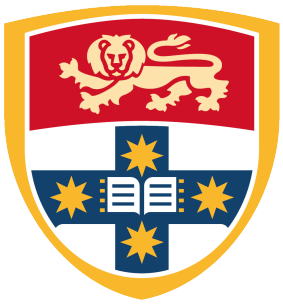
Vision Based Robotic Manipulation



THE UNIVERSITY OF
SYDNEY

Shreyash Annapureddy

Engineering

University of Sydney

2015

Contents

1 Introduction **6**

1.1 Background 6

1.2 Problem Statement 8

1.3 Scope/Focus Area 8

2 Literature Review **9**

2.1 Visual Servoing 9

2.2 Visual Servoing Architecture 10

2.2.1 Image Based Visual Servoing (IBVS) 11

2.2.2 Position Based Visual Servoing (PBVS) 15

2.2.3 Hybrid Visual Servoing 16

2.3 Summary 17

3 Computer Vision and Processing **19**

3.1 Camera Calibration 19

3.2 Feature Extraction for Object Recognition 27

4 Object Model and Depth Estimation **40**

5 Results **41**

5.1 Experimental Procedure 41

5.2 Depth Estimation 43
5.3 Pose estimation 50

6 Discussion **54**
6.1 Depth Estimation 55
6.2 Pose Estimation 56
6.3 Fusing PBVS and IBVS schemes 57

7 Conclusion and Future Work **58**

Abstract

Robotic systems are used everywhere in the manufacturing industry, robots such as those developed by ABB or Faunc are designed for high volume production. Robots designed for these purposes require specialised facilities requiring large sums of initial capital to set up such facilities. Baxter is a new generation of robots, a low cost robot used for low yield manufacturing that is capable of working in existing manufacturing facilities in collaboration with humans. Currently Baxter is used by teaching Baxter way-points in 3D space by a human, which Baxter then repeatedly follows, It would prove far more advantageous if Baxter has a dynamic method to reach and pick place objects using its on-board cameras in its hands. This thesis explores the use of camera data and models of desired objects in order to estimate the objects position in 3D space. The majority of this thesis focuses on the viability and how effective the method of using constrained based models of a desired object to estimate its position is. The system was first tested using a VGA webcam to verify if the optimisation models was able to estimate depth and position, which was then implemented on Baxter and tested.

List of Figures

2.1 Control loop for a Dynamic Look and Move system 11
2.2 Control loop for a Image Based Visual Servoing System 14
2.3 Control loop for a Position Based Visual Servoing System 16
2.4 Control loop for a 2.5D control scheme 16

3.1 Pin Hole Camera model 20
3.2 Corner extraction of the chess board 23
3.3 Grid extraction of the chess board 24
3.4 Position of Chessboard w.r.t to camera frame 26
3.5 Position of Chessboard w.r.t to global world frame 26
3.6 Re-projection error of pixels 27
3.7 Raw Image captured from Baxter's left hand camera 28
3.8 Gamma corrected, grey scaled and square extracted image 29
3.9 Pixel signal 30
3.10 Plotting points of an image using H,S and V as axis coordinates 30
3.11 Over-saturation of colours in an image 31
3.12 The Left is the FFT of the over-saturated image on the right 32
3.13 Discrete Fourier Transform of a square in a zero noise environment 33
3.14 2D convolving the over-saturated image with a low pass filter kernel 34
3.15 Extracting square from Image 36

3.16 Frequency response of the laplacian kernel 37
3.17 Edge sharpened image 38
3.18 Final result of object recognition 39
3.19 Object Extraction of other geometric shapes 39

5.1 Common apparatus and initial set-up for all experiments 42
5.2 Placement of the ruler on the gripper pincer. The empty gap from 30cm to the edge of the ruler is an additional 1.1cm measured using another ruler 43
5.3 Depth estimation error for gripper directly above the centroid of the square 45
5.4 Depth estimation error for gripper to the right of the square with an angled gripper 46
5.5 Depth estimation error for gripper to the left of the square with an angled gripper 47
5.6 Depth estimation error for gripper in front of the square with an angled gripper 48
5.7 Depth estimation error for gripper behind the square with an angled gripper 49
5.8 Starting position of Baxter 50
5.9 Pincer positioned down moving at 35% of δP 52
5.10 Pincer positioned at an angle moving at 35% of δP 52
5.11 Pincer positioned down moving at 12% of δP 53
5.12 Pincer positioned at an angle moving at 12% of δP 53

Chapter 1

Introduction

1.1 Background

Robotic systems come in a variety of configurations, sizes and complexity, depending on the end-users need. Moving forward, it is important to define what makes a system a robotic system, for a device to be considered a robot, it must complete tasks using the following three steps:

1. **Sense**: Taking measurements of the surrounding environment using sensors attached to the machine.

2. **Logic**: Algorithm to make sense of the sensor data and make decisions based on the data.

3. **Actuation**: Moving motors or solenoids that interact with the external world.

A common place where robots, more specifically, robotic arms, are used are in manufacturing plants with typical use cases being wielding, de-moulding, pick and place(P and P), and assembly. Robots used in these industries utilise simple range finding sensors to sense the positions of an object, and use simple binary logic to determine when to move

in a pre-programmed path. It is common for the robotic arms to have purpose designed end of arm tools(EOAT) to complete desired tasks, and example of this are EOATs used for de-moulding. Typically they are made of 6061 aluminium and made up of multiple pneumatic grippers on either side of a EOAT frame, with multiple range sensors to determine how far the grippers are from the moulded part. The EOAT also has multiple magnetic proximity sensors to determine the state of the grippers, either open or close. The robotic arm is then programmed by an operator, moving the arm to desired way-points, then records the way-point to be moved depending on a binary sensor signal from the moulding machine and range sensors.

Robotic arms are used, such as those listed in table 1.1, due to their speed and accuracy in accomplishing either medial repetitive tasks, or the task is too dangerous for humans. In order to achieve these goals, manufacturing firms are required to invest large sums of capital to design and build the EOAT, as well as new facilities to accommodate the robotic arms.

Table 1.1: Traditional Industrial Robots

Type	**Typical Use Case**
Cartesian Robots	Applying Adhesive, Welding, P and P
Scara Robots	Assembly, P and P
6-axis Robots	All of the above, de-moulding

Baxter is a new generation of dual arm robot in which humans and robots work co-operatively within the same workspace. Unlike the robots in table 1.1, Baxter can be easily integrated into existing factory floors, and is far easier to set up. This proves to be a great advantage for companies looking to modernise their manufacturing process, requiring fewer sensors and using cheaper EOAT. The main disadvantage of using Baxter is the lack of speed and accuracy in the motors of its arms, hence Baxter tends to be used in processes which do not require high degree of accuracy, tasks such as packaging and P

and P are suited for Baxter.

1.2 Problem Statement

Many of the robots used in industry rely on a rigid environment in order to accomplish their specific tasks, if, for example in a moulding and de-moulding robot, if the placement of the moulds to be placed or removed deviate slightly from the assumed programmed position, the whole system halts. Systems designed to compensate for error in object placements are expensive and require multiple sensors, in addition to high initial capital needed to design and build EOAT along side a facility to place the robot in. Therefore it would prove to be beneficial if it possible to reduce the cost of completing P and P tasks with minimal sensors with robots such as Baxter who have comparatively poorer motors. Since Baxter as eye in hand cameras, visual feed back control (visual servoing) method of control is chosen to control the movement of Baxter's arm.

1.3 Scope/Focus Area

Visual servoing is a well established method of controlling robotic arms that utilises image features to try and move towards an object. This thesis focuses on use of constrained based models of objects to try an estimate, using sequential quadratic programming (SQP) methods to estimate the objects position in 3D space (specifically its depth Z). The end goal of the thesis is for Baxter to observe a geometric object from various initial poses to then move its arm towards the centroid of the shape and pick it up.

Chapter 2

Literature Review

2.1 Visual Servoing

The majority of robotic application are operating in factories where the operating environment is modified to complement them. Due to the primary application in manufacturing and research, robots have surprisingly little impact in dynamic work environments. Such environments may include a packaging robot in grocery stores, or object placement tasks, where the objects to be manipulated are not well defined in 3d space. The primary limitation is due to a lack of sensory capability that is economical viable to implement. In order to place robots in the real world it is economically impractical to re-design the world around us to compensate for the use of robots. The ultimate goal for a roboticists is create sensor feedback system that is able to dynamically cope with a changing environment. One of the best systems that is able to cope with such a challenge is a human, using a visual feedback and known dynamics of the arm to grasp and manipulate objects. Thus vision is a powerful robotic system, as it allows the robot to make non-contact measurements, and in turn, with proper modelling and control, manipulate and place objects in the real world.

According to F.Chaumette and S.Hutchinson [1], visual servo control refers to the use

of computer vision data to control the motion of a robot. A more concrete definition provided by P.I.Corke is [2], Visual servoing is to use visual information to control the pose of a robots end effector relative to a target object or a set target feature. The use of visual data in a control feedback system was first tested by Y.Shirai and H.Inoue [3], prior to their work, majority of systems were based on an open loop system. An open loop system meant that the visual system and the robots movements are completely separate, in effect the system would first observe, then move. This meant that such systems accuracy heavily depended on the accuracy of the visual sensor and the robots motors. A method to increase the accuracy of both subsystems is to use vision data in a feedback control loop to control the position of an end effector, this is referred to as visual servoing. The term visual servoing was first coined by J.Hill and W.T.Park [4] to make the distinction between systems that use an open loop control.

2.2 Visual Servoing Architecture

Prior to understanding the different architectures within visual servoing, it is important to understand that setup configuration of the overall system is vital. This configuration can be broken up into two main setups, the first being the eye in hand where the camera is mounted onto the end effector of the robot. The second variant is hand to eye, where the cameras is externally mounted. The difference between the two setups is what the required observations are, the eye in hand is used to measure the relative distance between the end effector and target. The hand to eye setup is used to observe both the target and end effector.

Sanderson and Weiss [5] introduced a taxonomy of visual servoing systems, that is then used to categorise the system into 4 distinct control schemes, they are (in no particular order):

1. Dynamic Look and Move

2. Direct Visual Servo

3. Image Based Visual Servoing (IBVS)

4. Position Based Visual Servoing (PBVS)

5. Hybrid Visual Servoing

An example of Dynamic Look and Move can be seen in figure 2.1. This system uses vision to determine set points of the joint controller, using the joint feedback of the robots own controller to internally stabilise the system.

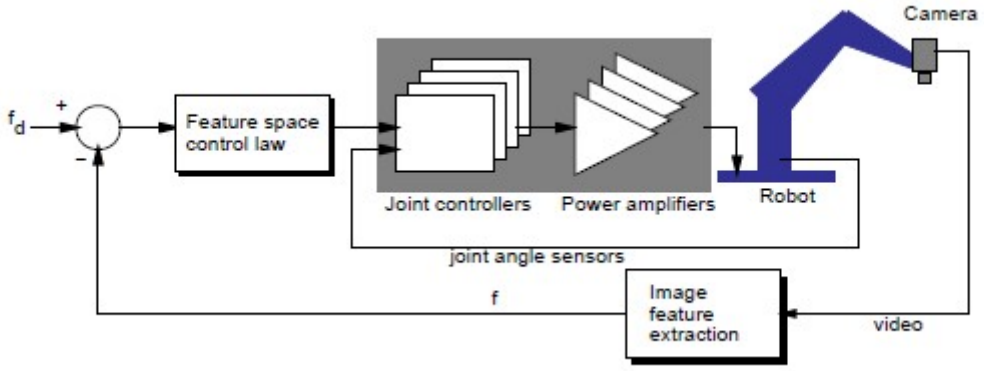


Figure 2.1: Control loop for a Dynamic Look and Move system

Direct visual servoing on the other hand, removes the robots controller and the placing the joint inputs directly with a visual servo controller. Though these controllers are used due to their simplicity, allowing robots to be considered as ideal Cartesian systems, research efforts have been put into improving IBVS, PBVS and Hybrid systems.

2.2.1 Image Based Visual Servoing (IBVS)

The aim of IBVS is to match the pose of object that is observed by the camera to a desired pose, this is done using specific features located on the image plane directly for

feedback [6][7]. Error is computed directly on the values that have been extracted from the features on a 2D image plane [8], this is why IBVS is also called 2D visual servoing.

