Robotic Catching and Manipulation Using Active Vision

by

Won Hong

Bachelor of Science, University of California at Berkeley, 1993

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1995

Author .....
Department of Mechanical Engineering
August 11, 1995

Certified by .....
Jean-Jacques E. Slotine
Associate Professor of Mechanical Engineering and Information Sciences
Thesis Supervisor

Accepted by .....
Ain A. Sonin
Chairman, Departmental Committee on Graduate Students

Robotic Catching and Manipulation Using Active Vision

by

Won Hong

Submitted to the Department of Mechanical Engineering
on August 11, 1995, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

Robot Hand/Eye Coordination and Active Vision are both fields which have enjoyed much attention. A variety of research has been completed which examines the use of vision to direct robot manipulation. Specifically, previous research at MIT has examined the task of combining vision and manipulation applied to the task of tracking and catching tossed balls in controlled environments. Building upon the foundations of this past research, this thesis presents work which incorporates a new *active* vision system which requires a minimally controlled environment and implements new methods for object tracking, robot/camera calibration, and new catching algorithms.

The system which is used here is composed of a seven degree of freedom cable driven arm and a ceiling mounted active vision system. The active vision system is composed of two color CCD cameras each mounted on two degree of freedom actuators. The vision processing is done using simple blob detection to locate color-keyed objects.

The goal of this research is to develop the control methods and additional algorithms required to complete successful catching of lightly tossed objects. The required elements to achieve successful catching are: (1) a vision system capable of locating objects, (2) controllers and robust tracking methods for the active vision system to stably track fast moving objects, (3) cross calibration between the vision system and manipulator, (4) path prediction for the tossed object, and (5) path generation for the manipulator to intercept the object.

This thesis addresses each of the required elements. Techniques for locating and robustly tracking objects using visual information are presented. Methods of cross calibration between the vision system and the manipulator are discussed. A recursive least squares algorithm for model-based path prediction of the tossed object is presented. And finally, methods for determination of safe catch points and new polynomial path generation techniques for the robot manipulator are discussed.

Experimental results for the application of the above algorithms to the task of catching free-flying spherical balls are presented. The system was tested on under-hand tosses from random locations approximately 1.5-2.5 meters distant from the arm. The average time of travel from leaving the hand of the tosser to successful catching is approximately 0.5 seconds. The best performance results were found to be 70-80% success for similar tosses.

Thesis Supervisor: Jean-Jacques E. Slotine
Title: Associate Professor of Mechanical Engineering and Information Sciences

Acknowledgments

For all the people who aided in this research, my heartfelt thanks. To Jean-Jacques Slotine and Ken Salisbury for their advice and guidance throughout the project. To Günter Niemeyer for his constant availability and willingness to help and his many thought provoking questions. To Akhil Madhani for his mechanical wizardry and helpful down-to-earth discussions.

I would also like to thank my parents. My father, SoonChul Hong, for his constant support throughout my life. My mother, BokHi Hong, for her frequent calls to see how I was doing. Also my brother, Paul Hong, for his encouragement.

Thanks also to my roommate over these past two years, Wesley Brewer, whose negative contributions to my research have made me a much better skier. Finally, last but *not* least, I would like to thank God.

This thesis documents research conducted at the Nonlinear Systems Laboratory and the Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. Support for this work has been provided in part by the Starr Foundation, by the Fujitsu Corporation, by NASA/JPL Contract No: 959774, by the Office of Naval Research, University Initiative Program Grant N00014-92-J-1814, by the Sloan Foundation, and by Furukawa Electric.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides an introduction to the contents of this thesis. The motivation and project goals for this research are presented first. Then, prior work on robotic catching with this system and similar projects at other sites are discussed. This chapter ends with a description of the organization of this thesis.

1.1 Motivation

There exist robots which are stronger, faster, more repeatable, etc. than humans. But with all these benefits, there are still dynamic tasks at which humans perform much better. One such task is catching. Human vision has a complexity which is not yet understood and the coordination and learning capabilities of humans are much greater than artificial methods which have been developed.

