ISSN 0280-5316
ISRN LUTFD2/TFRT--5694--SE

# Simulation of Visual Servoing in Grasping Objects Moving by Newtonian Dynamics

Jee Hui Wong

Department of Automatic Control
Lund Institute of Technology
October 2002

| | |
|---|---|
| **Department of Automatic Control**<br>**Lund Institute of Technology**<br>**Box 118**<br>**SE-221 00 Lund Sweden** | *Document name*<br>MASTER THESIS<br>*Date of issue*<br>October 2002<br>*Document Number*<br>ISRN LUTFD2/TFRT--5694--SE |
| *Author(s)*<br>Jee Hui Wong | *Supervisor*<br>Johan Bengtsson, Tomas Olsson, Rolf Johansson at LTH. Mathias Haage Computer Science, LTH. Martin Clarke at Imperial College<br>*Sponsoring organization* |

*Title and subtitle*

Simulation of Visual Servoing in Grasping Objects Moving by Newtonian Dynamics.
(Simulering av visuell följning för gripande av object med Newtonsk rörelsedynamik)

*Abstract*

Robot control systems and other manufacturing equipment are traditionally closed systems. This circumstance has hampered system integration of manipulators, sensors as well as other equipment, and such system integration has often been made at an unsuitably high hierarchical level. With the aid of vision, visual feedback is used to guide the robot manipulator to the target. This hand-to-target task is fairly easy if the target is static in Cartesian space. However, if the target is dynamic in motion, a model of the dynamics behaviour is required in order for the robot to track and intercept the target.
The purpose of this project is to simulate in a virtual environment to show how to organise robot control systems with sensor integration. This project is a simulation that involves catching a thrown virtual ball using a six degree-of-freedom virtual robot and two virtual digital cameras. Tasks to be executed in this project include placement of virtual digital cameras, segmentation and tracking of the moving virtual ball as well as model-based prediction of the virtual ball's trajectory.
Consideration have to be given to the placement of the virtual digital cameras so that the whole trajectory of the ball can be captured by both the virtual digital cameras simultaneously. In order to track the trajectory of the virtual ball, the image of the ball captured by the digital cameras has to be segmented from its background. Then a model is to be developed to predict the trajectory of the virtual ball so that the virtual robot can be controlled to align itself to grasp the moving virtual ball.

*Keywords*

*Classification system and/or index terms (if any)*

*Supplementary bibliographical information*

| | |
|---|---|
| *ISSN and key title*<br>0280-5316 | *ISBN* |

| | | |
|---|---|---|
| *Language*<br>English | *Number of pages*<br>60 | *Recipient's notes* |
| *Security classification* | | |

# Simulation of visual servoing in grasping objects moving by Newtonian dynamics

Wong Jee Hui

September 26, 2002

# Chapter 1

# Introduction

The vast majority of today's growing robot population operate in factories where the environment can be conditioned to suit the robots. For instance, tasks such as assembly are set up so that the robots know the exact position of the different objects to be grasped. Robots have had far less impact in applications where the working environment and object placement cannot be accurately controlled. This limitation is largely due to the lack of sensory capability in contemporary commercial robots. It has long been recognised that sensor integration is fundamental to increasing the versatility and application of robots. Cameras are an example of such sensors, providing visual capability for the robot on the environment.

Vision is a useful robotic sensor since it mimics the human sense of vision and allows for non-contact measurement of the environment. Robot controllers with fully integrated vision systems are now available. However, the accuracy of the resulting operation depends directly on the accuracy of the visual sensor and the robot manipulator. An alternative to increasing the accuracy of these subsystems is to use a visual-feedback control loop that will increase the overall accuracy of the system. The main idea is to use the cameras as sensors, and the data extracted from the images of the camera in a feedback loop. The goal is usually to align the robotic manipulator with the desired object in order that the object can be grasped by the robotic manipulator. Thus machine vision can provide closed-loop position control for a robot manipulator. This is referred to as "visual servoing". [1]

One problem when a new control system is built is the testing of the new control system. Three-dimensional computer modeling is a useful and economic way to simulate an experimental environment to understand the cause and effect of control techniques on the simulated environment. It is also quicker and easier to change parameters in a three-dimensional computer model than the real world. After obtaining successful implementation in the simulated environment, implementation can then be done on real objects in the real world with much smoother results expected.

This project is a collabration between the author and his partner, Woon Teck Her. The author's work consists mainly of interpreting and processing visual images captured by the two digital cameras, while his partner concentrated on developing a model to estimate the position and trajectory of the ball. Although our responsibilities in this project were distinctive, our efforts were not exclusive as we contributed to each other's field of work during the course of this project.

Three fundamental fields have been combined for the development of this master thesis, namely robotics, computer vision and three-dimensional computer modeling. These three fields have been popular topics for research and the automative industry. The author shall now attempt to give a brief historical accounts of the three fields as well as some basic introduction.

## 1.1 Robotics

A robot is:

> 'A reprogrammable, multi-functional manipulator designed to move materials, parts, tools or specialised devices through the various programmed motions for the performance of a variety of tasks.'
> *Robot Institute of America(1979)* [2]

The word 'robot' was coined by the Czech playwright Karel Čapek (see Figure 1.1) naturally from the Czech word, for forced labour. The use of the

Figure 1.1: Karel Čapek

word ‘robot’ was introduced to his play, Rossum’s Universal Robots (RUR), which opened in Prague in January 1921. In RUR, Capek poses a paradise, where the machines initially bring so many benefits but in the end bring an equal amount of blight. The robots then were not mechanical in nature though, but were created through chemical means. [3] It is interesting to note the deviational route taken by modern robots, in contrast to the original ‘robot’ dreamt by Čapek.

The term ‘robotics’ refers to the study and use of robots. The term was first coined and used by the Russian-born American scientist and writer Isaac Asimov. The word ‘robotics’ was first used in a short story published in 1942, Roundaround. [4]

After the technology explosion during the World War II, a historic meeting occured between two Americans, George C. Devol and Joseph F. Engelberg in 1956. The former was a successful inventor and entrepreuner while the latter was an engineer. The two were discussing the writings of Asimov over cocktail. Together they made a serious effort to develop a real, working robot. They persuaded Norman Schafler of Condec Corporation in Danbury that they had a basis for commercial success. Their first robot was nicknamed the ‘Unimate’ (see Figure 1.2). Engelberger started a manufacturing company named ‘Unimation’ while Devol wrote the necessary patents. Thus the first commercial company to make robots was formed and as a result

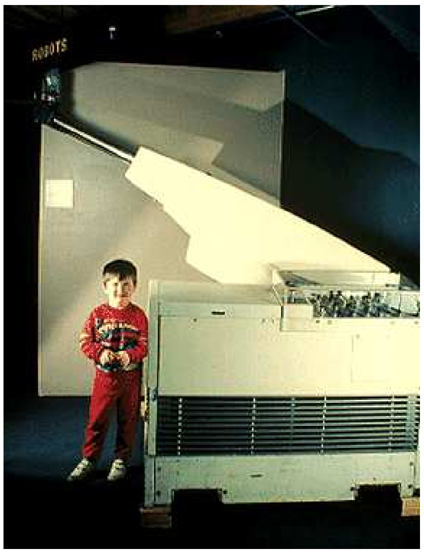


Figure 1.2: A picture of a boy with the model of an ‘Unimate’, the first commercial robot

of this, Engelberger has been called the ‘father of robotics’. The first Unimate was installed at a General Motors plant to work with heated die-casting machines. [5]

Modern industrial robots have increased in capability and performance through controller and language development, improved mechanisms, sensing and drive systems. In the early to middle of 1980s, the robot industry grew very fast primarily due to large investments by the automotive industry. Fully functioning androids (robots that look like human beings) are many years away due to the many problems that must be solved. However, real, working, sophisticated robots are in use today and they are revolutionising the workplace. These robots do not resemble the romantic android concept of robots. They are industrial manipulators and are really computer controlled “arms and hands”. Industrial robots are so different to the popular image that it would be easy for the average person not to recognise one. [6]

Today, robots with restricted sensor feedback are limited in the kinds of behaviour they can exhibit. However, this is the way they are currently used in industry. The needs for motion desciptions and operator interactions clearly show that robot control needs its own control techniques. [7]

Robot control techniques traditionally uses world coordinate (Cartesian) system as well as joint coordinate system to determine the position of the robot and the desired positions and trajectories. This works satisfactorily for static environments in industrial applications. However, most natural settings are not structured and not easy to model and they are bound to create problems in the process of controlling the robotic manipulator.

In this master thesis, a six degree-of-freedom industrial robot is guided to grasp a ball moving with three-dimensinal Newtonian dynamics in an unattended and less structured environment. The object of focus for implementing control will be solely on the robotic arm.

## 1.2 Computer Vision

In robotics, we require a feedback control scheme for a task involving the manipulation of three-dimensional objects. One easy way to acquire feedback information is through an external sensor. A robotic manipulator may use as feedback the information coming from different kinds of sensors. One such kind of external semsor is a camera, where image of the three-dimensional object is captured and then sent to the computer to be analysed. Hence computer vision comes into play.

Unlike robotics, computer vision until recently was still regarded as a field of research still in its infancy, not yet mature and stable enough to be considered part of a standard curriculum in computer science. A quick review of the field reveals that image processing and pattern recognition has achieved tremendous success in terms of delivering operational systems. Everyday barcodes are used in supermarkets and pattern recognition techniques are used for purposes such as identification, bill recognition and address recognition. Contrarily, full-scale computer vision applications that involves motion estimation, depth recovery and scene interpretation have achieved fairly limited progress. The goal of computer vision is to make computers understand and interpret visual information. [8]

Computer vision involves the capturing, understanding and processing of images. The tools needed by a computer vision system include hardware for acquiring and storing digital images in a computer, processing the images and communicating results to users or other automated systems. [9] The processing of information poses problems that are difficult to manage. For example, it is difficult to distinguish objects of different material or even of different geometrical shape because their image captured by the camera could be the same.

In computer vision, when the position of a three-dimensional object is to be determined using a two-dimensional image, the coordinate that indicates the depth of the object is difficult to be determined. An example of human processing of an image regarding the depth of an object is called "pictorial depth cue". A 'cue' may be the most familiar size, interposing or occlusion, shade or shaded area, size related to the horizon line, motion and motion parallax, binocular perception (stereoscopy). Stereoscopy is the study of corresponding images for recreating three-dimensional coordinates. Depth is calculated from the disparity between two or more images. [10]

A solution for improving depth information is to utilise several cameras in a configuration such that the desired target is in the view of all cameras involved. Most multi-camera configurations use two cameras, also called the stereo rig configuration. This configuration allows the system to obtain a complete Euclidean reconstruction of the observed objects. [11]

In this master thesis, the stereo rig configuration is adopted. Two digital cameras are used as external sensors to feedback the visual image of the environment, specifically the position and the trajectory of the ball.

## 1.3 Three-dimensional computer modeling

In recent years, computer graphics has made tremendous progress in visualising three-dimensional models. Many techniques have reached maturity and are being ported to hardware. This evolution causes an important demand for more complex and realistic models. The problem is that even though the tools that are available for three-dimensional computer modeling are getting more and more powerful, synthesising realistic models is difficult and time-consuming, not to mention very expensive. Many virtual objects are inspired by real objects and it would therefore be interesting to be able to acquire the models directly from the real object.

Researchers have been investigating methods to acquire three-dimensional information from objects and scenes for many years. In the past the main applications were visual inspection and robot guidance. Nowadays however the emphasis is shifting. There is more and more demand for three-dimensional models in computer graphics, virtual reality and communication. This results in a change in emphasis for the requirements. The visual quality becomes one of the main points of attention. Therefore not only the position of a small number of points have to be measured with high accuracy, but the geometry and appearance of all points of the surface have to be measured.

Due to the convergence of these different factors, many techniques have been developed over the last few years. Many of them do not require more than a camera and a computer to acquire three-dimensional models of real objects. What may have required a million pound computer a few years ago can now be achieved by a personal computer costing a few hundred pounds. The goal of three-dimensional computer modeling is to automatically extract a realistic three-dimensional model by freely moving a camera around an object. [12]

In this project, OpenGL is used for modelling three-dimensional objects in the virtual environment setup. OpenGL is an environment for developing portable, interactive three-dimensional graphics applications. OpenGL is a cross-platform standard for three-dimensional rendering and three-dimensional hardware acceleration. The software runtime library ships with all Windows, Macintosh Operating System, Linux and Unix systems. [13] More importantly for our project, OpenGL standard has language bindings for Microsoft Visual C++, which has passed a set of conformance test.

## 1.4 Visual servo control

Visual servoing is the fusion of results from many elemental areas, including robotics, computer vision and three-dimensional computer modeling. The task of visual servoing is to control a robot to manipulate its environment using vision, as opposed to just observing the environment.

Since the first visual servoing systems were reported in the early 1980s, progress in visual control of robots has been fairly slow, but the last few years has seen a marked increase in published research. This has been fueled by personal computing crossing the threshold that allows analysis of scenes at a sufficient rate. Prior to this, researchers required specialised and expensive pipelined pixel processing hardware. The task of visual servoing is to control the position of the robot's end-effector using visual information obtained from images.

There are two different kinds of visual servo control system, namely Postion-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS). For the former, features are extracted from the camera image and used to estimate the position of the target with reference to the camera. Using these values, an error between the current and the desired position of the robot is defined in the task space. In this way, PBVS neatly separates the computation of feedback signal and the estimation problems involved in computing position from the visual images obtained. For IBVS, control values are computed on the basis of image features directly. A disadvantage with IBVS is that there is no direct control over the Cartesian velocities of the robot manipulator. As a result, the robot can execute trajectories that are desirable in the image, but which are contorted in Cartesian space. [14]

In this master thesis, PBVS is adopted for the visual servoing of the robot. Visual feedback from two different digital cameras will be used to track the moving ball and image coordinate system will be introduced alongside Cartesian coordinate system as well as joint coordinate system. The two-dimensional image acquired from the digital cameras are converted into three-dimensional Cartesian coordinates. New Cartesian coordinates for estimating the expected position of the ball are then computed before being converted into joint coordinate system. The joint coordinates are then fed back to the robotic arm to be implemented so that the gripper of the robotic arm can align itself to grasp the ball.

