right] \tag{A.17}$$

Marquardt's second insight is that equations A.16 and A.15 can be combined by we define a new matrix α' by the following prescription:

$$\begin{aligned} \alpha'_{jj} &\equiv \alpha_{jj}(1+\lambda) \\ \alpha'_{jk} &\equiv \alpha_{jk} \qquad (j \neq k) \end{aligned} \tag{A.18}$$

and then replace both equation A.16 and A.15 by

$$\sum_{l=0}^{M-1} \alpha'_{kl} \delta a_l = \beta_k \tag{A.19}$$

When λ is very large, the matrix α' is forced into being *diagonally dominant*, so equation A.19 goes over to be identical to equation A.16.

Given the initial guess for the set of fitted parameters $\mathbf{a}$, the recommended Marquardt recipe is as follows:

1. Compute $\chi^2(\mathbf{a})$;
2. Pick a modest value for λ, say $\lambda = 0.001$;
3. Solve the linear equations A.19 for $\delta\mathbf{a}$ and evaluate $\chi^2(\mathbf{a} + \delta\mathbf{a})$;
4. If $\chi^2(\mathbf{a} + \delta\mathbf{a}) \geq \chi^2(\mathbf{a})$, *increase* λ by a factor of 10 (or any other substantial factor) and go back to 3;
5. If $\chi^2(\mathbf{a} + \delta\mathbf{a}) < \chi^2(\mathbf{a})$, *decrease* λ by a factor of 10, update the trial solution $\mathbf{a} \leftarrow \mathbf{a} + \delta\mathbf{a}$, and go back to 3

A.3 Curvature Scale Space Corner Detector

This is a corner detector based on the Curvature Scale Space (CSS) representation first proposed by Mokhtarian and Suomela [24]. In their papers and in a later paper by Mokhtarian and Mohanna [23], it was claimed that the method is an improvement over the 'traditional' methods such as the Harris Detector and the SUSAN Detector. The subsequent images used for demonstrating the method also shows that the CSS detector has a better chance of finding the right corners and less chance of false detection.

The CSS technique is suitable for recovering invariant geometric features such as curvature extrema or zero-crossing points of a planar curve at multiple scales. The curve Γ is first parameterised by the arc length parameter u:

$$\Gamma(u) = (x(u), y(u)) \tag{A.20}$$

An *evolved version* Γ_σ of Γ is defined by:

$$\Gamma_\sigma = (X(u,\sigma), Y(u,\sigma)) \tag{A.21}$$

where

$$X(u,\sigma) = x(u) \otimes g(u,\sigma) \qquad Y(u,\sigma) = y(u) \otimes g(u,\sigma)$$

where $g(u,\sigma)$ denotes a Gaussian of width σ, and *sigma* is also referred to as the *scale* parameter. The process of generating evolved versions of Γ as σ increases from zero to infinity is referred to as the *evolution* of Γ. In order to find curvature zero-crossings or extrema from evolved versions of the input curve, the curvature need to be computed accurately and directly on an evolved version Γ_σ. Curvature κ on Γ_σ is given by:

$$\kappa(u,\sigma) = \frac{\mathcal{X}_u(u,\sigma)\mathcal{Y}_{uu}(u,\sigma) - \mathcal{X}_{uu}(u,\sigma)\mathcal{Y}_u(u,\sigma)}{(\mathcal{X}_u(u,\sigma)^2 + \mathcal{Y}_u(u,\sigma)^2)^1.5} \tag{A.22}$$

where

$$\mathcal{X}_u(u,\sigma) = x(u) \otimes g_u(u,\sigma) \qquad \mathcal{X}_{uu}(u,\sigma) = x(u) \otimes g_{uu}(u,\sigma)$$

$$\mathcal{Y}_u(u,\sigma) = y(u) \otimes g_u(u,\sigma) \qquad \mathcal{Y}_{uu}(u,\sigma) = y(u) \otimes g_{uu}(u,\sigma)$$

The outline of the CSS corner detector is as following [23]:

- Extract edges from the original image;
- Extract image edge contours, filling the gaps and finding T-junctions;
- Use different scales of the CSS for contours with different lengths;
- Compute the absolute curvature on the smoothed contours;
- Smooth the absolute curvature function for long contours;
- Detect initial local maxima of the absolute curvature for short contours;
- Detect initial local maxima of the smoothed absolute curvature functions for long contours;
- Consider those local maxima as initial corners whose absolute curvature are more than twice as much as one of the neighbouring local minima;
- Track the corners down to the lowest scale for each contour to improve localisation;
- compare the T-junction corners to the corners found using the curvature procedure to unify close corners.

