ISSN 0280-5316
ISRN LUTFD2/TFRT--5695--SE

Model-based Visual Servoing Grasping of Objects Moving by Newtonian Dynamics

Teck Her Woon

Department of Automatic Control
Lund Institute of Technology
October 2002

Department of Automatic Control
Lund Institute of Technology
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
October 2002

Document Number
ISRN LUTFD2/TFRT--5695--SE

Author(s)
Teck Her Woon

Supervisor
Johan Bengtsson, Tomas Olsson, Rolf Johansson at LTH. Mathias Haage Computer Science, LTH. Martin Clarke at Imperial College

Sponsoring organization

Title and subtitle
Modelbased Visual Servoing Grasping of Objects Moving by Newtonian Dynamics.
(Modellbaserd visuell följning för gripande av object med Newtonsk rörelsedynamik).

Abstract
Robot control systems are traditionally closed system. With the aid of vision, visual feedback is used to guide the robot manipulator to the target in a similar manner as humans do. This hand-to-target task is fairly easy if the target is static in Cartesian space. However, if the target is dynamics in motion, a model of this dynamical behaviour is required in order for the robot to predict or track the target trajectory and intercept the target successfully. One the necessary modeling is done, the framework becomes one of automatic control.
In this master thesis, we present a model-based visual servoing of a six degree-of-freedom (DOF) industrial robot in the manner of computer simulation. The objective of this thesis is to manoeuvre the robot to grasp a ball moving by Newtonian dynamics in an unattended and less structured three-dimensional environment.
Two digital cameras are used cooperatively to capture images of the ball for computer vision system to generate qualitative visual information. The accuracy of the visual information is essential to the robotic servoing control. The computer vision system detects the ball in image space, segments the ball from the background and computes the ball in image space as visual information. The visual information is used for 3D reconstruction of the ball in Cartesian space. The trajectory of the thrown ball is then modeled and predicted. Several ball grasp positions in Cartesian space are predicted as the thrown ball travelling towards the robot. At that same time, the inverse kinematics of the robot is also computed and it steers the robot to track the predicted ball grasp positions and grasp the ball when the error is small. In addition, the performance and robustness of this model-based prediction of the ball trajectory is verified with graphical analysis.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
74

Recipient's notes

Security classification

1. Introduction

This master thesis addresses the problem of guiding a 6-DOF industrial robot to grasp an object moving by Newtonian dynamics in an unattended and less structured 3D environment. The thesis consists of the research on robotic control and computer vision, which both have been a popular topic in automotive industry and research.

The project is a collaboration work of my partner, Jee Hui Wong and me. My work in this project consists mainly of 3D image reconstruction and developing of a model to estimate ball grasp position as well as the trajectory of the thrown ball, while my partner, Jee Hui concentrated on interpreting and processing of visual information captured by two digital cameras.

1.1 Robotics

Since the late 1960s autonomous mobile robots have been subject to considerable research efforts by scientists all over the world. During this period they have developed various robot control techniques that enable the robots to navigate and operate robustly within their respective environments, using all kinds of different sensors [1].

This has helped to increase the number of robot being used in many manufacturing industries, where the robots are used to improve productivity and minimize variations in manufactured components thereby raising quality standards [13].

As robotic applications wider in many scientific fields, there exists an increasing demand to improve the capability and the flexibility of the robot. This could be due to the fact that most natural settings are not structured and not easy to model [7].

One way to increase the flexibility would be to integrate more sensors into the robot system [7]. However this has not proven cost effective for the bulk of robotic applications and may slow down system computation process [5].

Therefore, due to the lack of sensory capability and flexibility in many contemporary commercial robot systems, the robot applications are always restricted to operate in structured or designated environment [5].

This has become obvious that for robot to operate outside the designated environments where uncertainties may exist, a more flexible sensor is required.

This limitation has now mostly been resolved, thanks to the development of Computer Vision.

In this master thesis, research interest has been focused on the manipulation of a robotic arm.

1.2 Computer Vision

With the huge improvement of computing power and image processing techniques available, computer vision together with the use of digital camera has been made possible as a useful and flexible robotic sensor, since computer vision mimics the human sense of vision while digital camera allows for non-contact measurement of the environment [5]. Thereby it enables the use of robots in unstructured environment, where the robot workspace is unlimited.

Early studies in the field of computer vision, in the 1970's, were mainly motivated by problems of pattern recognition. Owing to the increasing attention paid to mobile robots, researchers have investigated more complex questions such as stereo vision and outdoor scene analysis. In complement, major work was done in the domain of the analysis of image sequences (dynamic vision). The original motivations might come from telecommunication (motion compensated coding) or military (target tracking, recognition of moving objects) applications [2].

In most robotic applications, external sensor is used to acquire feedback information for control scheme. The external sensor can be classified into two classes as physical contact sensor and non-physical contact sensor. Examples of the physical contact sensors are contact switch and force sensor while some example of the non-physical contact sensor are ultra-sound sensor, infra-red sensor, laser sensor and digital camera [3].

Computer vision involves the capturing, understanding and processing of images [3]. Digital camera is used to capture images for the computer vision system. The computer vision system analyses and process images to provide visual information for robotic manipulation.

A common problem of computer vision is to identify object of different materials and geometrical shape in image. The other common problem is to locate the depth of an object in image, which will be discussed further in Triangulation.
Triangulation is the studies of 3D reconstruction of corresponding images.
Depth is calculated from the disparity between at least two images [4].

In this master thesis, two digital cameras are used as external sensors to capture and feedback images of the robot environment, in which with specific attention to the position of the robot arm and the trajectory of the thrown ball.

1.3 Visual Servoing Control

Visual servoing is the combination of robotic control and computer vision. The task in visual servoing is to use visual information to control the pose of the robot's end-effector relative to a target object or a set of target features [5].

There are two classical approaches of visual servoing, namely Image-Based Visual Servoing (IBVS) or 2D Visual Servoing and Position-Based Visual Servoing (PBVS) or 3D Visual Servoing.

In the Image-Based Visual Servoing, the pose estimation is omitted and the control error function is computed in the image space. The IBVS approach does not need a precise calibration and modeling since a closed-loop scheme is performed. However, the stability is theoretically ensured only in the neighborhood of the desired position. Therefore, if initial and desired configurations are closed, IBVS is robust with respect to measurement and modeling errors. Otherwise, that is if desired and initial position are distant, the stability is not ensured and the object can get out of the camera field of view.

In the Position-Based Visual Servoing, the control error function is computed in the world or Cartesian space. Image features are extracted from the image and a perfect model of the target is used to determine its position with respect to camera frame. The main advantage of this approach is that it controls the camera trajectory directly in Cartesian space. The drawbacks are there is no control in the image space and the object may get out of the camera field of view during servoing and it is impossible to analytically demonstrate the stability of the system in presence of modeling errors. Indeed, the sensitivity of pose estimation algorithm with respect to calibration errors and

measurement perturbations is not available [6]. In addition, an accurate camera calibration is required.

The mixture of PBVS and IBVS is called Hybrid-Based Visual Servoing (HBVS), which may counteract the drawbacks of each other to render a better control.

In this master thesis, only the Position-Based Visual Servoing technique is applied. Therefore drawbacks of PBVS mentioned above will have to be resolved or emasculated in order to obtain good performance of 3D image reconstruction as well as servoing control.

1.4 Previous Projects

Prior to this project there have been a few relevant projects and thesis assignments carried out at the Department of Automatic Control in the University of Lund. These projects are:

1. "Vision based robotic grasping tracking of a moving object" by Michail Bourmpos. In this project, Image-based visual servoing was used and the goal was to track and grasp a ball that rolls linearly on the metallic board.

2. "Vision guided force control in robotics" by Tomas Olsson. In this project, a method for combining direct force control and visual servoing in the presence of unknown planar surfaces was presented. The goal of this project was to grasp a marker pen and draw shapes on the whiteboard. Extensive studies and useful information about vision feedback image-Jacobian and position of the cameras with hand-eye calibration can be obtained from this thesis.

3. "Computer vision and kinematic sensing in robotics" by Luis Manuel Conde Bento and Duarte Miguel Horta Mendonca. In this project, visual servoing with cameras of a moving feature point (the center of a cross) was studied. The goal was to keep the feature point centred inside the images captured by two cameras, mounted on a robotic manipulator. In this project, the cameras were initially calibrated. In addition, the thesis provides good background reading and useful information relevant to the thesis discusses in this paper.

4. "Virtual environment for development of visual servoing control algorithms" by Jose Luis de Mena. In this project, a virtual environment done in java was used and the performance of three different controllers to track and grasp a moving ball was studied. The three controllers were proportional gain controller, image-based Jacobian controller and hybrid visual servoing controller.

1.5 Thesis Outline

The thesis is organized as follows:

Chapter 2 describes the problems formulation.

Chapter 3 introduces robot virtual world and experimental setup.

Chapter 4 introduces overall system and Matcomm.

Chapter 5 describes feature point extraction and 3D reconstruction by triangulation.

Chapter 6 describes system identification, model-based prediction of ball gasp positions and inverse kinematics.

Chapter 7 investigates the graphical results, error analysis and model performance.

Chapter 8 Discussion.

Chapter 9 Conclusion.

Chapter 10 Future Work.

Chapter 11 References.

Chapter 12 Appendices.

2. Problems Formulation

The main goal of the project is to make the virtual robot to be able to catch a thrown ball moving by Newtonian dynamics in 3D, using the Position-Based Visual Servoing technique and a model that predicts the ball grasp position.

In other words, when a ball is thrown to the virtual robot, the robot system has to be able to predict the ball trajectory and estimate the ball intercept position accurately.

The ball trajectory is unknown and random. Every throw will have different initial velocity, initial (x, y, z) Cartesian coordinates and xz plane angle as well as xy plane angle. Thus it will result different ball trajectory.

Two virtual cameras are used as visual sensors in doing the project. The virtual cameras will capture several images on the ball as it travels toward the virtual robot. These images will be used as visual information to the computer vision system for image processing.

It is obvious that to do the project, a number of subtasks are required. Some basic problems of these subtasks are described briefly below.

2.1 Experimental Setup

To start the project, it is necessary to think about questions such as how and where should the two virtual cameras be placed, where should the virtual ball be thrown and in what direction to the virtual robot it should be thrown. All these are important because they will affect the virtual experimental process and the measurements taken. Furthermore, the virtual experimental setup should be an achievable setup that can be applied to the real

robot environment at the robot laboratory in the Department of Automatic Control, University of Lund. Therefore, the constraints of the virtual environment setup are the size of the robot laboratory and the structures in it.

2.2 Camera Placement, Adjustments and Calibration

Camera placement is important. A good camera placement will give sufficient and qualitative visual information that help the robot system to perform well.

For instance, if the two virtual cameras are to be placed directly facing each other, it is most likely that both virtual cameras will not be able to capture the whole ball trajectory. Thus it will result loss of visual information for image processing.

The other important issue over the virtual camera placement is called Triangulation. Triangulation is essential for 3D image reconstruction and it requires the two virtual cameras to be placed in such a way that both cameras can capture the whole ball trajectory simultaneously. The name Triangulation may comes from the way that the cameras are placed in relation to a focus point where it may most likely be lie at the direction of the ball trajectory.

To get a clear view of the scene, the camera lens need to be adjusted. Sometimes the camera placement is limited by the size of the laboratory, so in order to capture the whole scene of the ball trajectory the camera may need to be zoomed in or out.

It is also obvious that the robot system needs some information on the camera location in the laboratory and the camera setup. This information can be obtained by calibrating the cameras and the stereo system. The calibration process gives us information on where the

two cameras are located relative to the robot and the intrinsic parameters of the cameras [7].

2.3 Image Processing - Feature Points Extraction

The colour of the target, in our case is the ball needs to be 'obvious' to the computer vision system in image. 'Obvious' in the sense that the colour of the target should help the computer vision system to identify the target easily. The three most obvious colour to the computer vision system are Pure Red, Pure Green and Pure Blue. These three colours should contain highest number of colour pigment intensity.

Feature Point Extraction basically scans the image pixels. Each pixel consists of three colour pigments. Each colour pigment has colour intensity range from 0 ~ 127.

If 'unobvious' colour is used, it may cause some loss track of visual information on the ball i.e. sample loss.

In addition, the texture of the background and the lighting will affect the feature point extraction.

2.4 3D Reconstruction – Triangulation

Triangulation is an essential method for 3D image reconstruction and thus it helps to determine the Cartesian or world coordinate of the target. Two cameras are required for triangulation setup and both cameras have to be able to capture as much images of the target as possible simultaneously. Triangulation would fail if both cameras cannot capture images relative to the ball movement simultaneously.