F.Chaumette and S.Hutchinson [1] provide a tutorial to basic approaches behind visual servoing mathematics, in the paper, the aim of all vision based control schemes is to minimise the error which is defined as:

$$\mathbf{e}(t) = s(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^* \tag{2.1}$$

Where the vector $\mathbf{m}(t)$ is a set of image measurements (these may include co-ordinates of points of interest on an image, or image co-ordinates of the centroid of an object), and $\mathbf{a}$ is a set of additional parameters that is already known about the system. The vector $\mathbf{s}^*$ contains desired values of an image feature, since the goal of Baxter is to use move and manipulate fixed objects to a fixed goal, $\mathbf{s}^*$ is constant. Visual servoing schemes change depending on how $\mathbf{s}$ is designed, in the case of IBVS $\mathbf{s}$ is a set of features that is directly detected from an image.

A study by Corke [9] studied some the issues faced in visual servoing systems, and where researchers need to invest further time into. The paper suggests that it is known that robots cannot reach target positons within one set point, rather requiring interval or planned motion over many intervals between start and end position. In the paper Corke argues that for better performance of closed loop systems, computing and controlling the velocity results in easier control, hence considers visual servoing as a *steering* problem.

This means that equation 2.1 cannot be simply be based on image co-ordinates, hence a relationship between the time variation of $\mathbf{s}$ and the camera velocity $\mathbf{V_c} = (\mathbf{V_c}, \omega_\mathbf{c})$, where $\mathbf{V_c}$ is the linear velocity of the origin of the camera, and $\omega_\mathbf{c}$ is the instantaneous angular velocity of the camera frame. Therefore the relationship between the time variation of $\mathbf{s}$ and $\mathbf{V_c}$:

$$\dot{\mathbf{s}} = \mathbf{L_s}\mathbf{V_c} \tag{2.2}$$

Using equation 2.1 and 2.2, we can obtain a relationship between camera velocity and

time variation of the error:

$$\dot{\mathbf{e}} = \mathbf{L_e V_c} \tag{2.3}$$

Where $\mathbf{L_s} = \mathbf{L_e}$ is known as the image Jacobian matrix, which was first introduced by Weiss et al [10] (in literature also called, interaction matrix or feature Jacobian).

In order to formulate the image Jacobian, it is important to understand that any 3D point in space can be in the form. $\tilde{\mathbf{X}} = (X, Y, Z)$, and can be projected into 2D image space $\tilde{\mathbf{x}} = (x, y)$:

$$x = \frac{X}{Z} = \frac{(u - c_u)}{f\alpha} \tag{2.4}$$

$$y = \frac{Y}{Z} = \frac{(u - c_u)}{f} \tag{2.5}$$

Taking the time derivative of equation 2.4 and 2.5 we can determine the interaction matrix (proof is explained in greater detail in [1] and [2]):

$$\mathbf{L_x} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix} \tag{2.6}$$

When an image is taken, the 3D space is converted to 2D space, losing a dimension information, specifically the depth Z. Therefore solving the image Jacobian is challenging as the equation is has an unknown parameter Z, in order to overcome this an estimation must be made of Z.This issue was studied by N.P.Papanikolopouos and P.K.Khosla [11], where an adaptive approach was used to estimate the depth value, though this method provided results that were suboptimal when compared to their simulation. This was due image blurring when tracking objects, which created errors in the visual measurements. Another issue of updating the image Jacobian at each iteration of the control law is that, though the system may converge to a solution, it maybe that he solution is a local minima or reaching singularities [12]. Other approaches to mitigating the issue of unknown Z was researched by G.D.Hager and his team [13], where feedback formulations were developed which did not use the depth of the image. Another challenge in using

the image Jacobian is far more fundamental, as it is evident from equation 2.6 the image Jacobian is a non-invertible matrix, requiring the use of a pseudo-inverse of the matrix. Using the pseudo-inverse has its own inherent problems of reaching solutions which may lead to singularities in the image space.

The dis-advantage of using IBVS is the challenge in designing the controller, as the plant is non-linear and highly coupled, possibly requiring numerical methods such as Runge-Kutta to solve for the controller. Furthermore, IBVS is prone to perform poorly when required to make large translational or rotational motion [17]. When large rotations, specifically around the z axis, rather than rotating around the cameras z axis, the camera suffers a server translation along the optical (viewing direction) axis due to the control scheme.

The advantage of IBVS is that the control system is not susceptible to hand-eye mis-calibration (which is important for Baxter) as long as the feedback system is asymptotically stable. Figure 2.2 shows a visual representation of IBVS control loop.

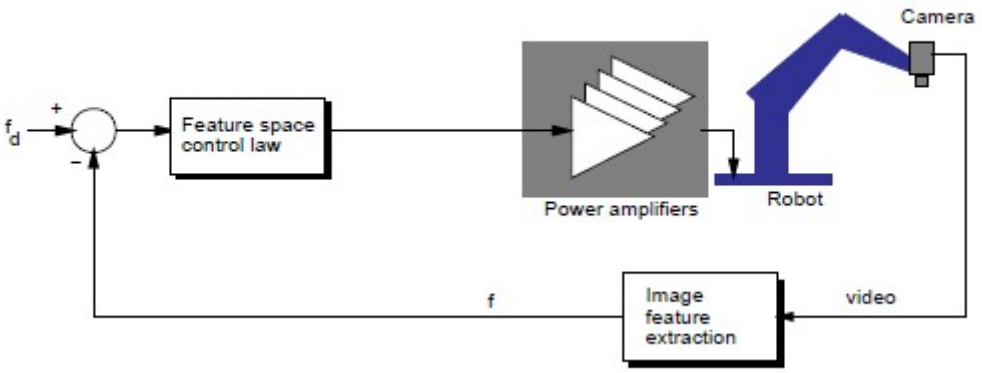


Figure 2.2: Control loop for a Image Based Visual Servoing System

2.2.2 Position Based Visual Servoing (PBVS)

According to Corke and Hutchinson [14], PBVS systems extract features from an image, and in turn, use the features to compute a (partial) 3D reconstruction of the Cartesian environment (usually with respect to some global world frame). The error in the control loop is computed in the Cartesian task space. Therefore PBVS is a model based control that requires prior knowledge/ model of both the target and the camera that is being used (camera models fall into two overarching group, the most commonly used is pinhole model, and another less commonly used is the thin lens model).

The most common estimation method that PBVS is implemented towards is pose estimation of a geometric object with respect to a desired model. Many solutions have been provided to such problems, and they fall under two categories, either analytic or least square. One such instance is work done by Y.Wang, J.Thunberg and X.Hu[20], the group focused on turning the PBVS problem into a convex optimisation problem, where the camera motion is constrained so that the object of concern is always in the field of view (FOV) of the camera, such that the error between the camera's position and desired position is minimised. The results of optimisation method for camera motion showed that the camera had to make two specific movements each time an estimation is made for path, the camera would first estimate and conduct rotational motion about the Z-axis then translate towards the object. The convex optimisation method used perspective changes and motion of the camera to determine the depth of the object.

The advantages of using PBVS is that control tasks and pose can be described using a Cartesian frame, the major disadvantage of PBVS is its sensitivity to calibration error, as this can lead to large error when reconstructing 3D images. Figure 2.3 depicts a visual representation of PBVS control loop. The dis-advantage of PBVS is that it is highly sensitive to camera calibration parameters [2], further more, though work on using convex optimisation has been conducted to estimate depth in IBVS [21](though implementation is difficult), it has not been tested thoroughly. on a PBVS control scheme.

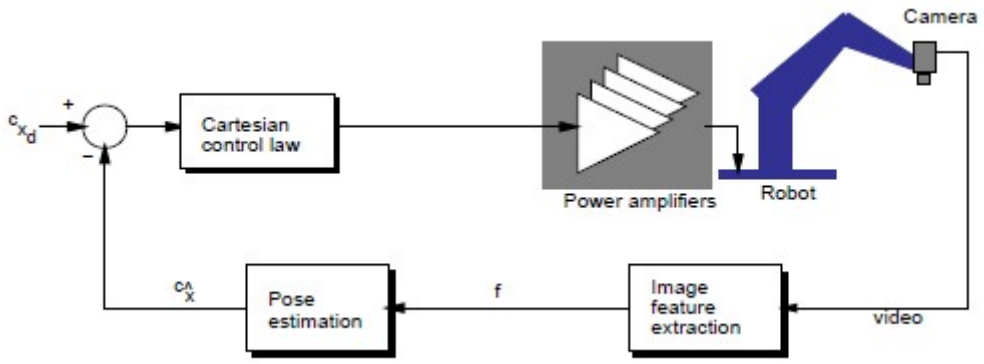


Figure 2.3: Control loop for a Position Based Visual Servoing System

2.2.3 Hybrid Visual Servoing

By fusing the advantages of IBVS and PBVS would prove to provide a more robust control scheme, Chaumette and Malis [15] propose such a new hybrid servoing method termed 2.5D visual servoing. The interesting aspect of 2.5D visual servoing is that it requires no

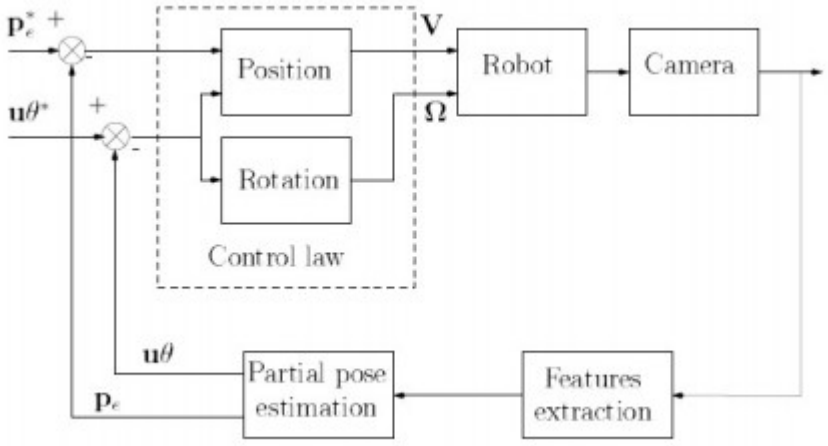


Figure 2.4: Control loop for a 2.5D control scheme

prior knowledge of the 3D structure that is being considered. Another major advantage of this from of hybrid servoing is that fact the image Jacobian used, is an upper block triangular matrix [16], this means that the Jacobian is only singular in degenerate cases

such as when $Z = 0, \frac{1}{Z} = 0$ and $\theta = \pm 2\pi$ ensuring a highly robust system.

The reason that this was achievable in a 2.5D servoing control method is because the image Jacobian has been decoupled into a translational and rotational component, this is evident when observing the control loop as shown in figure 2.4 [16].With 2.5D servoing it is stated in the paper that intrinsic parameters of the camera and calibration parameters are needed, although when experimented with calibration error, the control system still performed quite well, further outlining its robustness. Although the system is quote sensitive image noise (an example of this maybe blurring of the image during movement) than compared to IBVS. Experimental results show that 2.5D servoing offers a larger region of asymptotic stability than IBVS or PBVS [17], and also overcomes the major stability issue of the camera retreat faced by IBVS.

Other hybrid approaches have also been proposed to try and overcome some of the downfalls of IBVS and PBVS. Since IBVS proves to be a more popular approach (due to its in-sensitivity to calibration error) a partitioned approach has been proposed by Corke and Hutchinson [18]. Since IBVS suffers from the camera retreat problem, it would be better suited to decouple and partition the z component of the rotation and translation. Therefore the time derivative of $\mathbf{s}$ as mentioned in equation 2.2, changes to:

$$\dot{\mathbf{s}} = \mathbf{L_s V_c} = \mathbf{L_x y} V_x y + \mathbf{L_z} V_z \tag{2.7}$$

2.3 Summary

The early work of Y.Shirai and H.Inoue put in place the stepping stones to push visual servoing into what is today an advanced field of close loop control. The two commonly used approaches of IBVS and PBVS both have their respective advantages and disadvantages. PBVS (specifically using pose based estimation) estimates the position of the target object based on known target and camera models, also with known calibration parameters. The aim of PBVS to minimise the 3D model error based on a Cartesian world frame of

reference. IBVS on the other hand using image space to minimise the error between the observed pose and desired pose, the problem with IBVS is that it suffers from complex controllers design and high coupling. Though for most object manipulation or P and P problems IBVS is commonly used as the Z depth is solved over multiple iterations of its movements, PBVS can also be used if a robust enough method to estimate depth can be formulated. Since PBVS is an easier control scheme to implement, it would seem fruitful to pursuit to try and improve depth estimations methods, using convex optimisation methods.