Appendix B

Camera and Lenses Specifications

B.1 Camera

The camera used for the thesis is the Panasonic Digital Signal Processing 1/3" B/W CCD Cameras in of the WV-BP330 Series. The cameras incorporate a 1/3" interline transfer type CCD (447,000 pixels: effective) and achieve high sensitivity of 0.08 lx at F1.4, outstanding 570-line horizontal resolution and signal-to-noise ratio of 50 dB [27].

The key features of the camera are:

- 1/3" CCD pick-up element with 768 (H) x 582 (V) pixels provides a high 570 lines of horizontal resolution.
- A vertical digital 2H enhancement improves signal strength at the horizontal and vertical edges to provide a crisper images.
- Knee circuit effectively expands the dynamic range when viewing brightly illuminated objects.
- Outstanding 50 dB of signal-to-noise ratio provides natural images.
- Minimum scene illumination of 0.08 lx at F1.4
- A specially developed lens mount configuration accepts both C and CS mount

lenses, as well as both DC and Video voltage for ALC lenses.

- An advanced Electronic Light Control (ELC) enables the use of inexpensive fixed iris lenses for more cost effective systems. Gen-lock capability for system expendability, and other a variety

- Gen-lock of synchronisation modes (Line-Lock, VD2, Internal).

- WV-BP330 Camera Series are:

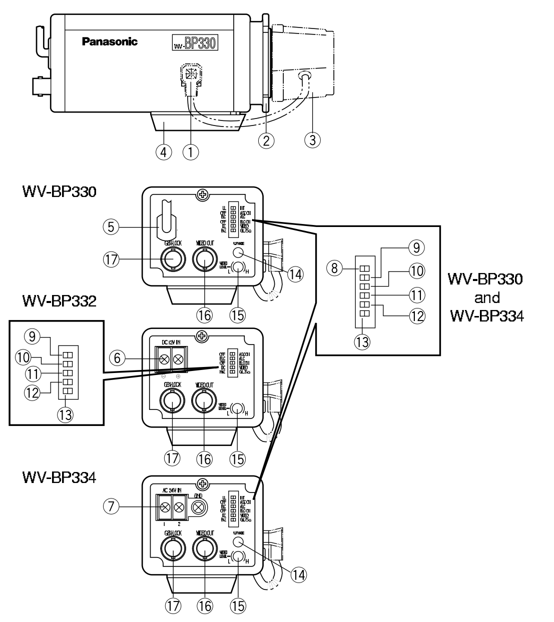
  - WV-BP330: 220-240V AC, 50Hz
  - WV-BP332: 12V DC
  - WV-BP334: 24V AC, 50Hz

- The maximum extensible coaxial cable length between the camera and the monitor is shown below.

Type of coaxial cable		3C-2V (RG-59/U)	5C-2V (RG-6/U)	7C-2V (RG-11/U)	10C-2V (RG-15/U)
Recommended maximum cable length	(m)	250	500	600	800
	(ft)	825	1,650,	1,980,	2,640

The major operating controls, accessories, specifications and appearance for the camera is shown on figure B.1.

MAJOR OPERATING CONTROLS



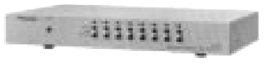
① Auto Iris lens Connector
② Flange-back Adjusting Ring
③ Lens (option)
④ Camera Mounting Screw Hole
⑤ Power Cord (only WV-BP330)
⑥ DC 12 V Input Terminal (only WV-BP332)
⑦ AC 24 V Input Terminal (only WV-BP334)
⑧ Synchronization Mode Selector (only WV-BP330/WV-BP334)
⑨ AGC ON/OFF Selector
⑩ Automatic Light Control / Electronic Light Control Selector
⑪ Back Light Compensation Mode Selector
⑫ Lens Drive Signal Selector
⑬ Gen-lock Termination Selector
⑭ Vertical Phase Control (only WV-BP330 and WV-BP334)
⑮ Video Level Control
⑯ Video Output Connector
⑰ Gen-lock Input Connector