The purpose of this research is to conduct an experimental investigation of the coordination of vision and manipulation for static and dynamic tasks. Specifically, this thesis discusses the control and use of a new *active* vision system and the coordination of this system with a seven degree of freedom manipulator arm. The combined system is applied to the static and dynamic tasks of grasping stationary objects and catching of free-flying objects.

The hopes are that this system will provide a foundation for future study on hand/eye coordination and robot learning.

1.1.1 Project Goals

The primary goals for this research are classified into two categories: (1) the incorporation of the new active vision system, the Fast Eye Gimbals (FEGs), and (2) the coordination of the vision and manipulation to achieve successful catching of free-flying balls in 3D in a minimally controlled environment.

Incorporation of Active Vision

Since the previous work on this system, one of the major changes has been in the vision system. The incorporation of the new active vision system involves design and implementation of controllers for the Gimbals and the design of algorithms to utilize the stereo vision information. This research applies traditional PD and PID controllers and an Adaptive controller to the Gimbals and studies the relative performance of these controllers for trajectory tracking. This research also develops methods for using visual information to locate and focus attention on objects of interest. These methods are then extended to the task of tracking fast moving objects. This introduces additional challenges. Accurate compensation for time delays in the system and object state estimation are required. This research presents an object tracking algorithm which attempts to satisfy these requirements and perform robust object tracking.

Hand/Eye Coordination for Dynamic Tasks

In order to coordinate the vision system and the manipulator, a cross calibration is required and interface routines are needed. This research presents a simplified approach to cross calibration. Using this calibration, simple static tasks such as grasping stationary objects can be completed. For dynamic tasks, additional techniques are required. The path of the tossed object needs to be predicted for the full duration of the toss. Then, algorithms for using the object path prediction to select satisfactory catch points are required. And finally, a path generation scheme must be designed to direct the manipulator to intercept the object. These algorithms are presented here and applied to the ultimate project goal for this research, successful catching free-flying tossed balls.

1.1.2 Applications of Research

The initial development of the active vision/manipulator system was towards successful hand/eye coordination in order to be able to locate and grasp static objects. The goal being the use of the system for autonomous planetary rovers which would be capable of collecting geological samples.

The catching task was considered to gain a better understanding of the numerous individual components required for successful catching. The resulting methods from this research could expanded upon to be used in many different areas. Methods for simple cross calibration, object tracking, recursive least squares fitting, and path planning are all presented here. A possible relevant task is intercepting and catching free flying objects in space using a manipulator arm.

The primary application for this research, which is directly realized, is the establishment of a platform for further study. This research has developed a system which can and will be used for further research into active vision, vision guided manipulation, and robot learning.

1.2 History

1.2.1 Past Research With This System

The Whole Arm Manipulator (WAM) has been previously used for robotic catching research [Hove, Slotine 91, Kimura et al 92]. Using the WAM, combined with two stationary black and white cameras, successful catching results were presented [Hove, Slotine 91]. Windowed least squares methods were used to fit ball data to a parabolic path with assumed gravitational acceleration in the z direction. Path generation for the arm was accomplished through use of a time-varying second order filter.

Using the same system configuration as above, adaptive visual tracking algorithms were added [Kimura et al 92] which provided better vision information which improved catching reliability. These new algorithms used a vision window to track objects using adaptive and nonlinear control and identification techniques. In addition, a recursive least squares fit approach was implemented to fit ball data to a second-order polynomial in x, y, and z with no assumptions on acceleration.

The WAM is again used here in conjunction with a new *active* vision system. The new system is comprised of two high resolution color CCD cameras, each mounted on two degree of freedom gimbals. A new vision processing system is also added which is faster and much less sensitive to background. A new end-effector for the WAM is also used. Both the vision system and the end-effector are described in more detail in Chapter 2.