Chapter 3

Computer Vision and Processing

Baxter has 3 cameras in total, 2 eye in hand cameras for each limb, and an additional camera mounted on it's head. The main camera that will be used is the left eye in hand camera of Baxter, the camera outputs images in Ipl format. Ipl format is an old format used by previous versions of Open-CV which is converted to an array format to be used my the current version of Open-CV. This chapter will discuss the camera calibration process of the left hand camera and image filtering methods applied to improve feature extraction.

3.1 Camera Calibration

Distortion is an inherent feature of all camera lenses, there are no camera lenses that can perfectly capture an undistorted image. In order to compensate for the lens distortion, the lens and CCD sensor are calibrated together in a method known as camera calibration. In high end cameras and lens, the intrinsic parameters of the camera are given in order to un-distort images.

When conducting camera calibration, a model of the camera must be assumed, the most common model assumed is what is known as the pinhole model, this is shown

in figure 3.1. From the figure it is evident that the pin hole camera model requires

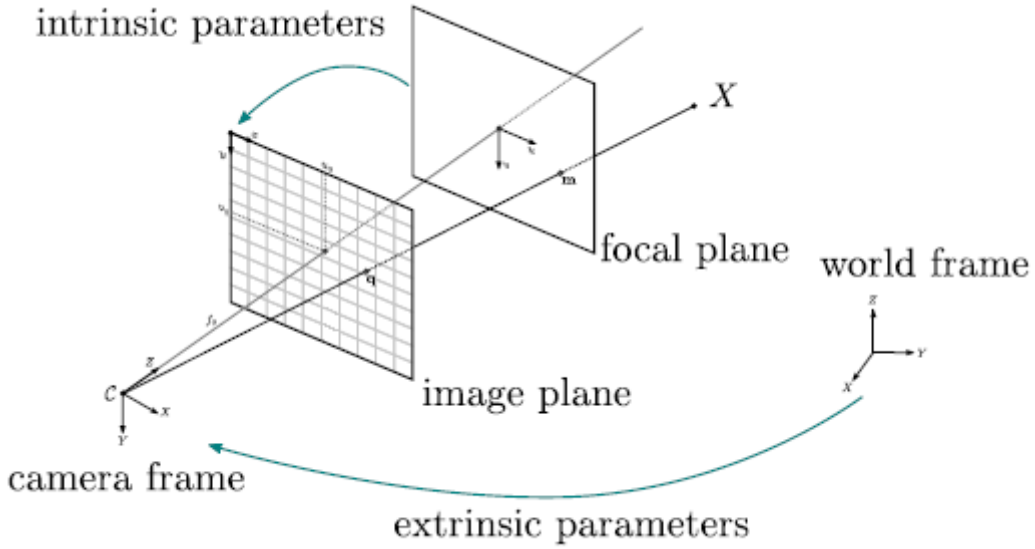


Figure 3.1: Pin Hole Camera model

two parameters to define a co-ordinate transform from the image plane to the camera frame, then from the camera frame to a world frame. These parameters are known as intrinsic and extrinsic parameters respectively, using these two parameters the camera is mathematically modelled as:

$$\begin{bmatrix} u \\ v \\ z_{world} \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix} \begin{bmatrix} x_{world} \\ y_{world} \\ 1 \end{bmatrix} \tag{3.1}$$

where $\begin{bmatrix} u & v & 1 \end{bmatrix}^T$ is position of a point in 2D space on the image plane and $\begin{bmatrix} x_{world} & y_{world} & z_{world} \end{bmatrix}^T$ is the point as seen on the image plane in 3D space, with respect to (w.r.t) some global frame of reference. $\mathbf{R}$ and $\mathbf{T}$ are the extrinsic that is the homogeneous transformation matrix that identifies the position of the camera frame w.r.t to some global frame. $\mathbf{A}$ is a square matrix that contains the intrinsic parameters of the camera, the general form of

the intrinsic parameter is given as:

$$\mathbf{A} = \begin{bmatrix} f_x & \tau & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

From intrinsic parameter matrix, their are 3 key sets of parameters that define the lens/camera, they are defined as:

- f_x & f_y represents the focal length in the x and y axis of the image plane in pixels. In order determine the conversion between pixel length and real world length, f_x & f_y can be written as:

$$f_x = F.mm_x \tag{3.3}$$

$$f_y = F.mm_y \tag{3.4}$$

  where F is the focal length of the camera in mm and mm_x & mm_y is the conversion factor between pixel to mm.

- τ is the skew parameter which defines the angle between the x and y axis of the image plane, also known as the aspect ratio. For most cameras, the aspect ratio is 1:1, therefore the skew parameter is also zero. Pixels can also be skewed if the image is gathered by a frame grabber, the pixel grid may possibly be skewed due to inaccuracies in the synchronisation of the pixel-sampling process. Since the images from acquired from Baxter are directly grabbed from the camera, this issue is of no concern.

- o_x & o_y are the optical center co-ordinates of the image frame w.r.t to the image frame co-ordinates (most imaging systems define the origin of the image frame to be the top left pixel of the image).

It is important to note that when conducting camera calibration, we either approach it has a linear or a non-linear problem, though the intrinsic parameters are non-linear in nature, in order to simply the mathematics and calibration process a linear approach has been used. When correcting for distortions during calibrations, their are two main distortions of concern, they are radial and tangential distortion.

Radial distortion occurs when light bends near the edges of a lens w.r.t to the center of the image. Formally, radial distortion changes the distance between the image center and a point on an undistorted ideal image, whilst maintaining the direction of the vector joining the two points. If the change in the vector distance is negative, it is indicative of barrel distortion, on the other hand, if the change in vector distance is positive it is indicative of a pincushion distortion. If the the distortion is a mixture of the two across the image, it forms a third type of distortion known as moustache distortion.

Radial distortion can be accounted and corrected by using the following equation:

$$x_{\text{corrected}} = x_{\text{distorted}}(1 + K_1 r^2 + K_2 r^4 + ...) \tag{3.5}$$

$$y_{\text{corrected}} = y_{\text{distorted}}(1 + K_1 r^2 + K_2 r^4 + ...) \tag{3.6}$$

Radial distortion can be modelled to any number of K radial distortion parameters, since the MATLAB Camera Calibration ToolBox is being used, it outputs a total of three K parameters, hence only three will be used.

Tangential distortion is a results when the camera CCD and the lens are not parallel, rather offset by some small angle. Similar to radial distortion, we can express the correction for tangential distortion as:

$$x_{\text{corrected}} = x_{\text{distorted}} + [2P_1 x_{\text{distorted}} y_{\text{distorted}} + P_2(r^2 + 2x^2_{\text{distorted}})] \tag{3.7}$$

$$y_{\text{corrected}} = y_{\text{distorted}} + [P_1(r^2 + 2y^2_{\text{distorted}}) + 2P_2 x_{\text{distorted}} y_{\text{distorted}}] \tag{3.8}$$

Where

$$r = \sqrt[2]{(x_{\text{distorted}} - o_x)^2 + (y_{\text{distorted}} - o_y)^2} \tag{3.9}$$

Typical camera calibration method involves using a chess board with squares of 30mm. Multiple images at different ranges and orientations are taken to improve intrinsic parameter estimation. When calibrating Baxter's left hand camera, 148 images were used by the MATLAB Camera Calibration ToolBox. The calibration process involves first, by hand, selecting the corners of the chess board from a calibration image as shown in figure 3.2.

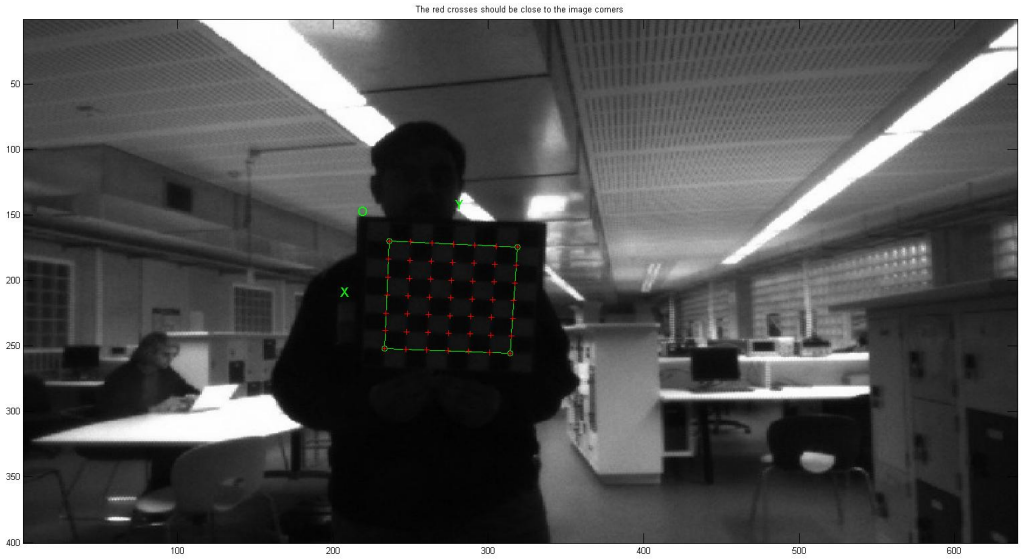


Figure 3.2: Corner extraction of the chess board

Which the toolbox automatically detects the location of each of the grids in the chessboard as seen in figure 3.3

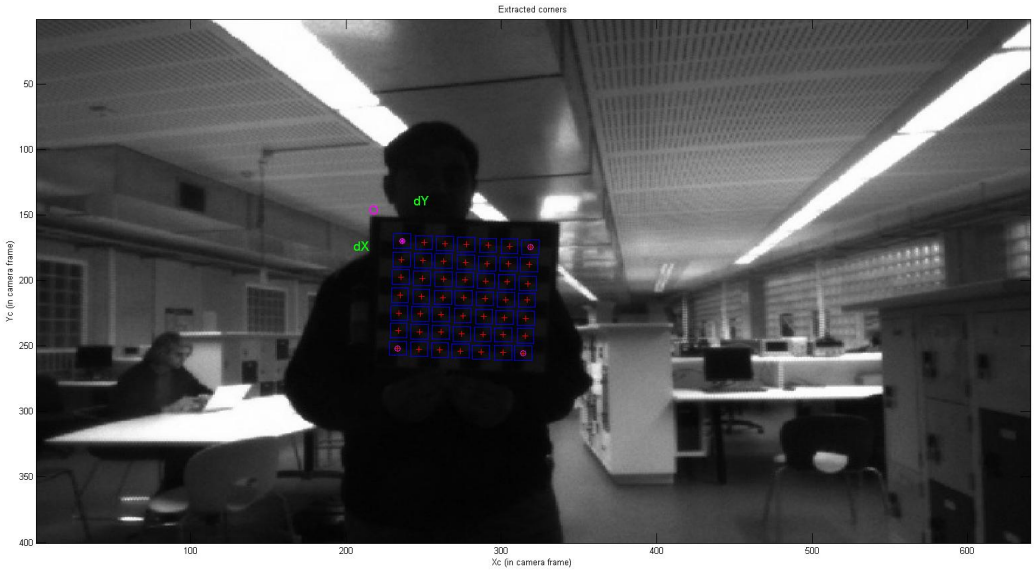


Figure 3.3: Grid extraction of the chess board

This process is repeated for all 148 images, which then the toolbox produces an initial estimate of the intrinsic parameters of the camera, due to the fact that a human has to select the corners of each image, the initial estimate is not the most accurate representation of the intrinsic parameters. The initial estimate of the intrinsic parameters are passed through an automated corner extraction method, from which a more accurate values of intrinsic parameters are determined, which for Baxter's left hand camera are:

$$\mathbf{A} = \begin{bmatrix} 409.20851 & 0 & 309.69054 \\ 0 & 408.54186 & 226.60469 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

$$\text{Radial Distortion} = \begin{bmatrix} 0.02784 & -0.05070 & 0.0000 \end{bmatrix} \tag{3.11}$$

$$\text{Tangential Distortions} = \begin{bmatrix} -0.00132 & 0.00435 \end{bmatrix} \tag{3.12}$$

It is evident from the intrinsic parameters that the camera does not suffer heavily from tangential distortion compared to the radial distortion, hence tangential distortion can be safely ignored when correcting the image. Studying the radial distortion coefficients, since the first value is positive it is assumed that Baxter's left hand camera suffers from slight pincushion distortion. The calibration toolbox also provides us with information regarding the re-projection error of the pixels from estimated positions using the intrinsic parameters and selected pixel positions. The toolbox also provides extrinsic plot of the estimated positions of the chess board w.r.t the camera. This allows us to verify if the calibration process yielded a fairly accurate estimation of the camera parameters. From figure 3.4 and 3.5 it was evident that estimated positions of the chessboard matches where they were held during calibration.