2.5 System Identification & Prediction of the Ball Grasp Position

As the Newtonian dynamic equations are nonlinear, therefore our system is nonlinear. In order to identify a nonlinear system with linear system identification method such as linear regression, decoupling and linearization of the system is required. In addition, three dimensions of the Newtonian dynamic equations have to be considered. Also in some cases unknown parameters are multiplied together, ways to split and identify them separately have to be found.

There are several ways for the robot to grasp the ball. An efficient and simple way has to be found. To avoid delay and has a successful grasping, predictions of the ball grasp position and the ball trajectory are required.

2.6 Different Object Has Different Default Coordinate System

Each object in the virtual world has its individual coordinate system, which does not correspond to the world coordinate system. In other words, the robot has its own coordinate system and so the camera and the robot gripper. The coordinate systems of these objects were set in the simulation when they were created.

Because of this difference in coordinate system of each object, we have to check and find out their default coordinate system one by one. This is essential when a specific orientation of the object is required. The checking was done frequently when we were doing the experimental setup. For example, if the orientation of the camera is unknown, we would not be able to point the camera to our desired direction.

3. Virtual World and Experimental Setup

The virtual environment that had been created in Visual C++ by Tomas Olsson from the Department of Automatic Control and Mathias Haage from the Department of Computer Science, Lund Institute of Technology was modified to suite our project setup.

The aim for the use of virtual world is to try to simulate the actual robot workspace environment as closely as possible.

The virtual world provides a free and thorough control of the robot workspace to the users and thus it allows various experiments and robotic studies to be carried out.

Moreover, the virtual robot can be more flexible than the actual robot during operations. Less precaution is needed when using the virtual robot than the actual robot.

After some surveying in the real robot laboratory and with consideration of the size of the robot lab, an experimental setup has been decided.

In the experimental setup, the virtual robot is located 4m away from the origin with its arm rotated 25 degree clockwise from the x-axis, while the ball is thrown from the origin with an initial height of 1m above the ground. Two virtual cameras are placed 1m backward from the origin and 1m away from each other. Virtual camera at the left side (called camera2) and virtual camera at the right side (called camera3) were rotated approximately 12 degrees clockwise and 12 degrees anticlockwise respectively. Both virtual cameras are 1m above the ground.

The experimental setup from the top view is briefly shown in figure 1.

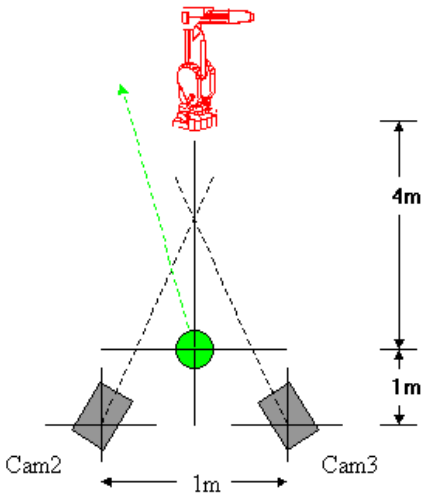

Figure 1: Experimental setup.

This kind of experimental setup has been convinced through simulations that it allows the whole ball trajectory to be captured simultaneously in images by the two cameras. As the reason, the 3D image reconstruction by Triangulation will be achievable and there will be no loss of visual information.

As mentioned before in the problem formulation, section 2.6 that different object has different coordinate system defined in the virtual environment, so the default coordinate system of each object used in the virtual world has to be found out in order to rotate the object to point it to the desired direction. These object default coordinate systems have been obtained and will be presented in this chapter.

In the virtual world, the default world coordinate system is defined as shown below in figure 2.

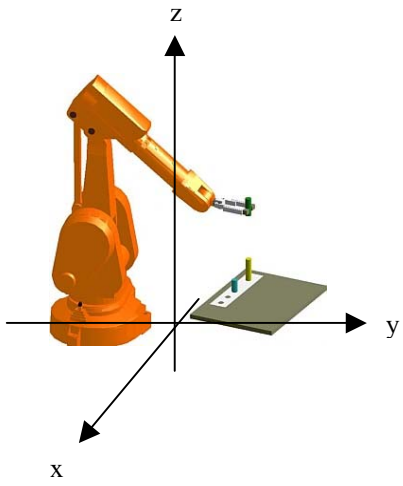

Figure 2: Default definition of the world coordinate system.

The x-axis represents the depth of the virtual world in the 2D image, the y-axis represents the horizontal level of the virtual world in the 2D image and the z-axis represents the vertical level or the height of the virtual world in the 2D image.

The followings introduce the actual equipments or objects in the robot laboratory, which are to be simulated in the virtual world.

3.1 Robot ABB IRB-2000

The robotic manipulator used in the robot lab is called IRB-2000. It has 6 degrees of freedom (DOF) and a precision of 0.1mm. The manipulator's joint 2, 3 and 5 are revolute joints while the other joint 1, 4 and 6 are cylindrical joints, see figure 3. The fact that this robot has 6 degrees of freedom allows it to reach any desired position and orientation within the workspace.

The robot is controlled from Matlab/Simulink, which sends control signals through the network - Matcomm.

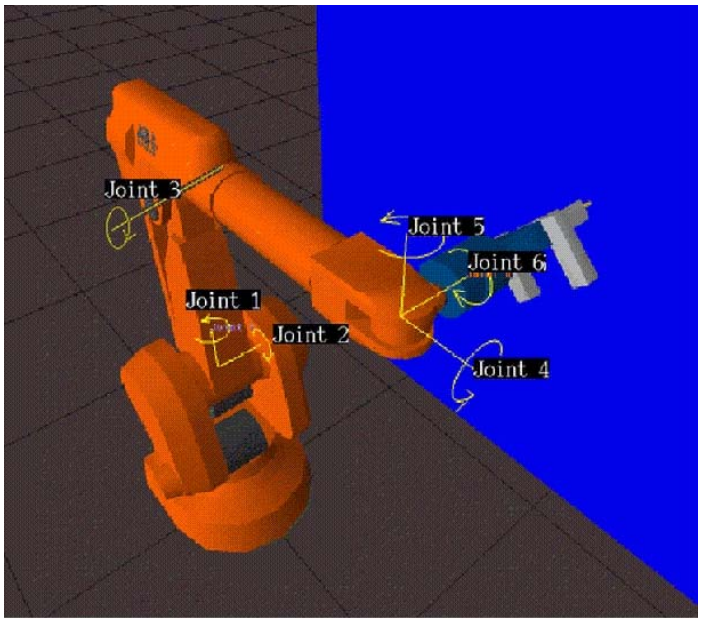

Figure 3: Six joints of the Irb2000.

The default coordinate system of the robot has been found relative to the world coordinate system as shown below in figure 4.

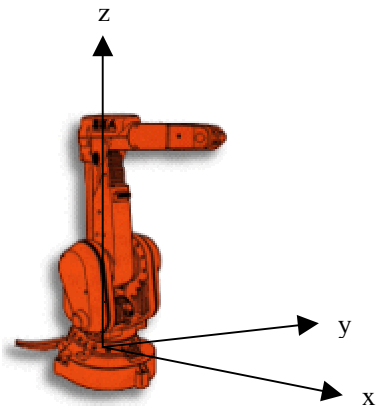

Figure 4: Default definition of the Irb2000 coordinate system.

A gripper (end-effector) is attached to the robot manipulator and it has a default coordinate system relative to the robot coordinate system as shown below in figure 5.

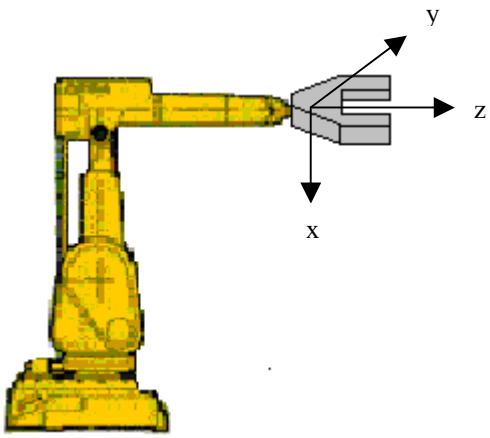

Figure 5: Default definition of the gripper coordinate system.

The Rt (Rotational and Translation) matrix consists information of the orientation and the position of an object. The Irb2000 in the virtual world has the Rt matrix as shown below.

$$\text{robot\_Rt} = \left[\begin{array}{cccc} 0.9064 & 0.4233 & 0 & -4 \\ -0.4233 & 0.9064 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

The 3-by-3 matrix of the first three rows and columns of the robot's Rt matrix represents the orientation of the robot while the 3-by-1 vector of the first three rows and the fourth column of the robot's Rt matrix represents the position of the robot.

For further information of the Rt matrix, see Appendix A.

3.2 Cameras

Two virtual cameras, namely camera 2 and camera 3 are used to capture images as visual information for image processing. The camera 1 is another virtual camera that is being used to observe the experimental environment during the simulation process.

The cameras in the robot lab are called Sonny DFW V-300, which utilizes the IEEE 1394 high performance serial bus to send non-compressed YUV digital data and allow us to use control functions such as colour tone, brightness, picture quality, white balance and automatic gain control (AGC). The camera signals are transmitted at 200 M-bits/sec, which means there is a transmission rate of 30 images per second between the cameras

and the image processing software. The picture format of 640 x 480 pixels (4:1:1) can be modified to 320 x 240 pixels (4:2:2), which is the resolution we used [4],[3].

The cables used to transmit the images from the cameras in the robot lab are called Fire Wire. It is a high speed, non-proprietary, scaleable digital serial bus that can sustain a transmission rate of 100, 200 and 400 M-bits/sec. The cable also enables true plug and play, which allows the user to connect new devices with the system switched on and the bus active [3],[4],[8].

The camera and its default coordinate system relative to the world coordinate system are shown below in figure 6.

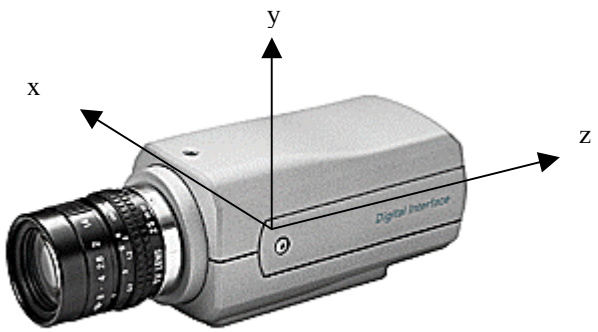

Figure 6: Default definition of the camera coordinate system.

The Rt matrix of the virtual camera 2 and camera 3 are found and shown below.

$$\text{cam2\_transform} = \left[\begin{array}{ccc|c} 0.2065 & 0 & 0.9785 & 1 \\ 0.9785 & 0 & -0.2065 & -0.5 \\ 0 & 1 & 0 & 1 \end{array}\right]$$

$$\text{cam3\_transform} = \left[\begin{array}{ccc|c} -0.2533 & 0 & 0.9674 & 1 \\ 0.9674 & 0 & 0.2533 & 0.5 \\ 0 & 1 & 0 & 1 \end{array}\right]$$

In the virtual world, the calibration of camera 2 and camera 3 is not required.

The calibration matrix of camera 2 and camera 3 used in our simulation program, Winrobot were taken from Tomas Olsson, who had used the calibration matrices while doing robot simulation in the virtual world. The parameters in the calibration matrices he used were obtained from calibration in the real world.

The focal length of the calibration matrices of our cameras was re-adjusted to zoom in or out the camera scene such that the two cameras can capture images of the full trajectory of the ball. This requirement is important for the 3D image reconstruction.

The intrinsic parameters of camera 2 and camera 3 after having their focal length re-adjusted are shown below:

$$\text{cam2\_K} = \begin{bmatrix} 456.2578 & 0 & 175.5991 \\ 0 & 485.4411 & 113.8451 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{cam3\_K} = \begin{bmatrix} 629.6043 & 0 & 194.6277 \\ 0 & 657.8963 & 152.6292 \\ 0 & 0 & 1 \end{bmatrix}$$

3.3 XML Ball and 3D Newtonian Dynamics of The Ball

The ball was created by Jee Hui Wong in Visual C++ and saved as a XML (Extensive Markup Language) file. The ball is a pure green sphere-a-like object having radius of 3 cm. In the XML, the ball consists of 24 vertex points, 18 quad-faces and 8 triangle-faces. Pure green is used for the ball colour as it is one of the obvious primary colour out of three that are used in video camera systems. The other two obvious primary colours are pure red and pure blue. Every pixel of an image consists of these three primary colour elements.