## 1.5 Thesis outline

The thesis is organised into various chapters as follows:

- Chapter 2: Problem Formulation
- Chapter 3: Camera Model
- Chapter 4: Experimental Setup
- Chapter 5: System Overview
- Chapter 6: Image Processing

- Chapter 7: Three-dimensional Reconstruction
- Chapter 8: Model-based Ball Grasping
- Chapter 9: Results
- Chapter 10: Discussion
- Chapter 11: Conclusion
- Chapter 12: Future Work
- Chapter 13: References
- Chapter 14: Acknowledgements
- Chapter 15: Appendices

# Chapter 2

# Problem Formulation

The main goal of this experiment is in controlling the virtual robot to grasp the thrown virtual ball moving in three-dimensional Newtonian dynamics, by using computer vision to analyse images captured from two virtual digital cameras.

## 2.1 Experimental Setup

A framework need to be set up to place the virtual robot, the virtual ball and the two virtual digital cameras at strategic positions in a virtual environment. The virtual ball needs to be programmed to move in the three-dimensional environment with Newtonian dynamics. The virtual environment needs to be constantly updated of the change in position and trajectory of the virtual ball as well as the position of the gripper of the robotic arm. A software program called Microsoft Visual C++ will be used to set up this virtual framework.

## 2.2 Camera Placement

Special attention is to be given to the placement of the two virtual digital cameras. It has to be ensured that image of every position of the vitual ball's trajectory can be captured by both the virtual digital cameras. The image on both virtual digital cameras have to be captured simultaneously to ensure that there is a good depth estimation of the virtual ball. This can be done by orientating the virtual digital cameras and adjusting the focal length of the virtual digital cameras.

## 2.3 Image Processing

The visual images obtained from the two virtual digital cameras will be sent to a computer to be analysed. The center point of the virtual ball is to be determined with reference to the images obtained from the two virtual digital cameras. The colour of the virtual ball have to be chosen so that the image of the ball can be segmented from its background. The method used for segmenting the image of the virtual ball from its background will be feature point extraction with thresholding. A software program called MATLAB will be used for the segmentation of the image of the virtual ball.

## 2.4 Three-dimensional Reconstruction

The stereo rig configuration setup of the two virtual digital cameras should solve the reconstruction problem. Depth can be calculated from the disparity between corresponding feature points found in both the visual images. The reconstruction problem reconstructs the three-dimensional world coordinate using the images from both the virtual digital cameras.

## 2.5 System Identification

The position of the gripper of the robotic arm is constrained to lie in the vertical plane one metre away in front of the robot. Linear system identification method such as linear regression is needed to identify the system, which is non-linear because the Newtonian dynamics of the ball are non-linear. With the initial velocity, intial angles and initial three-dimensional position identified recursively, the robotic arm have to be manipulated to align itself at the estimated position that the virtual ball will reach in the vertical plane specified.

# Chapter 3

# Camera Model

In the visual system of man, the process of image formation begins with light rays coming from the outside world and impinging on the photoreceptors in the retina of the eye. This chapter gives a short theoretical orientation of problems in computer vision. It shows image acquisition and the necessary processing methods to produce estimation of three-dimensional data.

## 3.1 The Human Eye

The human eye (see Figure 3.1) is approximately spherical with a diameter of 15 millimeters and light is sensed by the photoreceptors located in the retina at the back of the eye. In normal daylight conditions, the photoreceptors are active and the colour sensitivities are: 65% red, 33% green and 2% blue. The distance between the lens and the retina is approximately constant at 15 millimeters, so focusing is achieved by muscles that change the shape of the lens.

Understanding of the human eye provides the basic fundamentals into the theory of optics.

## 3.2 Basic Optics

As with many natural visual system, the process of image formation in computer vision begins with the light rays that enter the camera through an angular arpeture and hits a screen (image plane). The image plane is the camera's photosensitive device that registers light intensities. Most of these rays are the result of the reflections of the rays emitted by the light sources hitting object surfaces.

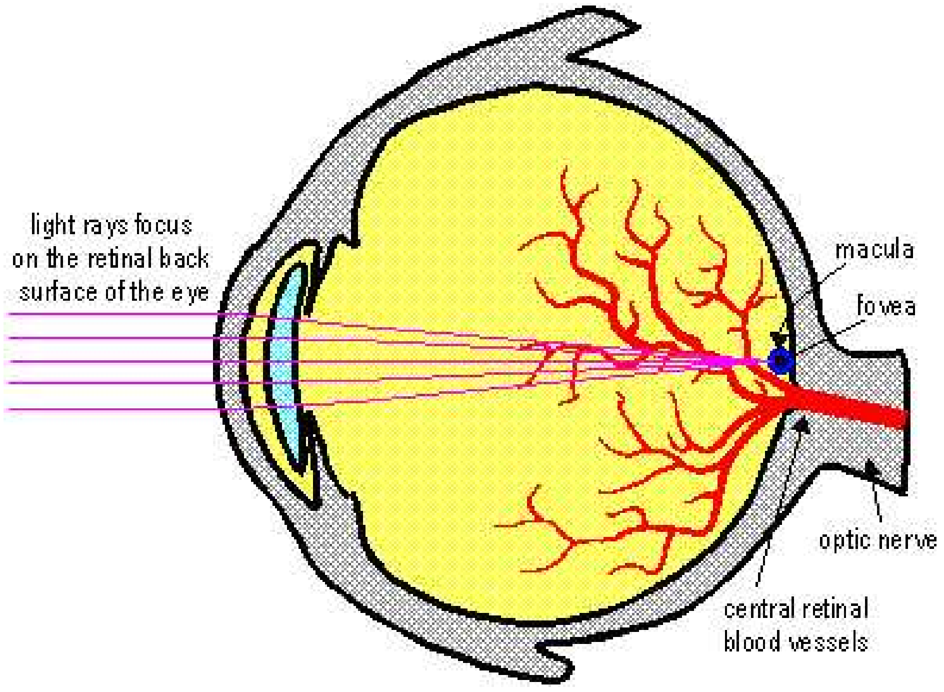


Figure 3.1: A human eye

Any single point of a scene reflects light coming from many directions, so that many rays reflected by the same point may enter the camera. In order to obtain sharp images, all rays coming from a single point of the scene, $P$ must converge onto a single point on the image plan, $p$. If this happens, the image of $P$ is in focus; if not, the image of $P$ is spread over a circle. There are two ways to make the image of $P$ in focus:

1. Reducing the camera's aperture to a single point, which is also called a pinhole camera. Thus only one ray from any given point in the scene can enter the camera and this creates a one-to-one correspondence between the visible points in the scene, rays and points on the image plan. This results in very sharp, undistorted images of objects at different distances from the camera.

2. Introducing an optical system composed of lenses and apertures, explicitly designed to make all rays coming from the same point in the scene converge onto a single image point.

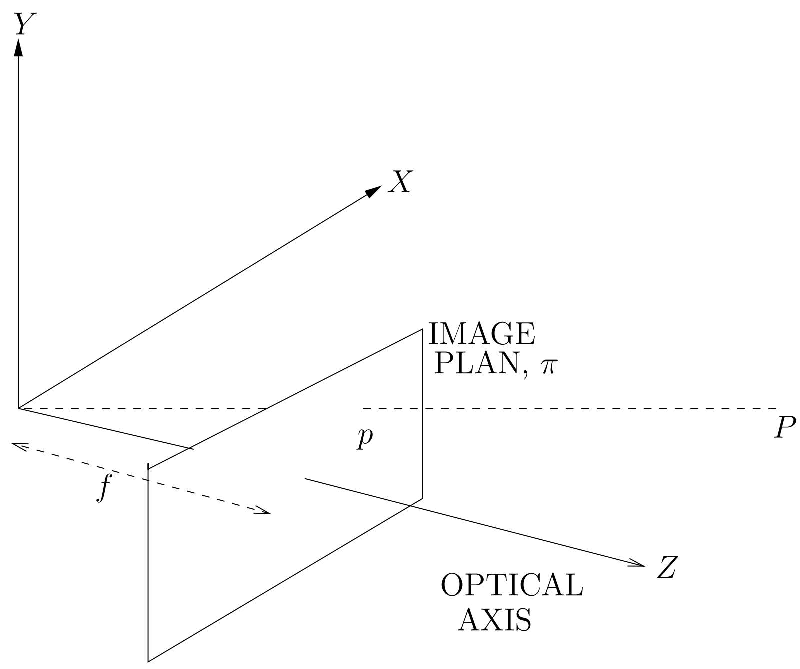


Figure 3.2: A pinhole camera

### 3.2.1 Pinhole Camera

A pinhole camera is a box that has a infinitely small hole, through which light enters and forms an inverted image on the image plane. Usually the image plane is placed between the focal point of the camera and the object so that the image is not inverted. This mapping of three dimensions onto two dimension is called perspective projection.

In Figure 9, a three-dimensional feature point, $P = (X\ ,Y\ ,Z)$ is projected onto an image plane with perspective rays originating at the center of projection (COP). The origin of the coordinate system is traditionally taken to be the COP. The focal length, $f$ is the distance from COP to the image plane along the optical axis. The optical axis is traditionally aligned with the z-axis. By geometry, the image coordinates $(u_i\ ,v_i)$ on the x-axis and y-axis respectively are related to the object coordinates of $(X_i\ ,Y_i\ ,Z_i)$ by:

$$u_i = f\,\frac{X_i}{Z_i} \quad \text{and} \quad v_i = f\,\frac{Y_i}{Z_i} \tag{3.1}$$

These equations can be easily expressed by introducing homogeneous transformation, which is a matter of placing Euclidean geometry into the projective geometry space. In homogeneous equations, the perspective projection onto the plane is given by:

$$\lambda \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{3.2}$$

where $\lambda = Z_i$ is the depth of the point found on the image plan of the camera.

An obvious disadvantage of a pinhole camera is its exposure time (length of time the image plane is allowed to receive light). A digital camera needs a minimum amount of light to register a clear image. As a pinhole camera allows very little light into the camera per unit of time, the exposure time of several seconds is needed to form the image. [15]

### 3.2.2 Optical System

An optical system (see Figure 3.3) instead can be regarded as a device that aims to produce the same image obtained by a pinhole camera, but by means of a larger aperture and a shorter exposure time. Morever, an optical system enhances the light gathering power. The optical behaviour of a lens is characterised by two elements:

1. An optical axis going through the center of the lens and perpendicular to the scene

2. Two special points called left focus, $F_l$ and right focus $F_r$ respectively. They are placed on the optical axis, on opposite sides of the lens and the same distance from the center of the lens. This distance is usually called focal length, $f$.

By construction, a thin lens deflects all rays coming from one side and parallel to the optical axis, onto the focus on the other side. Assuming the lens is relatively thin, it operates according to the following law:

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f} \tag{3.3}$$

where $u$ and $v$ are are the distance of a point on an object from the image plan and the distance of a point on the focused image from the image plan respectively.

The image formed by a lens is not ideal. Lenses used for imaging are usually compound lenses containing several simple lenses in order to achieve a compact optical system. Imperfections in the shape or alignment of the

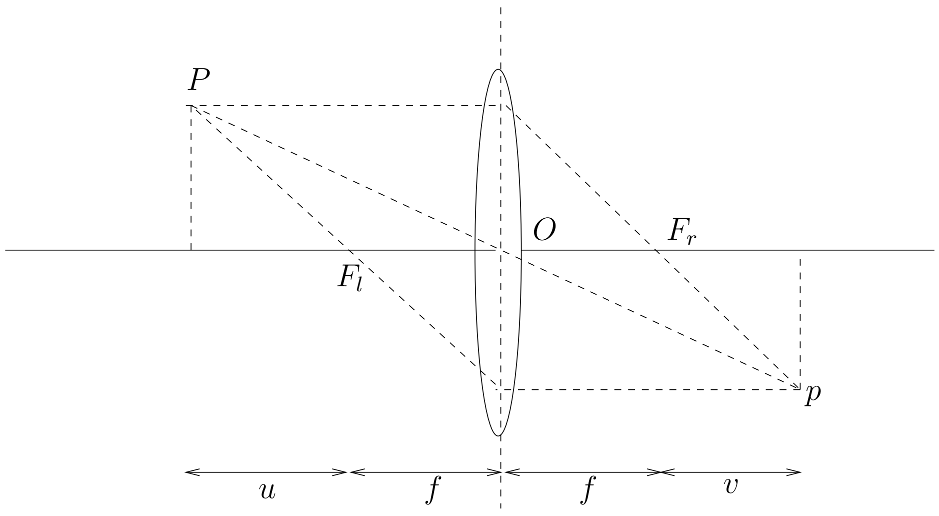


Figure 3.3: An optical system

simple lenses lead to degraded image quality. Non-idealities include aberrations that lead to image blur and geometric distortions cause the image to fall in wrong places. [16]

Since the experiment is done in a simulated three-dimensional computer modeled environment, the pinhole camera model is a sufficient model for the two virtual digital cameras used in this experiment.

## 3.3 Projective Geometry

Projective geometry is fundamental to the understanding of image analysis.

Although focal length is the most emphasised internal camera geometry parameter, there exists more complex parameters. For an ideal optically-modelled camera to deliver a perspective image, a mapping that completely characterise the camera model must be done.

### 3.3.1 Intrinsic Conditions

This mapping can be characterised completely by using the intrinsic parameters, which entails six of the following parameters:

> $f$ = focal length in pixels
> $u_p$ = camera's x-coordinate of the center of the projection
> $v_p$ = camera's y-coordinate of the center of the projection
> $\gamma$ = skew between the camera axes
> $\alpha$ = scaling of the image plane along the camera's x-axis
> $\beta$ = scaling of the image plane along the camera's y-axis

The intrinsic camera matrix $K$ includes all six of these intrinsic parameters for the camera model, which is effectively reduced to 5 since $\alpha$ and $\beta$ are dependent on each other. The intrinsic camera matrix $K$ is given as the following:

$$K = \begin{bmatrix} f\alpha & f\gamma & u_p \\ 0 & f\beta & v_p \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

### 3.3.2 Extrinsic Conditions

There are two extrinsic parameters for the camera model:

> $R_{(3\times3)}$ = orthogonal matrix that specifies the orientation of the image plane coordinate frame
> $t_{(3\times1)}$ = matrix that specifies the translation from the COP

Since $|R| = 1$, this corresponds to the matrix relation that $RR^T = R^TR = I$ and thus $R^{-1} = R^T$. The final projection equation for a point $(X_i\ ,Y_i\ ,Z_i)$ in Cartesian space for the perspective camera is given by:

$$\lambda \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = K \begin{bmatrix} R_{(3\times3)} & t_{(3\times1)} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{3.5}$$

where $\lambda$ is a non-zero scalar factor indicating the depth of a point on the image plan.