OPTIONAL ACCESSORIES

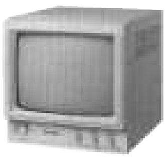
B/W Quad Unit
WJ-410

ALC Lenses
WV-LF4R5C3A
WV-LF9C3A
WV-LA210C3
WV-LA408C3
WV-LA908C3

Sequential Switchers
WJ-SQ208
WJ-SQ308

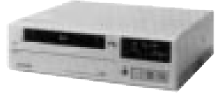


Video Monitors
WV-BM500
WV-BM900
WV-BM1410
WV-BM1700
WV-BM1900



Camera Housings
WV-7110A
WV-7120D
WV-7135
WV-7140

Time-Lapse VCRs
AG-6124
AG-6040
AG-6730
AG-TL300
AG-TL700



SPECIFICATIONS CCIR

Models	WV-BP330	WV-BP332	WV-BP334
Pick-up Device	768 (H) x 582 (V) pixels, Interline Transfer CCD		
Scanning Area	4.9 (H) x 3.7 (V) mm (Equivalent to scanning area of 1/3" pick-up tube)		
Synchronization	Internal, External, Line-locked or Multiplexed vertical drive (VD2) selectable		
Scanning System	2 : 1 interlace		
Scanning	625 lines / 50 fields / 25 frames Horizontal: 15.625 kHz Vertical: 50 Hz		
Horizontal Resolution	570 lines		
Video Output	1.0 V[p-p] CCIR composite 75 Ω / BNC connector		
Signal-to-Noise Ratio	50 dB (AGC OFF)		
Electronic Light Control	Equivalent to continuous variable shutter speed between 1/50 s and 1/10,000 s		
Minimum Illumination	0.08 lx at F1.4, AGC ON		
Lens Mount	C-mount or CS-mount selectable		
Ambient Operating Temperature	–10℃ - +50℃		
Ambient Operating Humidity	Less than 90%		
Power Source and Power Consumption	220-240V AC 50 Hz 4.5W	12V DC 250 mA	24V AC 50 Hz 3.5W
Dimensions (without lens)	67 (W) x 55 (H) x 123 (D) mm [2-5/8" (W) x 2-3/16" (H) x 4-13/16" (D)]		
Weights (without lens)	0.64 kg (1.4 lbs)	0.445 kg (0.98 lbs)	0.470 kg (1.04 lbs)

Specifications are subject to change without notice.
Dimensions and weight are approximate.
This product may be subject to export control regulations.

APPEARANCE

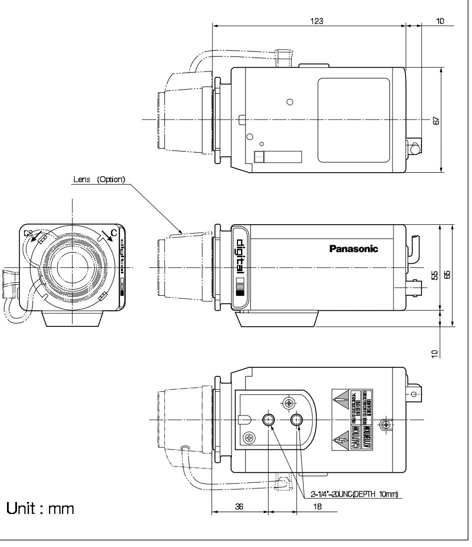


Figure B.1: Specifications for the camera [27].

B.2 Lens

The Lens used for the project is the Panasonic 2x Variable Focal Lens WV-LZA61/2, for use with 1/3-type CCD cameras in CCVE lens line-up. WV-LZA61/2, adjustable a desired focal length (3.8-8mm) and angular field of view (H: $35.6° - 73.6°$, V: $26.6° - 53.4°$) suited to fixed cameras [28]. Figure B.2 shows the specification and the measurements for the lens.