The ball and its default coordinate system relative to the world coordinate system are shown below in figure 7.

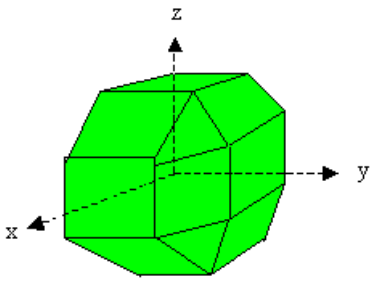

Figure 7: Default definition of the ball coordinate system.

The Rt matrix of the ball is found and shown below.

$$\text{ball\_Rt} = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

Our 3D Newtonian dynamics equations of the ball trajectory are:

$$X_k = -v_0 \cos(\theta)\sin(\phi)t_k + x_0 \quad (1)$$

$$Y_k = -v_0 \cos(\theta)\cos(\phi)t_k + y_0 \quad (2)$$

$$Z_k = v_0 \sin(\theta)t_k - \frac{1}{2}gt_k^2 + z_0 \quad (3)$$

where k = number of sample of the ball trajectory.

(X,Y,Z) is the Cartesian coordinates of the ball.

v_0 is the initial velocity of the ball trajectory.

(x_0,y_0,z_0) is the initial position of the ball trajectory in Cartesian space.

t_k is the time of a sample.

θ is the angle of the ball trajectory on the xz plane.

ϕ is the angle of the ball trajectory on the xy plane.

g is the gravitational constant of 9.82 ms^{-2}.

According to these equations, the ball trajectory was programmed in Visual C++. This program was written by my partner, Jee Hui Wong.

The side and the top view of the ball trajectory are shown below in figure 8.

The dotted line represents the ball trajectory.

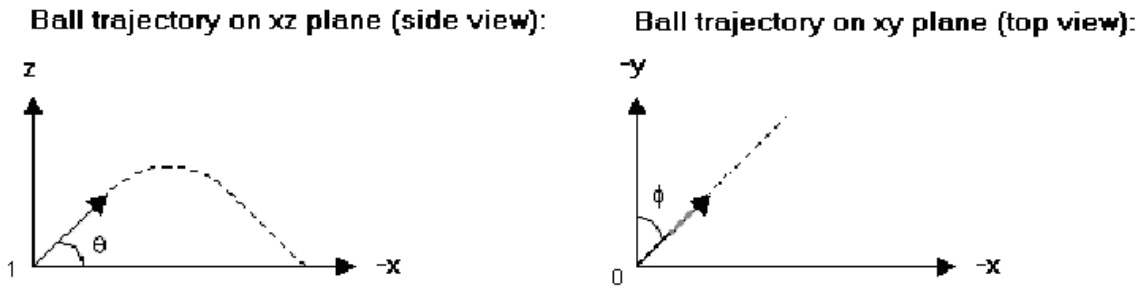

Figure 8: The side and the top view of the ball trajectory.

Equation (1) and (2) have negative initial velocity, it is because in our experimental setup the ball is thrown from the origin and it travels toward the negative region of X and Y Cartesian coordinates.

4. Overall System and Matcomm

4.1 Matcomm Network

Our communication between the virtual environment in Visual C++ window platform and the Matlab in UNIX platform is done using Matcomm.

Matcomm is a software that was developed in the Department of Automatic Control, University of Lund. This software uses the TCP/IP protocol to transmit data between computers in the network where different systems might exist. The reason to use Matcomm is that it is an easy and fast way to connect with another computer without the necessity of setting all necessary parameters. Data are sent through Matcomm in an array format using sockets.

In our system, Matcomm is used in the following ways as shown below in figure 9.

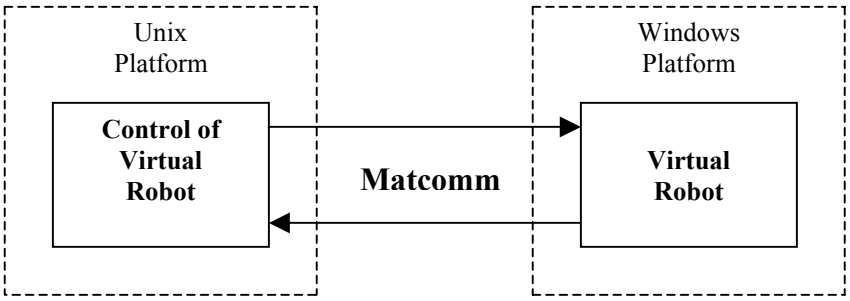

Figure 9: Matcomm communication link.

Images taken from the virtual environment (in visual C++ in the windows platform) are sent using Matcomm to the Matlab (in the unix platform) where all the computation for the control of the virtual robot is done. In the final computation, Matlab calculates the

joint angles of the robot and sent these joint angles through Matcomm to control and move the Irb2000 to the desire position. This simulation setup of communication is used because the real world communication setup at the robot laboratory in the Department of Automatic Control, Lund Institute of Technology is implemented in such a way.

The reason of this way of communication is used in the robot laboratory is because the robot Irb2000 is set only to communicate with the Unix machine (called Euler) while most control software are only worked in Window platform. Moreover, it is good to have two systems working so that tasks can be split and the tasks computation can be done faster.

4.2 The Overall System

The overall system is shown below in figure 10. The overall system can be divided into two main parts namely the software part and the hardware part. The software part is doing the control of the robot, which is done in Matlab (in the unix platform) and the hardware part is about the robot and the cameras.

The control of the robot is made of five subtasks. The five subtasks are image feature point extraction, 3D reconstruction, system identification, ball grasp position prediction and inverse kinematics.

The camera is used as a visual sensor. Images taken by the cameras are sent into the control of the robot for image feature point extraction (or image processing). In the process of the image feature point extraction, the position (u, v) of the ball in 2D image is

calculated, where u is the number of image pixels in horizontal from the origin to the target while v is the number of image pixels in vertical from the origin to the target.

The u and v of the two images, which each taken by one camera is then input to 3D reconstruction. The 3D reconstruction uses a method called Triangulation to reconstruct 3D image from two 2D images. In the process of Triangulation, the 3D world coordinate (X, Y, Z) of the ball is calculated using least square solution. The ball world coordinate is passed-on for system identification. In the process of system identification, the unknown system model parameters are computed using least square solution. The calculated unknown parameters are sent for ball grasp position prediction, where several possible ball trajectories and the position the ball might travel-by are predicted. The ball grasp position prediction calculates the predicted 3D world coordinate (X_p, Y_p, Z_p), which is then sent to the inverse kinematics to calculate the six joint angles so the six robot joints would move relatively the robot to the ball grasp predicted position in 3D.

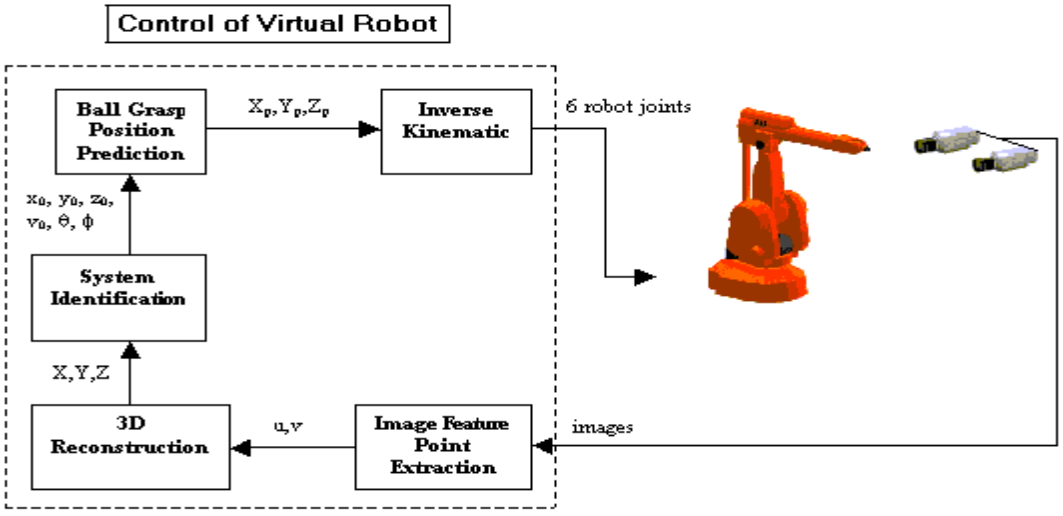

Figure 10: The overall system.

5. Feature Point Extraction and Triangulation

5.1 Feature point extraction

This section of work was done by Jee Hui Wong and is written with the contribution from him.

The computer vision system is required to extract visual information needed to perform servoing task. The two main operations of most computer vision system are detection and location. For instance, a target is first to be detected to check if it exists in the image and if it does exist its location is then to be obtained. Through the process, useful visual information is extracted.

The detection is done based on the image features of the object. The image features can be edges, points, corners, surfaces, lines, curves, etc.

Several method of the detection based on these image features have been proposed by researchers. See [3].

In our thesis, the point of the object's image feature is used for the detection and it is called feature point extraction.

The feature point extraction identifies the connected groups of pixels (which correspond to the points of the object) in a binary image that all have the same binary value. How good is the detection of these pixels depends on the threshold be used. The threshold is used to separate the target from the background, and to label them as object or as background.

Each pixel consists of three primary colour elements, namely Red, Green and Blue. The arrangement of the three primary colour elements is fixed. Each primary colour element has the colour intensity ranged from 0 to 127.

The fact to have the three primary colour elements in each pixel allows the detection to be carried out not only based on image feature point but also the colour.

Because the ball used is pure green, hence it is easy to be detected by setting the threshold to be equal to the highest green colour intensity, which is 127.

So the point of the object in image is detected if the pixel detected has the thresholds 0 for the Red colour element, 127 for the Green colour element and 0 for the Blue colour element.

The algorithm of the detection in Matlab scans the entire image (left to right, bottom to top), searching for occurrences of pixels of the same binary value.

The threshold limit for segmenting the ball from the background is:

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

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

$$(u_0, v_0) = \left(\frac{\sum u_{ball}}{\sum Ball(u) = 1}, \frac{\sum v_{ball}}{\sum Ball(v) = 1} \right)$$

where u_{ball} and v_{ball} are the u and v for when Ball(u,v) is equivalent to binary '1'.

5.2 3D Image Reconstruction by Triangulation

Triangulation is a method of 3D image reconstruction using two 2D images. The 3D image reconstruction is sometimes referred to as Stereovision, which determines the position of a target in Cartesian space by using Triangulation.

The relationship for the 2D and the 3D transformation can be expressed by the projection equation as shown below.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P_{3x4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where $\lambda = Z$ is the depth of the imaged point in the camera,

(u,v) is the object homogeneous image coordinates,

P = K[R|t] is a 3-by-4 projective transformation matrix. The K is the camera calibration matrix and the [R|t] is the camera rotation and transformation matrix.

(X, Y, Z) is the object homogeneous Cartesian coordinates.

The derivation of the projection equation can be seen in Appendix C.