The result of the matrix multiplication, $H_{3\times4} = \mathrm{K}\begin{bmatrix} R_{(3\times3)} & t_{(3\times1)} \end{bmatrix}$ is the calibration matrix, also known as the projective tranformation matrix. This transformation matrix gives the relation between points in the image plan and the pixels in the sampled image.

## 3.4 Camera Calibration

The key idea behind camera calibration is to write the projection equations linking the known coordinates of a set three-dimensional points as well as their projections and solve for the camera parameters. In order to know the coordinates of some three-dimensional points, camera calibration methods rely on one or more images of a calibration pattern. A calibration pattern is a three-dimensional object of known geometry located in a known position in space and generating features that can be located accurately.

Many methods for camera calibrations use a number of images of a planar calibration object to estimate the camera parameters. The images are taken from many different positions as well as orientations, and the calibration estimates both the intrinsic and extrinsic camera parameters. [17]

# Chapter 4

# Experimental Setup

A software program called ‘WinRobot’ was written Tomas Olsson, a doctoral student in the Department of Automatic Control, Lund Institute of Technology, Univeristy of Lund. ‘WinRobot’ was written using a platform in Microsoft Visual C++ and it uses OpenGL to model all the objects in ‘WinRobot’, as found in the robotic laboratory.

‘WinRobot’ is created from a scene graph and the scene graph is assembled from objects to define the geometry, lighting, location, orientation and appearance of visual objects. ‘WinRobot’ provides a free and uninhibited control of the robot workspace, thus enabling various experiments and robotic studies to be carried out.

## 4.1 Robotic Laboratory

The industrial robotic manipulator, ABB IRB-2000 (see Figure 4.1) is found in the robotic laboratory with Open Robot Control architecture developed at the Department of Automatic Control, Lund Institute of Technology, University of Lund. The robot has one built-in controller for each of the 6 joint angles. These controllers are cascaded Proportional, Integral and Derivative (PID) controllers, with an outer position loop around an inner velocity loop. The velocity signal used in the inner loop is obtained by differentiating and low-pass filtering the position signal. MATLAB interfaces for downloading and dynamically linking new control algorithms to the robot system as well as the integration of external sensors such as digital cameras. [18]

In the lab, there are also two Sony DFW-V300 digital cameras (see also Figure 4.1), which utilises the IEEE 1394 high performance serial bus to send non-compressed digital data and allows control functions such as colour tone, brightness, picture quality white balance and automatic gain control

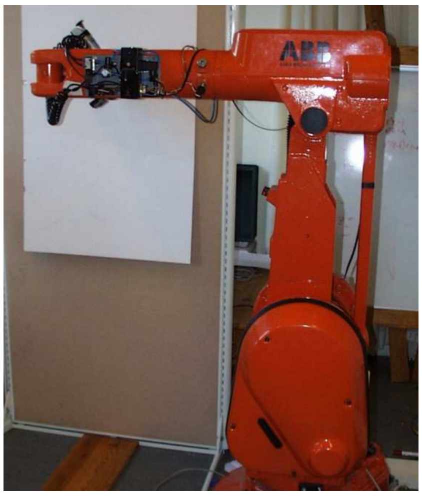


Figure 4.1: ABB IRB-2000 robot and Sony DFW-V300 camera

(AGC). The digital cameras can take about 30 images per second and the images are in the YUV format. Y represents the luminance of the images while U and V represents the colour. The picture format of 640 × 480 pixels (4:1:1) can be modified to 320 × 240 pixels (4:2:2). The images are read by the computer using a IEEE-1394 (FireWire) connection, using the Fire-i API from UniBrain. FireWire is a high-speed, non-propietry, scalable digital serial bus that can transport data rates of 100, 200 and 400 Mbits per second. FireWire enables true plug and play, which allows the user to connect new devices, with the system still switched on and the bus active.

## 4.2 Coordinate System

In robotics, most tasks are specified with respect to a specific coordinate. It is often a need to relate the position and orientation of the objects in the simulated environment to each other.

Each point in the world can be specified by three numbers, the X-coordinate, the Y-coordinate and the Z-coordinate of that point. These three numbers together: $(X, Y, Z)$ is known as a vector and it can specify any point in the three-dimensional space. A vector can also represent a relative movement in three-dimensional space. [19]

Usually the world is perceived as a Euclidean three-dimensional space. Euclidean transformations have six degrees of freedom, three for orientation

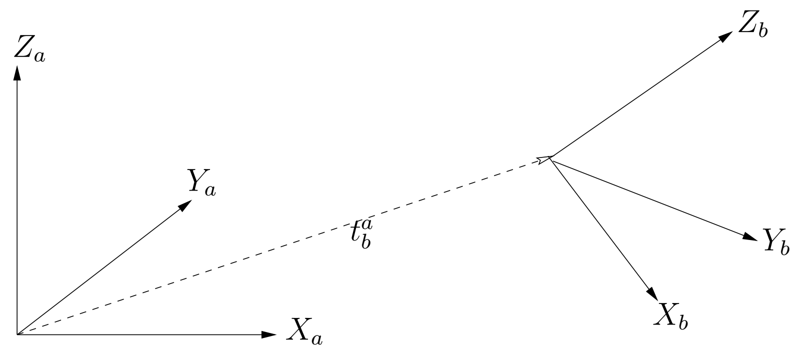


Figure 4.2: Transformation between frames

and three for translation. A Euclidean transformation has the following form:

$$\text{Rt} = \begin{bmatrix} R_{3\times3} & t_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{4.1}$$

Since $R$ is a rotation matrix, i.e. $|R| = 1$, then this transformation represents a rigid motion in space. This also corresponds to the matrix relation that $RR^T = R^T R = I$ and thus $R^{-1} = R^T$.

A pose, $T_b^a$ (see Figure 4.2) is used to change the coordinate system from frame $b$ to frame $a$. $T_b^a$ consists of a $3 \times 1$ vector, $t_b^a$ and a $3 \times 3$ orthogonal matrix $R_b^a$. $t_b^a$ is the vector from the origin of frame $a$ to the origin of frame $b$, expressed in the coordinate system of frame $a$. The column vectors in the matrix $R_b^a$ are the unit x-vector, y-vector and z-vector of the frame $b$, expressed in the coordinate system of frame $a$, where $R_b^a = (X_b^a)(Y_b^a)(Z_b^a)$.

To change the coordinates from frame $b$ to frame $a$, the following equation can be applied:

$$\begin{bmatrix} X^a \\ Y^a \\ Z^a \end{bmatrix} = R_b^a \begin{bmatrix} X^b \\ Y^b \\ Z^b \end{bmatrix} + t_b^a \tag{4.2}$$

The transformation matrix, $T_b^a$ can now be defined as:

$$T_b^a = \begin{bmatrix} R_b^a & t_b^a \\ 0_{1\times3} & 1 \end{bmatrix} \tag{4.3}$$

Due to the special structure of the matrix $T_b^a$, the inverse can be easily calculated as:

$$(T_b^a)^{-1} = \begin{bmatrix} (R_b^a)^T & -(R_b^a)^T t_b^a \\ 0_{1\times3} & 1 \end{bmatrix} \tag{4.4}$$

## 4.3 WinRobot

'WinRobot' consists of an industrial robot, Virtual IRB-2000 and a pair of virtual digital cameras.

### 4.3.1 Lighting

Lighting had to be first created in 'WinRobot' so that all the three-dimensional objects can be seen.

The lighting model incorporates three kinds of real world lighting reflection: ambient, diffuse and specular. Ambient reflection results from constant low level light in a scene, diffuse reflection is the normal reflection of a light source from a visual object while specular reflection is the highlight of a light source from an object. The lighting model is applied for each of the RGB colour components.

The array for lighting of room was defined in 'WinRobot' as:

GLfloat LightDiffuse = (1, 1, 1, 1)

GLfloat LightSpecular = (1, 1, 1, 1)

GLfloat LightAmbient = (0.5, 0.5, 0.5, 1)

The lighting was placed in three different coordinates of: $(0, 0, 4)$, $(5, -5, 4)$ and $(-5, 5, 4)$.

### 4.3.2 Virtual IRB-2000

The Virtual IRB-2000 (see Figure 4.3) is built by two large arms and a wrist. The virtual robot has six degrees of freedom, which means the end-effector of the robotic arm can be moved to any desired position and orientation within a task space. Joint 1, joint 4 and joint 6 of the Virtual IRB-2000 are cylindrical joints while joint 2, joint 3 and joint 5 of the Virtual IRB-2000 are revolute joints. The robot is controlled by using MATLAB to send the desired position of the robotic arm through the network.

The Virtual IRB-2000 is used to test algorithms for image processing and model-based control. Using Virtual IRB-2000, it is possible to avoid collisions and implementing bad trajectories in the robotic laboratory. It is a fast and safe benchmark test for the system.

The Virtual IRB-2000 was rotated 25 degrees from the negative x-axis towards the negative y-axis and the robot was placed 4 meters away from

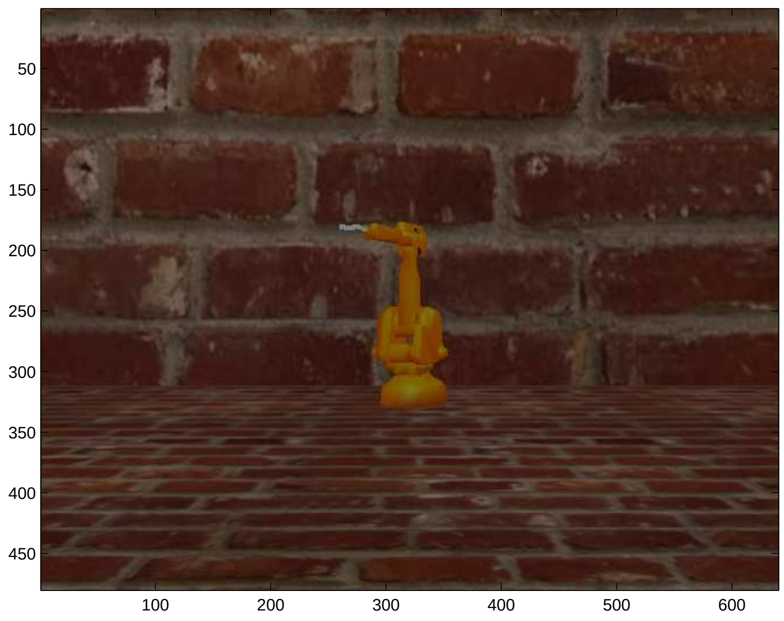


Figure 4.3: Virtual IRB-2000

the origin on the negative x-axis. The position of the Virtual IRB-2000 specified in three-dimensional Euclidean space is:

$$\mathrm{Rt}_{robot} = \begin{bmatrix} 0.9064 & 0.4233 & 0 & -4 \\ -0.4233 & 0.9064 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

A gripper is attached to the end-effector of the robotic arm and its coordinate system is relative to the coordinate system of the end-effector of the robotic arm.

### 4.3.3 Virtual Ball

A ball moving with Newtonian dynamics was needed in this experiment, so a three-dimensional virtual ball was designed using OpenGL. The design was written using Extensive Markup Language (XML). XML is a markup language for documents containing structured information. [20]

OpenGL only allows creation of two-dimensional triangles and quadrangles by specifying the coordinates for the vertex points of the triangles and quadrangles. The virtual ball was designed to be made up of 26 surfaces, of which 18 of the surfaces are squares and 8 of the surfaces are triangles. Since

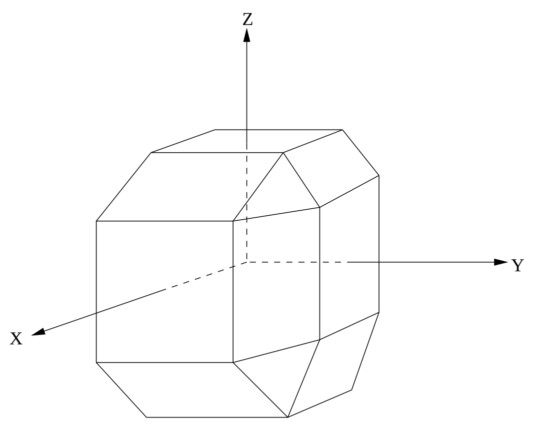


Figure 4.4: The virtual ball

the design of the virtual ball is to mimic a tennis ball in the real world, a diameter of 6 cm was set for the virtual ball (see Figure 4.4). In addition to that, the shading of the virtual ball were set to smooth so that it is more realistic. The virtual ball was set to be green in colour, just like a tennis ball in the real world.

The virtual ball's colour is very important for image processing in the simplified virtual environment. Colours are sensed as non-linear combination of long, medium and short wavelengths, which correspond to the three primary colours that are used in the camera system: Red(R), Green(G) and Blue(B). Therefore, for each pixel of image information, there is a matrix of colours (R, G, B).