1.2.2 Research at Other Sites

There is not much current research on robotic catching specifically, but there is a wealth of research in the same general area of applying manipulators and vision systems for dynamic tasks.

As a result of the European ROBAT championship of robot ping pong [Billingsley 83], there exist many robots capable of playing ping pong [Andersson 87, Hashimoto et al 87, Fässler et al 90]. The ping pong system at the Swiss Federal Institute of Technology (ETH) [Fässler et al 90] is a good example of the capabilities of the robot ping pong players. Their stationary two vision camera system boasts an average position determination error of approximately 0.5 mm and is used to track balls moving up to 7 m/s. They use thresholded vision processing which outputs information at 50 Hz. to detect a white ball upon a black background. Their vision system is combined with a seven degree of freedom manipulator which is capable of moving over distances of 0.3 m in 0.1 s. The ping pong system benefits from a controlled environment and a stationary vision system requiring extensive calibration which allows it to have extraordinary accuracy. In contrast, our system is used in a much less controlled environment and a simpler visual calibration method is implemented. But with an associated cost of a less accurate vision system.

Another robot system which combines vision and manipulation, and also incorporates learning, is the robot at Osaka University which is capable of playing Japanese badminton [Watanabe et al 95]. This system consists of a stationary two camera vision system and a five degree of freedom manipulator which uses a simple learning algorithm to increase the performance of the system. Our system does not currently use learning, but there are plans to incorporate learning in a number of areas. The future research on the current system will use learning to move away from model-based path prediction of the tossed object and

to modify catching algorithms so that more items may be learned rather than programmed.

There are additional works which are similar to sub-elements of our system, such as active vision systems and robust object tracking methods. Some examples of other active vision systems are TRICLOPs at the National Institute of Standards (NIST) [Fiala et al 94] and Prism-3 at Teleos Research [Nishihara, Thomas 94]. Some examples of object tracking research are [Woodfill et al 94, Wavering, Lumia 93, Kalata 84]. In contrast to our system, these other active vision systems often have short baseline distances and often have tilt, vergence, and "neck" degrees of freedom. That arrangement has the benefit that more accurate mounting may be achieved, the system is more portable, and feature correlation is easier since the cameras have similar perspectives. In contrast, our system has less accurate mounting, but benefits from its large baseline. In addition, our system provides greater flexibility because each camera is independent. They may be individually moved or additional cameras may be added.

Research is also being done on the use of visual feedback to directly guide manipulation and on uncertainty tolerant precise positioning systems. Research at University of Illinois at Urbana-Champaign [Castaño, Hutchinson 94] uses visual feedback to position the end of a manipulator along a visual constraint surface defined by a projection ray to achieve visual compliant motion. Other research at Yale University [Hager et al 95] also uses visual feedback for uncertainty tolerant precise positioning of a manipulator. The visual servoing methods they have developed can still accurately position the manipulator in the presence of large deviations in camera positioning. Currently, our system uses visual information, but we do not directly close a control loop by using visual feedback. In the future, we may explore the use of visual feedback for more precise positioning tasks.

1.3 Summary

This thesis is organized into three primary sections. The first section provides an introduction, the second section is the main body of the thesis, and the third section presents the results and conclusions.

The first section is comprised of this chapter and Chapter 2. This chapter has presented

the motivation, history, and goals for this research. Chapter 2 provides descriptions of the mechanical and computational hardware which comprise our system.

The second section is comprised of Chapters 3 and 4. Chapter 3 deals with the new component of our system, the Fast Eye Gimbals (FEGs). The dynamics, control, and use of the FEGs are presented. In addition, the coordination between the vision system and the WAM is also discussed. The algorithms presented in Chapter 3 are used subsequently in Chapter 4. Chapter 4 discusses the use of the FEGs and the WAM for the purpose of catching tossed objects. Path prediction methods and path generation methods are presented.