As the reason that two images are used for the Triangulation, therefore two projection equations are used. The derivation of Equation (12) from the two projection Equations (4) and (5) is shown below.

$$\lambda_1 \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = P_{A3x4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4}$$

$$\lambda_2 \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} = P_{B3x4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

By assigning the P matrix as shown, we get:

$$\begin{bmatrix} \lambda_1 u_1 \\ \lambda_1 v_1 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} P_{A11} & P_{A12} & P_{A13} & P_{A14} \\ P_{A21} & P_{A22} & P_{A23} & P_{A24} \\ P_{A31} & P_{A32} & P_{A33} & P_{A34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{6}$$

$$\begin{bmatrix} \lambda_2 u_2 \\ \lambda_2 v_2 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} P_{B11} & P_{B12} & P_{B13} & P_{B14} \\ P_{B21} & P_{B22} & P_{B23} & P_{B24} \\ P_{B31} & P_{B32} & P_{B33} & P_{B34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{7}$$

Solve the above equations we get:

$$\begin{bmatrix} \lambda_1 u_1 \\ \lambda_1 v_1 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} (P_{A11}X + P_{A12}Y + P_{A13}Z + P_{A14}) \\ (P_{A21}X + P_{A22}Y + P_{A23}Z + P_{A24}) \\ (P_{A31}X + P_{A32}Y + P_{A33}Z + P_{A34}) \end{bmatrix} \tag{8}$$

$$\begin{bmatrix} \lambda_2 u_2 \\ \lambda_2 v_2 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} (P_{B11}X + P_{B12}Y + P_{B13}Z + P_{B14}) \\ (P_{B21}X + P_{B22}Y + P_{B23}Z + P_{B24}) \\ (P_{B31}X + P_{B32}Y + P_{B33}Z + P_{B34}) \end{bmatrix} \tag{9}$$

Rearrange the above equations, we get:

$$\begin{bmatrix} (\lambda_1 u_1 - P_{A11}X - P_{A12}Y - P_{A13}Z) \\ (\lambda_1 v_1 - P_{A21}X - P_{A22}Y - P_{A23}Z) \\ (\lambda_1 - P_{A31}X - P_{A32}Y - P_{A33}Z) \end{bmatrix} = \begin{bmatrix} P_{A14} \\ P_{A24} \\ P_{A34} \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} (\lambda_2 u_2 - P_{B11}X - P_{B12}Y - P_{B13}Z) \\ (\lambda_2 v_2 - P_{B21}X - P_{B22}Y - P_{B23}Z) \\ (\lambda_2 - P_{B31}X - P_{B32}Y - P_{B33}Z) \end{bmatrix} = \begin{bmatrix} P_{B14} \\ P_{B24} \\ P_{B34} \end{bmatrix} \tag{11}$$

Combine the above two equations, we get our final equation for Triangulation as

$$\underbrace{\begin{bmatrix} -P_{A11} & -P_{A12} & -P_{A13} & u_1 & 0 \\ -P_{A21} & -P_{A22} & -P_{A23} & v_1 & 0 \\ -P_{A31} & -P_{A32} & -P_{A33} & 1 & 0 \\ -P_{B11} & -P_{B12} & -P_{B13} & 0 & u_2 \\ -P_{B21} & -P_{B22} & -P_{B23} & 0 & v_2 \\ -P_{B31} & -P_{B32} & -P_{B33} & 0 & 1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} X \\ Y \\ Z \\ \lambda_1 \\ \lambda_2 \end{bmatrix}}_{C} = \underbrace{\begin{bmatrix} P_{A14} \\ P_{A24} \\ P_{A34} \\ P_{B14} \\ P_{B24} \\ P_{B34} \end{bmatrix}}_{B} \tag{12}$$

Generally, the P matrix is known and to determine the object Cartesian coordinates, all we need is to know the image coordinates u_1, v_1, u_2, v_2 of the two images. The Cartesian coordinate system (X, Y, Z) of the object of two images is the same, but different image taken from different position or angle will have different image coordinates.

To solve equation (2), we simply do in Matlab

C = A \ B

This renders a least-square solution because A is not a square matrix, is singular and has more rows than columns. A rather similar least-square solution can also be obtained by using Matlab function's pseudoinverse, pinv(A) such as C = pinv(A)*B, see [9].

Note: in general the mathematical notation of pseudoinverse of the matrix A is written as A^{+}.

6. System Identification and Ball Grasp Position Prediction

6.1 System Identification

We realize that our Newtonian dynamic equations are non-linear and we want to identify the system for the unknown parameters such as v_0, x_0, y_0, z_0, θ and ϕ.

However, we have a problem to identify θ and ϕ from cosine or sine alone. We need to simplify the cosine and the sine in the equations. One basic way to simplify them is to assume

$\beta = \cos(\theta)$ and $\alpha = \cos(\phi)$

And by Trigonometric Identities such as

$\sin^2(\theta) + \cos^2(\theta) = 1$

We modify our Newtonian dynamic equations to become

$$X_N = -v_0\beta(\sqrt{1-\alpha^2})t_N + x_0 \quad (13)$$

$$Y_N = -v_0\beta\alpha t_N + y_0 \quad (14)$$

$$Z_N = v_0(\sqrt{1-\beta^2})t_N - \frac{1}{2}gt_N^2 + z_0 \quad (15)$$

Equations in this form are much easier to be identified.

There are several ways to identify a system model. One simple way is called Linear Regression, see [10].

The equation of Linear Regression is

$$y_N = \Phi_N \vartheta + e_N$$

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

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

N is the total number of sample.

y_N and Φ_N are measurable and ϑ is unknown and is to be identified.

We transform our system to the form of Linear Regression.

$$\underbrace{\left[X_k \quad Y_k \quad (Z_k + \frac{1}{2}gt_k^2)\right]}_{y_N} = \underbrace{\begin{bmatrix}1 & t_k\end{bmatrix}}_{\Phi_N} \cdot \underbrace{\begin{bmatrix} x_0 & y_0 & z_0 \\ -v_0\beta(\sqrt{1-\alpha^2}) & -v_0\beta\alpha & v_0(\sqrt{1-\beta^2}) \end{bmatrix}}_{\vartheta} \tag{16}$$

The unbiased estimate of θ, is calculated with the following equation.

$$\hat{\vartheta} = \Phi^+ y \qquad \text{where} \quad \Phi^+ = (\Phi^T\Phi)^{-1}\Phi^T$$

This equation renders least-square solution of $\hat{\vartheta}$.

Since we have known the value of the matrix, we can find out the value of all unknown parameters in the matrix. The value of the unknown parameters x_0, y_0 and z_0 can be determined straight forward from the matrix but other parameters are difficult to do so.

To determine unknown parameters v_0, β and α, the following equations have to be solved simultaneously.

$$a = -v_0 \beta(\sqrt{1-\alpha^2}) \tag{17}$$

$$b = -v_0 \beta\alpha \tag{18}$$

$$c = v_0(\sqrt{1-\beta^2}) \tag{19}$$

where a, b and c are the value from the matrix.

Therefore, from equation (18)

$$b^2 = v_0^2 \beta^2 \alpha^2 \tag{20}$$

from equation (17),

$$a^2 = v_0^2 \beta^2 (1-\alpha^2)$$

$$a^2 = v_0^2 \beta^2 - v_0^2 \beta^2 \alpha^2 \tag{21}$$

from equation (19),

$$c^2 = v_0^2 (1-\beta^2)$$

$$c^2 = v_0^2 - v_0^2 \beta^2 \tag{22}$$

Substitute (20) into (21), we get

$$a^2 = v_0^2 \beta^2 - b^2$$

$$v_0^2 \beta^2 = a^2 + b^2 \tag{23}$$

Substitute (23) into (22), we get

$$c^2 = v_0^2 - (a^2 + b^2)$$

$$v_0^2 = a^2 + b^2 + c^2 \tag{24}$$

Substitute (24) into (23), we get

$$(a^2 + b^2 + c^2)\beta^2 = a^2 + b^2$$

$$\beta^2 = \frac{a^2 + b^2}{a^2 + b^2 + c^2} \tag{25}$$

Substitute (23) into (20), we get

$$b^2 = (a^2 + b^2)\alpha^2$$

$$\alpha^2 = \frac{b^2}{a^2 + b^2} \tag{26}$$

Therefore,

$$v_0 = \sqrt{a^2 + b^2 + c^2} \tag{27}$$

$$\beta = \sqrt{\frac{a^2 + b^2}{a^2 + b^2 + c^2}} \tag{28}$$

$$\alpha = \sqrt{\frac{b^2}{a^2 + b^2}} \tag{29}$$

Remind that we have assumed $\beta = cos(\theta)$ and $\phi = cos(\alpha)$. Hence,

$$\theta = \cos^{-1}(\beta) \tag{30}$$

$$\alpha = \cos^{-1}(\alpha) \tag{31}$$

All unknown parameters have been determined now.

6.2 Ball Grasp Position Prediction

There are many methods to predict the ball grasp position in 3D Cartesian space. A simple way is proposed in this thesis, which it is done by fixing the X-coordinate and to predict the Y and Z coordinates as well as the sample time. This way to limit the X-coordinate makes the prediction much easier and in fact it simplifies a 3D prediction problem to become a 2D prediction problem.

The cameras will capture 20 images of the ball trajectory; each image taken is considered one sample.

Samples 1 to 6 of the ball trajectory are used for system identification to identify the model parameters. These parameters are then used in the prediction to predict the ball grasp position. As the reason that the x-coordinate is being fixed, therefore the y and z coordinates and the ball grasp time can be easily predicted. This whole thing repeats for the sample 1 to 7 and so on until sample 1 to 20. This means that there are 15 predicted ball trajectories and while the prediction is being done, the robot will move on the x-coordinate plane to 15 different predicted ball grasp position.

The more sample be used for the prediction will result more accurate ball grasp position to be predicted. This also means our error of the predicted ball grasp position is reducing as the prediction with more samples go on. So for the robot to be able to grasp the ball relies on how good is the prediction.

Through simulation this prediction has proved to be successful and it has rendered good performance with less error. The robot is able to grasp the ball on the fixed x-coordinate plane.

In the experiment, the x-coordinate of the ball grasp position is fixed at 1m away from the robot. So the robot has to be able to grasp the ball as the ball reaches the x-coordinate plane. From our Newtonian dynamic equations, by fixing the predicted x-coordinate, it is possible to predict the y and z coordinate as well as the time when the ball reaches the x-coordinate plane.

The predicted x-coordinate X_P is fixed to be equal to –3, which is 3m away from the origin and it corresponds to 1m away from the robot.

By rearranging our Newtonian dynamic equation the predicted time t_P can be calculated easily while the predicted y-coordinate and the predicted z-coordinate are to be calculated as usual way. These predicted equations are shown below:

$$t_P = \frac{X_P - x_0}{-v_0 \cos(\theta)\sin(\phi)} \tag{32}$$

$$Y_P = -v_0 \cos(\theta)\cos(\phi)t_P + y_0 \tag{33}$$

$$Z_P = v_0 \sin(\theta)t_P - \frac{1}{2}gt_P^2 + z_0 \tag{34}$$

6.3 Inverse Kinematics

The forward kinematics calculates the position and orientation of the robot end-effector with the input of the robot joint angles while the inverse kinematics calculates the robot joint angles with the input of the robot end-effector position and orientation [12].

In the project, the inverse kinematics is used to calculate the six robot joint angles given the predicted ball grasp position. These joint angles are then sent to the robot mechanism to move the robot end-effector to the ball grasp position.

A Matlab program called **invkin2400.m** was used for doing the inverse kinematics calculation. This program was written by Anders Robertson from the Automatic Control Department in the University of Lund.

Before doing the inverse kinematics, we need to convert the predicted ball grasp position (X_P, Y_P, Z_P) to be relative to the robot coordinate system as it is relative to the world coordinate.

The conversion from the world coordinate system to the robot coordinate system is shown below:

$$Robot\_coordinate = inv(robot\_Rt) * \underbrace{\begin{bmatrix} 0 & 0 & 1 & X_P \\ 0 & 1 & 0 & Y_P \\ -1 & 0 & 0 & Z_P \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\text{Gripper Rt matrix}}$$

where *robot_Rt* is a 4-by-4 matrix which represents the orientation and the position of the robot. The gripper Rt matrix represents the gripper orientation (the first three rows and columns) and the gripper position (the first three rows and the fourth column). Notice that the gripper position is the predicted ball grasp position. This is reasonable as we want to move the gripper to the predicted ball grasp position to catch the ball.

The calculated Robot_coordinate is a 4-by-4 matrix, from the matrix the ball grasp position (the first three rows and the fourth column) relative to the robot coordinate system is taken. The ball grasp position is in meter. However the calculation of the inverse kinematics requires input parameters to be scaled in millimeter. Therefore to convert meter to millimeter, the ball grasp position is multiplied by 1000.

In the inverse kinematics program, it is also required to input the gripper length, which is set to be 300 mm.

When the 6 robot joint angles in relation to the ball grasp position is calculated, the joint angles are all be scaled in radian. It is necessary to scale them all in degree as the robot has been programmed to take in joint angles in degree.