The initial position of the virtual ball was placed 1 metre away from the origin on the z-axis and the virtual ball was not orientated. The initial position of the virtual ball specified in three-dimensional Euclidean space is:

$$\mathrm{Rt}_{ball} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

The virtual ball's trajectory was modeled with three-dimensional Newtonian dynamics (see Figure 4.5) and the dynamic equations of the virtual ball in the x-coordinate, y-coordinate as well as z-coordinate can be seen below:

$$x = x_0 - v \times \cos\theta \times \sin\phi \times t \tag{4.7}$$

$$y = y_0 - v \times \cos\theta \times \cos\phi \times t \tag{4.8}$$

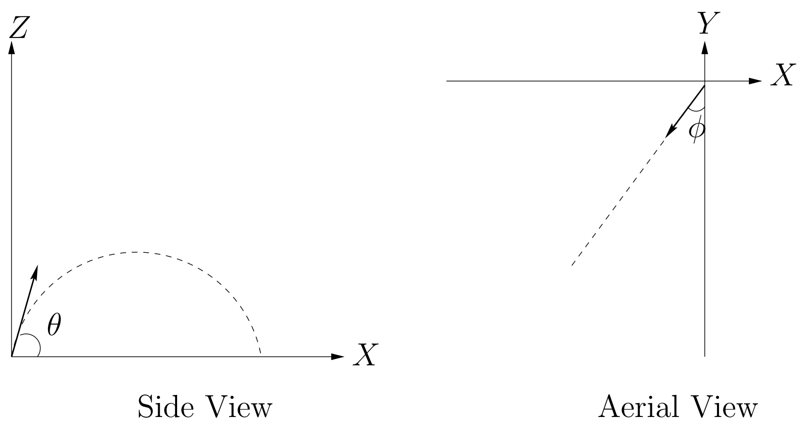


Figure 4.5: Aeriel and side view of the virtual ball's trajectory

$$z \quad = z_0 + v \times \sin\theta \times t - \frac{1}{2} \times g \times t^2 \tag{4.9}$$

where

$v$ = initial velocity of 7.5ms$^{-1}$

$\theta = 30°$

$\phi = \tan^{-1}(4)$

$t$ = time

$(x_0, y_0, z_0)$ = coordinate of $(0, 0, 1)$ for the initial position of the virtual ball

$g$ = gravitational constant of 9.82ms$^{-2}$

### 4.3.4 Virtual Digital Cameras

The two virtual digital cameras were labeled Camera 2 and Camera 3. Camera 1 was another virtual camera being used to observe the experimental environment during the simulation process. Both Camera 2 and Camera 3 were placed 1 meter away from the origin on the x-coordinate and 1.1 meter away from the origin on the z-coordinate. The difference between Camera 2 and Camera 3 is that they are placed -0.5 meter and 0.5 meter away from the origin on the y-coordinate respectively.

Camera 2 and Camera 3 were orientated as well as having their focal length readjusted so that the full trajectory of the ball could be captured by Camera 2 and Camera 3 simultaneously. The position of Camera 2 and Camera 3 in Euclidean form are shown below:

$$\mathrm{Rt}_{cam2} = \begin{bmatrix} 0.2065 & 0 & 0.9785 & 1 \\ 0.9785 & 0 & -0.2065 & -0.5 \\ 0 & 1 & 0 & 1.1 \end{bmatrix} \tag{4.10}$$

$$\mathrm{Rt}_{cam3} = \begin{bmatrix} 0.2065 & 0 & 0.9785 & 1 \\ 0.9785 & 0 & -0.2065 & 0.5 \\ 0 & 1 & 0 & 1.1 \end{bmatrix} \tag{4.11}$$

Camera 2 and Camera 3 did not require any calibration in the simulated environment. The intrinsic parameters of Camera 2 and Camera 3 after having their focal length readjusted are shown below:

$$\mathrm{K}_{cam2} = \begin{bmatrix} 456.2578 & 0 & 175.5991 \\ 0 & 485.4411 & 113.8451 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.12}$$

$$\mathrm{K}_{cam3} = \begin{bmatrix} 629.6043 & 0 & 194.6277 \\ 0 & 657.8963 & 152.6292 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.13}$$

# Chapter 5

# System Overview

## 5.1 MatComm

The MATLAB software program was found in the UNIX operating system, while Microsoft Visual C++ was found in the Windows operating system. This procedure was adopted because it mimics the same setup found in the robotic laboratory. MatComm reconciles the difference between the two operating system so that communication can be established beween the two different software programs based in two different platforms (see Figure 5.1).

MatComm is a software package that was developed at the Department of Automatic Control, Lund Institute of Technology, Univeristy of Lund. This software uses the Transmission Control Protocol (TCP) / Internet Protocol

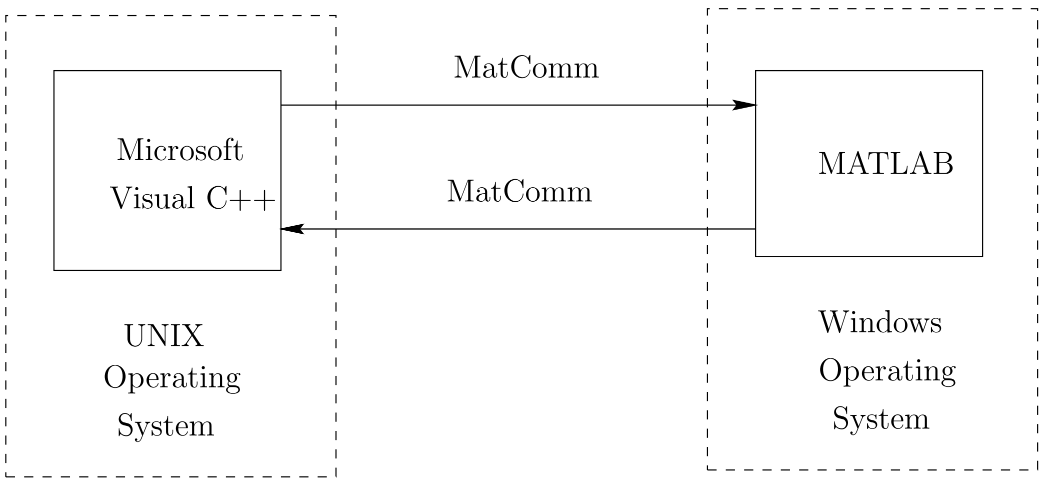


Figure 5.1: Communciation and control system of the virtual robot

(IP) to transmit data between computers on the network where different operating system might exist. Data are sent through MatComm in an array format using Berkeley sockets. MatComm can be used in both the UNIX and Windows platform. The reason that MatComm was used is because it is a fast and easy way to connect between computers without the necessity of setting all the parameters.

## 5.2 Overall System

The overall system can be seen in Figure 5.2.

The system is a closed-loop system that begins with the virtual digital cameras. The virtual digital cameras are used as visual sensors in the virtual environment. Images captured by the vitual digital cameras are sent from Microsoft Visual C++ to MATLAB using MatComm. Feature point extraction is applied on the images to determine the center of the virtual ball, $(u_0 , v_0)$ with reference to the two-dimensional images. Time stamped data is needed to calculate the exact time for the position of the virtual ball. After the images are processed, the coordinates of the virtual ball are then saved into an array that hold the time instances when the images were received.

$(u_0 , v_0)$ obtained from both the images are then used to reconstruct the three-dimensional coordinate of the virtual ball, $(X_k , Y_k , Z_k)$. The method used for three-dimensional reconstruction from two two-dimensional images is Triangulation. After six iterations of the process described, enough sample data are collected from Triangulation to start identifying the trajectory of the virtual ball. In the process of system identification, the unknown parameters are computed using linear regression. The calculated unknown parameters are then used to predict the virtual ball grasping position in the three-dimensional coordinate, $(X_p , Y_p , Z_p)$. Virtual ball grasping position prediction estimates the best position for the robotic arm to align itself to grasp the virtual ball. $(X_p , Y_p , Z_p)$ is then sent for inverse kinematics to convert the three-dimensional coordinate into robot joint coordinates. The robot joint coordinates are then sent back from MATLAB to Microsoft Visual C++ using MatComm.

Thus the joint angles of the six robot joints can be controlled to move the gripper attached to the end-effector of the virtual robot to the predicted virtual ball grasping position. The number of samples for the virtual ball to travel from its original position to the region of the predicted virtual ball grasping position was about 17 samples. Thus the process mentioned above was decided to be repeated for 20 samples.

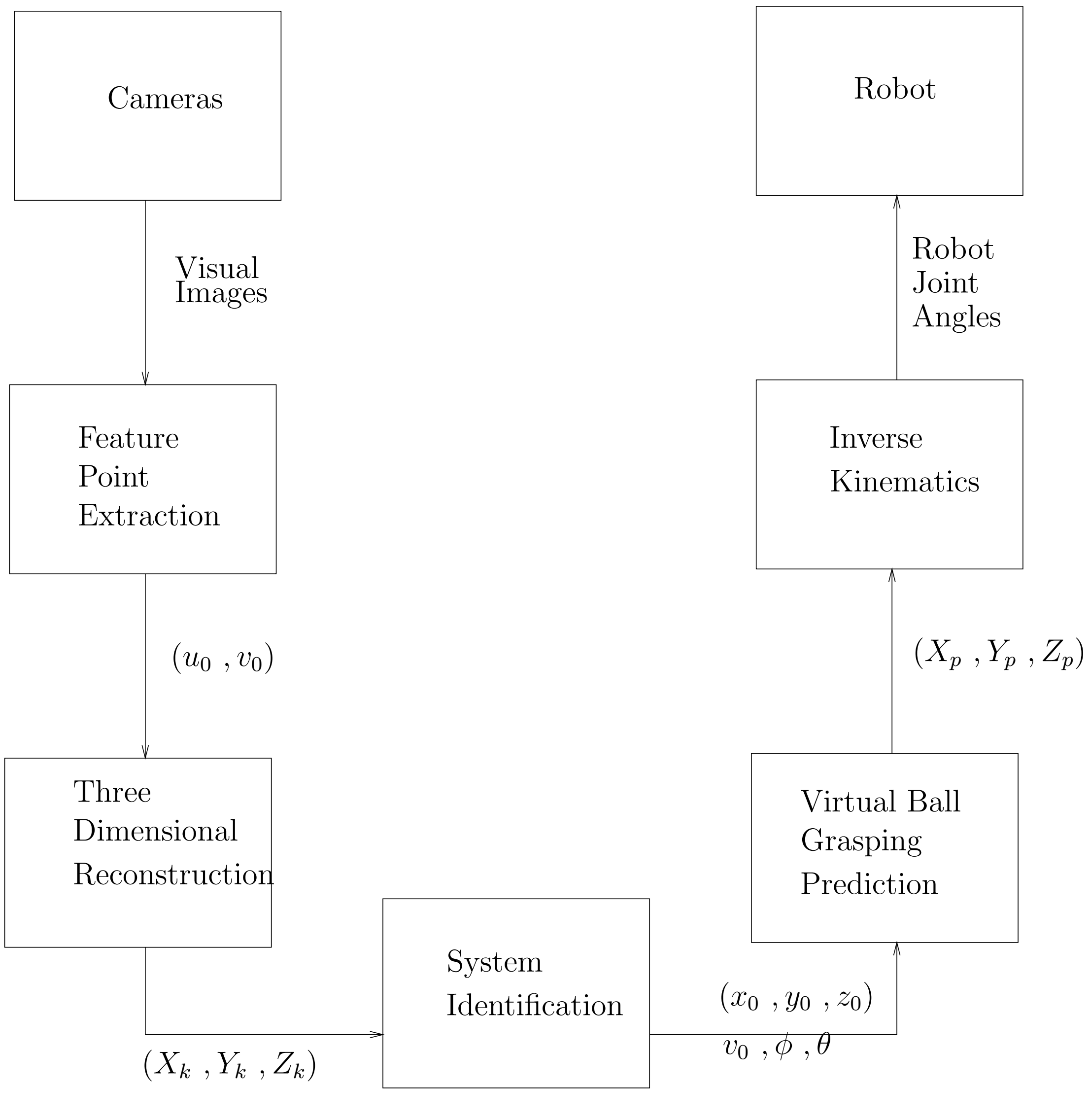


Figure 5.2: Overall System

# Chapter 6

# Image Processing

The sequence of operations for most computer vision system begins by detecting and locating some features in the input images. Image features can be edges, points, corners, surfaces, lines or curves.

## 6.1 Image Features

Image features are local, meaningful, detectable parts of the image. Meaningful means that the features are associated to interesting scene elements via the image formation process. Typical examples of meaningful features are sharp intensity variations created by the contours of the objects in the scene, or image regions with uniform grey level, for instance image of planar surfaces. Detectable means that location algorithms must exist, otherwise a particular feature is of no use. Different features are associated to different detection algorithms; these algorithms output collections of feature descriptors, which specify the position and other essential properties of the features found in the image.

Edge points are pixels at or around the image values undergoing a sharp variation. There are various reasons for interest in edges. The contours of potentially interesting scene elements such as solid objects, marks on surfaces and shadows all generate intensity edges. Morever, image lines, curves and contours are detected from chains of edge points. Finally, line drawings are common and suggestive images for humans.

In this project, feature extraction is an intermediate step, not the goal of the system. Features are not extracted just to obtain feature representation, but to locate the position of the virtual ball with reference to the visual images.

## 6.2 Images

The image constitution follows a three-colour model and its respective RGB components can be retrieved from the image buffer. MatComm is used to send the image buffer to MATLAB. The loop starts in the Microsoft Visual C++ and it sends image information from Camera 2 and Camera 3 all the time. Digital images obtained from virtual digital cameras are two-dimensional array of numbers and they were sent in the $320 \times 240$ picture format. The image information consists of the position of the virtual ball with reference to the visual images.

The two virtual digital cameras are synchronised to operate at 33.33 Hertz, just like the capability of the digital cameras in the robotic laboratory. This means the visual information obtained from both the digital cameras are about 30 milliseconds per sample. After the images are processed and the desired coordinates of the virtual ball extracted, the coordinates are then saved into an array that holds the time instances when the images were received.

## 6.3 Feature Point Extraction with Thresholding

Feature point extraction of the image of the virtual ball is done using thresholding, which can be thought as an extreme form of gray-level quantisation. Thresholding is the most common method applied to images that can be characterised as having bimodal histograms. These histograms are often associated with images that can contain objects and backgrounds having a significantly different average brightness. The goal is to separate the objects from the background, in this case the virtual ball from its background. The virtual ball is assigned a binary number ‘1’ while the background is assigned a binary number ‘0’.

The image constitution follows a three-colour model and its respective RGB components can be retrieved from the image buffer. The algorithm of detection programmed in MATLAB scans each pixel from the image information of Camera 2 and Camera 3 and segmentation is done for pixels with only pure green colour, which indicates the image of the virtual ball. The threshold limit for segmenting the image of the virtual ball from its background is:

$$\text{Ball}(u,v) = \begin{cases} 1 & \text{if } R(u,v) = 0, G(u,v) = 127 \text{ and } B(u,v) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.1}$$

The threshold limit was set to such a high level for segmenting only pure green colour for the virtual ball because the colour of the virtual ball was made to be irrelevant to the lighting in the simulated environment.

