NTNU – Trondheim
Norwegian University of
Science and Technology

Computer vision as an alternative for collision detection

Marius Aarvik Drangsholt

Master of Science in Cybernetics and Robotics
Submission date: June 2015
Supervisor: Amund Skavhaug, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

NTNU – Trondheim
Norwegian University of
Science and Technology

Computer vision as an alternative for collision detection

Marius Aarvik Drangsholt

June 2015

MASTER THESIS

Department of Engineering Cybernetics
Norwegian University of Science and Technology

Supervisor: Associate Professor Amund Skavhaug, ITK

Summary

The goal of this thesis was to implement a computer vision system on a low power platform, to see if that could be an alternative for a collision detection system. To achieve this, research into fundamentals in computer vision were performed, and both hardware and software implementation were carried out.

To create the computer vision system, a stereo rig were constructed using low cost Logitech webcameras, and connected to a Raspberry Pi 2 development board. The computer vision library OpenCV was used for interfacing with the stereo rig, and provide tools for image acquisition and stereo matching. The system was then put through a series of tests to evaluate the accuracy and refresh rate. For comparison reasons, an ultrasonic system were implemented with all the necessary hardware and software.

Based on the results obtained, a working implementation of a computer vision system was achieved. The system performed relatively good with respect to both the accuracy and the refresh rate, but will require some improvements depending on what type of applications the system are intended to be used in. The resulting computer vision system in this thesis does not provide any more functionality than the ultrasonic system in terms of collision detection, but it can be used as a foundation for future work to implement new features and more functionality.

The main contributions of this thesis is a computer vision subsystem that can be integrated for use into other applications directly. This system is also a foundation to be used for improvements and construction of a better system.

Sammendrag

Målet med denne avhandlingen var å implementere et datasyn-system på en lav-effektsplattform, for å se om det kan være et alternativ for kollisjonsdeteksjon. For å oppnå dette, fundamentale egenskaper ved datasyn ble undersøkt, og både maskinvare og programvare ble implementert.

For å lage datasyn-systemet ble et stereokamera konstruert ved å bruke et par billige Logitech webkameraer koblet til en Raspberry Pi 2. Datasynbiblioteket OpenCV ble brukt som grensesnitt mot stereokameraet, og som verktøy for å utføre bildetaking og stereosamsvar. Systemet ble så kjørt gjennom testing for å evaluere nøyaktigheten og oppdateringshastigheten. For sammenligning ble et ultralyd system implementert med all nødvendig maskinvare og programvare.

Basert på resultatene som er innhentet, en fungerende implementasjon av et datasyn-system ble oppnådd. Systemet fungerte relativt bra med hensyn på både nøyaktigheten og oppdateringsfrekvensen, men kan trenge noe forbedringer avhenging av hva slags applikasjoner systemer er tenkt brukt i. Det resulterende datasyn-systemet i denne avhandlingen gir ikke noe mer funksjonalitet utover et ultralyd system med hensyn på kollisjonsdeteksjon, men det kan brukes som et grunnlag for videre arbeid for implementasjon av nye funksjoner og funksjonalitet.

Bidraget med denne avhandlingen er et datasyn-system som kan bli integrert til bruk i andre applikasjoner direkte. Systemet er også et grunnlag for videre arbeid mot et bedre system.

Preface

This project has been conducted at the Department of Engineering Cybernetics at the Norwegian University of Science and Technology during the spring of 2015.

I would like to thank my supervisor, Associate Professor Amund Skavhaug, for providing valued guidance and advice during the project.

Trondheim, June 2015

Marius Aarvik Drangsholt

Contents

List of Figures

List of Tables

Abbreviations

UAS	Unmanned Aerial System
TOF	Time of Flight
SAD	Sum of Absolute Differences
SSD	Sum of Squared Differences
MSE	Mean Squared Error
MAD	Mean Absolute Deviation
NTNU	Norges teknisk-naturvitenskapelige universitet

Chapter 1

Introduction

1.1 Background and Motivation

Robotic systems has seen great progress over the last few decades, and the development of robots are expanding rapidly. The initial robotic development were seen in industries where robots could relieve humans in repetitive and heavy tasks. Today, robots have become more available and affordable, and we see robots in daily use such as vacuum cleaners, lawn mowers and robotic toys.

The need for improved autonomy in the robotic systems that is being developed is increasing. Many systems, such as UAS and autonomous cars, are being integrated with existing robots, machines and humans in daily use. When several players are coexisting in the same area, measures have to be taken to ensure safe operation. Collision detection is a way to increase safety, by having a system that can see other agents in the same area, and thus preventing them to crash into each other. The need for such a system is increasing, with more and more focus on size, power consumption and reliability.

This thesis is motivated by an interest in implementing computer vision as a system for collision detection, to see if that can be suitable for small autonomous applications. This is hoped to be achieved by investigating existing solutions, implement necessary hardware and software to run the computer vision system, and test this system in terms of accuracy and refresh rate.

1.2 Goal and Method

The goal of this thesis is to implement computer vision on a low power platform to see if that can be an alternative for collision detection. All necessary hardware and software to run a computer vision system will be implemented, and evaluation with regards to accuracy and refresh rate will be made. The results will then be presented in a user friendly format, functioning as a basic guide for choosing and comparing methods for collision detection for certain applications (e.g. autonomous cars, warehouse robots, hobby robots etc.). The main objectives included are:

1. Study relevant theory and fundamentals.
2. Present hardware and software used in this thesis.
3. Implement the computer vision system.
4. Test the computer vision system and evaluate the results.

To achieve these goals, background research into fundamentals of computer vision is necessary, as well as research into suitable hardware that can be used for implementation. The development of the hardware for the system will not be discussed in this thesis. Test procedures is also considered to be able to give an adequate baseline for comparison.

1.3 Structure of the Report

This report is intended for readers interested in options for collision detection systems on autonomous applications, and readers interested in implementing computer vision as an option for collision detection.

Chapter 2 presents the detection systems used in this thesis, as well as necessary background and theory for the systems.

Chapter 3 explains the implementation of both hardware and software for the systems presented in chapter 2.

Chapter 4 explains the test setup and test results of the implemented systems presented in chapter 2 and 3.

Chapter 5 presents a general discussion about the project work, as well as possible future work that can be done for improvements.

Chapter 6 concludes the work that has been performed in this report.

Chapter 2

Background and Theory

In this chapter, the methods for detection systems will be presented, and relevant background theory discussed. An ultrasonic system was chosen for use in comparison with the computer vision system. Available time limited the number of methods that could be implemented and used for testing and evaluation, since a large number of possible methods for collision detection exists. The primary criteria for choosing the ultrasonic method were based on usage in the real world, price and component requirements. Other methods could be used instead, but will not be considered in this thesis.

2.1 Ultrasonic

Ultrasonic obstacle detection is a widely used method for detecting objects and distances, and is used from cheap hobby projects, like the LEGO mindstorms robots, to advanced car parking systems. Ultrasonic obstacle detection adapts a range finding method to report a distance from the robot to an object. Ultrasonic range detection is suited for close range obstacle detection, due to the relative slow speed of sound and sound reflectance of solid objects.

2.1.1 Ultrasonic theory

Ultrasound is a sound pressure wave, with frequency greater than the human hearing. Ultrasound has the same physical properties as audible sound, but is

outside the human hearing range. Frequencies above 20kHz is considered ultrasonic [Kut91].

In a ultrasonic ranging system, a pulse is emitted from a transducer, which then switches to receiver mode and waits for a set time to detect an echo. If an echo is detected, the distance from the object can be found by multiplying the speed of sound with half the time for the pulse to be detected. An illustration can be seen in figure 2.1. Since the time measurement is from the pulse was sent to the echo is detected, the time measurement is halved to get the time from the pulse was sent to the pulse hit the object. The distance using TOF can be found by using the following equation:

$$D = \frac{c * t}{2} \tag{2.1}$$

Where c is the speed of sound, and t is the measured time in seconds. Treating air as an ideal gas, the speed of sound c can be found by using the equation [Ens88, FS78]

$$c = \sqrt{k * R * T} \tag{2.2}$$

Where k = ratio of specific heat = 1.4
R = gas constant = 286.9
T is the temperature in kelvin.
For a temperature of 20°C, the speed of sound is calculated to be 343.14 m/s. This value will be used throughout this thesis.

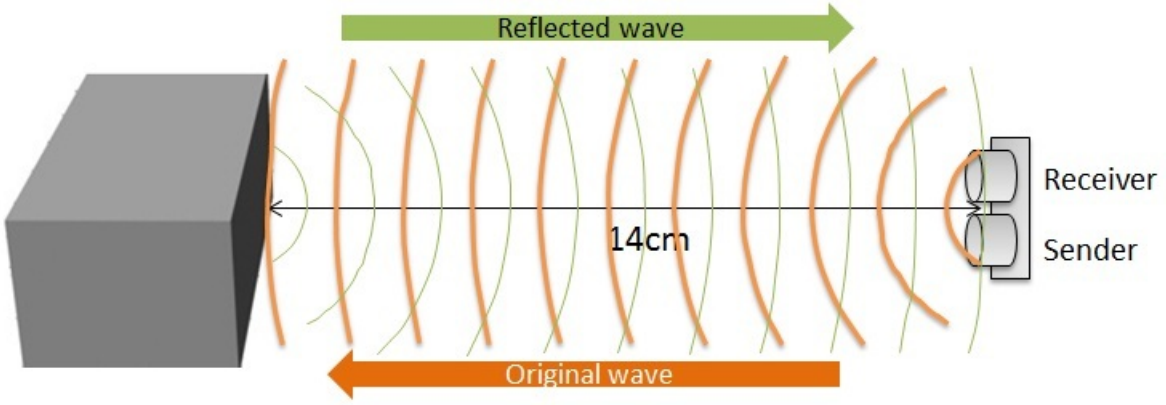


FIGURE 2.1: Illustration of ultrasonic operation[Tea]

2.1.2 Advantages

Using ultrasound as a method for distance measurements has several advantages over similar systems:

- Cost effective. A cheap sensor and a development board can get you started with a small detection system.
- For simple systems, implementation can be done relatively easy, as seen on the LEGO mindstorm robots.

A typical scenario where measurements is taken can be seen in figure 2.2. In this scenario the accuracy of the measured distance can be very good, as the object is directly opposite of the sensor.