The third section is comprised of Chapters 5 and 6. Chapter 5 present our experimental catching results. And finally, Chapter 6 closes out the thesis, containing recommendations for improvement of the system and directions for future research.

Chapter 2

System Description

This chapter provides a brief introduction to each of the components which comprise our system. Mechanical hardware is discussed first, followed by computational hardware and software, and vision processing hardware. Figure 2-1 shows the system configuration which is used for grasping and catching tasks.

2.1 The Arm, Eyes, and Hand

2.1.1 The Whole Arm Manipulator (WAM)

The Whole Arm Manipulator (WAM) is a four degree of freedom cable driven manipulator arm designed by Bill Townsend, Brian Eberman, and Dr. Kenneth Salisbury of the MIT Artificial Intelligence Lab [Salisbury 87, Salisbury et al 88, Townsend 88]. At full extension, the WAM is approximately 1.1 meters in length, depending upon the length of the end effector. Each of the four motors can output a maximum of 1.5 Nm and position signals are provided through 12 bit resolvers. The cable drive transmissions for each of the joints are two stage and provide velocity reductions between 1:20 and 1:30. The WAM is cable driven, therefore there is no backlash, and as a result, the whole arm may be used for force feedback. The WAM is controlled using an Adaptive Controller implemented by Günter Niemeyer [Niemeyer, Slotine 88, Niemeyer 90] which achieves a tracking performance of less than half a degree within 0.5 - 1.0 seconds of adaptation. Some additional items of note are the use of a differential in the second joint axis and the exceptional dynamic performance