The function for the inverse kinematics is shown below:

q = invkin2400(Robot_coordinate, front, noflip, gripper_length)*180/pi

7. Results

7.1 Graphical result of Camera Placement

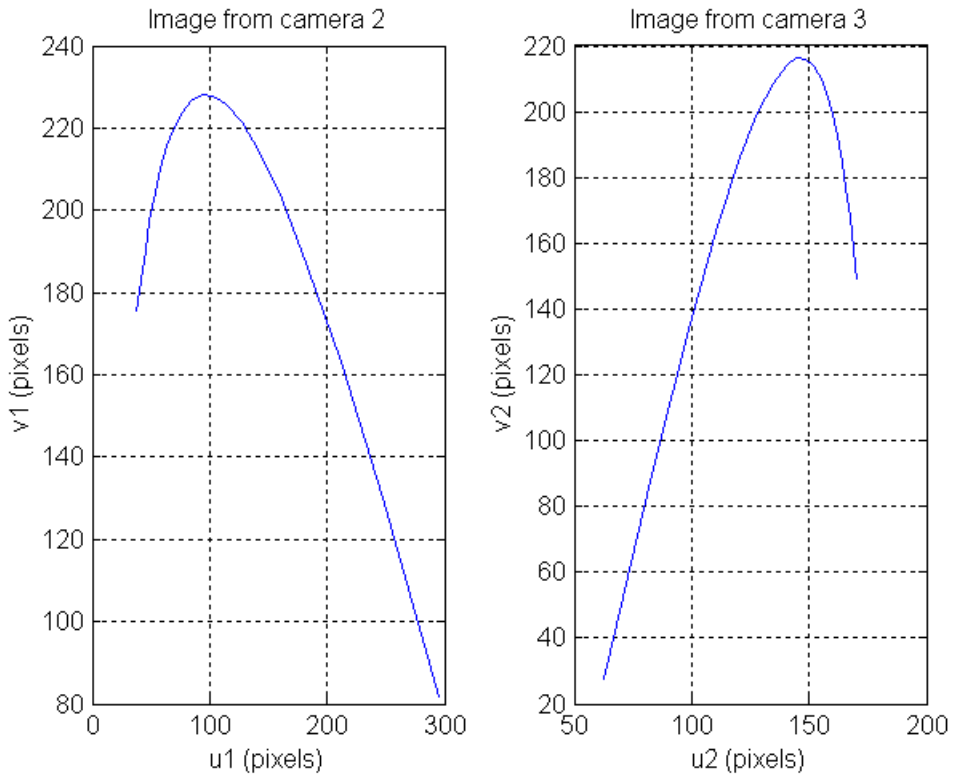

Figure 11: Images of ball trajectory captured by the two cameras.

The graphs shown in Figure 11 are the 2D image of the ball trajectory taken by each respective camera. The u and v are the image coordinates in pixels. From the two graphs we can see the whole of the ball trajectory from the start point to the end point, this indicates that our placement of the two cameras is perfect and this kind of placement allows 3D image reconstruction by Triangulation to be done. Hence, these two images are taken for 3D image reconstruction.

7.2 Graphical result of Feature Point Extraction

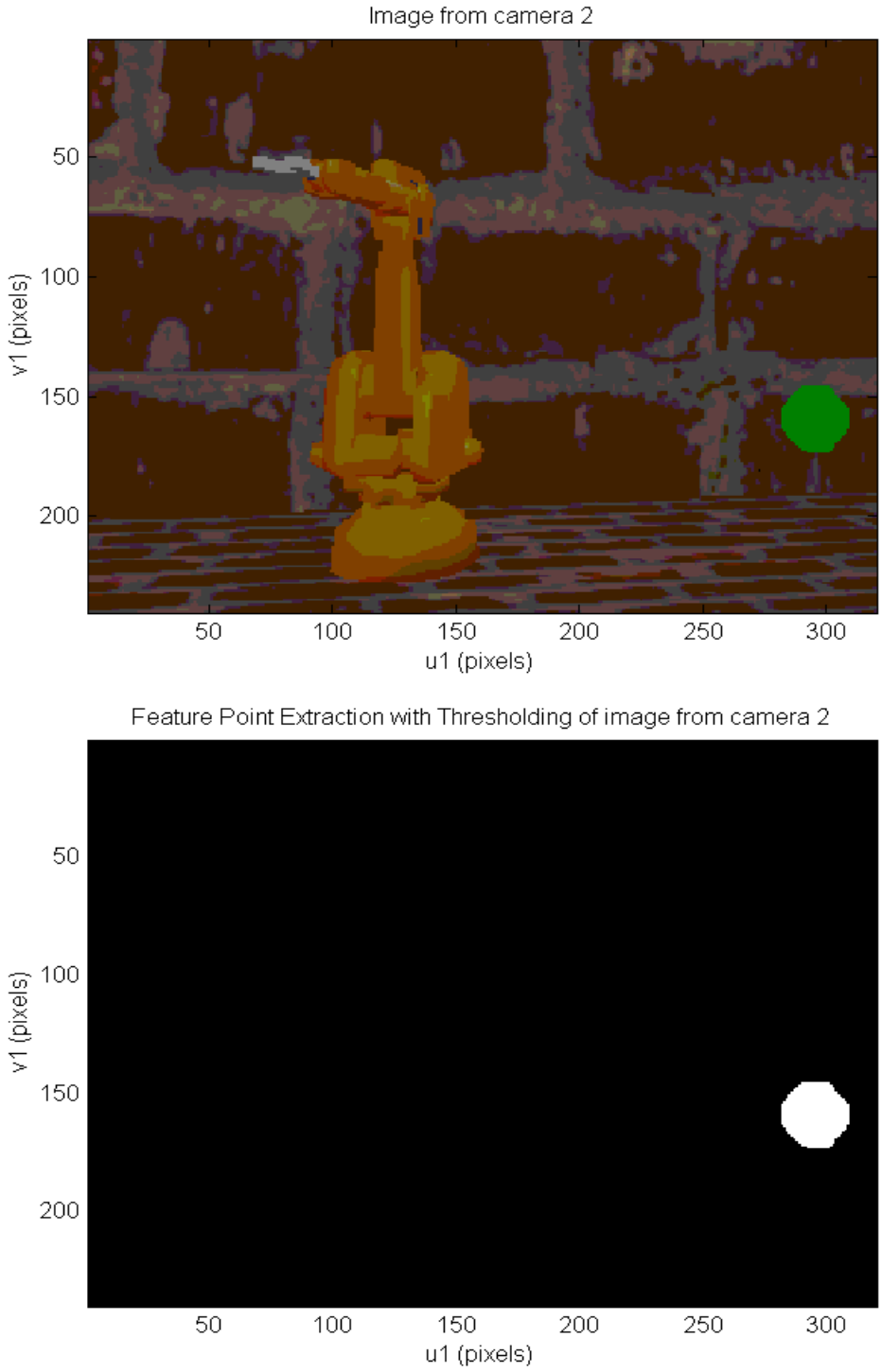

Figure 12: Image before and after Feature Point Extraction.

In figure 12, two images are shown. The image at the top is the image captured by the virtual camera while the image at the bottom is the image after Feature Point Extraction with Thresholding. The green ball in the first image (image at the top) was identified by Feature Point Extraction, which scanned the first image for image pixel contains pure green element. After the ball position in image has been identified, the ball was segmented from the background by setting a threshold limit. This process of finding and setting a proper threshold limit is called Thresholding.

7.3 Graphical results of 3D Reconstruction

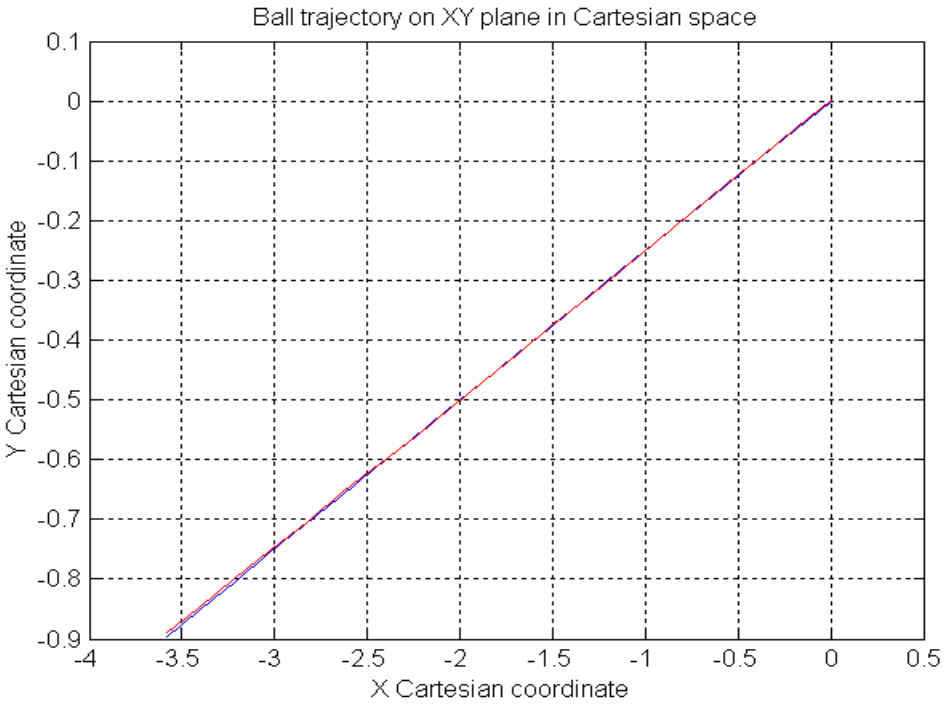

Figure 13: Ball trajectory on XY plane in Cartesian space.

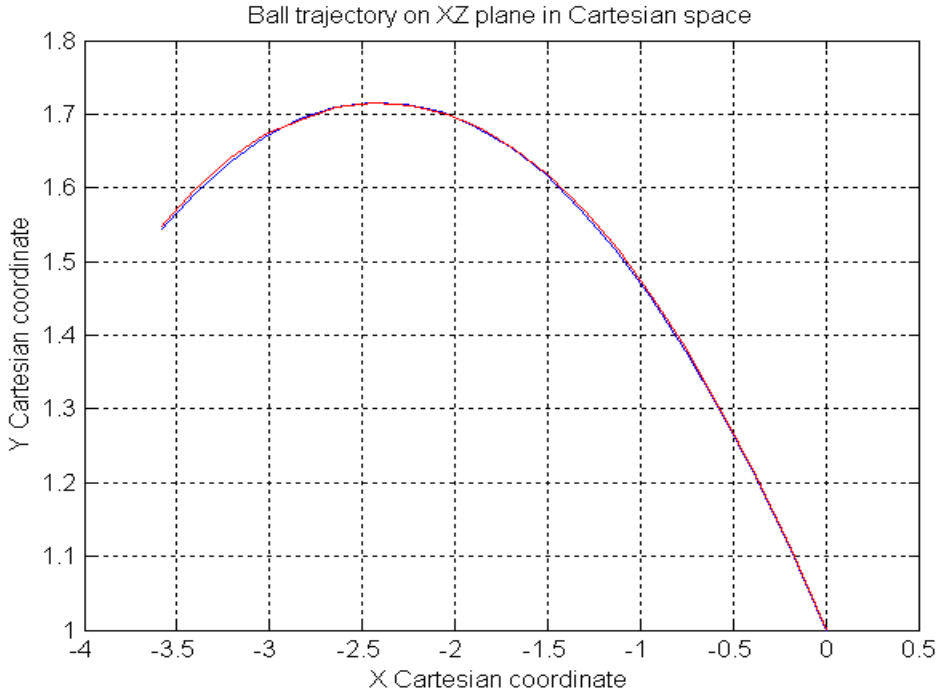

Figure 14: Ball trajectory on XZ plane in Cartesian space.

The graph shown in figure 13 is the comparison of the simulated (blue line) and the 3D reconstructed (red line) ball trajectory on xy plane in Cartesian space while the graph shown in figure 14 is the similar one but on xz plane.

From these two graphs we see how close is the 3D reconstructed ball trajectory to the simulated ball trajectory. This indicates the errors between the two trajectories are very small. Thus it shows good performance of our 3D reconstruction algorithm.