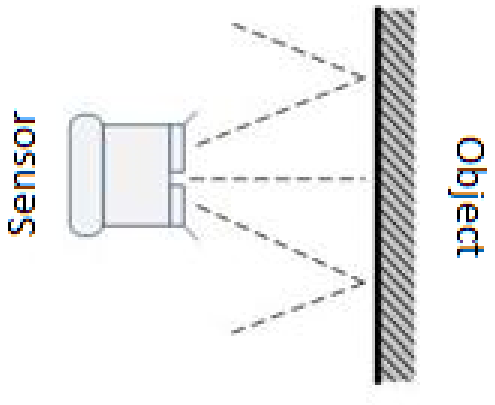


FIGURE 2.2: Illustration of object directly in front

2.1.3 Disadvantages

Accuracy of the measurement taken with a ultrasonic system is dependent of several factors. The speed of sound relies on temperature measurement, and a temperature difference of 10°C may give a noticeable difference in distance readings. If we for example have set a fixed temperature of 20°C, the speed of sound is found to be 343.14 m/s. For a distance of 5 meters, this gives us a time measurement of 0.0292s. If the temperature drops by 10°C, the speed of sound is found to be

337.24 m/s. For the same distance of 5 meters, this gives us a time measurement of 0.0297s. If we then still use 20°C for temperature in our equation, the calculated distance would be 5.1 meters, instead of 5 meters if the correct temperature value were to be used in the calculation.

The reflection of ultrasonic waves is an important factor to consider. As seen in figure 2.2, with the right angle, a good measurement can be taken due to the ultrasound reflectivity in objects. This however, can also be a challenge if the angle to the object changes drastically. In figure 2.3 and 2.4, two examples are shown were detecting objects is difficult.

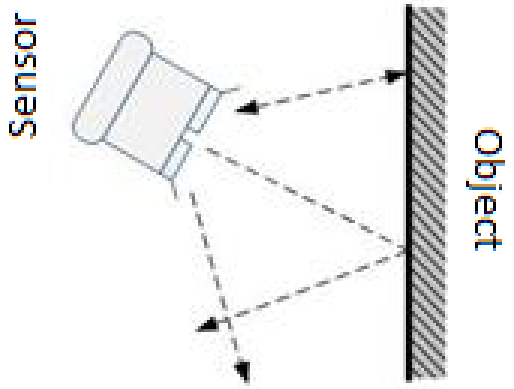


FIGURE 2.3: Illustration of angle leading to range error

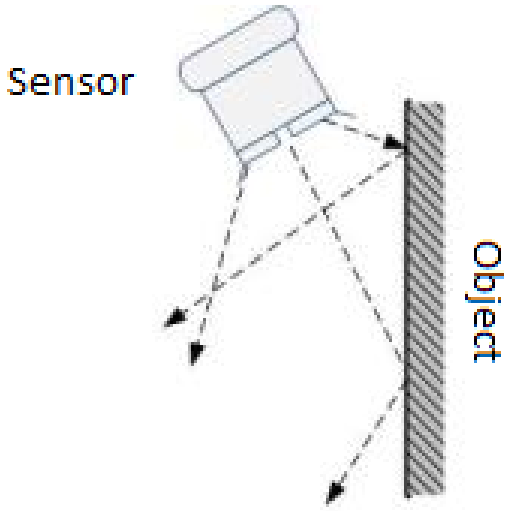


FIGURE 2.4: Illustration of angle leading to no detection

In figure 2.3, we can see that the sensor gets the measurement on the left side of the sensor, and not the middle line which get reflected off due to the angle. This results in a measurement that could be different from the optimal measurement in the middle.

In figure 2.4, the angle is so large that no energy is reflected back to the sensor, which in turn implies that the object is not detected.

2.2 Computer vision

Computer vision is a field in computer science which is often described as emulating human vision with machines. To be able to perceive the environment, the machine must be able to recognize different characteristics and properties in a place. This is done with analysis of images taken with a camera or multiple cameras, to try to identify different shapes, textures, colors etc. As small development boards and computers are getting more and more powerful, computer vision is becoming an interesting area to use for robotic applications.

2.2.1 Stereo vision

Human beings are in general provided with two eyes for vision, when combined provides better precision in the depth dimension than with only one eye. In computer vision, this can be implemented in the same way, by having two cameras take an image at the same time and from different angles to the object. With comparison of these two images, it is possible to get depth information in the form of disparities, and calculate distance from the cameras to the object.

2.2.2 Epipolar geometry

Epipolar geometry is the intrinsic projective geometry between two views [HZ04]. The use of epipolar geometry is motivated by the search of corresponding points between pictures. Two images of the same environment taken from separate positions, have numerous geometric locations between the 3D points and their 2D projections that leads to constraints between the image points.

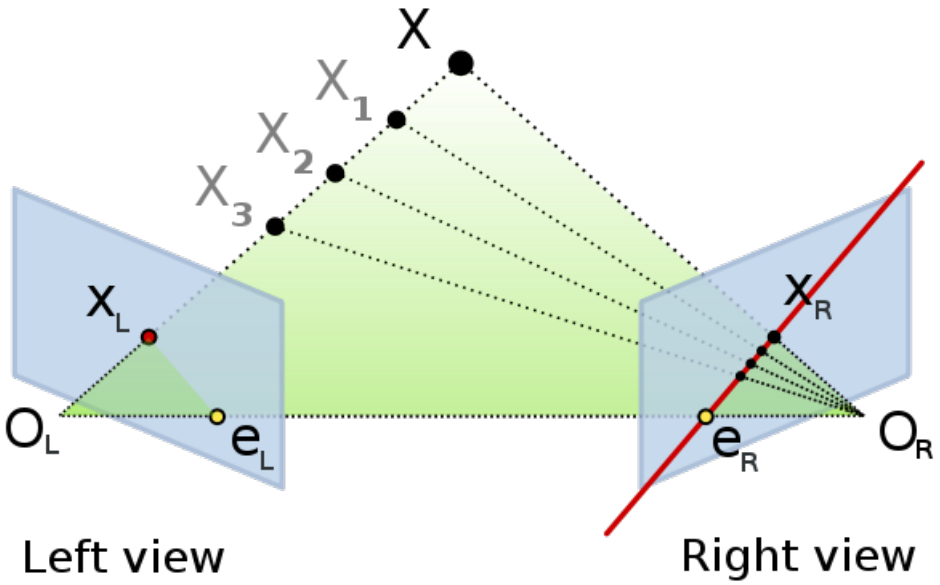


FIGURE 2.5: Illustration of epipolar view, by Arne Nordmann, CC BY-SA 3.0

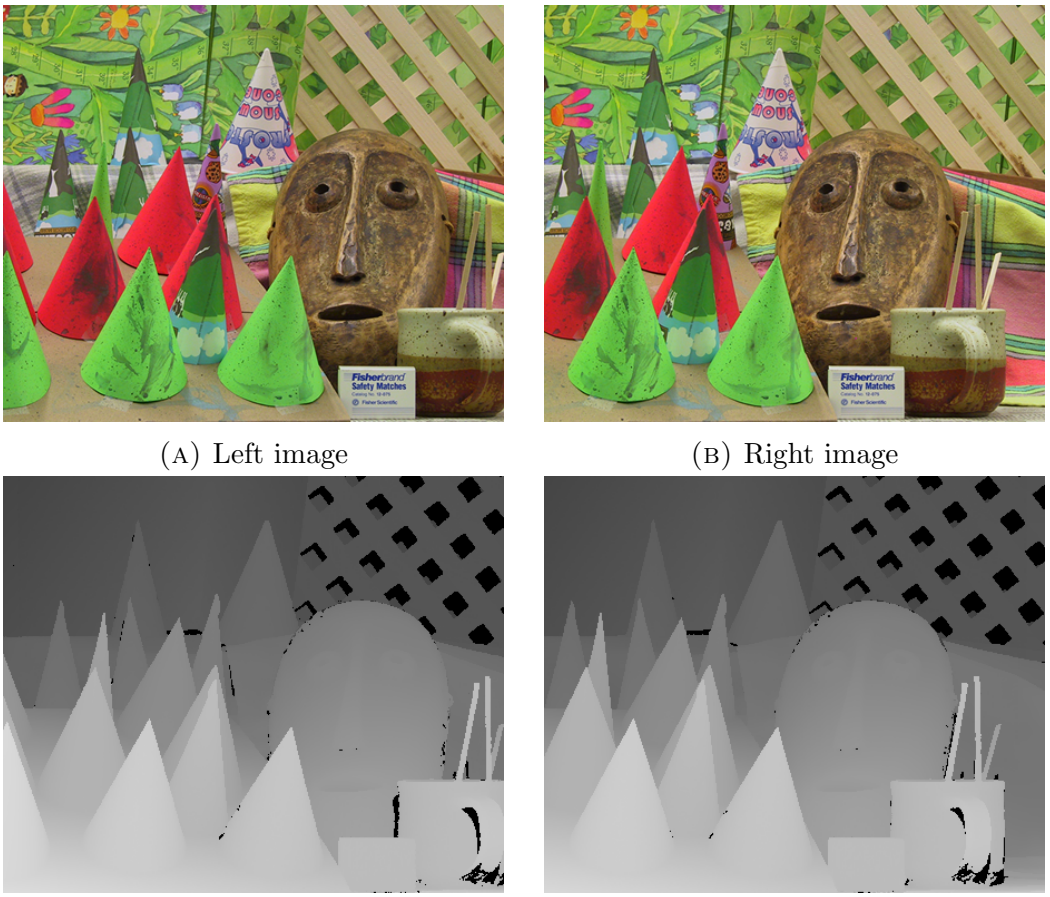
In figure 2.5, an illustration of epipolar view is shown. Each camera has a center of projection, called O_L and O_R, and corresponding projection planes. The point X has a projection to each projective planes, which is called X_L and X_R. The epipoles, e_L and e_R, is points intersecting between the image projective planes, and the line connecting the camera centres. The plane connected by the viewed point X, and the camera centres is called the epipolar plane, and the lines $X_L e_L$ and $X_R e_R$ are called the epipolar lines.

When we see a point that is projected onto an image plane, we have no information as to how far away that point is, just that it is on the line formed by the camera center, O_x, and the projection point X_x. This is a consequence of not knowing the distance to the object with only one camera. If we look at figure 2.5, the left camera only see the point X_L, which is the point X projected onto the left projection plane. The actual point X could be anywhere on the line made by $O_L X_L$, and the line does contain point X, but also other points. What is of interest, is to see what this line looks like projected onto the right image plane. This is in fact the epipolar line $X_R e_R$.

If we know the position of the cameras relative to each other, and the rotation of the cameras, we know that each projection point that is observed in one image plane, must lie on a known epipolar line in the other image plane. This gives us an epipolar constraint, and we can test to see if two points correspond to the same observed point. If such a correspondance exist, we can calculate the position of the point by using triangulation.