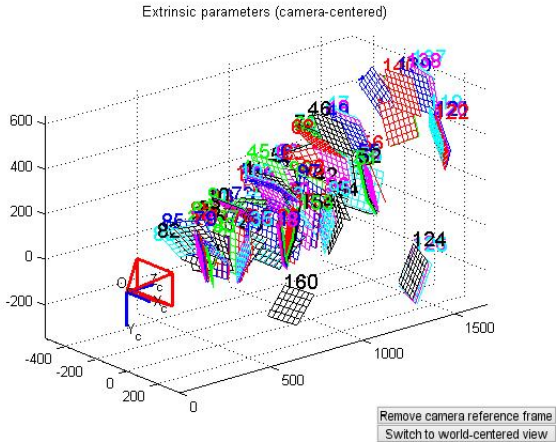


Figure 3.4: Position of Chessboard w.r.t to camera frame

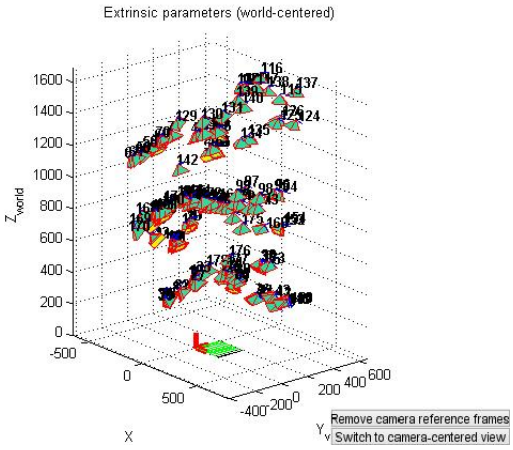


Figure 3.5: Position of Chessboard w.r.t to global world frame

The provided re-projection in figure 3.6 error shows a fairly compact region of pixel error, with an average re-projection error of $\begin{bmatrix}0.28107 & 0.30173\end{bmatrix}$ in the x and y axis respectively.

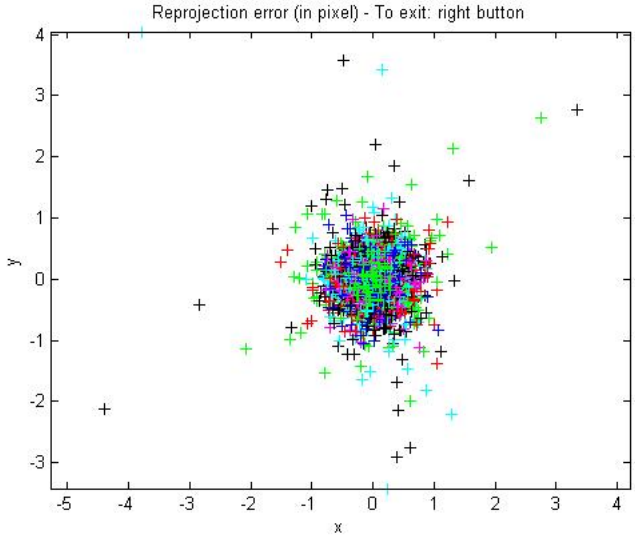


Figure 3.6: Re-projection error of pixels

3.2 Feature Extraction for Object Recognition

Visual servoing relies on observing and extracting desired features within in an image, for this to be achieved the object to be tracked must be visually distinct from the environment. To make the feature extraction simpler, the objects to be tracked are simple geometric shapes such as triangles, squares, pentagons etc. Although for the case of this thesis the square will be the main focus, although the application and methodology is the same for all other geometric shapes. The colour green was chosen for the object as the background that the object is going to be in is white (any bright colour can be used).

Image processing methods such as convolution (which will be discussed later down the report) or colour range detection can be implemented from scratch, but tend to be slow in processing and time consuming to create code, hence OpenCV, a python image processing API is used for ease of programming and fast compute times.

The process begins by first capturing the frame from the Baxter's left hand camera,the captured image is in Ipl image format which is used by an older version of OpenCV. Since a later version of OpenCV (under the api name of cv2) is used, the image is converted

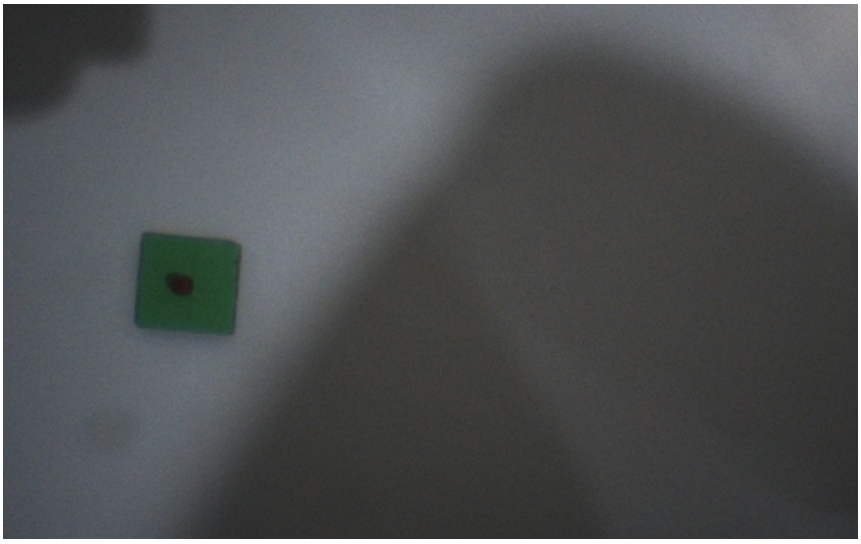


Figure 3.7: Raw Image captured from Baxter's left hand camera

from Ipl to an array format. The frame that is grabbed is in BGR colour space which is converted to HSV (Hue, Saturation and Value or Brightness). The HSV colour space is an ideal space to use to detect colours as only one channel (Hue) contributes to colour, while the other two channels effect the "intensity" of the colour.

The next step is to over saturate any colours in the image, and under-saturate and blacken any part of the image that is not a color. This ensures that when we want to extract a specific colour within the image it can be easily found. The over saturation process first begins by gamma correcting the image, gamma correction is achieved by applying equation 3.12.

$$V_{out} = V_{in}^{\lambda} \tag{3.13}$$

A λ value usually ranges between 0 and 1, where a lower lambda results in an image that increasingly brightened, or white-washed. Upon fine tuning the λ value, 0.6 was chosen which was a good compromise between brightening the image whilst maintaining the features in the image. The compromise is validated when studying pixel transition from the white/grey background to the green square. The signal as shown in figure 3.9

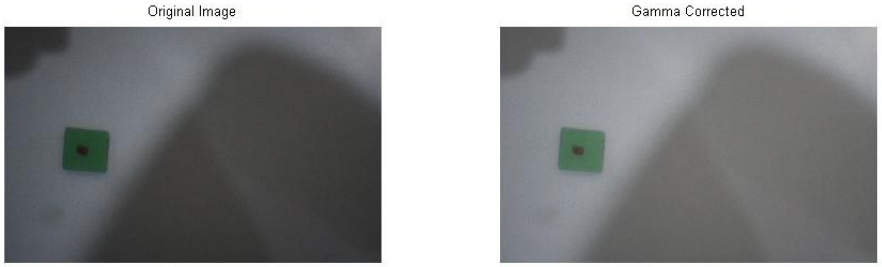


Figure 3.8: Gamma corrected, grey scaled and square extracted image

proves that the pixel intensity signal is only shifted up in intensity whilst maintaining the shape of the signal. The purpose of gamma correction is not just to brighten the image, but to also reduce the variance in the saturation of the image. The background is known to be white or grey, this, according to the HSV colour wheel, only occurs for low saturation values, irrespective of Hue. This is important as the gamma correction retains colour information of the image as shown in figure 3.10. Therefore, it is necessary to determine some method to distinguish whether the saturation value seen is due to a coloured object or due to unwanted background noise. This distinction is made by applying a dynamic threshold for the saturation channel, so that all unwanted background can be suppressed. From the HSV colour wheel it is evident that for any colour to be visible for a given hue, the saturation must exceed the minimum value for which white and grey colours occur. This means that if a colour is visible in an image, the saturation will inherently be larger than the minimum saturation value that defines a white or grey colour. Since the geometric objects of concern are coloured green, in a white/grey background, the

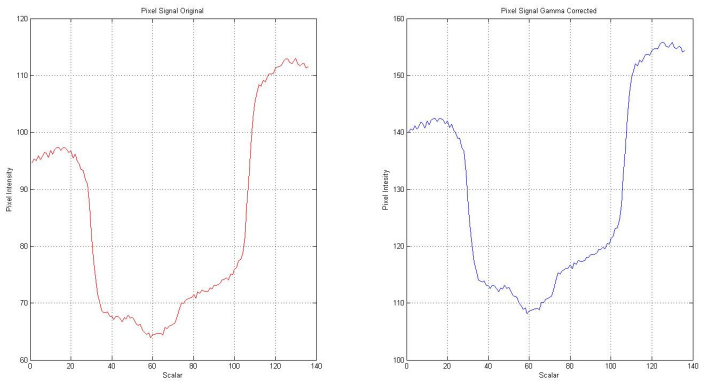


Figure 3.9: Pixel signal

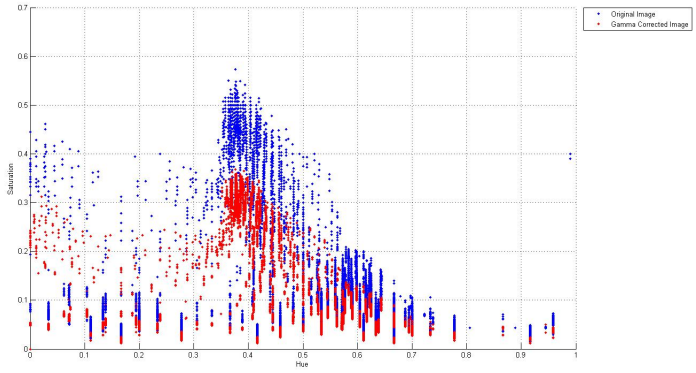


Figure 3.10: Plotting points of an image using H,S and V as axis coordinates

saturation value for the desired colour will always lie between the max and minimum saturation values of the image. Therefore if we divide the difference between the max and minimum saturation values in the image by some tuning factor ϵ, any saturation value beyond this threshold is indicative of a colour and must be over-saturated.

$$\frac{Saturation_{max} - Saturation_{min}}{\epsilon} \tag{3.14}$$

If any of the saturation value falls below the threshold, the corresponding pixel's brightness and saturation is reduced so that the pixel colour becomes black. Upon tuning the ϵ parameter as seen in equation 3.13, the optimal value found was 2.25, any higher for the

epsilon value added random artefacts to the image, any lower the threshold became too large and colours were not accurately over saturated under certain lighting conditions. The result of the over-saturating image is shown in figure 3.11 It is evident from figure

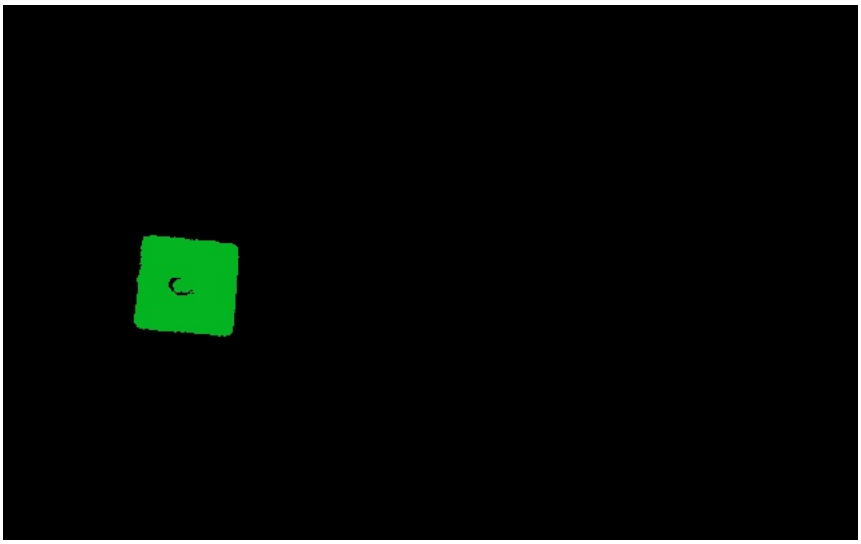


Figure 3.11: Over-saturation of colours in an image

3.11, during the process of over-saturation some dimensional accuracy is lost as the edges of the square have become jagged. In order to smooth the noise caused by the jagged edges, it is necessary to determine the type of noise that is being observed. This can be achieved by studying the image in the frequency domain by performing a 2D Discrete Fourier Transform. A 2D Discrete Fourier transform is defined as

$$X\left[u,v\right] = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} x\left[m,n\right] e^{-j2\pi(\frac{um}{M}+\frac{vn}{N})} \tag{3.15}$$

where $x\left[m,n\right]$ is the intensity of the pixel at index m,n. Note that M and N are the number of rows and columns in the image respectively.