The position of the center of the virtual ball with reference to the visual image obtained from the digital cameras are calculated as shown below:

$$(u_0, v_0) = (\frac{\sum u_1}{\sum \text{Ball}(u) = `1'} \, , \, \frac{\sum v_1}{\sum \text{Ball}(v) = `1'}) \tag{6.2}$$

where $u_1$ and $v_1$ are the u and v for when Ball($u$ , $v$) is equivalent to binary '1'.

# Chapter 7

# Three-dimensional Reconstruction

This section of work falls under the responsibility of my partner, Woon Teck Her.

In humans, the three-dimensional perception of the world is due to the interpretation that the brain gives of the computed differences in the retinal position between items, also called disparity. The disparities of all the image points form the disparity map, which can be displayed as an image. Given a number of corresponding parts of both the images, it is possible to obtain the three-dimensinal location and structure of the observed objects.

The disparity measures the difference in position between corresponding points of two images. Depth is inversely proportional to disparity as moving objects further away seem to move slower than the nearer ones.

The three-dimensional reconstruction that can be obtained depends on the amount of priori knowledge available on the the parameters of the system. Since the intrinsic and extrinsic parameters are known for the stereo rig configuration in this experiment, three-dimensional reconstruction becomes straightforward and reconstruction by triangulation is the simplest case.

## 7.1 Stereo Rig Configuration

The geometry of stereo rig configuration, known as epipolar geometry is shown in Figure 7.1.

The figure shows two pinhole cameras, their projection centers $O_L$ and $O_R$ as well as their image planes, $\pi_L$ and $\pi_R$. Each camera identifies a three-dimensional reference frame, the origin that coincides with the projection centers and and the z-axis aligned with the optical axes. The vectors $P_L =$

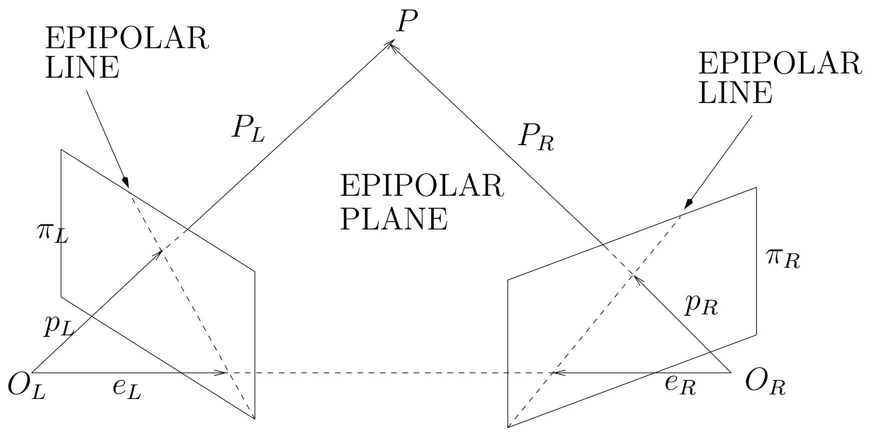


Figure 7.1: The epipolar geometry

$(X_L\ ,Y_L\ ,Z_L)$ and $P_R = (X_R\ ,Y_R\ ,Z_R)$ refer to the same three-dimensional point, $P$. $P$ is thought of as a vector in the left and right camera reference frames respectively. The vectors $p_L = (x_L\ ,y_L\ ,z_L)$ and $p_R = (x_R\ ,y_R\ ,z_R)$ refer to the projections of $P$ onto the left and right image planes respectively.

The name epipolar geometry is used because this points at which the line through the centers of projection intersects the image planes are called epipoles. The epipoles are denoted by $e_L$ and $e_R$. The plane identified by $P$, $O_L$ and $O_R$, is called a epipolar plane while the lines where the epipolar plane intersects the image planes are called conjugated epipolar lines. The practical importance of epipolar geometry stems from the fact that an epipolar plane intersects conjugated epipolar lines.

The important fact of epipolar constraint is that corresponding points must lie on conjugated epipolar lines. Thus it establishes a mapping between points in the left image and lines in the right image as well as points in the right image and lines in the left image. Since all rays include the projection center by construction, thus all epipolar lines go through the epipole.

## 7.2 Triangulation

Triangulation is the method used is this project for reconstructing a three-dimensional image from two two-dimensional images.

As two images are used for triangulation, two projective equations are used for Camera 2 and Camera 3 respectively and they as follow:

$$\lambda_A \begin{bmatrix} u_{0A} \\ v_{0A} \\ 1 \end{bmatrix} = H_{A3\times4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \text{for Camera 2} \tag{7.1}$$

$$\lambda_B \begin{bmatrix} u_{0B} \\ v_{0B} \\ 1 \end{bmatrix} = H_{B3\times4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \text{for Camera 3} \tag{7.2}$$

By combining and solving the equations we can obtain:

$$\begin{bmatrix} \lambda_A u_{0A} \\ \lambda_A v_{0A} \\ \lambda_A \\ \lambda_A u_{0A} \\ \lambda_A v_{0A} \\ \lambda_A \end{bmatrix} = \begin{bmatrix} H_{A11}X + H_{A12}Y + H_{A13}Z + H_{A14} \\ H_{A21}X + H_{A22}Y + H_{A23}Z + H_{A24} \\ H_{A31}X + H_{A32}Y + H_{A33}Z + H_{A34} \\ H_{A11}X + H_{B12}Y + H_{B13}Z + H_{B14} \\ H_{B21}X + H_{B22}Y + H_{B23}Z + H_{B24} \\ H_{B31}X + H_{B32}Y + H_{B33}Z + H_{B34} \end{bmatrix} \tag{7.3}$$

With the six equations, the they can be manipulated to become the homogeneous form of:

$$\underbrace{\begin{bmatrix} -H_{A11} & -H_{A12} & -H_{A13} & u_{0A} & 0 \\ -H_{A21} & -H_{A22} & -H_{A23} & v_{0A} & 0 \\ -H_{A31} & -H_{A32} & -H_{A33} & 1 & 0 \\ -H_{B11} & -H_{B12} & -H_{B13} & 0 & u_{0B} \\ -H_{B21} & -H_{B22} & -H_{B23} & 0 & v_{0B} \\ -H_{B31} & -H_{B32} & -H_{B33} & 0 & 1 \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ \lambda_A \\ \lambda_B \end{bmatrix}}_{D} = \underbrace{\begin{bmatrix} H_{A14} \\ H_{A24} \\ H_{A34} \\ H_{B14} \\ H_{B24} \\ H_{B34} \end{bmatrix}}_{E} \tag{7.4}$$

Since the matrix $H$ is known and the center point of the virtual ball $(u_0, v_0)$ can be obtained after image processing has been applied onto the visual images obtained from Camera 2 and Camera 3, thus we can obtain $D$ with the following equation:

$$D = C^{\dagger}E \tag{7.5}$$

where $C^{\dagger}$ is the pseudoinverse of C.

This renders a least square solution for $D$ as $C$ is not a square matrix.

# Chapter 8

# Model-Based Ball Grasping

This section of work falls under the responsibility of my partner, Woon Teck Her.

## 8.1 Newtonian Dynamic Equations

The Newtonian dynamic equations for the virtual are non-linear. The unknown parameters to be identified include: $v, x_0, y_0, z_0, \theta$ and $\phi$.

Let

$$\alpha = \cos\phi \tag{8.1}$$

$$\beta = \cos\theta \tag{8.2}$$

Using the trigonometric identity of: $\sin^2\psi + \cos^2\psi = 1$, the Newtonian dynamic equations can be modified to become:

$$X_k = x_0 - v\beta(\sqrt{1-\alpha^2})t_k \tag{8.3}$$

$$Y_k = y_0 - v\beta\alpha t_k \tag{8.4}$$

$$Z_k = z_0 + v(\sqrt{1-\beta^2})t_k - \frac{1}{2}gt_k^2 \tag{8.5}$$

where $X_k, Y_k, Z_k, t_k$ and $g$ are known parameters.

The method used to identify the Newtonian dynamic equations is Linear Regression. [21]

## 8.2 Linear Regression

The equation of linear regression is:

$$y_N = \Phi_N\vartheta + e \tag{8.6}$$

where

N = total number of samples

$y_N = [y_1 \ , y_2 ... y_N]^T$ is the vector of observations

$\Phi_N = [\varphi_1 \ , \varphi_2 ... \varphi_N]^T$ is the regression vector

$\vartheta$ = matrix with unknwon parameters to be identified

$e = [e_1 \ , e_2 ... e_N]^T$ is the error

Interpreting the Newtonian dynamic equation in the form of Linear Regression gives us:

$$\underbrace{\begin{bmatrix} X_k & Y_k & Z_k + \frac{1}{2} g t_k^2 \end{bmatrix}}_{y_N} = \underbrace{\begin{bmatrix} 1 & t_k \end{bmatrix}}_{\Phi_N} \underbrace{\begin{bmatrix} x_0 & y_0 & z_0 \\ -v\beta\sqrt{1-\alpha^2} & -v\beta\alpha & v\sqrt{1-\beta^2} \end{bmatrix}}_{\vartheta} \tag{8.7}$$

The unbiased estimate of $\vartheta$ is calculated using the following equation:

$$\widehat{\vartheta} = \Phi^{\dagger} y_N \tag{8.8}$$

where $\Phi^{\dagger} = (\Phi_N^T \Phi_N)^{-1} \Phi_N^T$ is the pseudoinverse of $\Phi$.

Since $\vartheta$ is a $2 \times 3$ matrix, a minimum of six samples are needed before the Newtonian dynamic equations can be identified. The unknown parameters $x_0, y_0$ and $z_0$ are easy to be identified as they can be obtained straight from the matrix $\vartheta$. The other unknown parameters involve some mathematical manipulation.

Let

$$I = -v\beta\sqrt{1-\alpha^2} \tag{8.9}$$
$$J = -v\beta\alpha \tag{8.10}$$
$$K = v\sqrt{1-\beta^2} \tag{8.11}$$

Solving the equations simultaneously, we obtain:

$$v = I^2 + J^2 + K^2 \tag{8.12}$$
$$\theta = \cos^{-1}\left(\sqrt{\frac{I^2 + J^2}{I^2 + J^2 + K^2}}\right) \tag{8.13}$$
$$\phi = \cos^{-1}\left(\sqrt{\frac{J^2}{I^2 + J^2}}\right) \tag{8.14}$$

## 8.3 Virtual Ball Grasping Position Prediction

The virtual ball grasping position was set as 1 meter in front of the robot, which corresponds to 3 meters away from the origin on the x-coordinate. A minimum of six samples were needed before the virtual ball grasping position can be predicted. The six samples are needed to identify the six unknwon parameters of the Newtonian dynamic equations. With the x-coordinate fixed, the time, y-coordinate and z-coordinate of the predicted virtual ball grasping position can be easily calculated as shown below:

$$X_p = -3 \tag{8.15}$$

$$t_p = \frac{X_p - x_0}{-v \cos\theta \sin\phi} \tag{8.16}$$

$$Y_p = y_0 - v \cos\theta \cos\phi t_p \tag{8.17}$$

$$Z_p = z_0 + v \sin\theta t_p - \frac{1}{2} g t_p^2 \tag{8.18}$$

The virtual ball grasping position prediction was done for 20 samples, 15 samples in total if the first 5 samples were excluded.

## 8.4 Inverse Kinematics

There are two kinds of kinematics: one is called forward kinematics and the other is called inverse kinematics. The forward kinematics calculate the position and orientation of the robot end-effector with the input of the robot joint angles while the inverse kinematics calculate the robot joint angles with the input of the robot end-effector position and orientation. [21]

In order to move the gripper attached to the end-effector of the virtual robot to the predicted virtual ball grasping position, the inverse kinematics is used. A MATLAB program called **invkin2400.m** was used for doing the inverse kinematics calculation. This program was written by Anders Robertsson from the Department of Automatic Control, Lund Institute of Technology, University of Lund. The function for the inverse kinematics is:

$q$ = invkin2400(robot_coordinate,1,1,gripper_length)

Before doing the inverse kinematics, the predicted virtual ball grasping position has to be converted to be relative to the robot's coordinate system. The three-dimensional coordinates of the predicted virtual ball grasping position too has to be converted into millimeters. The conversion can be seen below:

$$\text{robot\_coordinate} = (\text{Rt}_{robot})^{-1} \begin{bmatrix} 0 & 0 & 1 & X_p \times 1000 \\ 0 & 1 & 0 & Y_p \times 1000 \\ -1 & 0 & 0 & Z_p \times 1000 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8.19}$$

The measurement for gripper_length was to be specified in milimeters and it was set to 350 milimeters. The output of the function, q were six joint angles of the virtual robot in radians and they have to be converted to degrees before being sent to the virtual robot via MatComm. The conversion can be seen below:

$$q(\text{degrees}) = q(\text{radians}) \times \frac{180^\circ}{\pi} \tag{8.20}$$

# Chapter 9

# Results

Figure 9.1 shows the experimental setup in the simulated virtual environment. The experimental setup closely mimics the robotic laboratory environment in terms of room dimension.

The top picture in Figure 9.2 shows the image obatined from Camera 2 when the virtual ball is placed at its initial position, also the image obtained from Camera 2 for the first sample. The bottom picture in Figure 9.2 displays the result after feature point extraction with thresholding has been applied to the image. The image of the of the ball was assigned a binary number '1', which renders the image of the virtual ball to be white while the background was assigned a binary number '0', which renders the background to be black. It can be observed that the image of the virtual ball after feature point extraction with thresholding is at the exactly the same position as the image of the virtual ball initially obtained from Camera 2. Thus the algorithm for feature point extraction with thresholding was successful.

This was done for Camera 3 as well and the whole process was repeated for 20 samples. Figure 9.3 shows the graphs of two-dimensional images of the center of the virtual ball captured by Camera 2 and Camera 3. From both the graphs obtained, we can observe the whole trajectory of the virtual ball from its initial position to the last (twentieth) sample. This indicates that that the placement of the cameras were good. These two images were used for three-dimensional reconstruction.