1/3-TYPE CAMERA/LENS COMBINATION CHART

Model		**WV-LZA61/2**
Type of Lens		1/3-type Variable Focal Lens
Focal Length		3.8 - 8 mm
Panasonic 1/3-type CCD Cameras	**B/W Cameras** **WV-BP550 series** **WV-BP330 series** **WV-BP140 series**	YES
	Colour Cameras **WV-CP460 series** **WV-CP240 series**	YES

SPECIFICATIONS

Image Size	ø6 mm [4.8 (H) x 3.6 (V) mm]
Focal Length	3.8 - 8 mm
Maximum Aperture Ratio	1:1.4 (Wide), 1:1.8 (Tele)
Angular Field of View	H: 35.6° ~ 73.6° V: 26.6° ~ 53.4°
Focusing Range	1.2 m - ∞
Mount	CS-mount, 1-32UN
Ambient Operating Temperature	–10°C - +50°C
Dimensions	60.5 (W) x 53 (H) x 45.3 (D) mm
Weight (approx.)	75 g

APPEARANCE

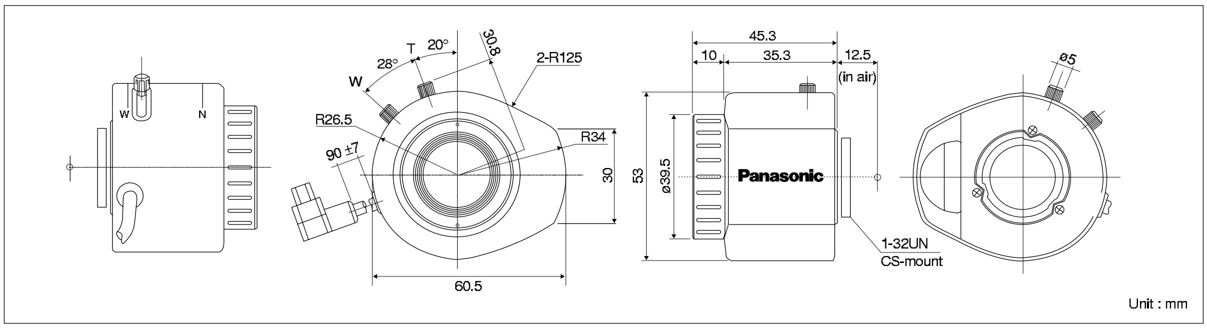


Figure B.2: Specifications for the lens [28].

Appendix C

Camera Calibration Parameters

The list of variables after calibration can be separated into two groups: intrinsic parameters and extrinsic parameters [3]. The former are variables related to the camera at large, for example the focal length of the camera, the image centres etc. The latter variables are those associated with the individual images, such as the rotation and translation.

Since we are only concerned with the intrinsic parameters for the use in the various calculations, the extrinsic parameters will not be discussed. The intrinsic parameters calculated by the calibration program are:

- **Focal Length**: The focal length is calculated with the unit in pixels, the values are stored in a 2×1 vector of `fc`,
- **Principle Point**: These are the centre coordinate of the image, stored in a 2×1 vector of `cc`,
- **Skew Coefficient**: The skew coefficient defines the angle between the x and the y axes and is stored in the scalar `alpha_c`, and
- **Distortion**: These are the image distortion coefficient in radial and tangential directions and are stored in a 5×1 vector `kc`.

Let P be a point in space of coordinate vector $XX_c = \begin{bmatrix} X_c & Y_c & Z_c \end{bmatrix}^T$ in the camera reference frame. The intrinsic parameters (`fc`, `cc`, `alpha_c`, `kc`) will relate the point to a point on the image plane in the following steps:

Let x_n be the normalised (pinhole) camera image position:

$$x_n = \begin{bmatrix} X_c/Z_c \\ \\ Y_c/Z_c \end{bmatrix} = \begin{bmatrix} x \\ \\ y \end{bmatrix}$$

and $r^2 = x^2 + y^2$. After adding in the distortion, the new normalised point coordinate x_d is:

$$x_d = \begin{bmatrix} x_d(1) \\ \\ x_d(2) \end{bmatrix} = (1 + \mathtt{kc}(1) \times r^2 + \mathtt{kc}(2) \times r^4 + \mathtt{kc}(5) \times r^6) \times x_n + dx$$

where dx is the tangential distortion vector given by:

$$dx = \begin{bmatrix} 2 \times \mathtt{kc}(3) \times x \times y + \mathtt{kc}(4) \times (r^2 + 2x^2) \\ \\ \mathtt{kc}(3) \times (r^2 + 2y^2) + 2 \times \mathtt{kc}(4) \times x \times y \end{bmatrix}$$

Therefore, the 5-vector `kc` contains both radial and tangential distortion coefficients (observe that the coefficient of 6th order radial distortion term is the fifth entry of the vector `kc`). The tangential distortion is due to "de-centring", or imperfect centring of the lens components and other manufacturing defects in a compound lens.