Applying the Discrete Fourier Transform equation to figure 3.11, results in Figure 3.12. The figure highlights a distinct, dominant, low frequency signal (in the shape of a cross) surrounded by some low and high frequency noise. In order to determine what noise

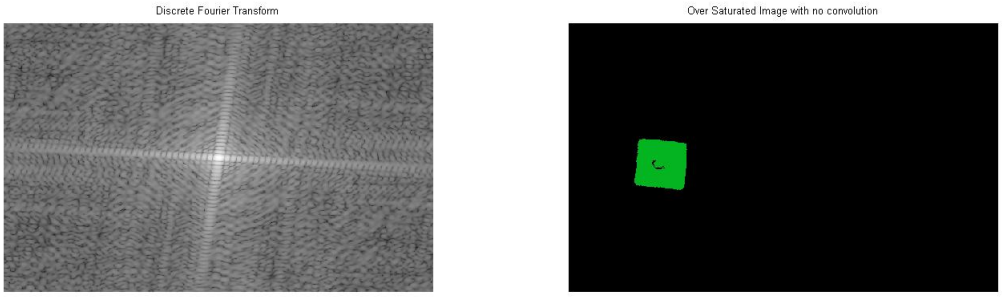


Figure 3.12: The Left is the FFT of the over-saturated image on the right

needs to be filtered it is necessary to have an ideal Fourier image to compare to. When analysing images in the frequency domain, geometric shapes have a distinct features that are intrinsic to the specific geometry. This proves to be a powerful tool as the objects of desire in this thesis are all geometric shapes. Since the geometry of concern is a square, the base line for an ideal Fourier transform can be constructed by making a white square on a black background,An example of this is shown in figure 3.13.

From examining figure 3.13 and 3.12, it is evident that the jagged edges can be classified as high frequency noise, as the corners of figure 3.13 have far less white intensity compared to figure 3.12. In order to filter the high frequency noise, the image will be filtered in the spatial domain using a 2D convolution, with a specific kernel. Mathematically, 2D convolution is defined as:

$$g\left[m,n\right] = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} x\left[k,l\right] h\left[m-k,n-l\right] \tag{3.16}$$

where $x\left[k,l\right]$ are the coordinates of a pixel in the image, and $h\left[m,n\right]$ is a 2D kernel

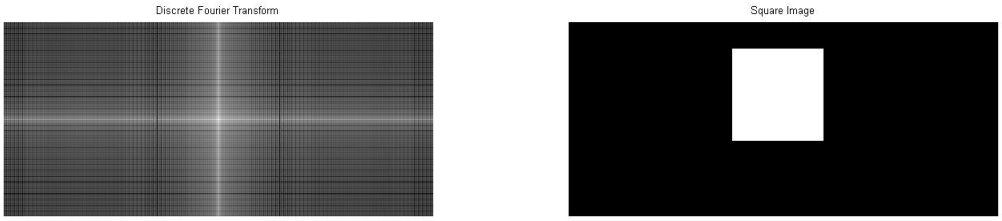


Figure 3.13: Discrete Fourier Transform of a square in a zero noise environment

that acts as the filter, different $h\left[m,n\right]$ will produce different filtering kernels. From the Fourier analysis of the image, a low pass filter kernel is needed, which from signal processing is nothing but a kernel that averages a region of pixel intensity values. Therefore A low pass filter kernel can be defined by the following equation:

$$\frac{1}{(2+b)^2}\begin{bmatrix}1\\b\\1\end{bmatrix}\begin{bmatrix}1 & b & 1\end{bmatrix} \tag{3.17}$$

where b is the parameter that determines the cut-off frequency. in order to determine the optimal b value for the kernel, figure 3.11 was passed through several different low pass filter kernel with several different b values. For each b value the corresponding Fourier transform was studied and compared to figure 3.13. The results of different low pass filter kernels can be found in A.1 to A.5 of the appendix, but from the testing process it was found that $b = 2$ provided the best result that matched closely to figure 3.13, as shown in the figure 3.14 The entire process of over-saturating the image for easier colour extraction

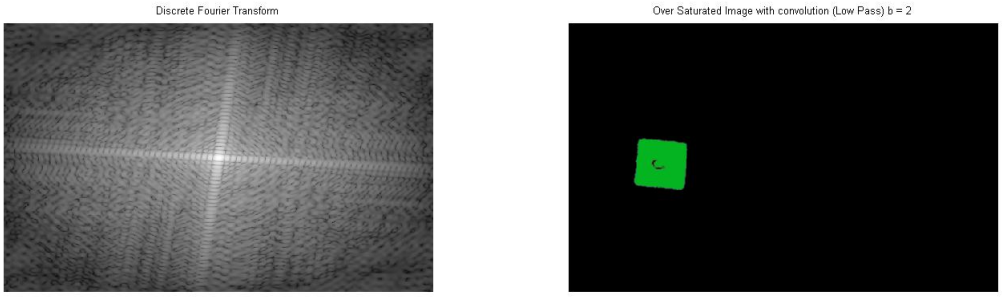


Figure 3.14: 2D convolving the over-saturated image with a low pass filter kernel

can now be formalised into a simple algorithm in A.2.

procedure OVERSATURATEIMAGE(baxterImage)
    $H, S, V = splitChannel(baxterImage)$
    $H = (\frac{H}{255.0})^{\lambda}$ ▷ λ is the value as defined in equation 3.13
    $S = (\frac{S}{255.0})^{\lambda}$ ▷ λ is the value as defined in equation 3.13
    $V = (\frac{V}{255.0})^{\lambda}$ ▷ λ is the value as defined in equation 3.13
    $Sat_{max} = max(S)$
    $Sat_{min} = min(S)$
    $threshold = \frac{Sat_{max} - Sat_{min}}{2.25}$
    if $anypixelvalueinS < threshold$ **then**
        $S[pixelIndex] = 0$
        $V[pixelIndex] = 0$
    end if

13: **if** $anypixelvalueinS >= threshold$ **then**
14: $S[pixelIndex] = 250.0$
15: $V[pixelIndex] = 180.0$
16: **end if**
17: $H = 2DConvolve(H, LowPasskernel)$ ▷ Kernel to be used is that of equation 3.17
18: $S = 2DConvolve(S, LowPasskernel)$
19: $V = 2DConvolve(V, LowPasskernel)$ ▷ OpenCV has cv2.filter2D, an inbuilt function that performs convolution given a kernel
20: $overSaturatedImage = merge(H, S, V)$
21: **end procedure**

Algorithm 1: Over-saturate Colour algorithm

Now that a robust method of over-saturation has been implemented, the next major step is to determine the geometry that is observed in the image. OpenCV has a method known as `findContours`, this method works by trying to find the least number of vertical and horizontal lines needed to approximate a closed boundary. This method is a problem if the geometry is tilted, this is because `findContours` will try yo estimate a sloped line with large number data points which make horizontal and vertical lines. To overcome this concern, the large number of data points are used to estimate a line of best fit for each edge, this is accomplished by using the Douglas Peucker algorithm (Which OpenCV has an inbuilt function for called `approxPolyDP`). In applying the Douglas Peucker algorithm, the large data set is reduced to the least number of data points needed to define and y edge boundary, these data points being the vertices of the shape.

It is important to note that `findContours` works on binary images, hence the desired colour must be extracted from the image followed by a binary threshold. Since the image has been over-saturated looking for colours can be done very easily, since the colour green is of concern, a range of H,S and V values correspond to the colour green. Upon fine tuning

the values that following ranges were determined to be optimal in estimating green.

$$50 \leq \textbf{Hue} \leq 70$$
$$100 \leq \textbf{saturation} \leq 255$$
$$100 \leq \textbf{Brightness} \leq 255$$

In order to remove any noise that maybe still present in the image,two processes known as morphological opening and morphological closing are implemented. Morphological opening operates by first eroding then dilating the image, removing small sudden fluctuations in the image. Morphological closing is applies the reverse of the opening, the image dilated then eroded, filling up any holes that may be in the image. The result of extracting the shape from figure 3.11 is shown below in figure 3.15.

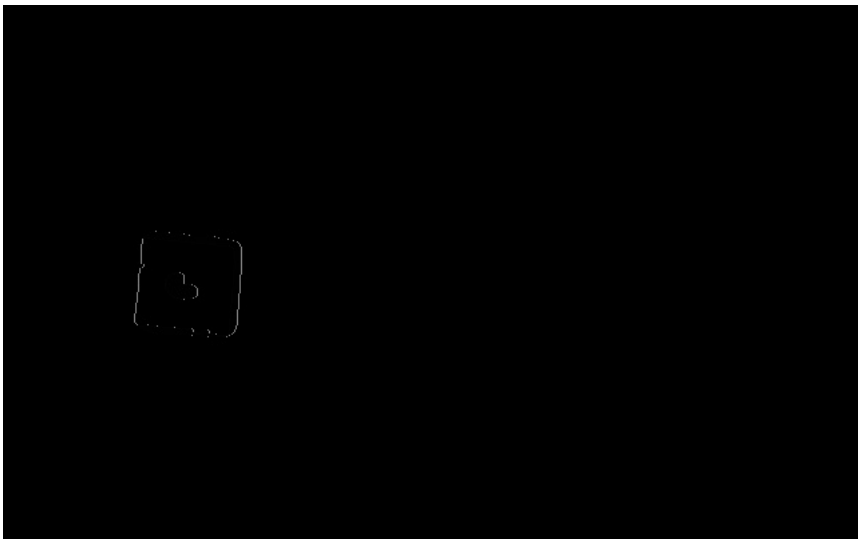


Figure 3.15: Extracting square from Image

From the figure the corners and edges are not well defined, this can be rectified by 2D convolving the image with a laplacian kernel. By 2D convolving the image with this kernel the edges of the image are enhanced, a typical kernel used for this process (and

also what was used to sharpen images in this thesis) is:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

The kernel sharpens edges because the frequency response of kernel in figure 3.16 shows that high frequency signals are amplified, in the case of an image, the transition between the edge to the background is a high frequency signal.

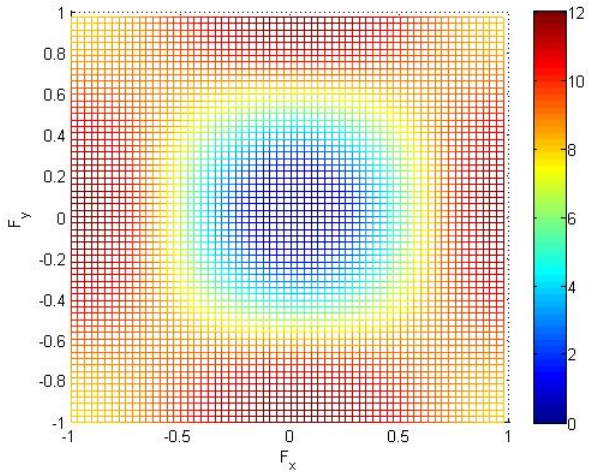


Figure 3.16: Frequency response of the laplacian kernel

Therefore the edge transition is amplified making the edge of the image sharper, as shown in figure 3.17

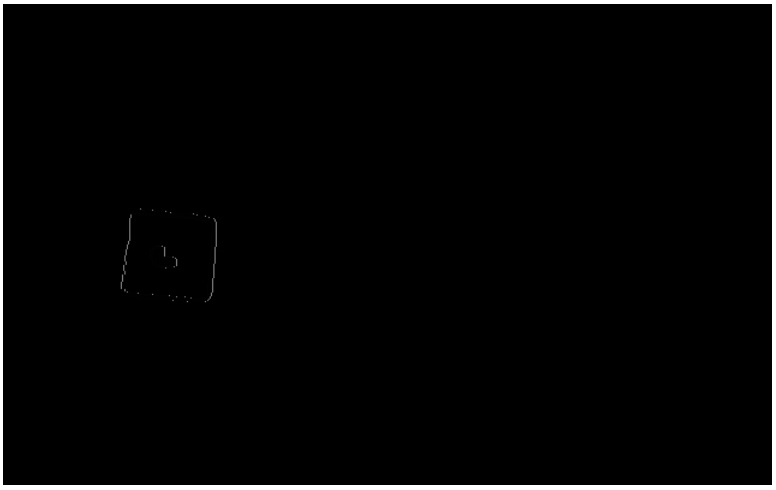


Figure 3.17: Edge sharpened image

With the edge sharpened image, it is passed through the `findContours`, which may return multiple closed boundaries due to noise, in order to filter out the redundant boundaries, the area of the closed boundary is checked. If the closed boundary area is below a threshold it can be ignored as noise, if the area is larger than a set threshold, it is almost certain it is due to a geometry that is observed in the image.