Figure 9.4 shows the parameters identified for the Newtonian dynamic equations. The blue line indicates the parameters identified for different sampled data while the red line indicates the actual parameters of the Newtonian dynamic equations. It can be observed that the identified parameters are very close to the actual parameters. Thus the identification on the Newtonian dynamic equations was good.

Figure 9.5 shows the the three-dimensional reconstruction of the position

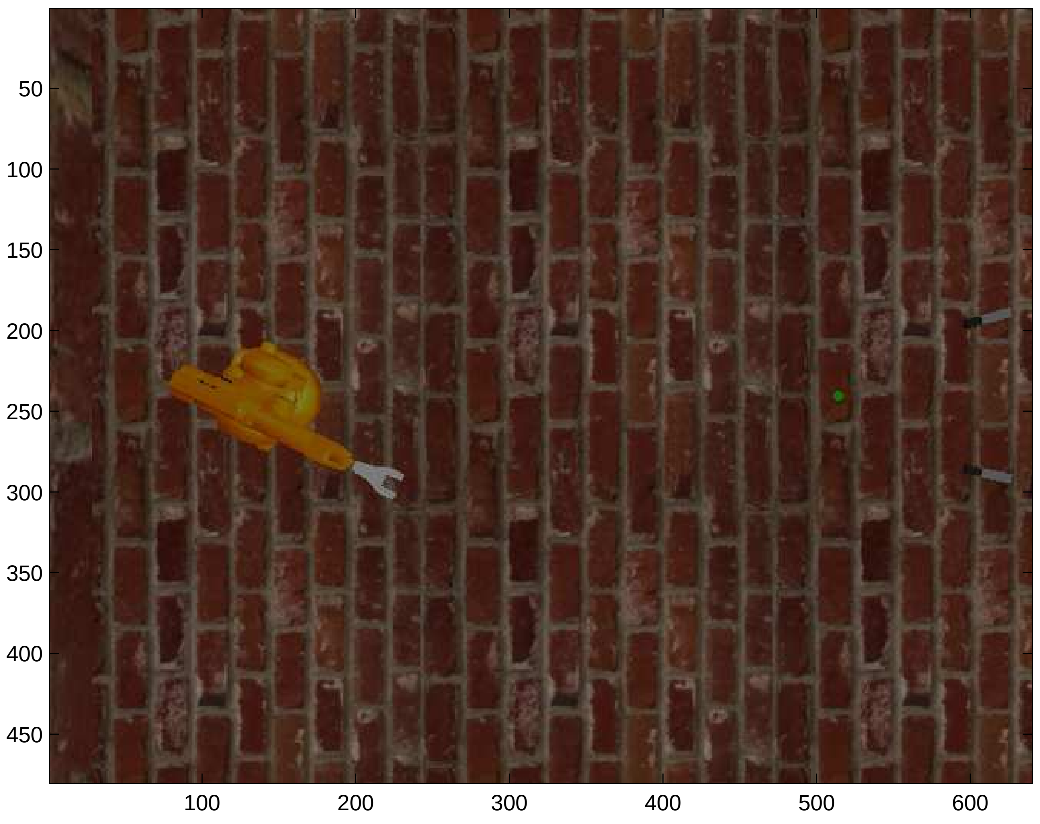


Figure 9.1: Aerial view of the experimental Setup

and trajectory of the center of the virtual ball. The reconstructed three-dimensional trajectory of the virtual ball was compared to the actual trajectory of the virtual ball and the error was found to be very small. Thus the three-dimensional reconstruction of the trajectory of the vitual ball was successful.

Figure 9.6 shows the predicted virtual ball grasping positions with the Newtonian dynamic equations identified for different sampled data.The first predicted virtual ball grasping point (the sixth sample) is denoted by a red '*'. The green dotted lines traces up to the last predicted virtual ball grasping point. The actual virtual ball grasping position is denoted by a red '*.

Figure 9.7 shows the computed errors between the predicted virtual ball grasping position for different sampled data and the actuall virtual ball grasping position. It can be observed that the error is structured and the magnitude of the error is very small. Thus the predicted virtual ball grasping position is good.

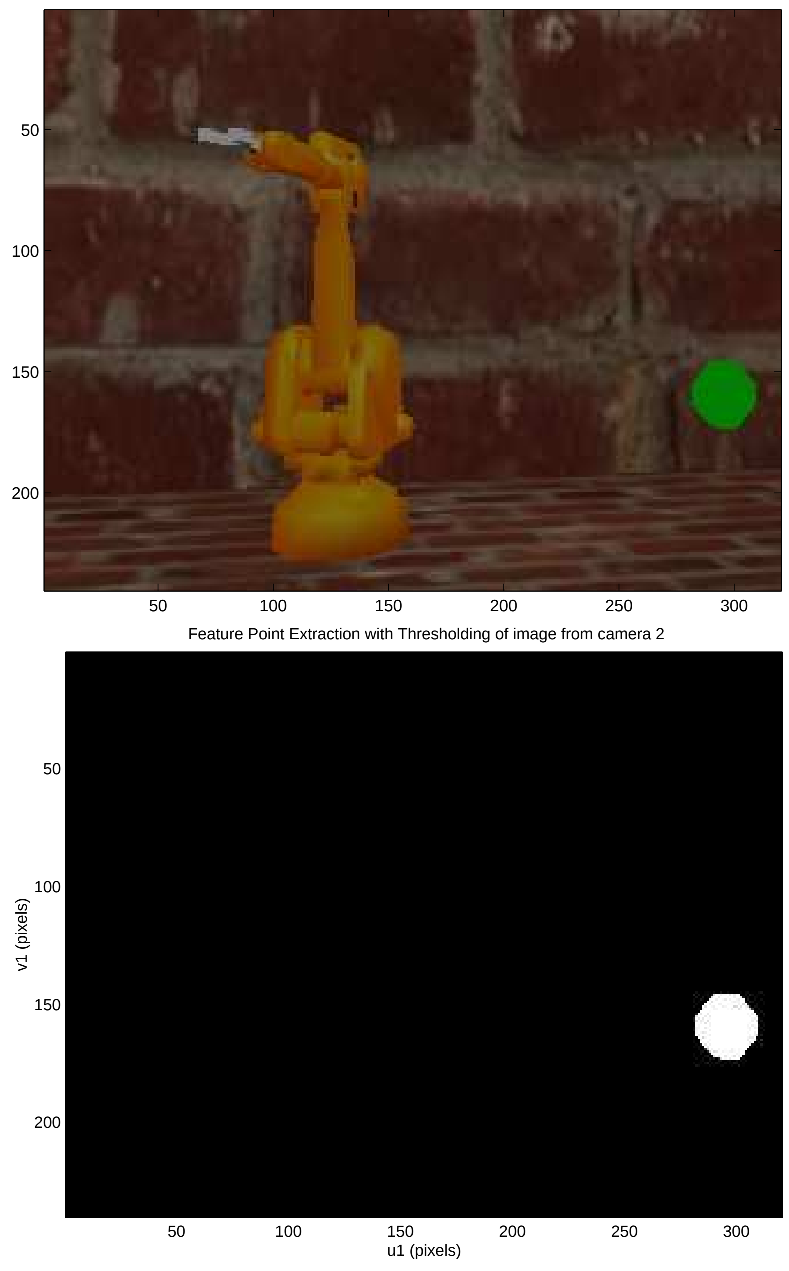


Figure 9.2: Initial image of Camera 2 after feature point extraction with thresholding

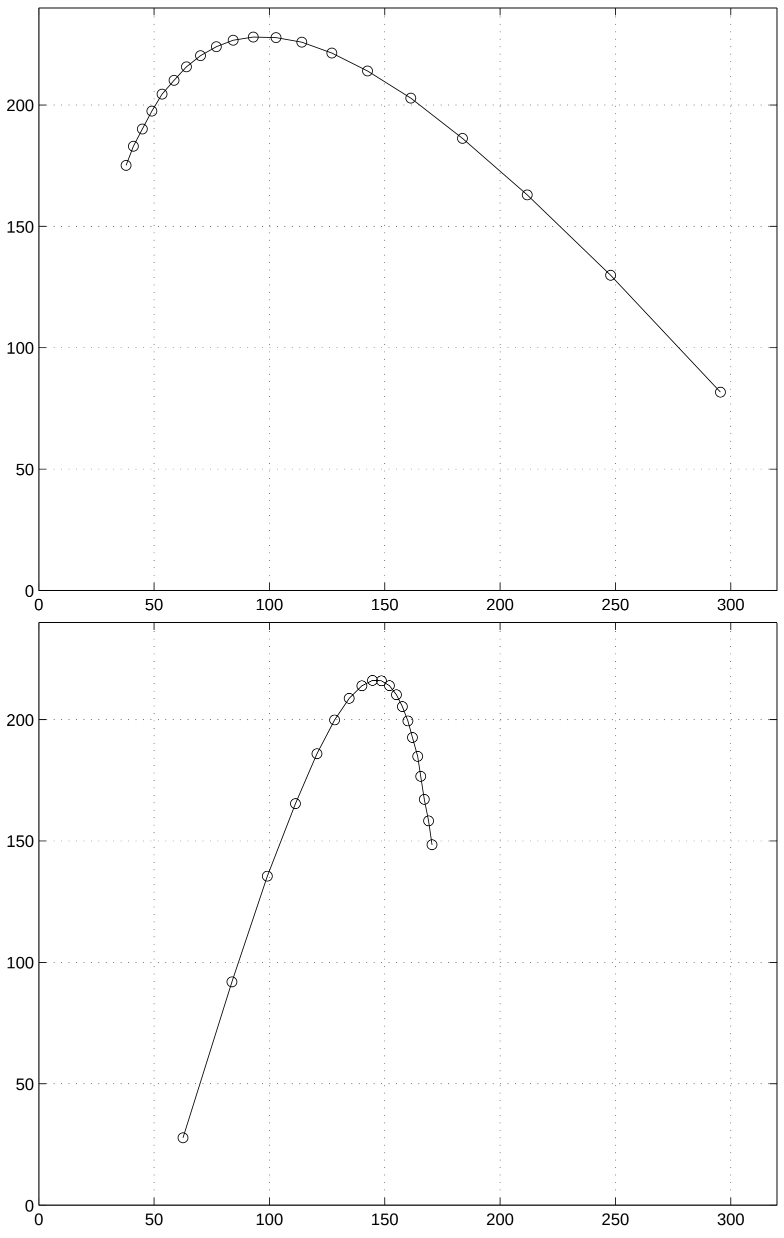


Figure 9.3: Trajectory tracking of the center of the virtual ball captured by Camera 2 and Camera 3

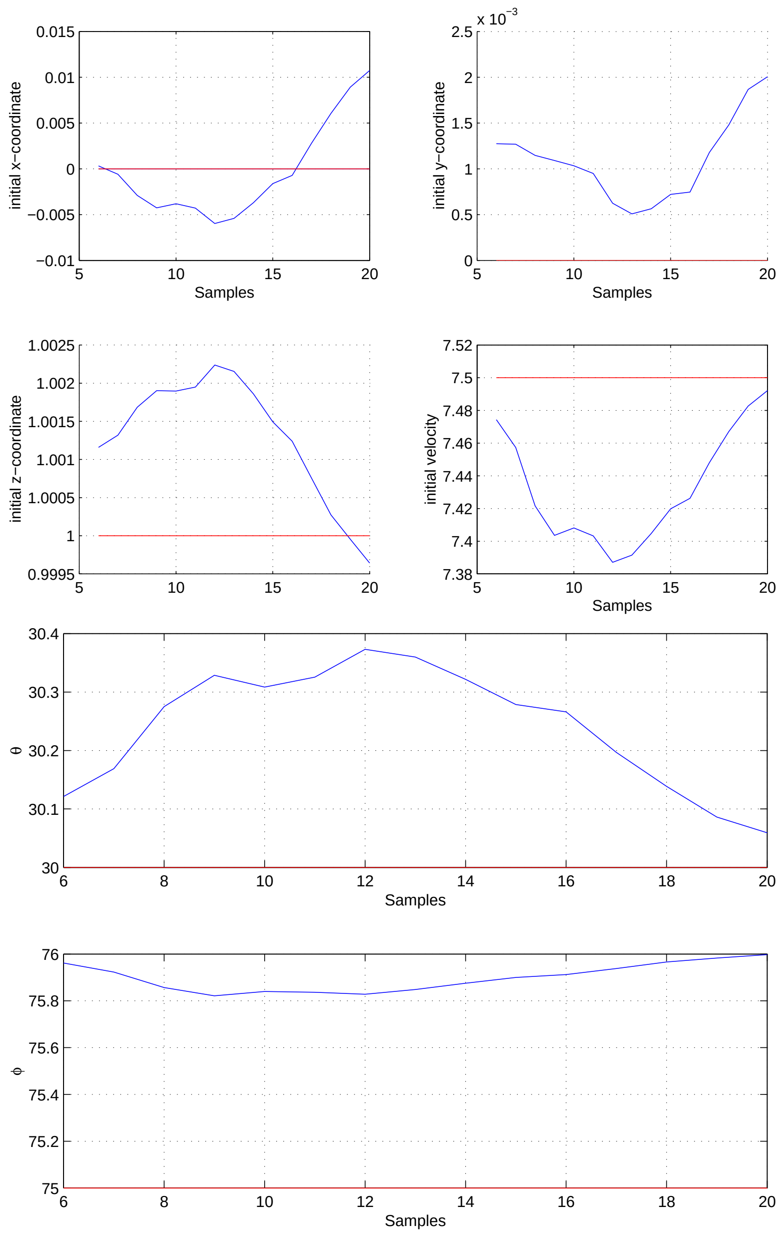


Figure 9.4: Identification of the parameters for the Newtonian dynamic equations

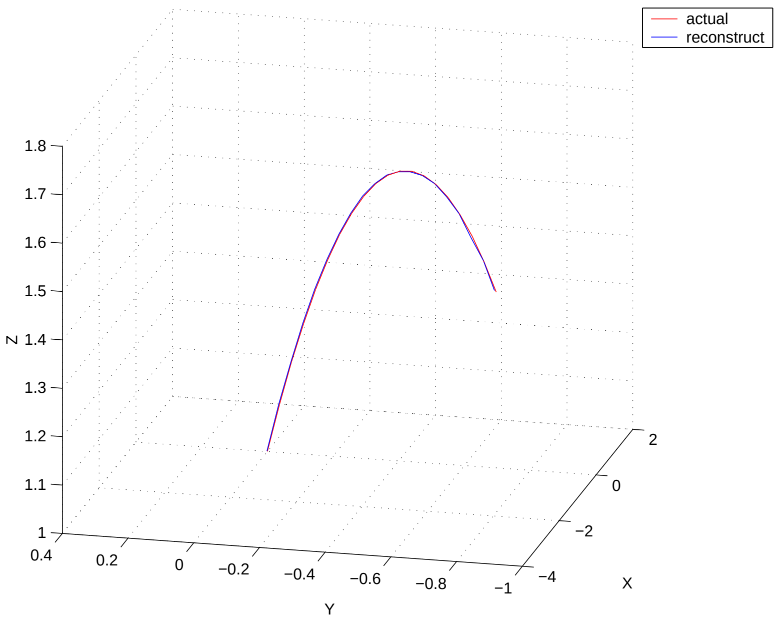


Figure 9.5: Three-dimensional reconstruction of the trajectory of the virtual ball