2.2.3 Disparity map

When finding corresponding features in the left and right picture, we can also look at the difference in the location of those features in both images. This difference is called the disparity [Qia97]. By doing this for every pixel, we get a disparity map or depth map. Each point in the disparity map represent the distance between a point in the reference picture (e.g. the left picture) and the corresponding point in the non-reference picture (e.g. the right picture). Figure 2.6 shows an example of disparity maps produced by a pair of images. The images are provided by the University of Tsukuba and made available on the Middlebury Vision website [SS03].



(A) Left image (B) Right image

(C) Left reference disparity map (D) Right reference disparity map

FIGURE 2.6: Stereo image and disparity map of cones.

In this example, figure 2.6a and 2.6b constitutes the stereo pair, with their respective disparity maps shown in figure 2.6c and 2.6d. The left disparity map is made with the left image as reference, and the right disparity map is made with the right image as reference. The disparity maps can easily be examined, as close objects are brighter than distant objects.

Creating the disparity map is done by giving the corresponding point a disparity value based on the distance difference. Corresponding points with no distance between them are placed at infinity and given a disparity value of 0. For points with distance 1, a disparity value of 1 can be given, and so on. The disparity values are often scaled to a set range, e.g. 0-255, to make it easier to view the disparity map and inspect it visually.

Converting to distance

Assuming we now have a stereo rig with two cameras with coplanar image planes, a known distance between the cameras, and equal focal lengths, we can find the distance to the point with the following equation:

$$Z = \frac{f * b}{d} \tag{2.3}$$

Where Z is the distance, f is the focal length, b is the baseline and d is the disparity value [JKS95]. We see that the depth is inversely proportional to the disparity between the images.

2.2.4 Stereo matching

To be able to produce the disparity map, and calculate the distance to the object, we need to implement a stereo matching algorithm. Stereo matching is used to find the corresponding pixels in an image pair used for the disparity map. A common approach in computer vision is to use a block matching algorithm [HIG02]. This approach divides the current frame into blocks, and slides the block over a search area in the other image to find the position where the blocks are most similar (i.e. where the minimum error in difference occurs). This gives us the disparity values used to create the disparity map.

To find these similarities between the blocks, a cost function is implemented. In this thesis the SAD method is used, due to the low matching time and ease of implementation. Other cost functions can be used, such as SSD, MSE, MAD, etc, but they typically require more computational power than SAD [HS07].

SAD:

$$SAD = \sum_{i=0}^{n} |(L_i - R_i)| \tag{2.4}$$

Chapter 3

Hardware and Software Implementation

This chapter will cover the various hardware and software used for evaluating the detection methods listed in chapter 2, and the implementation of said systems.

3.1 Hardware

This section presents the various hardware used for the testing carried out in this thesis.

3.1.1 Raspberry Pi 2

The main platform used in this thesis is the new Raspberry Pi 2 from Raspberry Pi Foundation. Both the computer vision module and the ultrasound sensor will interface with this board. This represent a low power ARM platform, and was chosen for its popularity in small robotics application, wide user community, low power requirement and relatively low cost. A list of specifications is given in table 3.1.

CPU:	Broadcom BCM2836 Arm7 Cortex-A7 (900MHz, 4 cores)
RAM:	1GB LPDDR2 SDRAM
USB:	4x USB 2 ports
GPIO:	40 pin
Ethernet:	Wired port
Storage:	Micro SD card slot
OS:	Raspbian (Debian based distribution)
Price:	$35 (as of 01.03.2015)

TABLE 3.1: Raspberry Pi 2 specifications

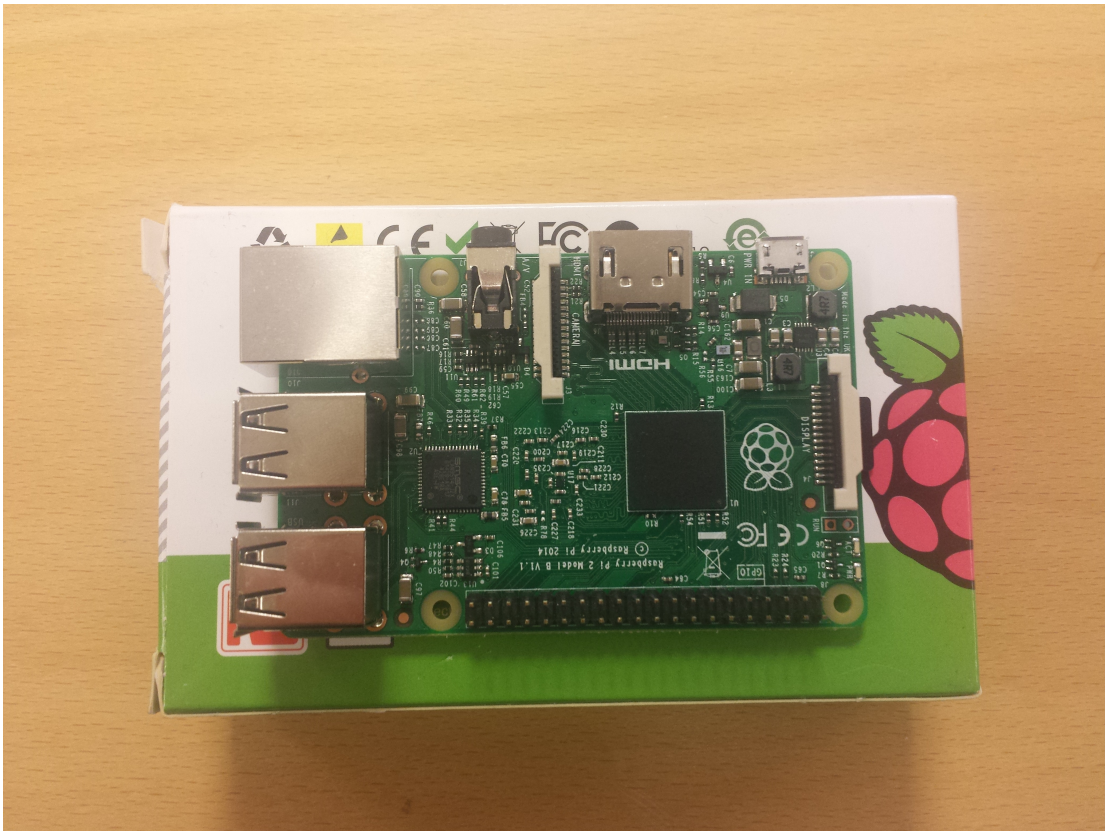


FIGURE 3.1: Raspberry pi 2

3.1.2 Desktop computer

A Dell Optiplex 990 desktop computer, and two monitors, was provided by NTNU to be used with the project work. This computer was used for writing code and text, upload code to the Raspberry Pi 2, and receive results from the Raspberry Pi 2 to display on the screens.

3.1.3 Logitech C920

The camera used for the computer vision system is the Logitech C920 webcamera, which is a standard USB webcamera used in everything from internet chatting to home surveillance. This camera model was chosen based on reviews online, and availability at NTNU. Logitech webcameras in general also has decent driver support in both linux and windows, making it easy to interface with. The C920 works out of the box on the default Raspbian distribution for Raspberry Pi. Other web-cameras may also be suited, but driver support should be considered before trying other models. A list of specifications is given in table 3.2.

Resolution:	1920 x 1080 maximum
USB:	USB connector
Video formats:	YUYV, H264 and MJPG
Price:	$70 (as of 01.03.2015)

TABLE 3.2: Logitech C920 specifications

FIGURE 3.2: Logitech C920

3.1.4 HC-SR04

For the ultrasonic system, the HC-SR04 sensor were used. This is a cheap ranging sensor consisting of two transducers, one transmitter and one receiver. This sensor is used in many different robotic projects, and are often found in starter kits for

robotic applications. This sensor was chosen based on previous experience and use in projects, as well as its performance and price. A list of specifications is given in table 3.3.

Working frequency:	40kHz
Min range:	2 cm
Max range:	400 cm
Resolution:	0.3 cm
Measure angle:	15°
Price:	$3 (as of 01.03.2015)

Table 3.3: HC-SR04 sensor specifications

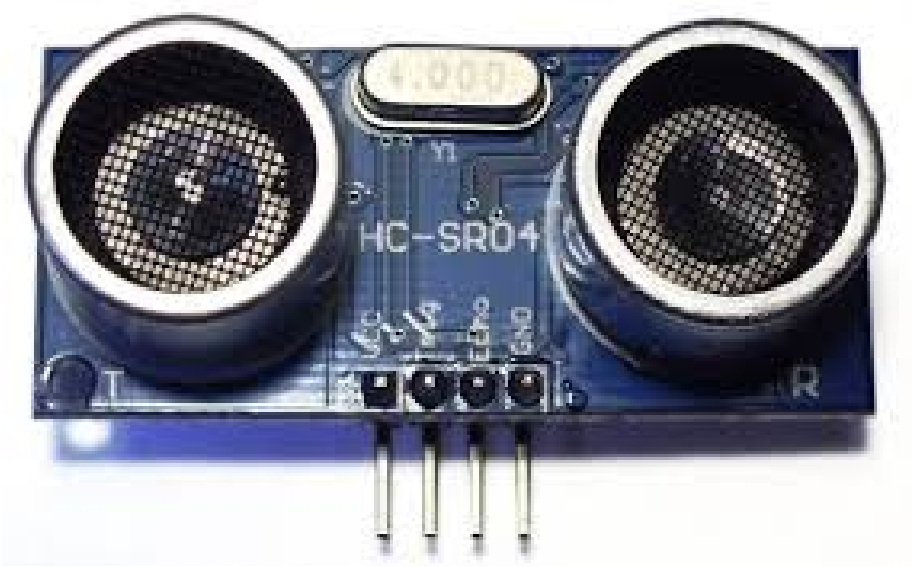


Figure 3.3: HC-SR04

3.2 Software

This section presents the main software used in this thesis.

3.2.1 OpenCV

OpenCV (Open Source Computer Vision Library), is an open source software library developed by Intel and Itseez, and focuses on computer vision and machine learning [BK08]. The library provides many functions for the necessary graphical operations needed to create computer vision applications. The version of OpenCV used in this thesis is 2.4.11, and being free to use under the open-source BSD license makes it a suitable choice for computer vision projects. In this thesis, OpenCV has been used for the following tasks:

- Set resolution for the web-cameras
- Capture images from the web-cameras
- Calibrate the web-cameras
- Convert images to gray scale
- Implements a Block Matching algorithm used for stereo matching
- Display both images from camera and disparity map after stereo matching

3.2.2 Operating systems

Raspbian

Raspbian is a Debian based operating system, optimized for the Raspberry Pi hardware. This is the default operating system that is installed on the memory card when buying a Raspberry Pi. It was decided to keep this as the operating system for the Raspberry Pi 2, since hardware accelerated floating point operations are enabled by default on this distribution.