Once the test condition is passed the Douglas Peucker algorithm is applied to the contour resulting in minimum number of data points (which are vertices) required to define a closed boundary, regardless of orientation. By counting the number of vertices we can estimate what geometry is being observed, the result of the object detection can seen in figure 3.18.

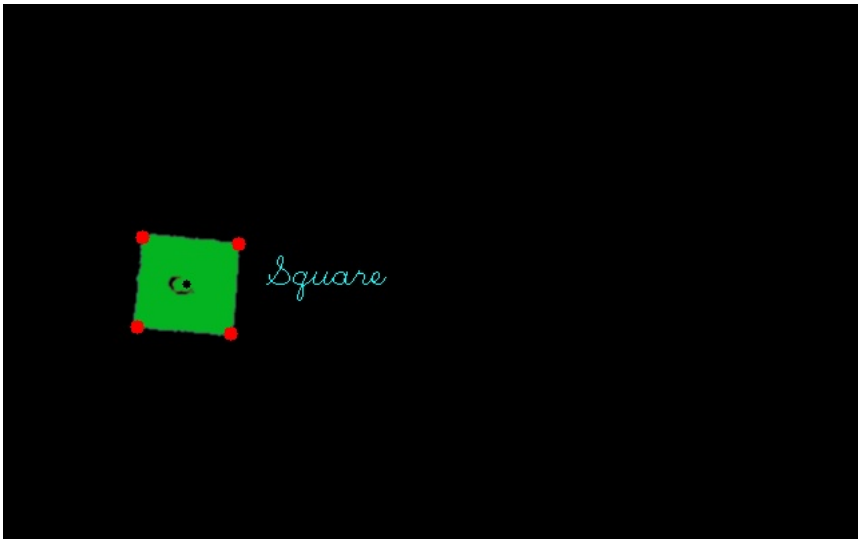


Figure 3.18: Final result of object recognition

The object recognition is not limited to only a square, as it can be shown in figure 3.19, that other geometric shapes can also be detected.

(a) Octogon (b) Triangle

Figure 3.19: Object Extraction of other geometric shapes

Chapter 4

Object Model and Depth Estimation

Chapter 5

Results

5.1 Experimental Procedure

In order to study the effectiveness and validity of using non-linear optimisation methods for pose and range estimation, a repeatable experimental procedure must be implemented. Prior to discussing the experimental procedure, a list of the apparatus needed to conduct the experiment.

- Baxter by Rethink Robotics
- A desktop with Ubuntu version 12.04 LTS or later
  - Kernel Linux 3.5.0-54-generic
  - GNOME 3.4.2
  - 2GB Ram
  - Intel Core 2 Duo 6700 running at 2.66GHz
- Robot Operating System (ROS), version: groovy
- Router for a common network for both Baxter and the desktop

- standard Ethernet connectors to connect Baxter and the desktop to the same network router
- Green coloured geometric objects (During experimentation a Square was used)
- White table
- 30cm Rule

To test the effectiveness of the system , 2 separate experiments were conducted to study the various aspects of the system. The 2 experimental testes:

1. Depth Estimation
2. Speed and Accuracy

The experimental set-up varied slightly from experiment to experiment, although the common string through out all experimentation is the use of the white table that is places in front of Baxter, and a the green square placed anywhere on the white table. It is important to note that the square, though it can be placed anywhere on the table, it must still be in the field of view of Baxter's camera.

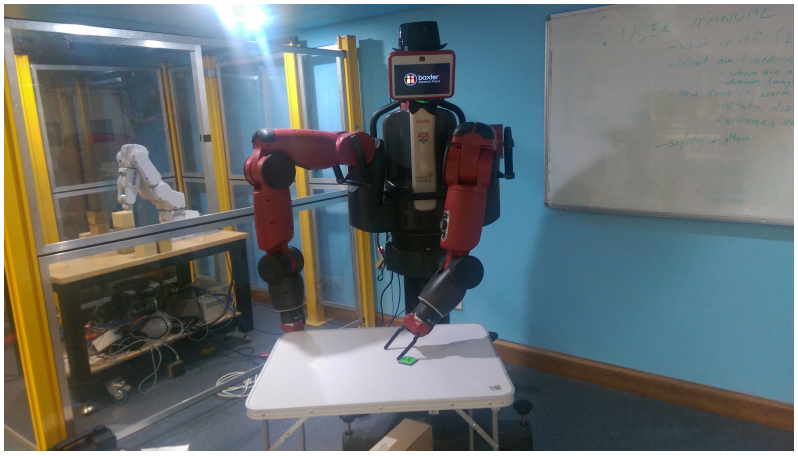


Figure 5.1: Common apparatus and initial set-up for all experiments

5.2 Depth Estimation

Due to the position of the gripper and camera on Baxter's arm, a ruler could not be placed in the center of the gripper as it would obstruct the view of the camera entirely. To compensate for this another method was employed, where a rule was attached to one of the gripper pincers so that rule protruded out 161mm from the tip of the gripper.

Figure 5.2: Placement of the ruler on the gripper pincer. The empty gap from 30cm to the edge of the ruler is an additional 1.1cm measured using another ruler

To test the accuracy of the depth estimation, Baxter's arm was moved in such that the middle of the gripper coincided with the centroid of the square, and using the rule as means to approximate a distance of 161 mm from the center of the gripper to the square. This test was conducted on the square with Baxter's arm in 4 unique poses:

1. The arm directly above the square, with the rule perpendicular to the square.
2. The arm to the right of the square (relative to the viewer) with the gripper pointing towards the square.
3. Similar to the set-up 2 but the arm is to the left of the square.
4. The gripper ahead of the square pointing towards the square

5. The gripper behind the square pointing towards the square

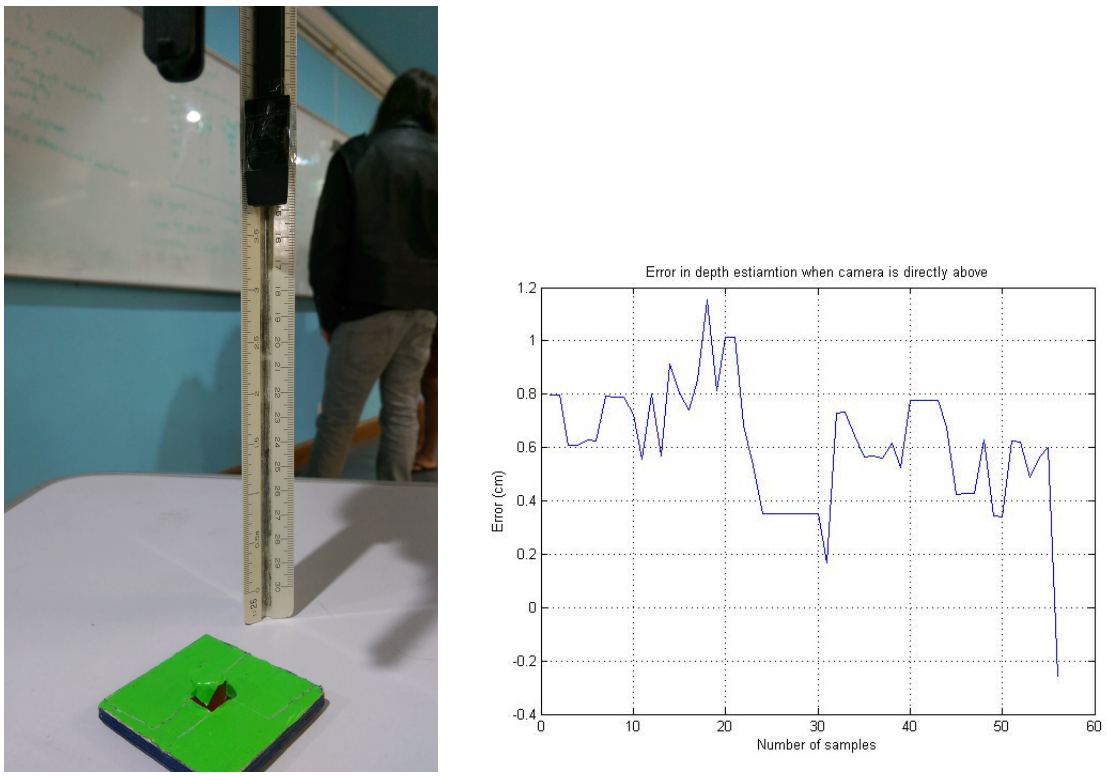
Baxter would observe the square for each pose listed above and collect 56 samples.

The description of the various poses maybe confusing, but images of the set-up will be shown further along the next sub sections to clear up any misunderstandings.

It is important to note a downside of using this method depth estimation, as the depth measurement will only be accurate if the line between the two pincers is parallel to the edges of the square. This is because it ensures that the rule is always perpendicular to atleast one axis of the square(Where the origin of square is at the centroid). If depth estimation is conducted on an angular approach where the line between the pincers is no longer parallel to the edges of the square, then the rule is no longer perpendicular to any of the squares' axis. Therefore, depending on the angle of approach, the distance from the centroid to the center of the grippers will higher or lower than 161mm. Therefore only test case 1, 4 and 5 will yield results that can quantitatively be used to measure the depth estimation accuracy. Test case 3 and 4 can only be qualitatively analysed to verify if the depth estimation is on the right track of what is expected, hence will not hold a huge weight when verifying the accuracy of the depth estimation. This will be made evident once again when examining the images of the experimental set-up.

Directly Above

The experimental set-up is shown in figure 5.3, the gripper placed above the square centroid.



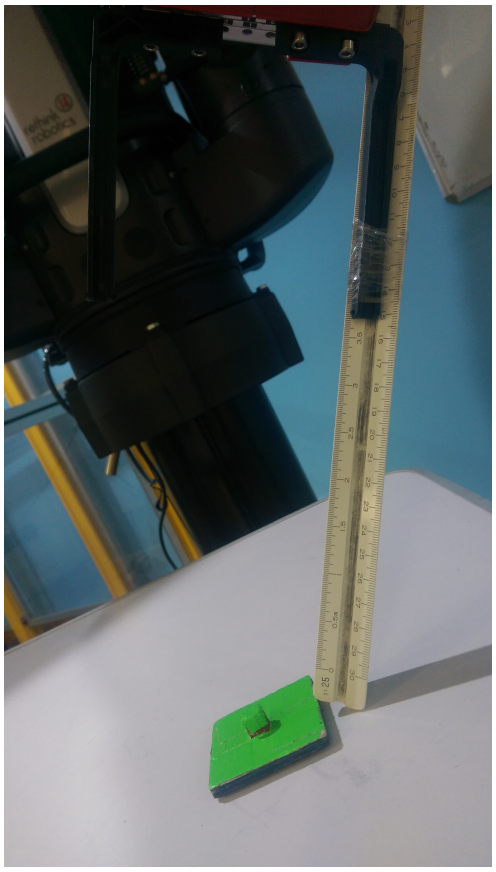
(a) Measurement Setup

(b) Error in depth estimation

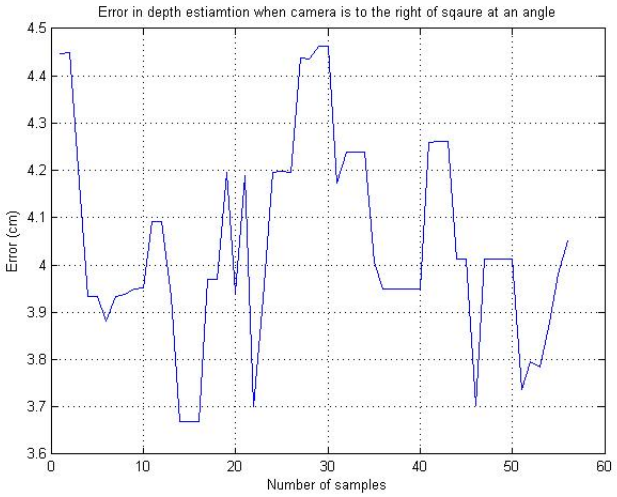
Figure 5.3: Depth estimation error for gripper directly above the centroid of the square

Left and Right Angled Approach

The experimental set-up shown in figure 5.4, the gripper placed to the left of the square, with the gripper facing the centroid of the square.