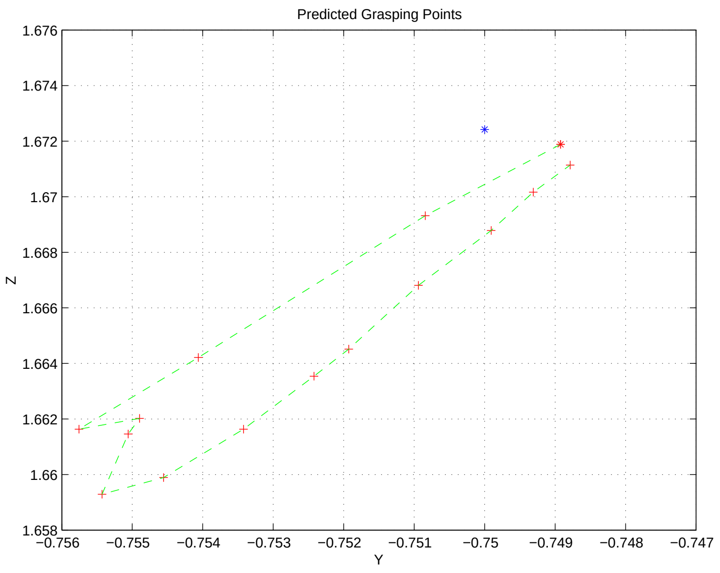


Figure 9.6: Predicted virtual ball grasping positions

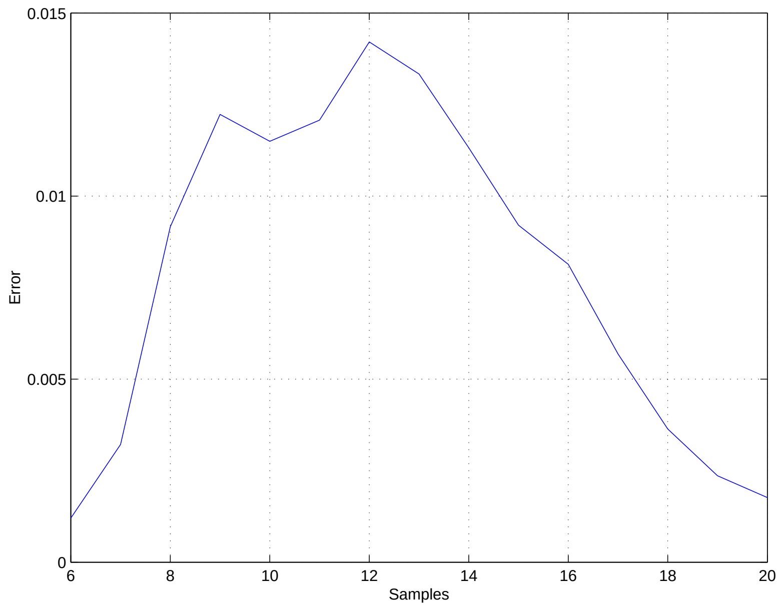


Figure 9.7: Error between predicted virtual ball grasping positions and the actual ball grasping position

# Chapter 10

# Discussion

When two different software programs are used to communicate with each other, discrepancies are bound to exist between them as they are written on different platforms and for differing applications. These discrepancies require solutions to overcome them in order that the two software programs can communicate effectively. In this project, three major discrepancies were found between Microsoft Visual C++ and MATLAB.

The first discrepancy was the different definition of 'char' in both the software programs. Feature point extraction with thresholding was initially done in Microsoft Visual C++ before being sent over to MATLAB via MatComm. The problem was that every pixel of the camera image is made up of a character, defined as a 'char'. The 'char' is signed and is made up of 256 different characters, ranging from –128 to 127 in Microsoft Visual C++. The 256 characters of 'char' in MATLAB are unsigned and ranges from 0 to 255. Thus when MATLAB receives the image information from Microsoft Visual C++, half the information was lost and the image of the virtual ball could not be detected in the visual images. To overcome this discrepancy, it was decided to have image processing done in MATLAB instead using only the unsigned 'char' ranging from 0 to 127 to restore the visual images.

Microsoft Visual C++ defines the origin of the pixels of the image at the top left of the image. As $u$ and $v$ increases, the pixel in question becomes further right and further down respectively. Conversely, MATLAB defines the origin of the pixels at the bottom left of the image. As u and v increases, the pixel in question becomes further right and further up respectively. Thus a simple mathematical equation was manipulated to overcome this discrepancy between the camera's y-coordinate:

$$v(\text{MATLAB}) = 240 - v(\text{Microsoft Visual C++}) \tag{10.1}$$

The virtual digital cameras in Microsoft Visual C++ were modeled using

OpenGL coordinate system while the images in MATLAB mimics the images obatined from the digital cameras in the robotic laboratory. The coordinate system adopted by OpenGL is different from the real world's coordinate system. The y-axis and z-axis of the OpenGL's coordinate system have to be inverted to overcome this discrepancy.

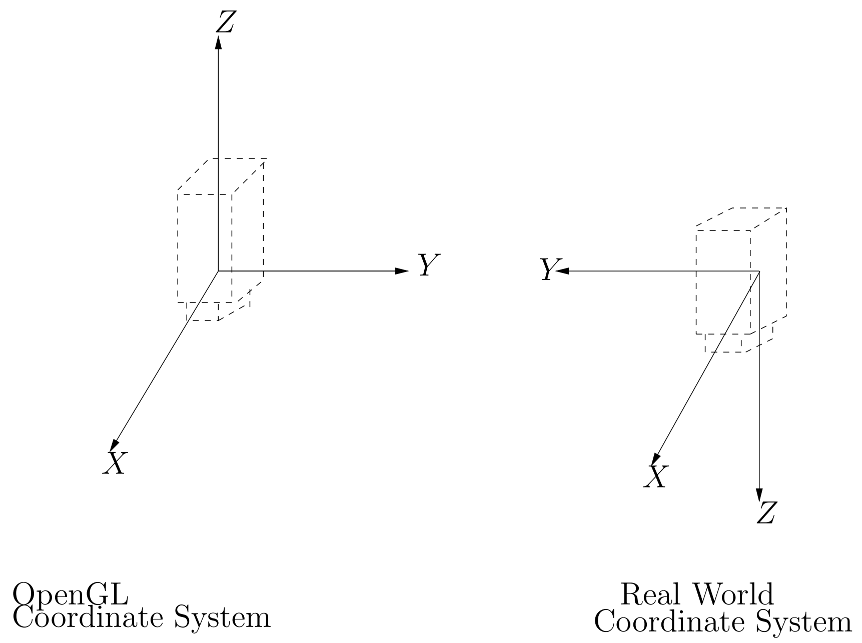


Figure 10.1: Inversion from OpenGL coordinate system to real world coordinate system

# Chapter 11

# Conclusion

The subject of this master thesis has been the application of position-based visual servoing in a simulated virtual environment. The task chosen to illustrate this invloves using a virtual robot with six degree-of-freedom to grasp a virtual ball moving in three-dimensional Newtonian dynaimcs. Virtual digital cameras were used to capture the whole trajectory of the ball simultaneously. The virtual digital cameras did not require calibration.

The images captured by the virtual digital cameras were sent from Microsoft Visual C++ to MATLAB in order to be processed. The image processing method used was feature point extraction with thresholding. There were various problems encountered during this process as different conventions were adopted for both the software programs in definition of characters, image coordinates and camera coordinate system. These problems were solved using mathematical manipulations. The difficulty did not lie in solving the problems, but in detecting them during image processing.

Triangulation was used to reconstruct the virtual ball in Cartesian coordinate from two-dimensional images obtained from the virtual digital cameras. Linear regression is then used to identify the trajectory of the ball so that the virtual ball grasping position can be predicted. The predicted virtual ball grasping position was then converted to robot joint angles before being sent to the robotic arm from MATLAB to Microsoft Visual C++.

Errors between the predicted virtual ball grasping positions and the actual virtual ball grasping position were found to be structured. The results were considered satisfactory as the maximum error was found to be less than 1.5 millimeters. This experiment was successful as the gripper attached to the end-effector of the virtual robot managed to grasp the virtual ball moving in three-dimensional Newtonian dynamics.

# Chapter 12

# Future Work

Since the experiment was successful in the simulated virtual environment, various methods could be included to make the experiment even better. One of them is to increase the robustness of the system by introducing errors into the images obtained from the virtual digital cameras. Then a controller such as a PID controller can be implemented to minimise the errors to increase the robustness of the system.

After that, the system can then be implemented in the robotic laboratory. In the robotic laboratory, the digital cameras will have to be calibrated. Various calibration techniques could be used to obtain the intrinsic and extrinsic parameters of the camera.

Feature point extraction with thresholding can still be used for image processing but new thresholds would have to be found for the ball as the colour of the virtual ball was only given a simple colour of pure green in the virtual environment. In real life, a tennis ball might look green on the surface, but it is a actually combination of red and blue colour as well, which cannot be seen by the naked eye.

# Chapter 13

# References

1 *A Tutorial on Visual Servo Control*, Seth Hutchinson, Gregory Hager and Peter Corke, IEEE Transactions on Robotics and Automation, Vol.12, No. 5, October 1996, pages 651-670

2 *ThielTech*,
http://www.acad.sunytccc.edu/CAPS152/mthiel/Robotics.htm

3 *The Author of Robots Defends Himself*, Karl Capek, Lidove Noviny, 19 June 1935, translation: Bean Comrada

4 *Asimov's Laws for Robotics: Implications for Information Technology*, Roger Clarke, Part 1 and Part 2, Computer, December 1993, pages 53-61

5 *RRG/Learn More/History*,
http://www.robotics.utexas.edu/rrg/learn_more/history

6 *Robotics FAQ Table of Contents*,
http://www.frc.ri.cmu.edu/robotics-faq

7 *State-space Identification of Robot Manipulator Dynamics*, Rolf Johansson, Anders Robertsson, Klas Nilsson and Michael Verhagen, Mechatronics, No. 10, 2000, pages 403-418

8 *Computer Vision Research at Penn State*,
http://vision.cse.psu.edu/intro.html

9 *Introductory Techniques for 3-D Computer Vision*, Emanuele Trucco and Alessandro Verri, Prentice Hall, 1998

10 *Vision Based Robotic Grasping Tracking of a Moving Object*, Michail Bourmpos, Master thesis ISRN LUTFD2/TFRT–5675–SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, September 2001

11 *Virtual Environment for Development of Visual Servoing Control Algorithms*, José Luis de Mena, Master thesis ISRN LUTFD2/TFRT–5686–SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 2002

12 *Introduction*
http://www.esat.kuleuven.ac.be/ pollefey/tutorial/node4.html

13 *Overview of OpenGL*
http://www.opengl.org/developers/about/overview.html

14 *A New Partitioned Approach to Image-Based Visual Servo Control*, Peter Corke and Seth Hutchinson, IEEE Transactions on Robotics and Automation, Vol.17, No. 4, August 2001, pages 507-515

15 *Computer Vision and Kinematic Sensing in Robotics*, Luis Conde Bento and Duarte Mendoca, Master thesis ISRN LUTFD2/TFRT–5670–SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, June 2001

16 *Visual Control of Robots: high-performance visual servoing*, Peter Corke, John Wiley & Sons Inc., 1996

17 *Vision Guided Force Control in Robotics*, Tomas Olsson, Master thesis ISRN LUTFD2/TFRT–5676–SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, October 2001

18 *Robotics Laboratory – Dept. Automatic Control*
http://www.control.lth.se/ robot/

19 *3D Theory — Coordinate Systems*
http://www.martinb.com/threed/solidmodel/coordinatesystems/index.htm

20 *XML.com: What is XML?*
http://www.xml.com/pub/a/98/10/guide1.html#AEN58

21 *System Modeling and Identification*, Rolf Johansson, Prentice Hall, 1993

22 *Industrial Robot Programming*, Klås Nilsson, PhD thesis ISRN LUTFD2/TFRT–1046–SE, May 1996, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden

# Chapter 14

# Acknowledgements

I would like to thank Imperial College of Medicine and Science, University of London, United Kingdom for giving me this opportunity to write my master thesis in Lund Institute of Technology, University of Lund, Sweden under the Erasmus exchange programme. I would like to thank Dr. Martin Clark from the Department of Electrical and Electronic Engineering, Imperial College of Science and Medicine for making the necessary arrangements prior to me making this unforgettable journey.