7.4 Graphical Results of System Model Identification

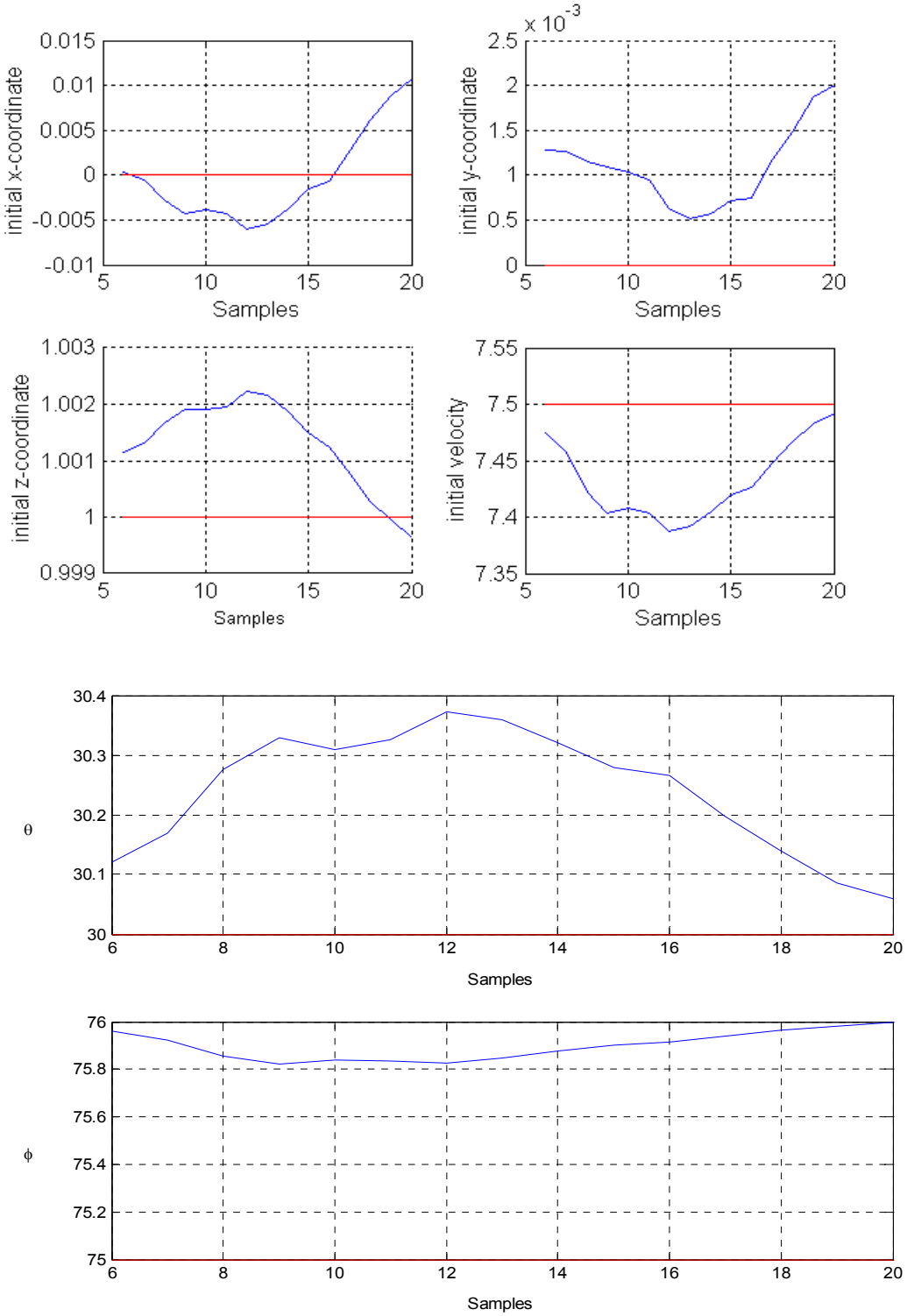

Figure 15: Performance of system identification.

The graphs shown in figure 15 indicate the performance of our system identification. The vertical axis of the graphs is the system model parameters to be identified with Linear Regression method. The horizontal axis of the graphs is the number of sample been used to identify the respective system parameter.

The red lines show the set parameter value of the simulated ball trajectory while the blue lines show the results of the system identification obtained with the respective number of sample.

From the graphs, our system identification is working well and renders good performance as we can see the error between the blue lines and the red lines is relatively small.

7.5 Graphical result of Model-based Prediction

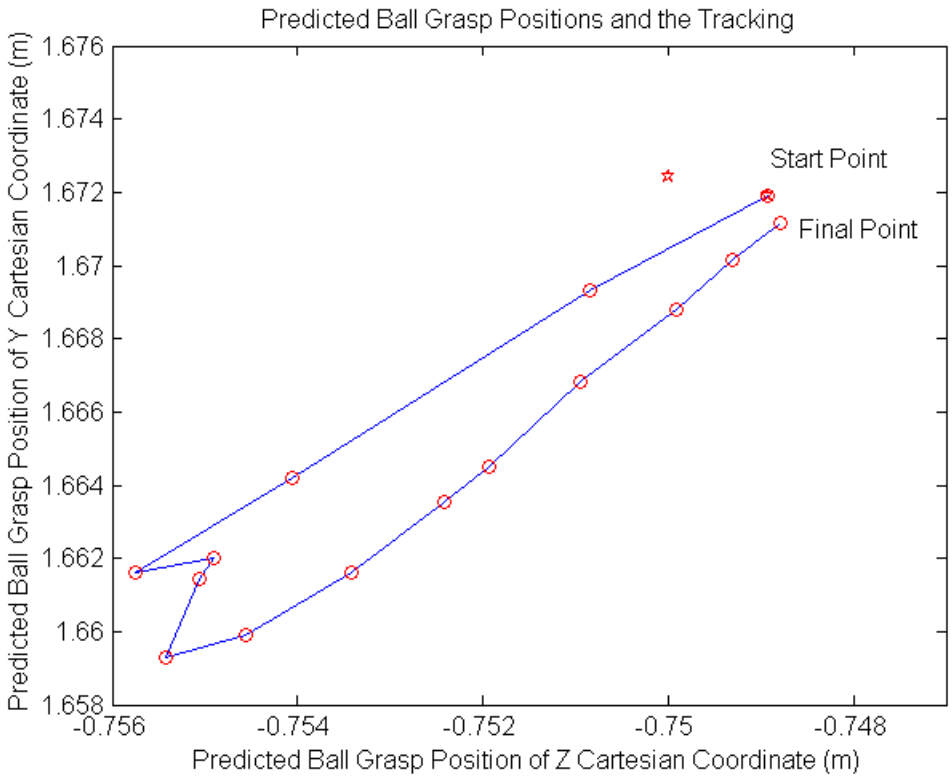

Figure 16: Predicted ball grasp positions and the tracking.

The graph shown in figure 16 shows all the predicted ball grasp position (red circle in the graph) and the tracking (blue line in the graph) done by the robot end-effector from the start point until the end point where the ball is finally grasped. The red star in the graph is the actual position where the robot should grasp the ball. Though the variation between predicted grasp points looks very large in the graph but they are actually relatively small in scale.

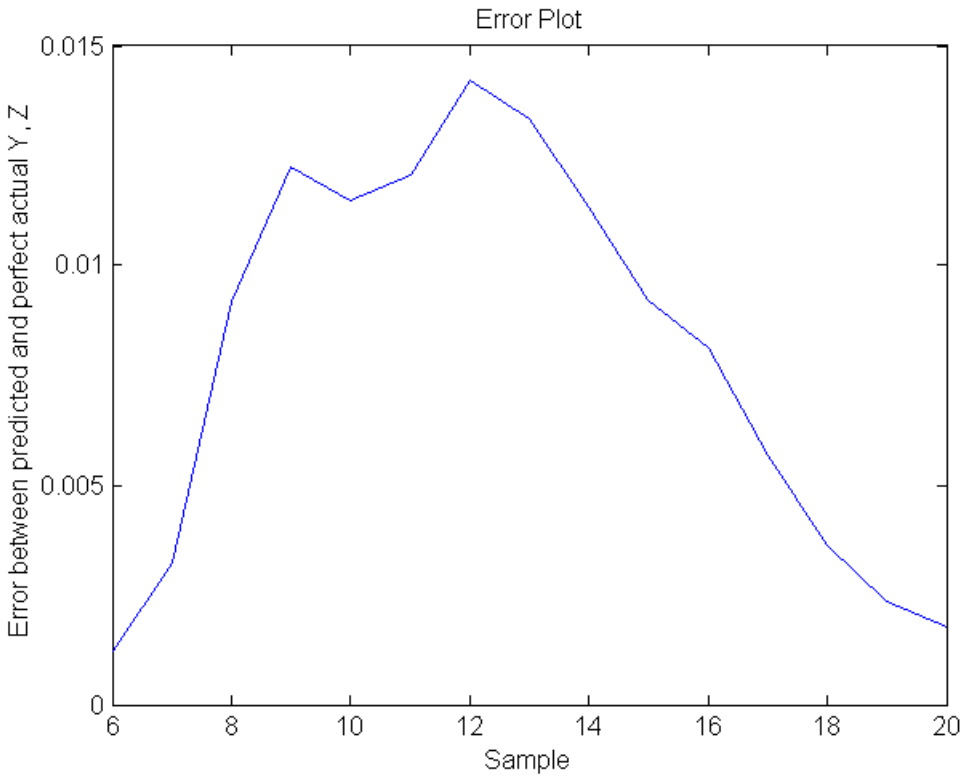

Figure 17: Error between predicted Y,Z and actual Y,Z.

The graph shown in figure 17 is the plot of error (between the predicted and the actual Y, Z Cartesian coordinates) against samples. Though the graph shows a high peak and seems to have very large error, they are actually very small in scale. The highest error shown at sample 12 is merely 0.012.

As the error is small, we can therefore say that our model-based prediction renders good perfomance.

7.6 Graphical Analysis of Robustness.

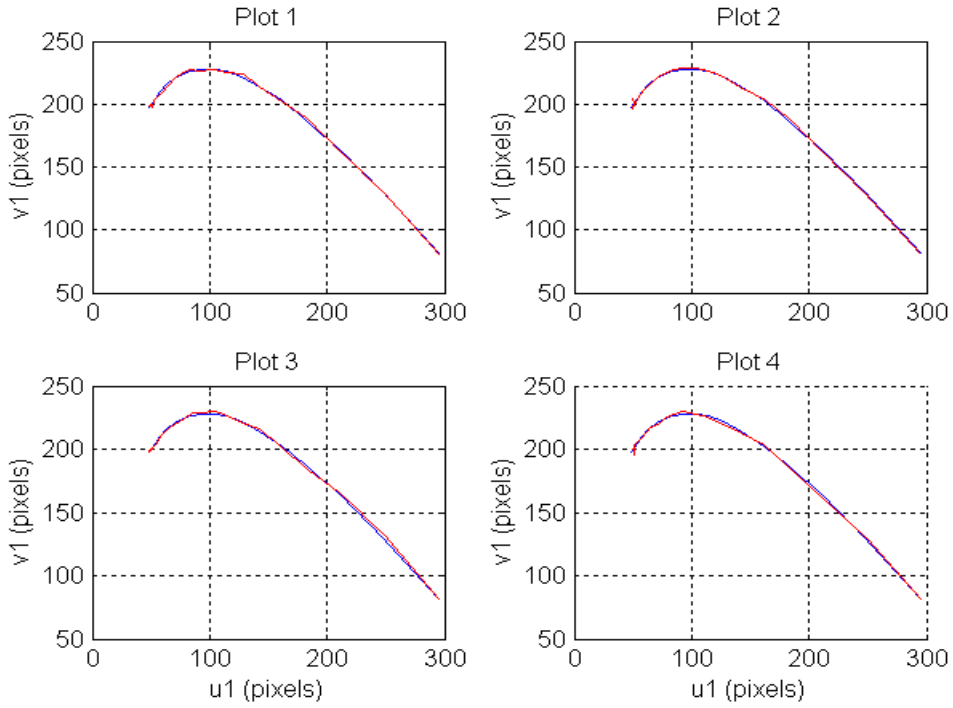

Figure 18: Robustness analysis 1.

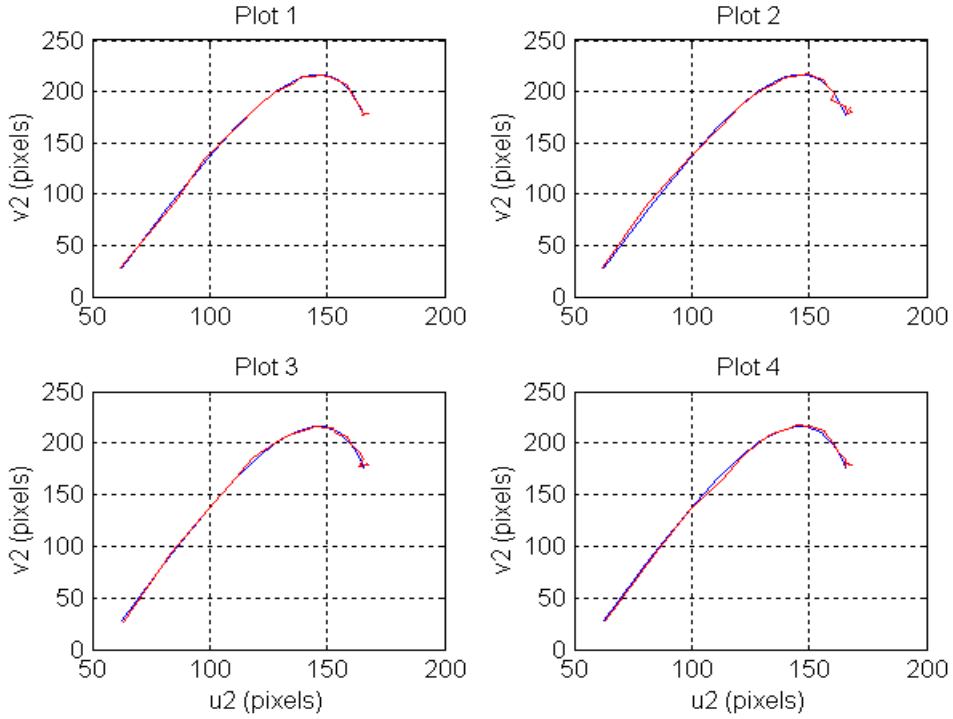

Figure 19: Robustness analysis 2.