(a) Measurement Setup

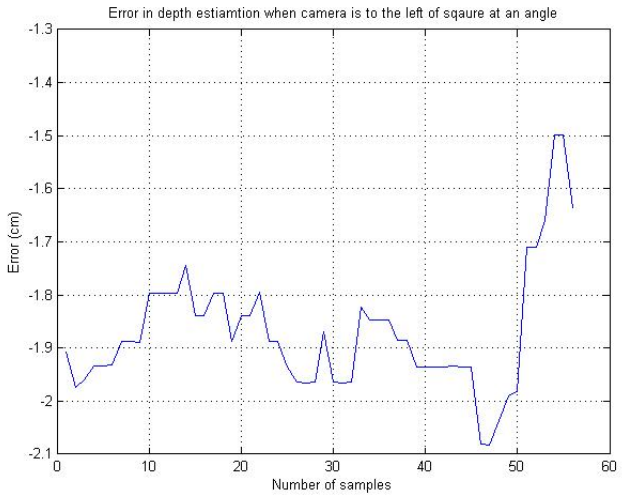


(b) Error in depth estimation

Figure 5.4: Depth estimation error for gripper to the right of the square with an angled gripper

The figure 5.5 shows a similar experimental set-up to figure 5.4 except this time, the gripper to the left of the square

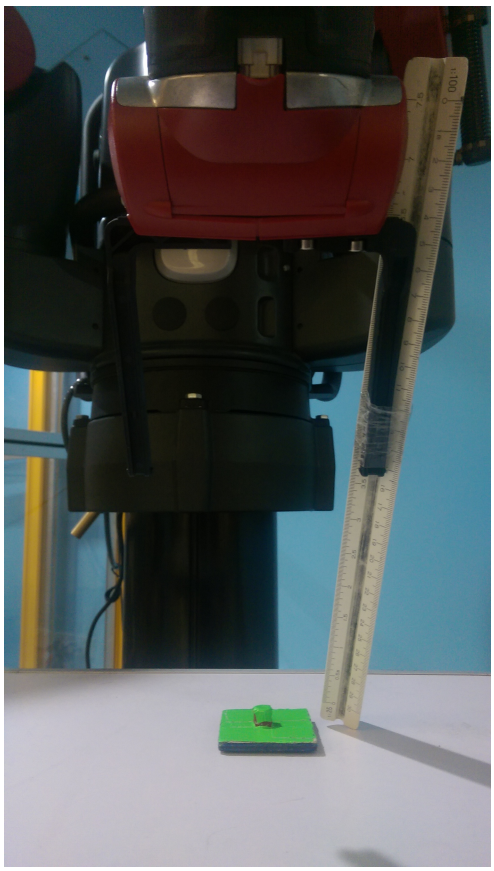
(a) Measurement Setup



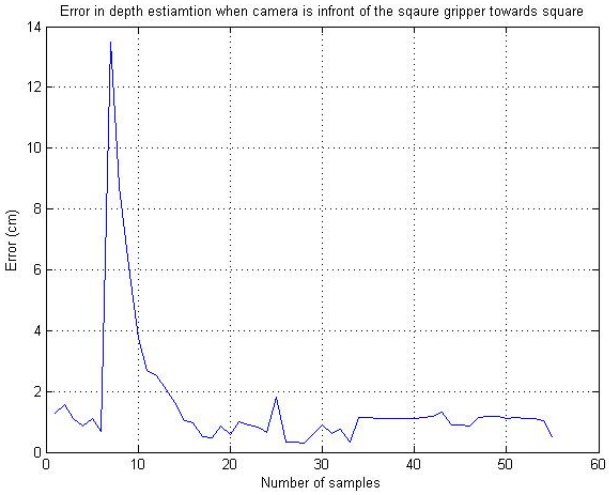
(b) Error in depth estimation

Figure 5.5: Depth estimation error for gripper to the left of the square with an angled gripper

In-front and Behind Angled Approach



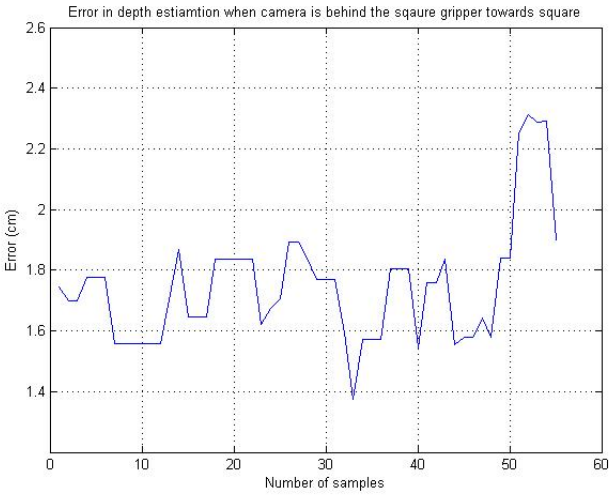
(a) Measurement Setup



(b) Error in depth estimation

Figure 5.6: Depth estimation error for gripper in front of the square with an angled gripper

(a) Measurement Setup

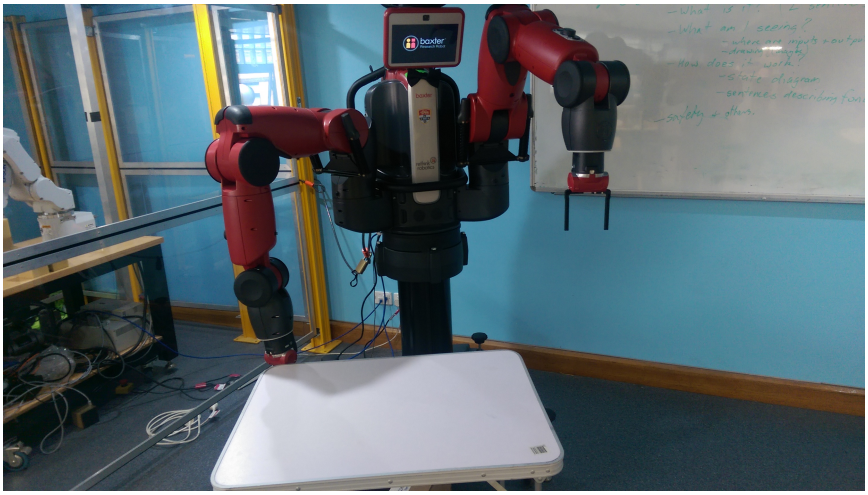


(b) Error in depth estimation

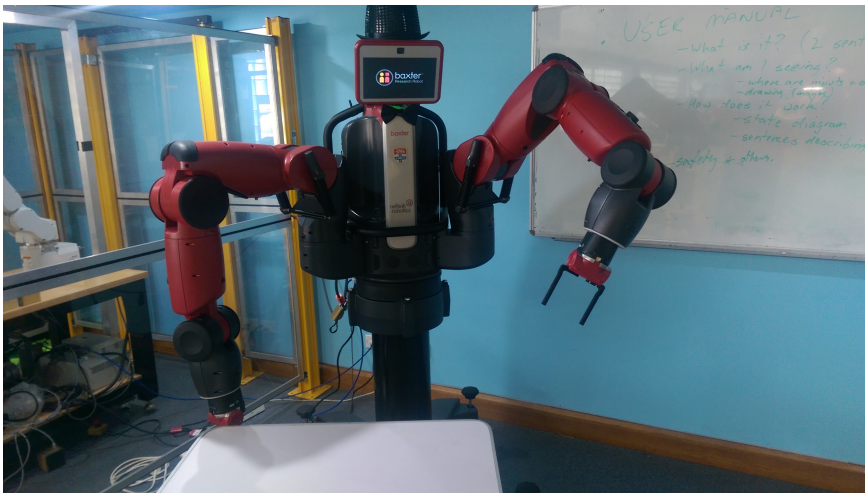
Figure 5.7: Depth estimation error for gripper behind the square with an angled gripper

5.3 Pose estimation

In this experiment the optimisers pose estimation is being tested, noting an time when was not able to reach the target. The experiment is also looking at effects of movement rate, and how that effects the error in trying to reach the square, and the time taken. Baxter's arm was placed in two different starting poses as shown in figure 5.8.



(a) Gripper Positioned Down



(b) Gripper Angled

Figure 5.8: Starting position of Baxter

Next, the square block is placed randomly on the table, at which point Baxter begins to move its arm towards the square.The error in this experiment is defined as how far away the square's centroid is from the origin of the gripper axis, where 3 error terms are determined:

1. Distance to the square's centroid from gripper axis origin
2. x axis error w.r.t to gripper axis origin
3. y axis error w.r.t to gripper axis origin

This experiment is then repeated for each pose, and each pose running at two different speeds:

1. each step is 35% of the total δPosition(P) vector required to move the gripper from its current to final position
2. each step is 12% of the total δP vector required to move the gripper from its current to final position

The following figures 5.9 and 5.10 are results for the gripper moving at 35% of total δP for two starting poses as seen in figure 5.8.