Once distortion is applied, the final pixel coordinates `x_pixel` $= \begin{bmatrix} x_p & y_p \end{bmatrix}^T$ of the projection of P on the image plane is:

$$x_p = \mathtt{fc}(1) \times (x_d(1) + \mathtt{alpha\_c} \times x_d(2)) + \mathtt{cc}(1)$$

$$y_p = \mathtt{fc}(2) \times x_d(2) + \mathtt{cc}(2)$$

Therefore, the pixel coordinate vector `x_pixel` and the normalised (distorted) coordinate vector x_d are related to each other through the linear equation:

$$\begin{bmatrix} x_p \\ y_p \\ 1 \end{bmatrix} = \mathtt{KK} \begin{bmatrix} x_d(1) \\ x_d(2) \\ 1 \end{bmatrix}$$

where KK is known as the camera matrix, and defined as follows:

$$\mathtt{KK} = \begin{bmatrix} \mathtt{fc}(1) & \mathtt{alpha\_c} \times \mathtt{fc}(1) & \mathtt{cc}(1) \\ 0 & \mathtt{fc}(2) & \mathtt{cc}(2) \\ 0 & 0 & 1 \end{bmatrix}$$

In addition to computing estimates for the intrinsic parameters `fc`, `cc`, `kc` and `alpha_c`, the toolbox also returns estimates of the uncertainties on those parameters. The MATLAB variables containing those uncertainties are `fc_error`, `cc_error`, `kc_error`, `alpha_c_error`. Those vectors are approximately three times the standard deviations of the errors of estimation.

Bibliography

[1] Ahuja, N. and Abbott, A. L. "Active Stereo: Integrating Disparity, Vergence, Focus, Aperture, and Calibration for Surface Estimation", *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. 15, No. 10, pp. 1007-1029, 1993

[2] Barnard, S. T. and Thompson, W. B. "Disparity Analysis of Images", *IEEE Trans. Patt. Anal. Machine Intel.*, Vol. PAMI-2, No. 4, pp. 333-340, July 1980

[3] Bouguet, J. "Camera Calibration Toolbox for Matlab", *http://www.vision.caltech.edu/bouguetj/calib_doc/index.html* updated October 10th, 2002

[4] Broggi, A.; Bertozzi, M.; and Fascioli A. "Self-Calibration of a Stereo Vision System for Automotive Application", *Proc. of the 2001 IEEE Int. Conf. on Robotics & Automation*, pp. 3698-3703, May 2001

[5] Canny, J. "A Computational Approach to Edge Detection", *IEEE Trans. Patt. Anal. Machine Intell.*, Vol. PAMI-8, No. 6, pp. 679-698, Nov. 1986

[6] Charnley, D.; Harris, C. G.; Pike, M.; Sparks, E. and Stephens, M. "The DROID 3D Vision System - algorithms for geometric integration", Tech. Rep. 72/88/N488U, Plessey Research Roke Manor, 1988

[7] Chojnacki, W.; Brooks, M. J.; and van den Hengel, A. "Fitting surfaces to data with covariance information: Fundamental methods applicable to computer

vision", Tech. Rep. TR03-99, version 2.1, Department of Computer Science, University of Adelaide, Auguest 1999

[8] Chojnacki, W.; Brooks, M. J.; van den Hengel, A.; and Gawley, D. "A Fast MLE-Based Method for Estimating the Fundamental Matrix", *Proc. APRS/IEEE Workshop on Stereo Image and Video Processing*, pp. 33-36, December 2000

[9] Corke, P. "Real-time Stereo Vision for Height Estimation on an Autonomous Helicopter", *Proc. APRS/IEEE Workshop on Stereo Image and Video Processing*, pp. 19-23, December 2000