Windows

The desktop computer came pre-configured with windows 7 from the IT-department at NTNU. It was decided to use this due to previous experience with this operating system, and not wanting to use time to replace it. Other operating systems could be used as well, and is mostly based on personal preference.

3.2.3 Miscellaneous

PuTTy

PuTTY is an open-source terminal emulator, which is used to communicate with the Raspberry Pi 2 using SSH. The main use of the terminal window is to login to the Raspberry Pi 2 without the need for a dedicated monitor and keyboard, and launch code on the Raspberry Pi 2 from the desktop computer.

Xming

Xming is a X Window System Server for windows, and provides a display system that can be used with PuTTY to display graphical applications from the Raspberry Pi 2, on the desktop computer. This removes the need for a monitor connected to the Raspberry Pi 2 completely, as everything can be controlled and displayed on the desktop computer.

Notepad++

Notepad++ is a text editor, which is used for writing and displaying source code. It supports multiple open files, displayed as tabs instead of separate windows, and syntax highlighting for the source code making easier to look at.

FileZilla

FileZilla is a graphical FTP software, which were used to transfer files between the Raspberry Pi 2 and the desktop computer.

3.3 Implementing ultrasound sensor

The ultrasonic sensor has a relatively straight forward process for implementation. The HC-SR04 sensor used in this thesis, has four pins for interfacing and is connected to the GPIO pins on the Raspberry Pi 2. A description of the GPIO header on the Raspberry pi 2 can be seen in figure 3.4.

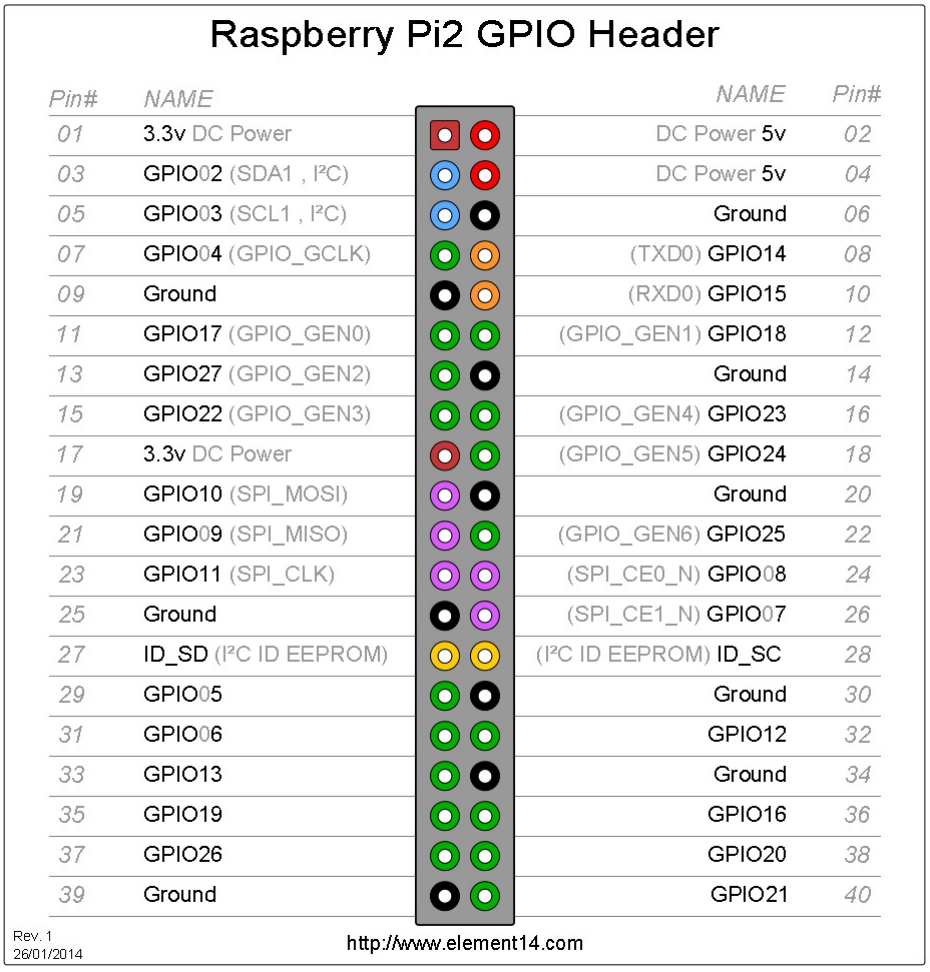


FIGURE 3.4: Raspberry Pi 2 GPIO, from element14.com

The pins on the HC-SR04 sensor for ceonnecting with external equipment is as follows:

- Voltage
- Ground
- Echo

- Trig

The HC-SR04 operates at 5V, that can be found directly on the Raspberry Pi 2 on pin two and four. Ground can be connected to any ground pin. The Trig and Echo pins are input and output pins respectively on the sensor, and can be connected on any GPIO pin on the Raspberry Pi 2. One problem that arouse during testing was that the GPIO pins on the Raspberry Pi 2 expects a 3.3V signal max, and the HC-SR04 sensor sends out a 5V signal. To solve this problem, a voltage divider was integrated in the circuit. A connection schematic can be seen in figure 3.5, and the ultrasonic system can be seen in figure 3.6.

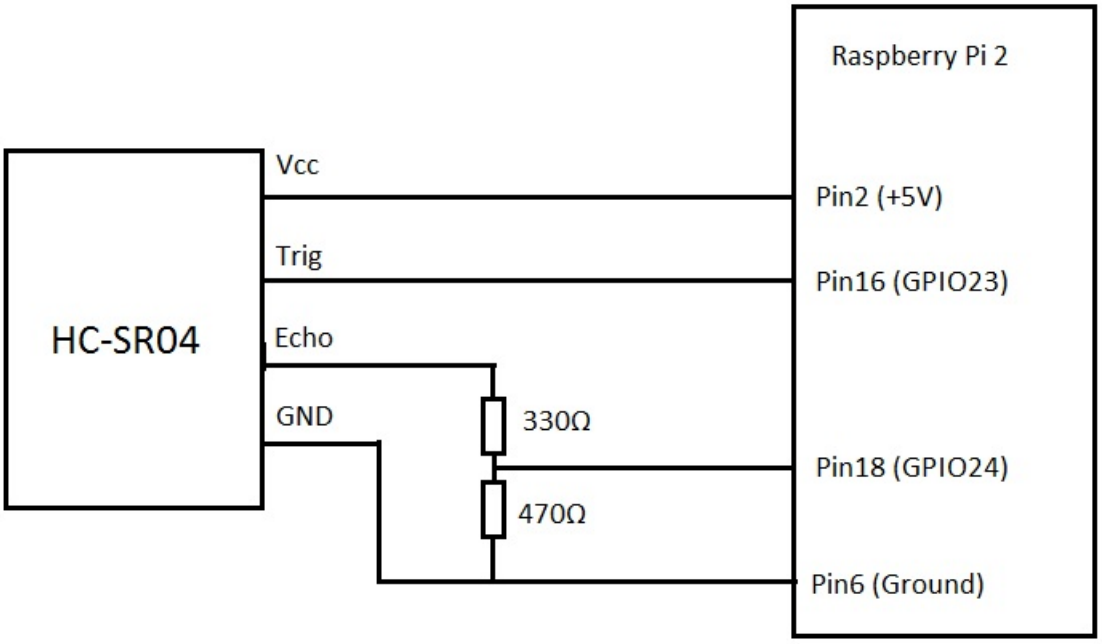


FIGURE 3.5: Connection schematic

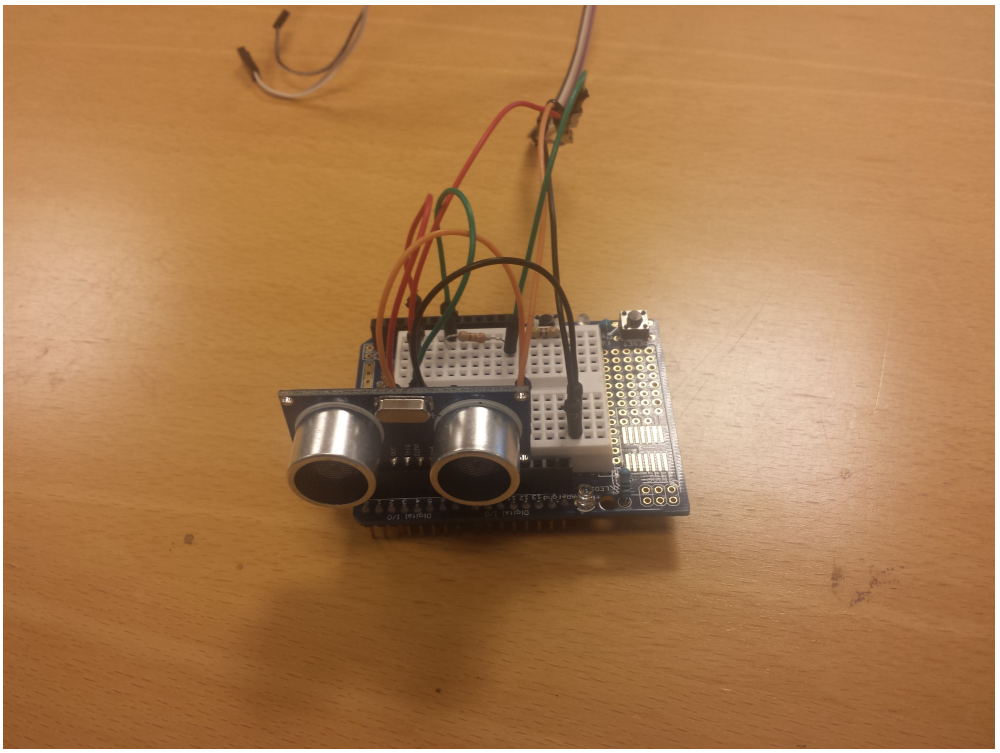


FIGURE 3.6: Ultrasonic system

Interacting with the sensor

The datasheet [Fre] specify how interaction with the HC-SR04 should be done with regards to triggering a signal, and waiting for a reply. Figure 3.7 shows the timing diagram used for the sensor.