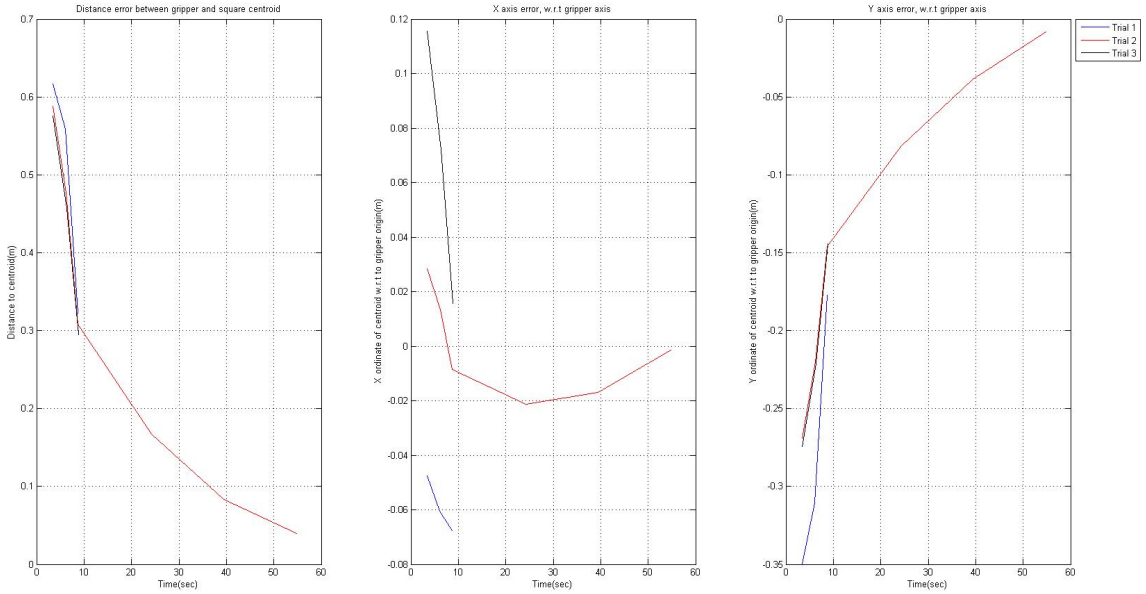


Figure 5.9: Pincer positioned down moving at 35% of δP

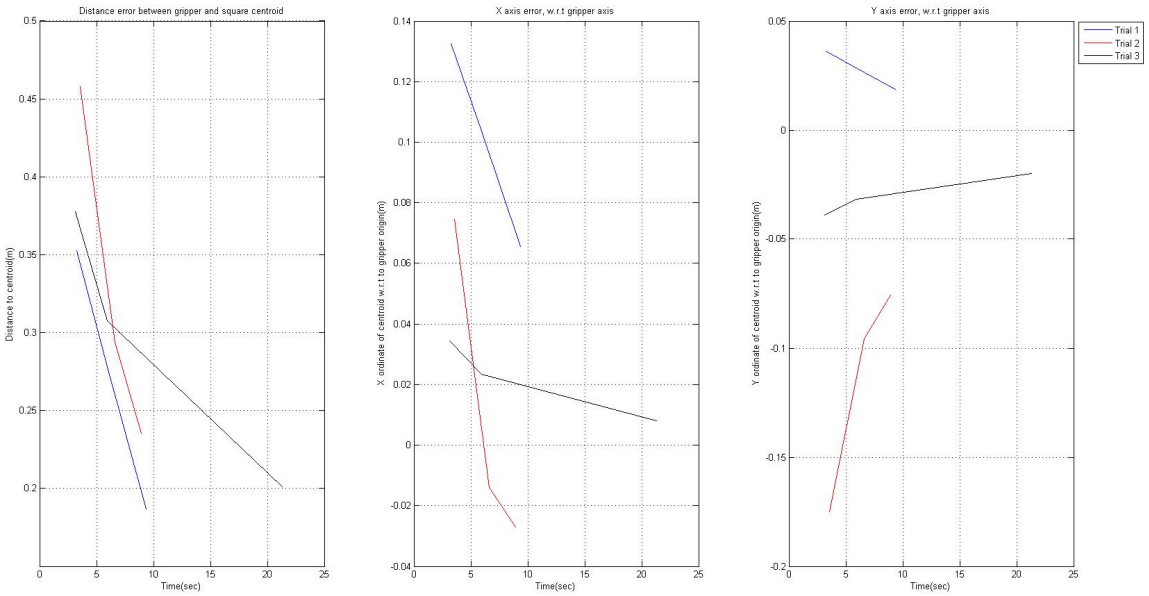


Figure 5.10: Pincer positioned at an angle moving at 35% of δP

The following figures 5.11 and 5.12 are results for the gripper moving at 12% of total δP for two starting poses as seen in figure 5.8.

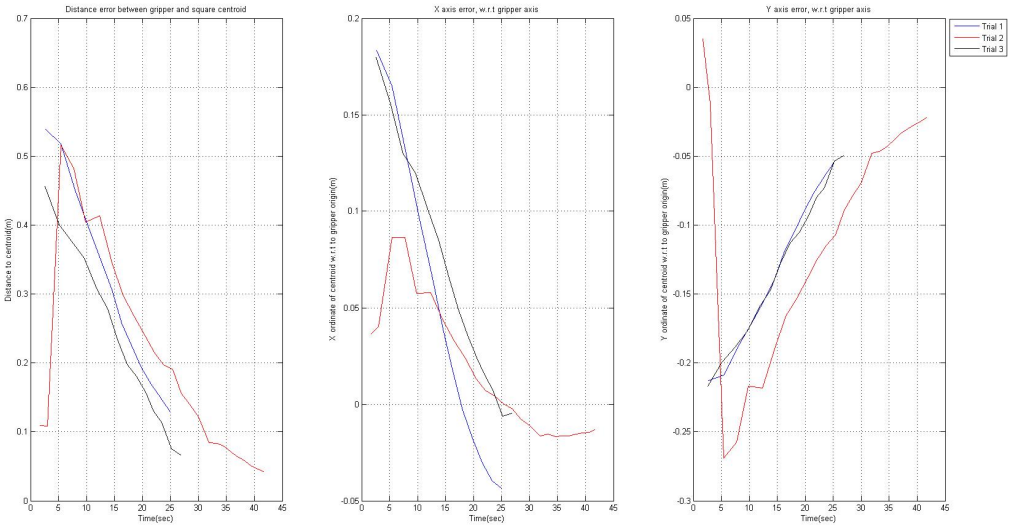


Figure 5.11: Pincer positioned down moving at 12% of δP

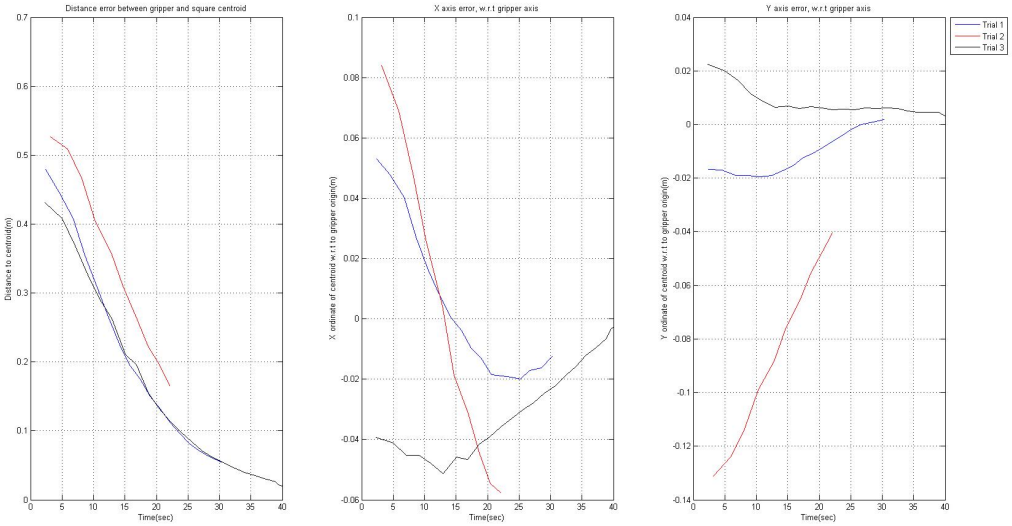


Figure 5.12: Pincer positioned at an angle moving at 12% of δP

Chapter 6

Discussion

The experimental results provided some interesting insight into using non-linear optimisation methods for depth and pose estimation. It is important to note when controlling Baxter's arm, Baxter's on-board controller were used, where commands to move the robotic arm were sent to the controller. All programming used for image processing and commanding Baxter was done on Python using OpenCV package, and all data gathered were analysed using MATLAB.

When computing the homogeneous transformation matrix to transform coordinates from camera space to Baxter space, the quaternion and end-effector position are used to determine the transform. This was a nice way to determine the transforms to move towards to the square. The problem of using this method, what ever the starting quaternion was used, the entire arm would maintain the same orientation as it moved. If the gripper was angled, the gripper would maintain the same angle as it moved to the square. This meant occlusion and collisions with the table would occur if the starting pose was not ideal.

During experimentation of the system, the object detection algorithm at times would have trouble detection vertices if the image after processing was very noisy, detecting more or less vertices than are observable. To mediate the issue a quick and simple high

pass filter was applied to remove any vertices that are detected close to each other. In doing so reduced the max height at which Baxter's arm can go before the square is no longer able to be detected.

6.1 Depth Estimation

The depth estimation for instances when Baxter's rule was perpendicular to the axis of the square which are figure 5.3, 5.6 and 5.7 provided fairly accurate results, with some constant offset that was present in all three results. The offset is possibly down to two issues, the first being un-distorting the raw image; when conducting the camera calibration, the radial and tangential distortion parameters were initially thought low enough to be safely ignored. By ignoring the un-distortion process, the error when computing the non-linear optimisation must have accumulated over time. The second possible source of the offset is translation matrix calculated to determine a relative vector between the gripper and camera. Since no data sheet that was detailed enough was available to determine the dimensions between the gripper and camera, it had to be estimated using a rule and measuring tape. Due to the awkward contours between the camera an gripper, the measurements were not accurate enough causing the constant error offset. The average error for figure 5.3, 5.6 and 5.7 are:

- Directly Above: 0.6197 cm
- In Front: 1.5486 cm
- Behind: 1.7476 cm

Where as the error for figure 5.4 and 5.5 were significantly higher, but as mentioned in chapter 5,the data can only be analysed qualitatively. Studying the pose of the gripper in 5.4, the center of the gripper is located higher than where the rule is protruding. This means that we expect the depth estimation to be higher than 161mm, which is what is

observed. Similarly for figure 5.5, the center of the gripper is located lower that where the rule protrudes, hence it is expected that the depth estimation to be lower than 161mm, which is observed.

Overall the depth estimation using a non-linear optimisation method produces results the appear to be viable ensuring that the distortion (OpenCV has inbuilt methods given intrinsic parameters can un-distort the image) and translation matrix issues are remedied. The reason it can be said the method is viable is because the position accuracy of Baxter is ±5mm, so if the error falls between this range, Baxter can fairly accurately determine how far to move to grab the square.

6.2 Pose Estimation

Studying the results for large δP movements (figure 5.9), when the gripper is facing down as shown in figure 5.8(a) is converging towards a zero error, where trial 2 closely reaches the centroid of the square, where trial 1 and 3 where unable to reach the square due the square partially leaving the frame of the image, hence the short and abrupt stops in error measurements. The error when the gripper is positioned as shown in figure 5.8(b) is significantly greater, this was because all tests failed to reach the square due to occlusion of the square caused by the gripper. Although it is important to note that once again, the system error is tending towards zero.

When the step size between each movement was reduced and the experiment was ran again, the error reduced far smoother than compared to figure 5.9 and 5.10, and also reached far closer to the square before occlusion or partial parts of the square left the image. One of the biggest source of error that caused the square to partially leave the image frame for either step size was the large error in the x-axis of the centroid w.r.t to gripper axis.

It is important to note that a smaller step size yielded an error that was far lower

than large step sizes, this is due to the fact that the camera is allowed to take samples of the square. In turn running a larger number of optimisation algorithm resulting in more accurate estimation. The down side is that the process takes far longer, the average time to complete the task for 12% δP is 31 seconds, where as the larger steps take an average of 18 seconds(The 60 second time taken in figure 5.9 for distance error trial 2, was because the square was partially cut off in the image frame, but due to the backlash on Baxter's motor the square came back into the frame).

6.3 Fusing PBVS and IBVS schemes

The current scheme of control is a dynamic look and move method, with aspects of PBVS where the features of the image are extracted, then using the Cartesian controller of Baxter arm to move it to a desired position. As mentioned by Corke and Hutchinson, PBVS control methods accuracy is highly dependent on the calibration parameters and knowing the kinematics of the robotic arm. This was made evident in the results that were gathered for both depth and pose estimation. Though the optimiser produced fairly accurate estimates of the pose and depth w.r.t to the camera, due to poor transformation matrix estimate from the camera to the gripper, and failing to un-distort the image, the final position that arm moves will deviate from the expected result.

An improved control scheme that can be applied to control Baxter's arm is to combine the feature extraction method of dynamic look and move with IBVS. The inherent draw back of IBVS is the lack of depth (z) information when controlling the robotic arm, since we have a model of an object, the non-linear optimisation method can be used to estimate the depth and initial pose for IBVS control scheme. This is so that we no longer rely on the Cartesian controller of Baxter, but solely use a control scheme that is dependent on what is happening the features in the image.

Chapter 7

Conclusion and Future Work

The goal of the thesis was to study how effective non-linear optimisation methods can be used for known models, to estimate the pose and depth so that a robotic arm can reach a modelled target. Chapter 5 and 6 show that the optimisation method holds promise in estimating the depth and pose of the object, and its ability to move towards a target.

The optimisation method does have disadvantages in the way it has been applied in the control scheme, as the optimisation is not being applied to live camera feed, rather an image is taken after each movement. This means that the system is highly dependent on Baxter's controller and how quickly the arm is able to move, as the control in this thesis was purely based in Cartesian space using position vectors. To improve the speed and error a controller must be developed from scratch, in particular as mentioned in chapter 6 use an IBVS control scheme. IBVS control scheme is less susceptible to calibration factors, as the control scheme is based in the image frame, controlling the robotic arm based on how the features in the image move.

The non-linear optimisation method was only tested on geometric objects, future work could focus on the use the optimisation method for more irregular shapes. The challenge in using this method for irregular shapes is selecting appropriate features to try and optimise, possible focus maybe using machine learning algorithms to select desired

features to optimise, in doing so create a more robust and general method to estimate the pose and depth of an object

Since most robots such as Baxter are used for pick and place task, this thesis only explored pick aspect of the task, future extension from this thesis would be to place objects at desired locations that form par of a simple assembly. Specifically using tactile feedback via torque readings from the joints of Baxter's arm to place objects correctly in an assembly task.