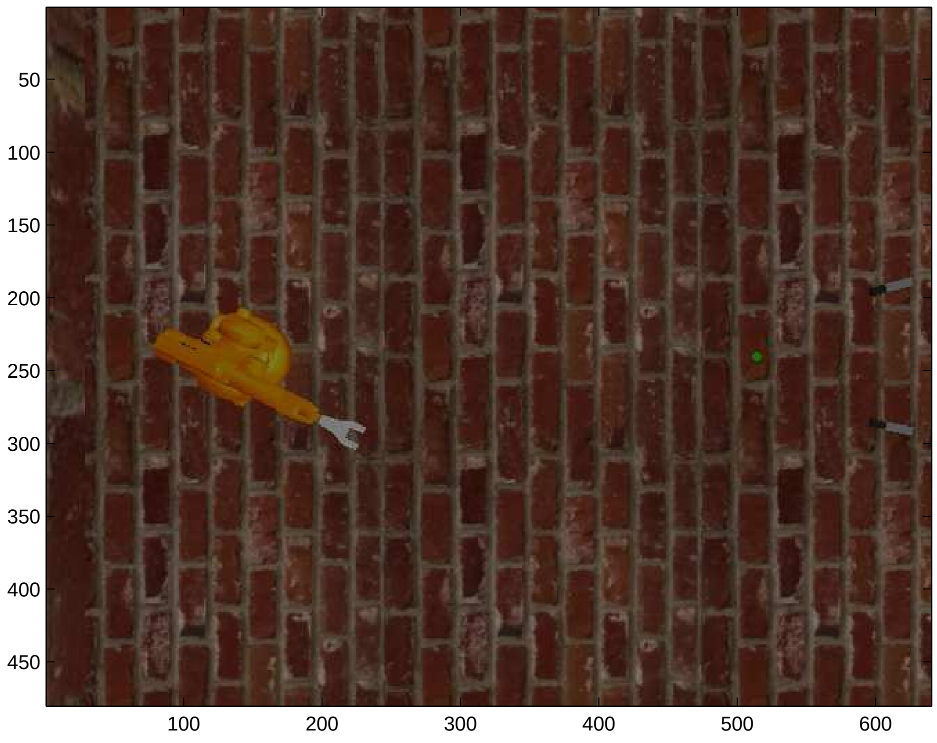
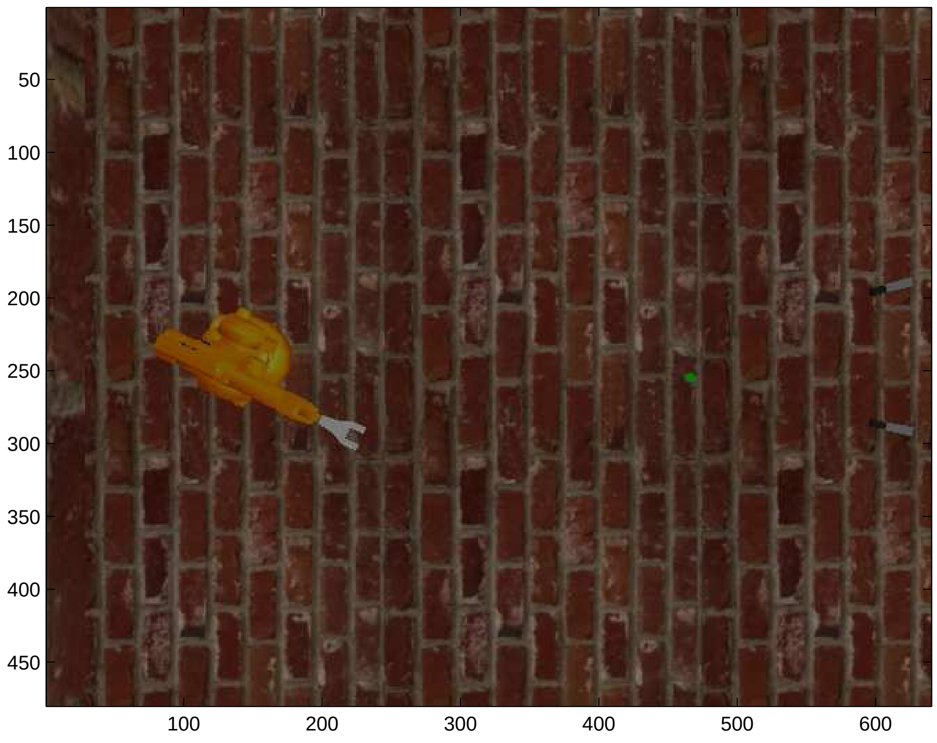
I would like to thank Professor Rolf Johansson from the Department of Automatic Control, Lund Institute of Technology for his kind guidance and concern both over my academic work and my personal welfare. I would like to express my sincere gratitude to Mathias Haage from the Department of Computer Science, Lund Institute of Technology as well as Johann Bengtsson and Tomas Olsson, who are both from the Department of Automatic Control, Lund Institute of Technology for their patience and helpfulness in assisting me, especially during the problem phases of this project. I would like to thank Leif Andersson and Anders Robertsson for their technical expertise and support during the course of this master thesis.

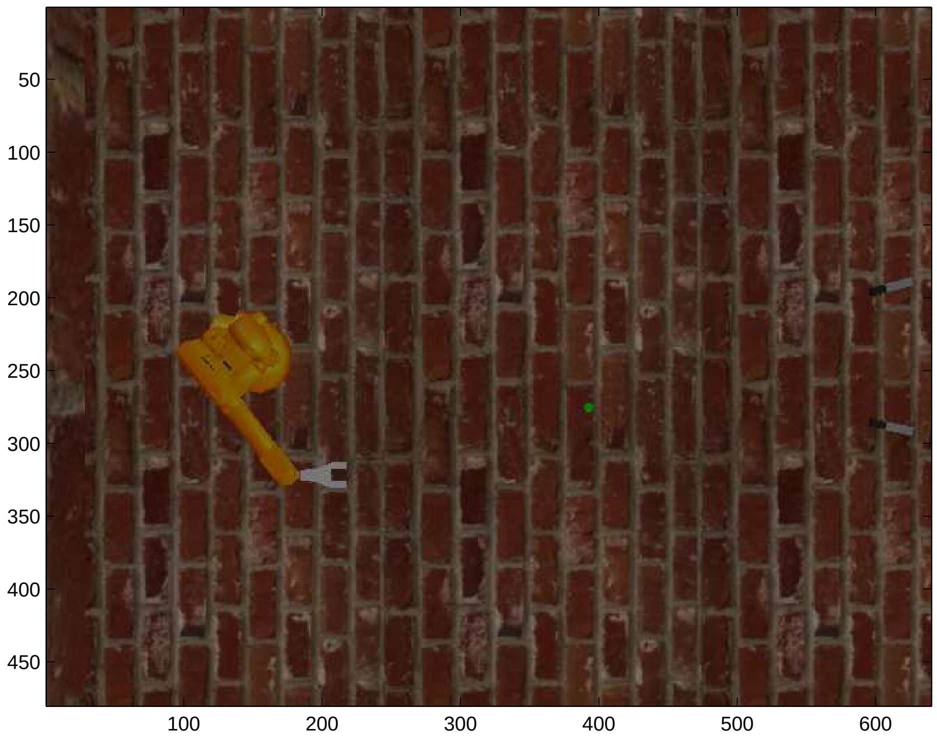
Finally, I would like to thank Woon Teck Her, my partner who is also from Department of Electrical and Electronic Engineering, Imperial College of Science and Medicine for toiling tirelessly with me for the completion of this master thesis.

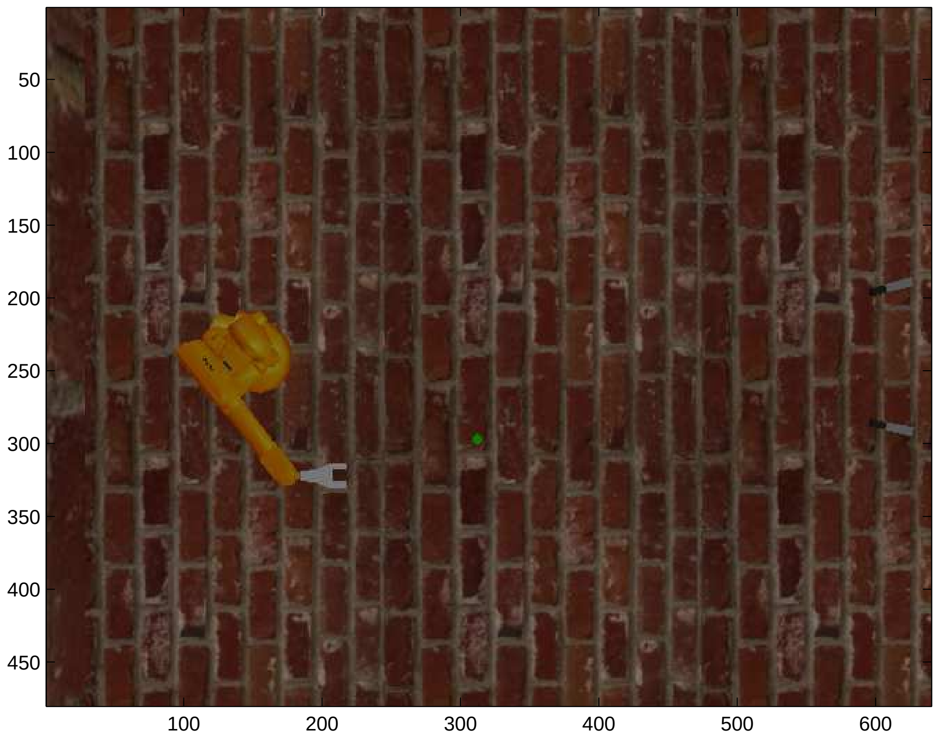
# Chapter 15

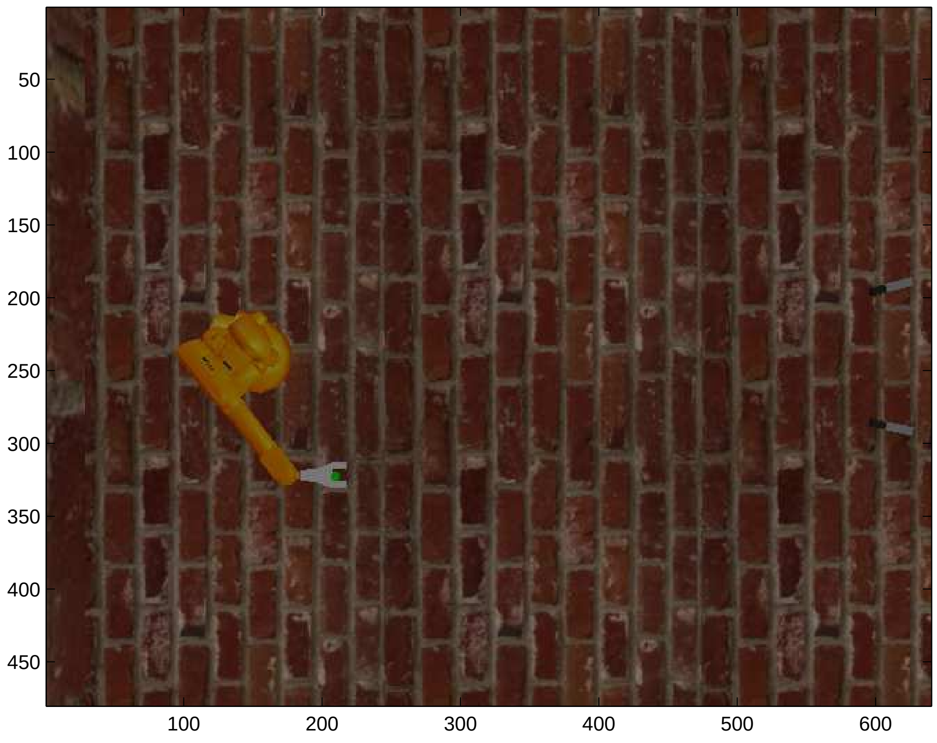
# Appendices

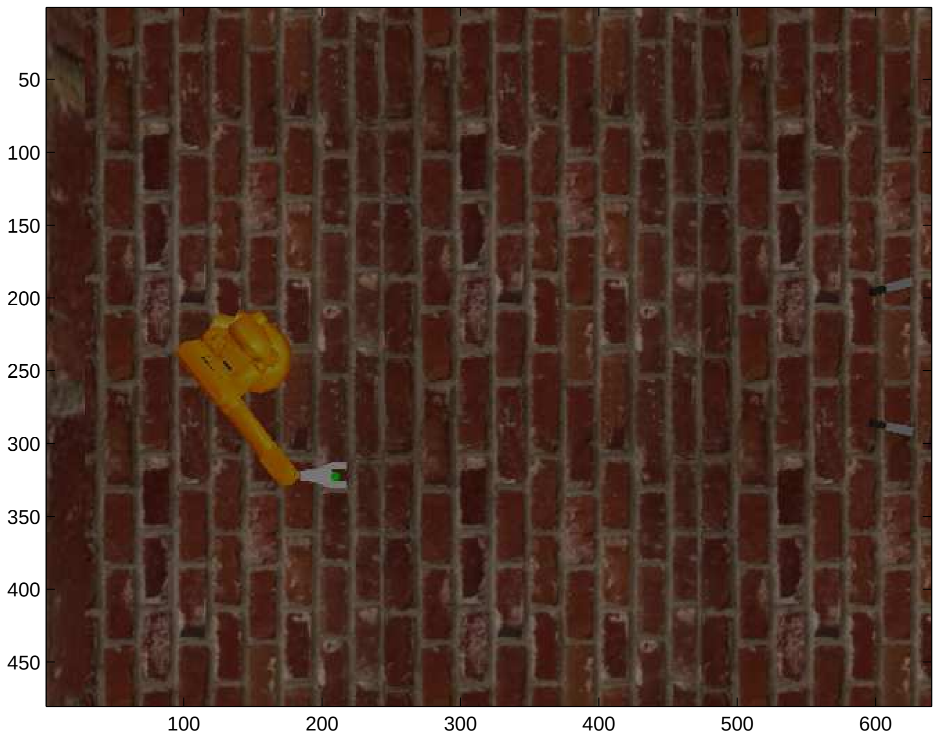
Pictures on the next three pages shows the simulation of the experiment. The position of the virtual ball and the six joint angles for Virtual IRB-2000 are updated for sample 1, sample 4, sample 8, sample 12, sample 17 and sample 20 in chrolonogical order.


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600

50
100
150
200
250
300
350
400
450
100
200
300
400
500
600


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600


50
100
150
200
250
300
350
400
450
100
200
300
400
500
600