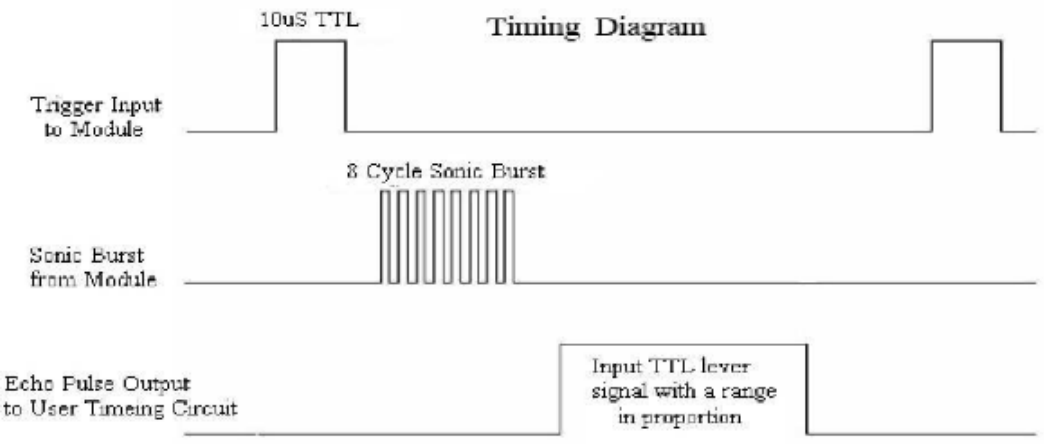


FIGURE 3.7: Timing diagram

The Raspberry Pi 2 sets a 10μs triggering signal on GPIO23 according to the connection schematic, and then starts a timer on to record the waiting time for the echo to return on GPIO24. When the echo returns to the ultrasonic sensor, the echo pin will go high and stop the timer. Depending on how long the wait time recorded by the timer is for the echo pin to go high, determines the calculated range. A basic block diagram showing how the ultrasonic system operates is shown in figure 3.8.

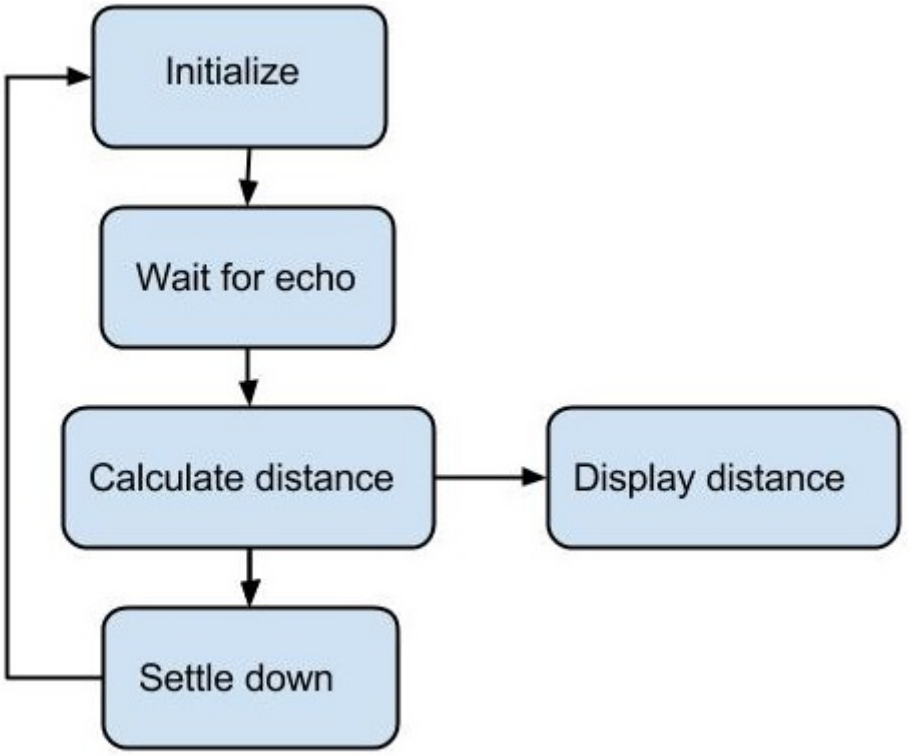


FIGURE 3.8: Ultrasonic block diagram

3.4 Implementing computer vision

Implementing a computer vision module requires several steps to reach a point where you can extract depth information from images. In this thesis, the following steps have been done to create the computer vision module:

- Create stereo rig
- Calibrate stereo rig
- Use OpenCV library to interact with the stereo rig and obtaining depth information used to calculate distance to objects

A basic block diagram showing how the computer vision module operates is shown in figure 3.9.

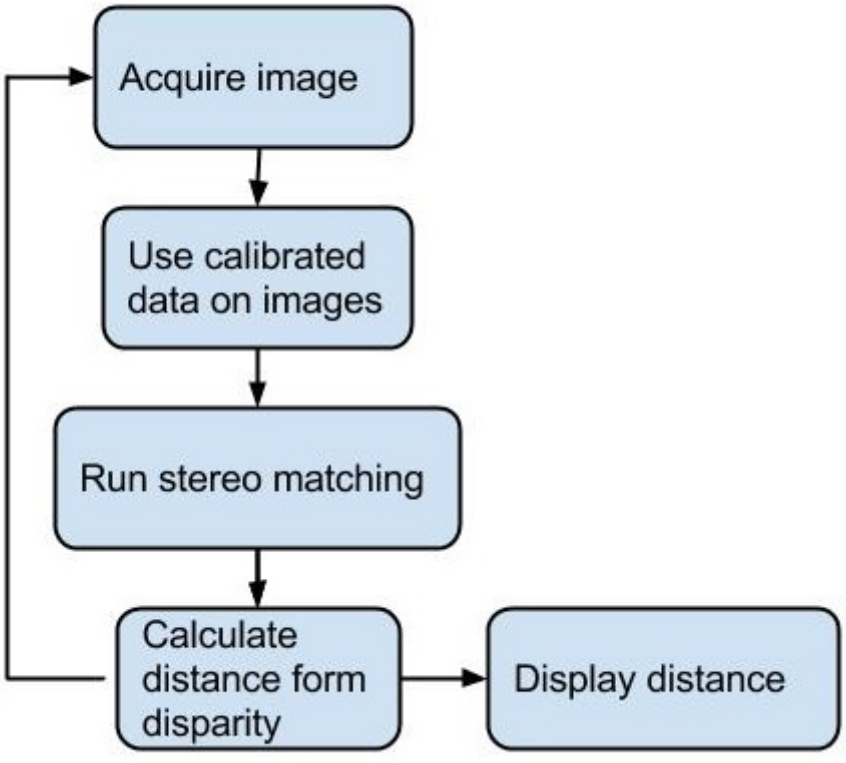


FIGURE 3.9: Computer vision block diagram

Stereo rig

The stereo rig used in the computer vision module to get image pairs was constructed using a pair of Logitech C920 webcameras. The stereo rig were constructed by mounting the two cameras on a firm plate taken from an old computer case, making it portable as well as sturdy to movement. The cameras were glued on the plate with a distance between the lenses of 10 cm, and taped down to ensure that they did not move during testing. A more suitable stereo rig should be considered for usage other then prototyping and testing. Figure 3.10 shows the stereo rig.

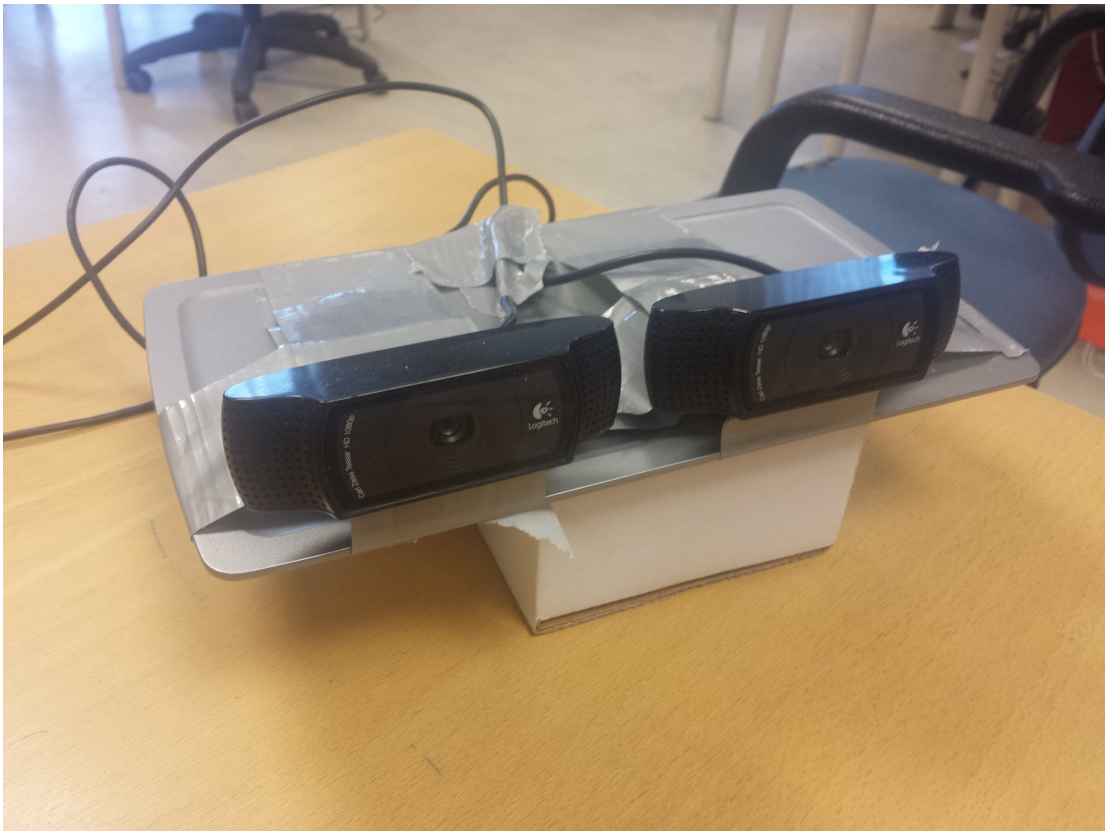


FIGURE 3.10: Stereo rig

Calibration

When manufacturing cameras and camera lenses, there could be some distortion presenting itself due to inaccurate manufacturing processes, and cost savings in producing the cameras. The webcameras used in this thesis, the Logitech C920, is a relatively cheap webcamera, and would need calibrating in order to work around the distortion problem.