The robustness of the overall system is studied by adding noise (normal distributed random numbers) into the two images captured by the two virtual cameras respectively. Four experiments (by adding noise) to each image of the two images were carried out.

The four graphs shown in figure 18 show the four experiments of the ball trajectory with and without noise, captured by the camera 2 in image. The red graphs are the ball trajectory with noise while the blue graphs are the ball trajectory without noise.

The same for figure 19 but it is the four experiments of the ball trajectory with and without noise, captured by the camera 3.

From all the graphs shown in figure 18 and 19, we can see that the variation and the difference between the trajectory with noise and the trajectory without noise is small and by simulations we have seen the virtual robot is able to grasp the ball at most of the time in the presence of noise, therefore we can conclude that our overall visual servoing system does show some sights of robustness.

7.7 Visual Images from The Virtual World

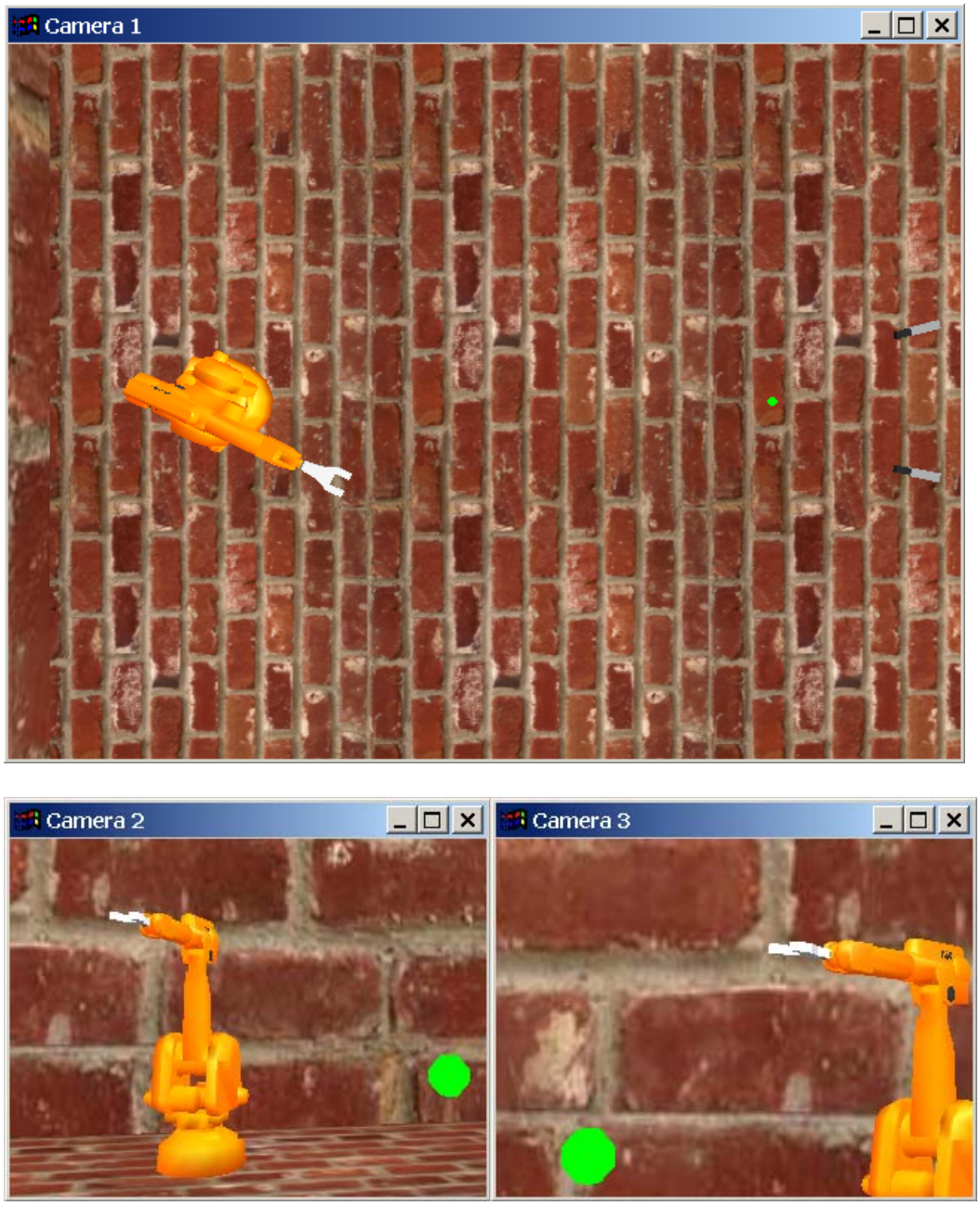

Figure 20: Starting stage of simulation.

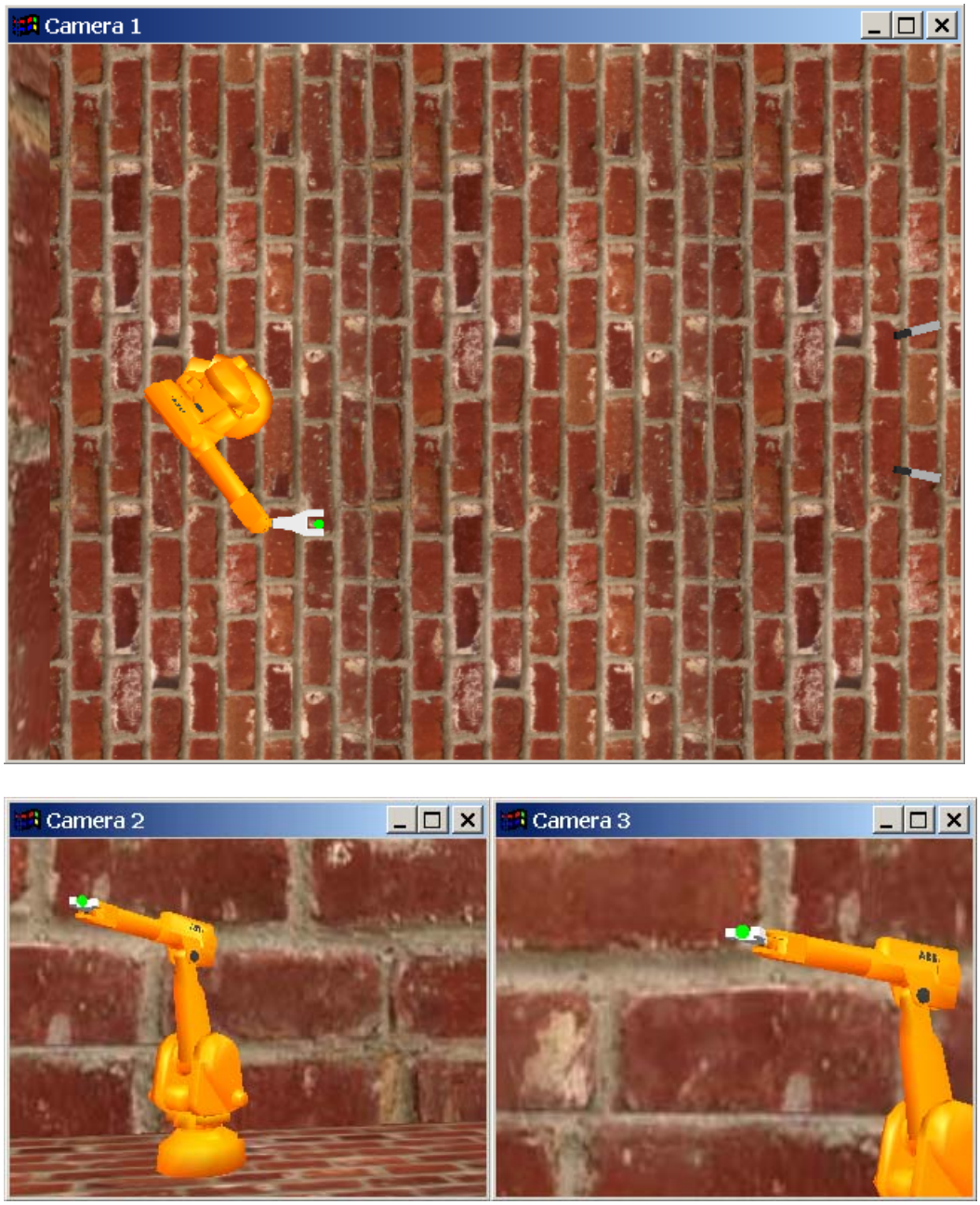

Figure 21: Robot Irb2000 grasps the ball.

The starting stage of the simulation is shown in figure 20. Camera 1 shows the experimental setup from the top view. Camera 2 and Camera 3 show the initial position of the ball in the respective images.

The way that the virtual robot grasps the ball is shown in figure 21. These images are the evidence to show that our system actually works.

8. Discussions

8.1 Different Software Environment Different Definition of Coordinate System

A rather unexpected and confused problem arose when we were transferring object data from the virtual environment in Visual C++ to the Matlab environment. The problem was caused by the different definition of the coordinate system taken by different software environment. So Visual C++ and Matlab both have different definition of object coordinate system.

This problem was analyzed and solved by flipping or inverting the y-axis and z-axis of the object coordinate system in Visual C++ before it is sent over to the Matlab environment for computation.

8.2 Different Software Environment Different Definition of Image Origin

An unusual problem occurred when we were doing the Image Processing and trying to send images from Visual C++ to Matlab environment. The problem is about the different definition of the image coordinate origin in the Visual C++ environment and the Matlab environment. After investigations, it has been found that Matlab defines the image origin at the bottom left corner of the 240x320 image, whereas in Visual C++, the image origin is defined at the top left corner of the 240x320 image.

As the reason, in order to transfer image from Visual C++ to Matlab environment, the v of the image coordinate (u,v) of the image from Visual C++ has to minus 240, which is the total amount of pixel in the vertical of the image.

8.3 Different Software Environment Different Structure of Character Arrangement

We encountered a problem of losing image information when we were doing Feature Point Extraction in Matlab environment with image data from Visual C++ environment. This problem was due to the different structure of pixel character of image arranged in Visual C++ and Matlab environment. The 256 characters in Visual C++ is 'signed' and is arranged from –128 to 127, whereas the 256 characters in Matlab is 'unsigned' and is arranged from 0 to 256. As the reason, half of the image information is lost when an image is sent from Visual C++ to Matlab. This problem is overcome by having the image processing done in Matlab using only the unsigned character ranging from 0 to 127 to restore the image.

9. Conclusion

In this project we have presented the computer simulation of model-based visual servoing control of a 6 DOF industrial robot.

The project is made up of several subtasks. These subtasks are Feature Point Extraction, 3D image reconstruction, system model identification, model-based prediction of ball trajectory and ball grasp position.

Problems described in the problem formulation related to these subtasks have been solved.

Other unexpected problems we have encountered throughout this project are discussed in chapter 8. These problems are mainly software related and have all been solved through fault-findings.

By many simulations, we have shown that our system is able to control the robot end-effector to grasp a ball moving by Newtonian dynamics in 3D. This has proven that our system is working well and also has fulfilled the main objective of the project. The performance and the robustness of the system have also been tested by graphical analysis.

From the results shown in chapter 7, we can conclude that we have managed to build a system, which is rather robust and renders high performance.

10. Future Works

It is true that there is a lot of space for improvement on the experiment performed. Currently the visual servoing of the robot is done with just the model-based prediction and has no controller. Although the system has been proven to have little robustness by simulation, it is still quite weak to resist noises and perturbations in the real world. Therefore the first priority for future work is to build a controller to minimize errors and improve the robustness of the system. Controller such as PID controller is suggested. It is simple, easy to tune and be able to give good performance.

After that, when the system is strong enough to resist noises and perturbations, implementation of the system to the real robot in the robot laboratory may then be carried out.

Improvement can also be done on the way that the grasping is performed. Grasping at a fixed position or a coordinate plane may not be the ideal solution, but it has proven to be the most simple and effective way to grasp the ball in the project.