[10] Dimo, Gajendra Ganesh; *A Panoramic Imaging System for the HSV*, BE thesis, University of Sydney, November 1997

[11] Fisher, B. "Geometric Framework for Vision I: Single View and Two-View Geometry Andrew Zisserman" *http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/EPSRC_SSAZ/epsrc_ssaz.html* Robotics Research Group, University of Oxford, Updated: April 16, 1997

[12] Gavrila, D. M.; Franke, U.; Wohler, C; and Gorzig, S. "Real-Time Vision for Intelligent Vehicles", *IEEE Instrumentation & Measurement Magazine*, pp. 22-27, June 2001

[13] Golub, G. H. and Van Loan, C. F. *Matrix Computations* Johns Hopkins University Press, 1983

[14] Gibson, J. J. *The Perception of the Visual World*, Cambridge, MA: Riverside, 1950

[15] Harris, C. and Stephens, M. "A Combined Corner and Edge Detector", *Forth Alvey Vision Conf.*, pp. 147-151, 1988

[16] *http://www.acfr.usyd.edu.au/projects/research/ute/main.html* High Speed Vehicle (HSV) Project

[17] Jain, R; Kasturi, R; Schunch, B. G. *Machine Vision*, McGraw-Hill, New York, 1995

[18] Julesz, B. *Foundations of Cyclopean Perception*, Chicago, IL: Univ. Chicago Press, 1971

[19] Mactier, T. G. *Urban Road following using a single CCD camera*, BE thesis, University of Sydney, November 2001

[20] Matrox Imaging *Matrox Meteor-II Installation and Hardware Reference* Matrox Electronic Systems Ltd., Manual no. 10577-101-0301, April 2000

[21] McLauchlan, P. F. "Gandalf: The Fast Computer Vision and Numerical Library" *http://gandalf-library.sourceforge.net/tutorial/report/report.html* Imagineer Software Ltd., Updated April 29, 2002

[22] Mohr, R. and Triggs, B. *Projective Geometry for Image Analysis*, A Tutorial given at ISPRS, Vienna, July 1996

[23] Mokhtarian, F. and Mohanna, F. "Enhancing the Curvature Scale Space Corner Detector" *Proc. Scandinavcan Conf. on Image Analysis* pp. 145-152, Bergen, Norway, 2001

[24] Mokhtarian, F. and Suomela, R. "Robust Image Corner Detection Through Curvature Scale Space" *IEEE Tran. Patt. Anal. Machine Intell.* Vol. 20, No. 12, pp. 1376-1381, Dec. 1998

[25] Nguyen, M. and Graefe, V. "Object Manipulation by Learning Stereo Vision-Based Robots", *Proceedings of the 2001 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Maul, Hawaii, USA, pp. 146-151, Oct. 29 - Nov. 03, 2001

[26] *http://www.vision3d.com/stereo.html* Optometrists Network, accessed 9th March, 2002

[27] Panasonic, *Digital Signal Processing 1/3" B/W CCD Cameras, WV-BP330 Series* Panasonic, Matsushita Communication Industrial Co., Ltd. Japan

[28] Panasonic, *2x Variable Focal lens WV-LZA61/2* Panasonic, Matsushita Communication Industrial Co., Ltd. Japan

[29] Physics, School of *Experimental Physics Notes for Intermediate Physics* School of Physics, University of Sydney, 1999

[30] Pollefeys, M. (2000) *Tutorial on 3D Modeling from Images* In conjunction with ECCV 2000, Dublin, Ireland

[31] Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; and Flannery, B. P. *Numerical Recipes in C++, The Art of Scientific Computing* Cambridge University Press, Cambridge, UK, 2nd ed., 2002

[32] SICK *Technical Description of LMS 200 / LMS 211 / LMS 220 / LMS 221 / LMS 291 Laser Measurement Systems* SICK AG, Auto Ident, Germany, June 2000

[33] Smith, S. M. and Brady, J. M.; "SUSAN - a new approach to low level image processing" *Int. J. of Comp. Vision* Vol. 23, No. 1, pp. 45-78, May 1997

[34] Whitehorn, M.; Vincent, T.; Debrunner, C.; Steele, J.; "Stereo Vision in LHD Automation", *IEEE*, pp. 1388-1395, 2001