Distortion most commonly presents itself as a radial distortion, due to the symmetry of the lens [JW76]. The radial distortions are commonly classified as either barrel distortion or pincushion distortion. With barrel distortion, the edges appear further away than they really are, and with pincushion distortion, the edges appear nearer than they really are. Figure 3.11 shows an example barrel distortion, and figure 3.12 shows an example of pincushion distortion.

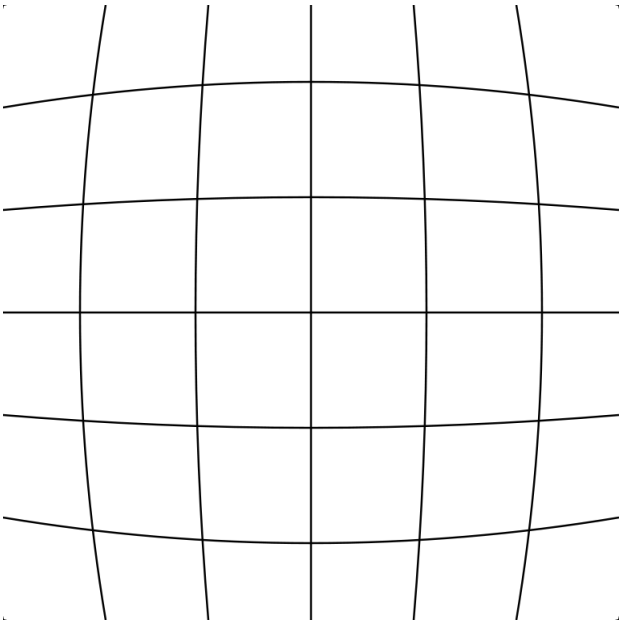


FIGURE 3.11: Illustration of barrel distortion [Wola]

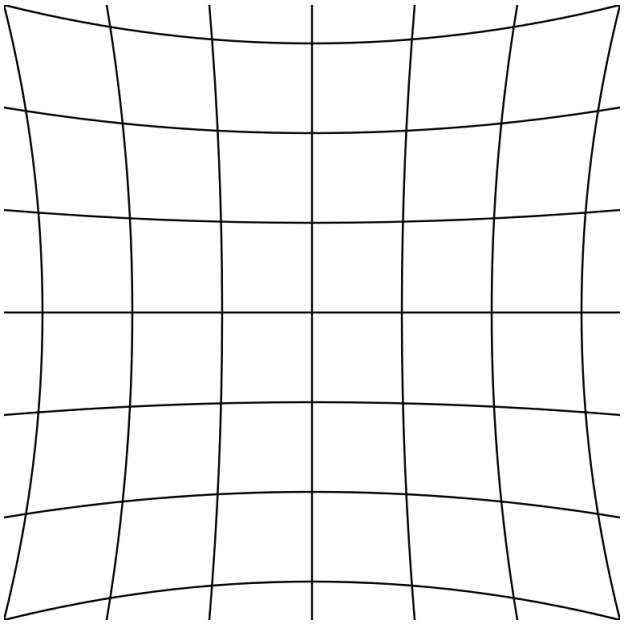


FIGURE 3.12: Illustration of pincushion distortion [Wolb]

OpenCV has built in functions to measure the distortion of the cameras, and calculate the intrinsic and extrinsic properties of the cameras. These properties can then be used to correct and undistort the images taken by the webcameras.

A standard procedure that OpenCV supports is using a chessboard for calibration. This is used because the corners in the board are easy to locate using computer vision algorithms, and the geometry has a simple design. Since we know the number of vertical and horizontal corners, as well as the distance between the corners on the chessboard we are using, the calibration code from OpenCV can be used to determine the properties of the cameras. Figure 3.13 shows the chessboard used in this thesis, which is a 9x6 chessboard with 2.6 cm between the corners, and figure 3.14 shows the result of the calibrated images.

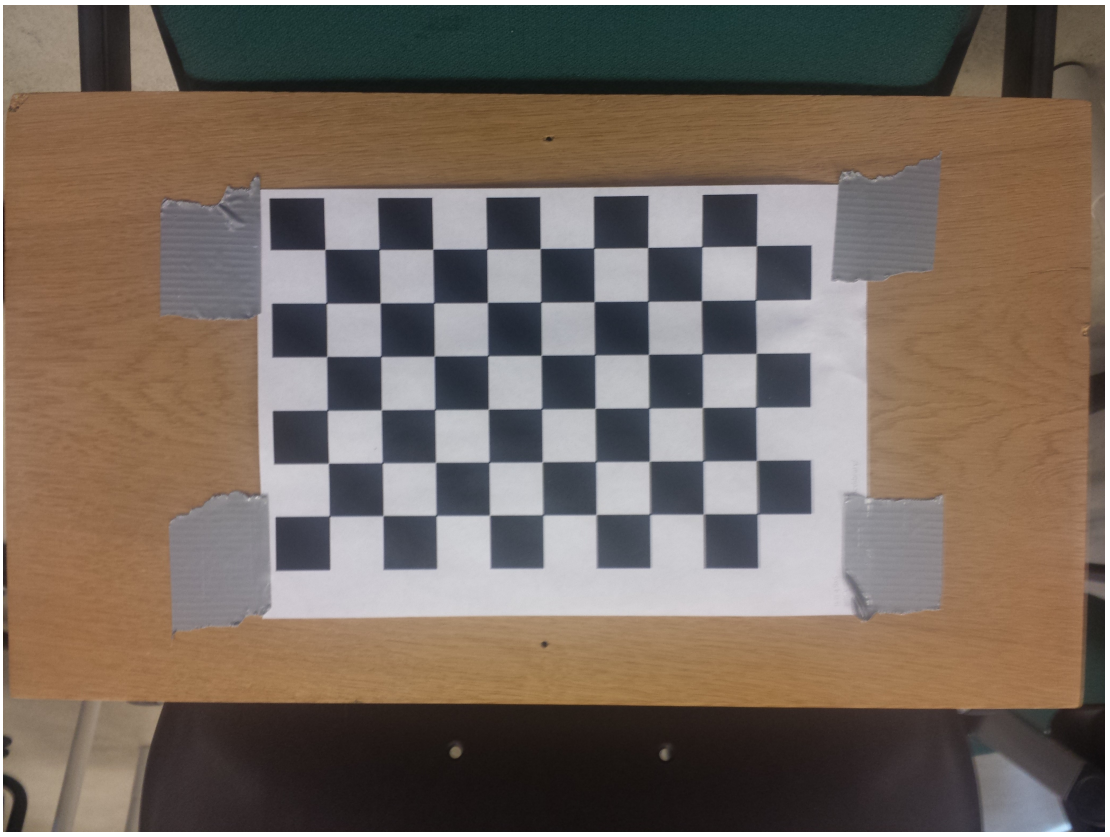


FIGURE 3.13: Chessboard used for calibration

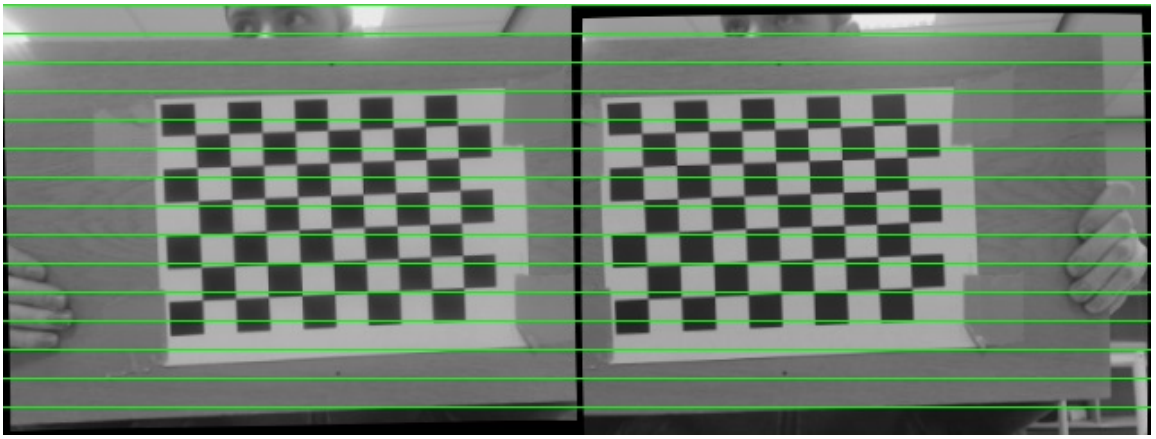


FIGURE 3.14: Calibrated images

3.5 Main code parts

This section will highlight some main parts in the code used to implement the computer vision system. A more detailed explanation can be found at [BK08] and looking at the source code appended.

Capture images

The camera properties must be set according to the hardware specification on the camera models used, and the hardware limitation on the host board. The height and width of the image taken can be set by using the following code:

cvSetCaptureProperty()

Acquiring images is done with the following code:

cvQueryFrame()

Both these actions must be done on both cameras in order to get correct results.

Calibrate

Calibrating is done using a chessboard pattern, as shown in figure 3.13. The corners are detected by using the following code:

findChessboardCorners()

Which generates matrices with the extrinsic and intrinsic properties for the cameras used.

Stereo matching

Using the images acquired and the properties found by calibrating the cameras, the last step is to run the stereo matching. First, the images are rectified using the following code:

cvRemap()

or

initUndistortRectifyMap()

The images are then ready to be processed by the stereo matching algorithm, which can be used with the following code:

cvFindStereoCorrespondenceBM()

3.6 Short user manual

For the computer vision module, a short user manual is provided below. This can be used to get the system up and running in a short amount of time.

1. Obtain all necessary hardware. The main items are the stereo rig and the Raspberry Pi 2, with accessories to be able to power up the items.

2. Connect the stereo rig to the Raspberry Pi 2. Using a default Linux distribution, drivers for Logitech webcameras are provided. The source code used in this thesis is appended on a digital file, as shown in appendix A.

3. Assuming the use of the source code used in this thesis, the only necessary requirements are installation of OpenCV on the Raspberry Pi 2, and compiling the source code. This will not be covered in this thesis. The source code are appended on a digital file as shown in appendix A.

4. For first time implementation, calibration should be run before the stereo matching program. This is done by taking 20 or more image pairs of a chessboard, where the entire chessboard is shown in both images, with different angles and distances for each image pair. When the image pairs are acquired, upload the images to "home/calibration/images". After the images are uploaded, navigate to "home/calibration" in a terminal window, type "./calibrate" without the quotes and press enter. After a short while, the resulting calibration files are produced. These files must be placed in "home/stereomatching/calibrationfiles" for use in the main program.