Furthermore, the time taken of the Feature Point Extraction may be reduced by applying Window-based tracking technique, see [5].

Although the system is not perfect and requires improvement, we must emphasize on the fact that the system is a very good base for future investigations of visual servoing.

11. References

[1] Georg von Wichert.
"Can robots learn to see?", Siemens Corporate Technology, Intelligent Autonomous Systems, ZT IK 6, D-81730 Munich, Germany, June 1998.

[2] Bernard Espiau, Franqois Chaumette and Patrick Rives.
"A new approach to visual servoing in robotics", IEEE Transactions on robotics and automation, vol. 8, no. 3, June 1992.

[3] Luis Manuel Conde Bento and Duarte Miguel Horta Mendonca.
"Computer vision and kinematic sensing in robotics", Master Thesis, Department of Automatic Control, University of Lund, June 2001.

[4] Michail Bourmpos.
"Vision based robotic grasping tracking of a moving object", Master Thesis, Department of Automatic Control, University of Lund, September 2001.

[5] Seth Hutchinson, Gregory D. Hager and Peter I. Corke.
"A tutorial on visual servo control", IEEE Transactions on robotics and automation, vol. 12, no. 5, October 1996.

[6] Youcef Mezouar and Franqois Chaumette.
"Path planning in image space for robust visual servoing", IEEE International Conference on robotics and automation, April 2000.

[7] Tomas Olsson.
"Vision guided force control in robotics", Master Thesis, Department of Automatic Control, University of Lund, October 2001.

[8] www.alliancevision.com /produits/cameras/sony_dfw_v300.htm

[9] MATLAB, high-performance numeric computation and visualization software reference guide, The Math Works, Inc., August 1992.

[10] Rolf Johansson.
"System modeling and Identification", Prentice Hall, January 1998.

[11] George Weiss.
"System Identification", lecture handouts, MSc in control systems, Imperial College, University of London.

[12] Anders Olsson and Sara Liljenborg.
"Identify of a surface with robot force control", Master Thesis, Department of Automatic Control, University of Lund, November 2000.

[13] Abhijit Nagchaudhuri, Sastry Kuruganty and Asif Shakur.
"Introduction to mechatronics concepts in a robotic course using an industrial SCARA robot equipped with a vision sensor", Mechatronics 12, p.g. 183 – 193, 2002.

[14] John J. Graig.
"Introduction to Robotics Mechanics & Control", Addison – Wesley, 1986.

[15] Lennart Ljung.
"System Identification Theory For The User", Prentice – Hall Information And System Science Series, 1987.

12. Appendices

Appendix A: Cartesian coordinate transformations

In robotics, all objects are rigidly attached to a coordinate system, or frame. In our case, these objects are the cameras, the robot, the end-effector and the objects to be grasped by the robot.

The coordinate system helps to describe the position and orientation of an object in space. All positions and orientations are described with respect to some reference coordinate system, usually the world coordinate system or with respect to other Cartesian coordinate systems, which are defined relative to the world system [14].

We will often need to relate the position and orientation of the frames / objects in the workspace to each other, or transform coordinates from one frame to another [7]. If we want to change coordinate system from frame b to frame a, we need the transformation matrix, T_b^a, which describes the location of frame a relative to frame b. The transformation matrix, T_b^a consists of a 3-by-1 translation vector, t_b^a and a 3-by-3 orthonormal rotation matrix that satisfying

$$(R_b^a)^T R_b^a = I$$

$$\det(R_b^a) = 1$$

The vector t_b^a is the vector from the origin of a to the origin of b, expressed in the coordinate system of a, see figure 22. The column vectors in the matrix R_b^a are the unit x, y and z vectors of the frame b, expressed in the coordinate system of a as

$$R_b^a = \left((X_b)^a, (Y_b)^a, (Z_b)^a\right)$$

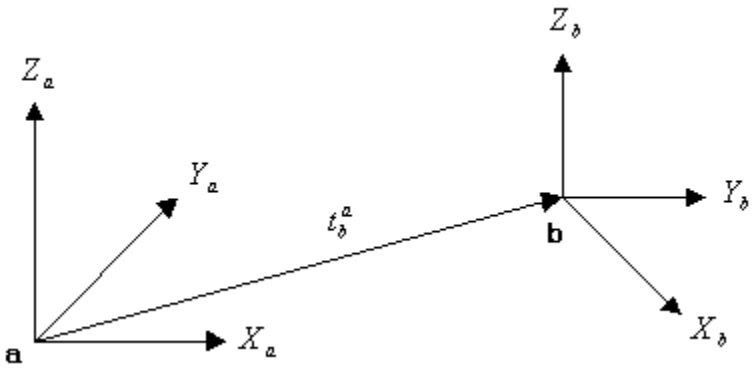

Figure 22: Transformation between frames.

The equation for the transformation from frame b to frame a is

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

By using homogeneous coordinates, the equation is rewritten as

$$\begin{bmatrix} X^a \\ Y^a \\ Z^a \\ 1 \end{bmatrix} = \left[\begin{array}{ccc:c} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \begin{bmatrix} X^b \\ Y^b \\ Z^b \\ 1 \end{bmatrix}$$

where we let

$$T_b^a = \begin{pmatrix} R_b^a & t_b^a \\ 0_{1x3} & 1 \end{pmatrix} = \left[\begin{array}{ccc:c} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right]$$

Therefore

$$\begin{bmatrix} X^a \\ Y^a \\ Z^a \\ 1 \end{bmatrix} = T_b^a \begin{bmatrix} X^b \\ Y^b \\ Z^b \\ 1 \end{bmatrix}$$

or simply

$$Q^A = T_b^a Q^b$$

We can also combine several transformation in series

$$Q^a = T_b^a Q^b = T_b^a T_c^b Q^c = T_c^a Q^c$$

or invert the transform

$$Q^b = T_a^b Q^a = (T_b^a)^{-1} Q^a$$

where the inverse transformation can be computed easily

$$(T_b^a)^{-1} = \begin{pmatrix} (R_b^a)^T & -(R_b^a)^T t_b^a \\ 0_{1x3} & 1 \end{pmatrix}$$

Appendix B: How to use Matcomm functions

Matcomm is a software that was developed at the Department of Automatic Control in the Lund Institute of Technology. This software uses the TCP/IP protocol to transmit data between computers, in a network where different systems might exist.

The commands for the usage of the matcomm functions in the Unix system are:

ID = matcomm('server', process)

- To create a server called by process e.g."apple" that listens for incoming requests. Process is a string containing the name of the target process. "ID" is the reference number to use for subsequent access to the communication line.

ID = matcomm('open', target, process)

- To open a communication line with process e.g. "apple". Target e.g. "euler" is the host to connect.

matcomm(ID, data)

- To transmit data called "data" to the "ID".

data = double(matcomm(ID))

- To read information in the form of double from "ID".

matcomm('close', ID)

- To close communication line with "ID".

The commands for the usage of the matcomm functions in the Windows system are:

MatCommWin32Int( );

- To initialize the WINSOCK2 library and it must be called before using any Matcomm functions in WIN32.

matcomm_id = MatCommOpen(char host, char process, int timeout);

- To open a communication line to the selected process e.g. "apple" in the host e.g. "euler.control.lth.se". The call terminates after timeout e.g. 60 seconds.

MatCommWriteDouble(matcomm_id, double data, int rows, int columns);

- To transmit data in the form of double, which has the matrix dimensions with rows and columns with the reference called "matcomm_id".

MatCommReadDouble(matcomm_id, double data, int rows, int columns);

- To read data in the form of double, which has the matrix dimensions with rows and columns with the reference called "matcomm_id".

MatCommClose(matcomm_id);

- To close communication line with "matcomm_id".

Appendix C: Perspective Projection of a pinhole camera

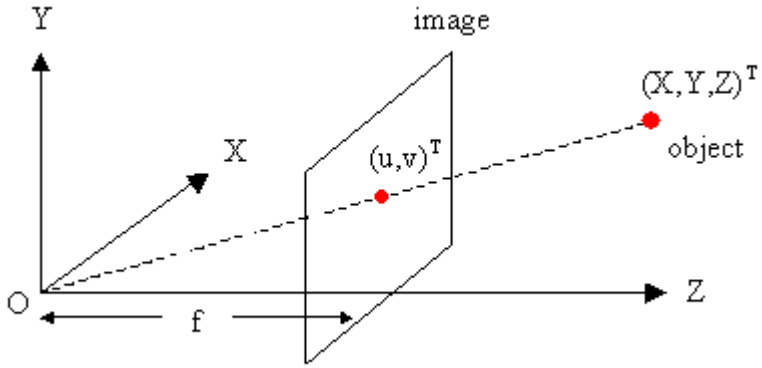

Figure 23: The pinhole camera model.

The most common camera model is the pinhole or perspective camera. It is a box that has an extremely small hole through which light enters and projects an inverted image on the camera back plane. To simplify things, we usually model a pinhole camera by placing the image plane between the focal point of the camera and the object, so that the image is not inverted. This mapping of three dimensions onto two is called perspective projection [3]. In figure 23, an object in Cartesian space is projected onto an image plane with perspective ray originating at the center of projection, O. The distance between the center of projection and the image plane along the principal axis or optical axis (Z-axis) is the

focal length, f. The u and v are the image coordinates while the X, Y and Z are the Cartesian coordinates of the object.

The projection equations for a point $(X,Y,Z)^T$ in Cartesian space in the perspective camera are given by

$$u_i = f\frac{X}{Z} \qquad\qquad v_i = f\frac{Y}{Z}$$

These equations can be easily expressed by introducing homogeneous transformations, which is a matter of placing Euclidean geometry into the projective geometry space.

The projection equations are rewritten as

$$\lambda\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where $\lambda = Z$ is the depth of the imaged point in the camera.

To transform between image-plane coordinates $(u,v)^T$ to pixel-coordinates in the camera, six extra camera parameters called intrinsic parameters are required. These parameters describe the Charged-Coupled Device (CCD) in the camera and allow us to describe non-quadratic pixels and skew [7].

These six intrinsic parameters are the focal length in pixels, f, the x-coordinate of the center of projection, u_0, the y-coordinate of the center of projection, v_0, the scaling of the image plane along the u, α, the scaling of the image plane along the v, β, the skew between the optical axes, γ [3].

The new camera model becomes

$$\lambda\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f\alpha & f\gamma & u_0 & 0 \\ 0 & f\beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\lambda\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = P_{3x4}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where P is a 3-by-4 projective transformation matrix of the camera.

As P has the relationship, P = K(R|t) with the intrinsic camera matrix, K and the extrinsic camera matrix (R|t), the above equation can also be written as

$$\lambda\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f\alpha & f\gamma & u_0 \\ 0 & f\beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\lambda\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K_{3x3}(R \mid t)_{3x4}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

where R of (R|t) is a 3-by-3 orthonormal rotation matrix (or orientation matrix of the camera) and t of (R|t) is a 3-by-1 3D translation vector (or position vector of the camera). See Appendix A for further information of the (R|t) matrix.

Acknowledgement

In sincerity and great gratitude, I would like to thanks Professor Rolf Johansson from the Department of Automatic Control in the University of Lund and Dr. Martin Clark from the Department of Electrical Engineering in Imperial College, University of London for giving me this memorable and invaluable opportunity to undertake my master thesis as an ERAMUS student in the Department of Automatic Control, University of Lund, Sweden.

During the course of this project, I was supervised by three experienced PhD students, namely Mathias Haage, Johan Bengtson and Tomas Olsson. Mathias is from the Department of Computer Science and is an expert in programming languages while Johan and Tomas are from the Department of Automatic Control, both of them are knowledgeable in robotic control. In specific, Johan is also well versed in system identification.

In time of despair in doing the project, they have been very helpful to me. Without their ingenious guiding, unconditional help and patience I could not be possibly to accomplish this thesis work successfully and exploit my interest in robotics. I have to admit I have gained profusely from them and would like to give thanks and offer praise to them.

Furthermore, I would like to thank my project partner, Jee Hui Wong. He has been a great partner of mine, being work together with him was my pleasure. I am grateful that

he has gone through this with me. Also I must thanks Leif Anderson for he has been very kind to help me in resolving any computing problems I encountered.

Finally, thanks to my parents. Without their emotional and financial support, I might not be achieved this far and have all these!

"We give thanks to God always for you all, making mention of you in our prayers"

From the holy bible, Thessalonians 1:2.

"Gravitation can not be held responsible for people falling in love" of research.

Albert Eistein

"To me there has never been a higher source of earthly honor or distinction than that connected with advances in science."

Isaac Newton

END.