5. When the calibration process is done, the stereo matching program can be run. This is the main program used in this thesis. Navigate to "home-/stereomatching" in a terminal window, type "./stereomatching" without the quotes and press enter. This brings up a display showing both cameras, as well as the resulting disparity map. In the terminal window, the distance to an object identified by the pointer is displayed.

Chapter 4

Testing and Results

In this chapter a discussion of the test procedures carried out on the computer vision and ultrasonic systems are provided, and the results obtained from the testing is discussed. In section 4.1, the test setup is described for the systems. In section 4.2, the results from the testing is presented.

4.1 Test setup

The tests performed is run to investigate some basic properties of a collision detection system, to see how well the implemented systems perform. In this thesis, the test procedures will examine the following properties in a collision detection system:

- The refresh rate.
- The accuracy.

In a basic and simple collision detection system, we check to see if anything is closer than a set threshold; if it is, then either stop movement or change direction. In such a system, both the refresh rate and the accuracy is important information to have when developing applications that uses a collision detection system. This information will limit how fast the application can move based on the refresh rate of the collision detection system, and how much error margin must be implemented due to errors in distance readings.

4.1.1 Refresh rate testing

Both systems in this thesis will be tested with regards to the refresh rate, to see how often a measurement reading can be taken. The test is implemented to take time measurements in certain parts of the code, and then calculate after each reading how much time was used on the separate parts, as well as the total running time. This will provide data that can be used to compare other systems to the computer vision system implemented in this thesis, or other components in similar computer vision systems. In this thesis, time measurements will be acquired before and after each of the items listed for the systems:

Ultrasonic

- Initializing.
- Waiting for echo.
- Calculating distance.
- Settling down.

Computer vision

- Acquire image.
- Use calibrated data on images and convert to grayscale.
- Run stereo matching.
- Calculate distance from disparity.

For the ultrasonic system, the reading that is expected to differ most is the duration of waiting for echo, as that is directly dependent on the distance to the object that is measured. The other readings are mostly determined by the datasheet of the sensor, and the computational power of the Raspberry Pi 2.

For the computer vision system, the readings are expected to stay mostly the same, since to the only variable changing is the objects distance to the cameras. If more objects were to be introduced in the scene, the time to run the stereo matching algorithm would change due to more similarities being calculated. This will not be looked at in this thesis.

4.1.2 Accuracy testing

Both systems will be tested with regards to the accuracy, to see how much difference there is in calculated distance by the systems compared to measured distance by hand to an object at a set distance. The test is implemented by placing a number of objects at known distances, and run each system to see what the calculated distances are, and then compare the results.

The objects used for testing were:

1. A toolbox
2. A bottle
3. A backpack
4. A shelf plate

This selection of objects provides different shapes and sizes for the systems to be tested against, and provides a decent foundation for comparison. Other objects may be chosen, but the objects used in this thesis were considered good enough for testing the ultrasonic and computer vision systems. Figure 4.1 shows the objects.

FIGURE 4.1: Objects used for testing accuracy

All of these objects will be placed individually at the distances listed below and then each system will perform a distance reading to the objects.

- 0.5 metres
- 1 metres.
- 1.5 metres.

This setup procedure provides a decent foundation for comparison both to the systems tested in this thesis, as well as external systems. The following conditions were not considered used or tested due to time limitations:

- Other objects in terms of size and shape.
- Other distances.

4.2 Results

In this section, the results obtained from testing the systems will be discussed.

For the refresh rate, the results from each system will be presented in separate tables, due to different components and timing measurements as listed in chapter 4.1.1. A comparison of the total running time for each system will be presented in a common table.

The accuracy results for each system will be presented in a common table for each test as described in chapter 4.1.2.

4.2.1 Refresh rate

The refresh rate testing is divided into six subsections to fully give a description as to how well the systems perform in terms of speed. A description of the subsections is as follows:

- Test 1 presents the setup given in chapter 4.1.1 for the ultrasonic system at a distance of 0.5 metres.
- Test 2 presents the setup given in chapter 4.1.1 for the ultrasonic system at a distance of 1 metres.
- Test 3 presents the setup given in chapter 4.1.1 for the ultrasonic system at a distance of 1.5 metres.
- Test 4 presents the setup given in chapter 4.1.1 for the computer vision system at a distance of 0.5 metres.
- Test 5 presents the setup given in chapter 4.1.1 for the computer vision system at a distance of 1 metres.
- Test 6 presents the setup given in chapter 4.1.1 for the computer vision system at a distance of 1.5 metres.

At the end the total running time for each system is presented. This is gathered from the results in test 1 through 6 for each system, and combined in a total

running time.

Test 1

Initializing	Waiting for echo	Calculating distance	Settling down
10 μs	2.9 ms	52 μs	60 ms

TABLE 4.1: Refresh rate for ultrasonic system at 0.5 metres

Test 2

Initializing	Waiting for echo	Calculating distance	Settling down
10 μs	5.9 ms	51 μs	60 ms

TABLE 4.2: Refresh rate for ultrasonic system at 1 metres

Test 3

Initializing	Waiting for echo	Calculating distance	Settling down
10 μs	8.8 ms	53 μs	60 ms

TABLE 4.3: Refresh rate for ultrasonic system at 1.5 metres

Test 1 through 3 shows the time measurements taken for the ultrasonic system at the different distances. The initializing time is the same at every test, and is as expected due to the specification for the ultrasound sensor. This could be reduced by choosing a more high end ultrasound sensor, but for general purposes this value is so low that it would not be noticeable.

The waiting for echo is increasing linearly for each test, and confirms the theory of the relation between distance and the speed of sound. This value would be the same regardless of which ultrasound sensor were used.

Calculating the distance takes roughly the same time for each test, and is limited by the computational power of the platform used for interfacing with the ultrasound sensor. The deviation from 51-53μs may be caused by hardware limitations in timing measurements on the Raspberry Pi 2, but the values are so close

that it has no real world impact. As with the initializing time, this value is so low that using a more powerful platform would not result in a noticeable speed-up on the overall system.

The settling down time is the most interesting value, as it is by far the highest one. The datasheet suggest a settling down time of around 50-70ms, due to some hardware limitations presented in the design. What was interesting to observe was that by lowering the settling down time to anywhere below 60ms, the sensor would report random distance readings ranging from 0-200. By choosing a different ultrasound sensor, this limitation could be improved, and is something that should be considered for future implementations.

Test 4

Acquire image	Use calibrated data on images	Run stereo matching	Calculate distance from disparity
37 ms	45 ms	136 ms	7.5 ms

TABLE 4.4: Refresh rate for computer vision system at 0.5 metres

Test 5

Acquire image	Use calibrated data on images	Run stereo matching	Calculate distance from disparity
37 ms	46 ms	135 ms	7.4 ms

TABLE 4.5: Refresh rate for computer vision system at 1 metres

Test 6

Acquire image	Use calibrated data on images	Run stereo matching	Calculate distance from disparity
37 ms	48 ms	137 ms	7.6 ms

TABLE 4.6: Refresh rate for computer vision system at 1.5 metres

Test 4 through 6 shows the time measurements taken for the computer vision system at the different distances. The acquire image time is the same at every test, and is to be expected due to limitations in the camera hardware and available bandwidth on the connection bus. Choosing other camera models could possible change this value.

Use calibrated data on images duration is roughly the same for each test, and describes the time taken for the platform to remap the images taken with the calibration data obtained as discussed in chapter 3.4. The small deviation in duration may be caused by the different images that is taken from the different distances. This step could be avoided to a certain degree by choosing a stereo rig with camera models specifically made for this purpose, where calibration would be kept at a minimum. This is something that should be considered for future implementations.

Running the stereo matching algorithm is the most time consuming task in the computer vision system. This is limited by the computational power on the platform, as well as how much similarity has to be calculated in the image sets. The deviation in duration may be accounted for by the different disparity values produced from the image sets taken from the different distances. This is something that a more powerful platform could improve on, but often when choosing a more powerful platform, the size and cost would increase.

Calculating the distance takes roughly the same time for each test, and is limited by the computational power on the platform used. As with the stereo matching duration, this value could be improved by choosing a different platform, but is so low that it would not result in a noticeable speed-up.

Total running time

System	Total running time at 0.5 metres	Total running time at 1 metres	Total running time at 1.5 metres
Ultrasonic	62.962 ms	65.961 ms	68.863 ms
Computer vision	225.5 ms	225.4 ms	229.6 ms

Table 4.7: Total running time for the systems

The total running time is shown above for both systems. The ultrasonic system is much faster than the computer vision system at every distance tested in this thesis. The ultrasonic system range from 62ms per reading to 69ms per reading, depending on distance to the object. Compared to the computer vision system, the ultrasonic system is about 3.5 times as fast. How this affects the application using the systems depends entirely on the requirements for the application. For a slow pacing unit, both the ultrasonic and computer vision system provides fast enough readings, but for faster applications, the ultrasonic may be a better choice.

4.2.2 Accuracy

The accuracy testing is divided into four subsections representing the four objects the tests were run on. A figure is included in each subsection showing how the object was placed, and a table with the results from each distance reading is presented. A description of the subsections is as follows:

- Test 7 presents the results obtained from testing on the toolbox as described in chapter 4.1.2.
- Test 8 presents the results obtained from testing on the bottle as described in chapter 4.1.2.
- Test 9 presents the results obtained from testing on the backpack as described in chapter 4.1.2.
- Test 10 presents the results obtained from testing on the shelf plate as described in chapter 4.1.2.

Test 7

FIGURE 4.2: Scene for toolbox

System	Calculated at 0.5 metres	Calculated at 1 metres	Calculated at 1.5 metres	Avg error percent
Ultrasonic Error	50.15 cm 0.3 %	100.85 cm 0.85 %	150.34 cm 0.226 %	0.46 %
Computer vision Error	51.62 cm 3.24 %	101.23 cm 1.23 %	150.39 cm 0.26 %	1.6 %

TABLE 4.8: Accuracy for toolbox

Test 8

FIGURE 4.3: Scene for bottle

System	Calculated at 0.5 metres	Calculated at 1 metres	Calculated at 1.5 metres	Avg error percent
Ultrasonic Error	50.32 cm 0.64 %	100.54 cm 0.54 %	150.91 cm 0.61 %	 0.6 %
Computer vision Error	50.87 cm 1.74 %	100.84 cm 0.84 %	150.46 cm 0.3 %	 0.96 %

TABLE 4.9: Accuracy for bottle

Test 9

FIGURE 4.4: Scene for backpack

System	Calculated at 0.5 metres	Calculated at 1 metres	Calculated at 1.5 metres	Avg error percent
Ultrasonic Error	50.71 cm 1.42 %	100.47 cm 0.47 %	150.78 cm 0.52 %	 0.8 %
Computer vision Error	51.52 cm 3.04 %	101.43 cm 1.43 %	151.32 cm 0.88 %	 1.8 %

TABLE 4.10: Accuracy for backpack

Test 10

FIGURE 4.5: Scene for shelf plate

System	Calculated at 0.5 metres	Calculated at 1 metres	Calculated at 1.5 metres	Avg error percent
Ultrasonic Error	50.87 cm 1.74 %	100.13 cm 0.13 %	150.21 cm 0.14 %	0.67 %
Computer vision Error	51.24 cm 2.48 %	100.45 cm 0.45 %	151.58 cm 1.05 %	1.33 %

TABLE 4.11: Accuracy for shelf plate

Test 7 through 10 shows the accuracy results for both systems according to the test setup given in chapter 4.1.2.

In test 7, both systems were able to detect the object, and calculate a distance. The distance that the ultrasonic system calculated were overall a bit more accurate than the computer vision system, especially at closer range. At 0.5 metres, the ultrasonic system calculates the distance with an error of 0.3%, whereas the computer vision system calculates the distance with an error of 3.24%. This difference decreases as the distance increases, and at 1.5 metres, the ultrasonic system calculates the distance with an error of 0.226%, and the computer vision system calculates the distance with an error of 0.26%. The computer vision system shows a larger deviation in the readings than the ultrasonic system. The accuracy increases with larger distance, and comes close to the ultrasonic system at 1.5 metres. This may be caused by the properties of the stereo rig constructed, and may be a limitation in the Logitech webcameras as well as the calibration method.

In test 8, the accuracy for the computer vision system changes a bit compared to test 7, and gets closer to the ultrasonic system. At 0.5 metres, the ultrasonic system has an error of 0.64%, and the computer vision system has an error of 1.74%. When the distance increase to 1.5 metres, the computer vision system delivers better accuracy than the ultrasonic system, with an error of 0.3%, versus 0.61% error for the ultrasonic system. This could be caused by the reflective properties of the bottle compared to the other objects used for testing in this thesis. What was interesting to notice when testing on the bottle was that the ultrasonic system required much more precise positioning to be able to get good readings, especially at 1.5 metres. This could explain that the computer vision system was able to get better accuracy at that distance. As with test 7, the same pattern with error deviation occurs, in that the computer vision system gets better accuracy with larger distance.

Test 9 and 10 shows the same as test 7, where the ultrasonic system generally had better accuracy than the computer vision system. The computer vision system had a bit more trouble calculating the distances on the object in test 10 compared to the ultrasonic system. This could be caused by the object shape, where as the large, flat shelf provided a decent structure for the ultrasonic wave to bounce off, the computer vision seemed to struggle with finding similarities when

running the stereo matching algorithm.

Both systems delivered readings with an average error below 2% in all the tests, which in most applications using such low cost components would be considered good enough. The computer vision system was less depending on positioning than the ultrasonic system, especially with smaller objects such as the bottle and at greater distances, but had a bit higher error in the readings. The computer vision system had a larger error deviation than the ultrasonic system, with the highest error being 3.24%. The deviation is small enough for general applications, that there is no real disadvantage to using the computer vision system. For more professional applications, steps to reduce the error and error deviation should be taken.

What the best choice is, is entirely dependent on what type of application the systems are to be used on, and in which type of environment the application is supposed to be operating in. Based on the results obtained in this thesis, a computer vision system can be used for collision detection.

Chapter 5

Discussion

This chapter will present a general discussion on the project and work process used to implement and test the detection systems, and present some ideas for future work.

5.1 Goal and method

The goal was to implement computer vision on a low power platform, with all necessary hardware and software, and then compare it against an ultrasonic system to see if it could be a viable option for a collision detection system. The main focus was on creating a runnable system, with as simple as possible modules to get to a working point, before focusing on further improvement to the systems. This turned out to be a good choice, since implementation of both hardware and software took more time than anticipated getting to a point with runnable systems.

Looking back at the project, it may have been more advantageously to focus on smaller parts of the overall system, instead of implementing a complete system from scratch in the limited time available. With focus on smaller parts of the system, we may have been able to obtain better results from specific parts that could have been used for a more commercialized product, but implementing a complete system that was runnable was an interesting task and provides a basis for further work to improve on. By implementing a complete system, this thesis also provides a guide for a complete implementation that can be used by others.

5.2 Main results

The goal was to implement computer vision on a low power platform, with both hardware and software to see if it could be an alternative for collision detection. The resulting system managed to calculate distances to objects placed at different lengths and with different shapes, with sufficient accuracy and speed compared to an ultrasonic system, meaning the goal of implementing the computer vision system on a low powered platform has been achieved.

The computer vision system implemented in this thesis was able to calculate all the distances from the test setup as shown in chapter 4.2 with small variations in refresh rate and accuracy. The results are displayed in tables under chapter 4.2 for a user friendly presentation that can serve as a foundation for comparing when improving the system or creating other systems.

The stereo rig that was created during this thesis performed quite well considering the low price for the web cameras and what applications they are usually used in. Overall accuracy was not quite as good as the ultrasonic sensor, but not too far off considering the construction of the stereo rig. The downside to making the stereo rig, including the accuracy errors, is the need for calibration using software. This is a time consuming part as seen in chapter 4.2, and is something that could be worked on to reduce. The alternative was to buy a pre-made and pre-calibrated stereo rig, but they usually cost upwards of $2000. This was not considered beneficial for this thesis, but could be considered for another implementation.

The resulting implementation of the computer vision system must be seen as a prototype. As it stands now, it could be used directly on applications undergoing testing and developing, or it could be used in a laboratory setting with focus on improvement to the system. It is not a ready to use commercial system for applications both in functionality and aesthetics, as it uses parts and solutions that has not undergone enough testing for commercial approval. This is a consequence of the focus on creating a runnable system within the set time limit.

5.3 Future work

As it stands now, the computer vision system implemented in this thesis does not provide any more functionality than what a basic ultrasonic system does, in terms of collision detection. For further work, two forks is outlined as possible routes to follow, improving on the functionality on the existing system, and focusing on design and user friendly operation.

5.3.1 Improving functionality

The computer vision system implemented in this thesis provides fundamental functionality for a collision detection system. With computer vision, several features could be added for improved functionality. Some ideas for features that can be added to the system in this thesis is as follows:

- Object detection.
- Distance to multiple objects.
- A tracking mechanism that enables following of specific objects.

5.3.2 Improving design

The stereo rig built for the computer vision system consists of cheap webcameras from Logitech, and is fastened to a plate holding them inplace. For a commercial product, a more permanent solution should be considered, and may include different cameras and mounting options. A stereorig that can be moved around without risking movement to the placement of the cameras, as well as a rig that can be mounted easily is desirable. A case for the entire system, with both the stereo rig and the Raspberry Pi 2 can be designed and built.

Chapter 6

Conclusion

This thesis is the result of the project work conducted at NTNU during the spring 2015, and has investigated the possibility of implementing a computer vision system on a low powered platform.

A stereo rig were constructed using Logitech web cameras, and connected to a Raspberry Pi 2. The computer vision library OpenCV was used for implementing calibration and stereo matching on the Raspberry Pi 2, and resulted in a runnable system. A short user manual is also provided for the computer vision system to get it up and running in a short amount of time.

The test procedure investigated two properties in a collision detection, the refresh rate and the accuracy. The test setup for both is described in the thesis, and the results obtained presented in dedicated tables for each test. In addition to the computer vision system, an ultrasonic system was implemented to use for comparing the results obtained from testing the computer vision system. The results for both systems were then presented in a user friendly format which others can use as a basis for usage and comparison. Based on the implemented computer vision system in this thesis, and the results obtained through testing, computer vision may be a good alternative for a collision detection system.

The main contributions of this thesis is a computer vision subsystem that can be integrated for use into other applications directly. This system is also a foundation to be used for improvements and construction of a better system.

Appendix A

Digital attachment

A.1 Source code

Various source code and example code.

A.2 Papers

Background literature used in the thesis.

A.3 Images

Images used in the thesis.

Bibliography

[BK08] G. Bradski and A. Kaehler. *Learning OpenCV*. O'Reilly Media, Inc, 2008.

[Ens88] Dale Ensminger. *Ultrasonics: fundamentals, technology, applications*. Marcel Dekker, New York, 1988.

[Fre] Elec Freaks. Ultrasonic ranging module hc - sr04, accessed 2015. www.micropik.com/PDF/HCSR04.pdf.

[FS78] V. M. Faires and C. M. Simmang. *Thermodynamics*. Macmillan, 1978.

[HIG02] H. Hirschmüller, P. R. Innocent, and J. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision*, vol 47:229–246, 2002.

[HS07] H. Hirschmüller and D. Scharstein. Evaluation of cost functions for stereo matching. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[HZ04] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[JKS95] R. Jain, R. Kasturi, and B. G. Schunck. *Machine vision*. MacGraw-Hill, 1995.

[JW76] F. A. Jenkins and H. E. White. *Fundamentals of Optics*. Macmillan, 1976.

[Kut91] Heinrich Kuttruff. *Ultrasonics fundamentals and applications*. Elsevier Applied Science, New York, 1991.

[Qia97] Ning Qian. Binocular disparity and the perception of depth. *Neuron*, 18:359–368, 1997.

[SS03] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol 1:I–195–I–202, 2003.

[Tea] TeachEngineering. The source of this material is the teachengineering digital library collection at www.teachengineering.org. all rights reserved. accessed may 2015. https://www.teachengineering.org/view_activity.php?url=collection/nyu_/activities/nyu_soundwaves/nyu_soundwaves_activity1.xml.

[Wola] WolfWings. Barrel distortion by wolfwings - own work. licensed under public domain via wikimedia commons, accessed may 2015. http://commons.wikimedia.org/wiki/File:Barrel_distortion.svg#/media/File:Barrel_distortion.svg.

[Wolb] WolfWings. Pincushion distortion by wolfwings - own work. licensed under public domain via wikimedia commons, accessed may 2015. http://commons.wikimedia.org/wiki/File:Pincushion_distortion.svg#/media/File:Pincushion_distortion.svg.