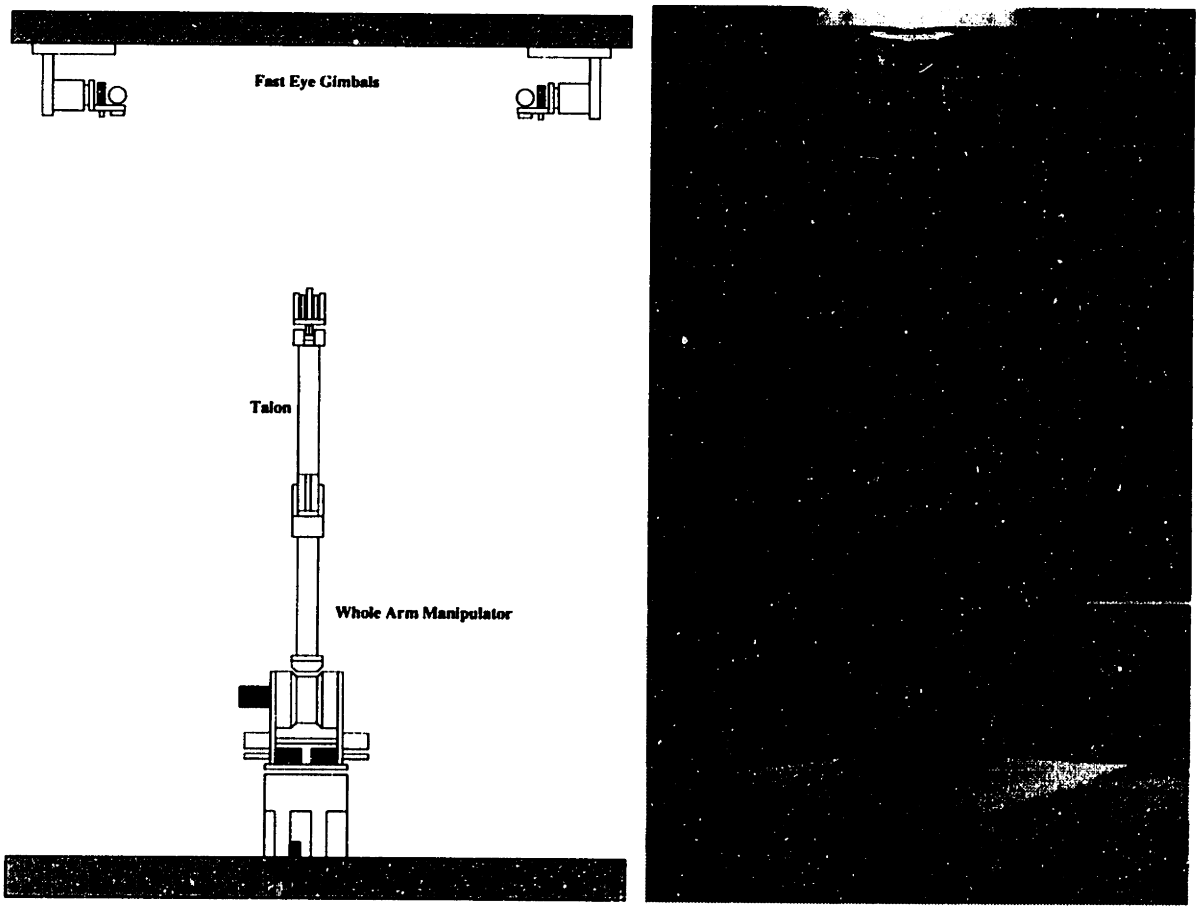


Figure 2-1: The system configuration used for catching has the FEGs mounted to ceiling rafters approximately seven feet from the floor and three feet behind the WAM.

which has allowed the end point of the WAM to achieve accelerations up to 25 times the acceleration of gravity (with no mass at the endpoint).

2.1.2 The Fast Eye Gimbals (FEGs)

Active Vision versus Static Vision

The previous vision system used two stationary cameras with large fields of view. The current system uses two cameras mounted on actuated gimbals. The choice to use an active vision system over a stationary system has associated advantages and disadvantages.

The advantages to the use of an active vision system primarily relate to focus of attention. The active vision system allows for the use of larger focal length lenses which provide much more detail. The pixel resolution for the same size object is much greater

due to the smaller field of view. In addition, by having a smaller field of view, attention may be more focused to a smaller region, neglecting a majority of background noise. With actuated cameras, the object of interest may be centered in the image, reducing the effects of lens distortion. Also, with actuated cameras, depending upon the range of motion of the actuator, in practice, a wider overall range of view is obtained.

The disadvantages to the use of an active vision system primarily relate to accuracy and increased difficulty. There is increased complexity when using an active vision system, such as gear ratios and misalignments and mounting errors, which result in more sources of error. Accurate calibration of an active vision system is therefore more difficult. As a result, often the accuracy is less than a stationary vision system. Finally, there is the difficulty of having to tracking moving objects to maintain them in the field of view and re-acquiring or locating new objects requires a searching routine.

Overall, the advantages of the active vision system out weigh the disadvantages. With better algorithms, the disadvantages of the active vision system can be minimized.

The Fast Eye Gimbals (FEGs)

The Fast Eye Gimbal (FEG) was designed by Nitish Swarup and Akhil Madhani as a two degree of freedom cable driven gimbal for active vision applications [Swarup 93]. The FEG has a range of $\pm$ 90 degrees on the first axis and $\pm$ 45 degrees on the second axis. The mass of the lens limits the maximum velocity and acceleration of the FEG. The current camera and lens weigh approximately 1 pound, with which the FEG has exceeded accelerations of 200 rad/s^2. The cameras are high resolution 768x493 color CCD cameras which output interlaced NTSC signals at 60 Hz. The cameras have an adjustable shutter speed, white balance, auto gain control, and can synchronize with an external trigger. The lens is currently a 50 mm focal length lens allowing for approximately a $\pm$ 5 degree field of view. The small field of view allows for better accuracy in position determination, but introduces difficulties in high performance tracking and searching. The cameras are mounted to the FEGs such that the focal point of the camera closely coincides with the center of rotation[1]. A stationary third camera with a fish-eye lens may be added in the

[1]Inaccuracies in mounting can be approximately compensated for if mounting offset distances are known.

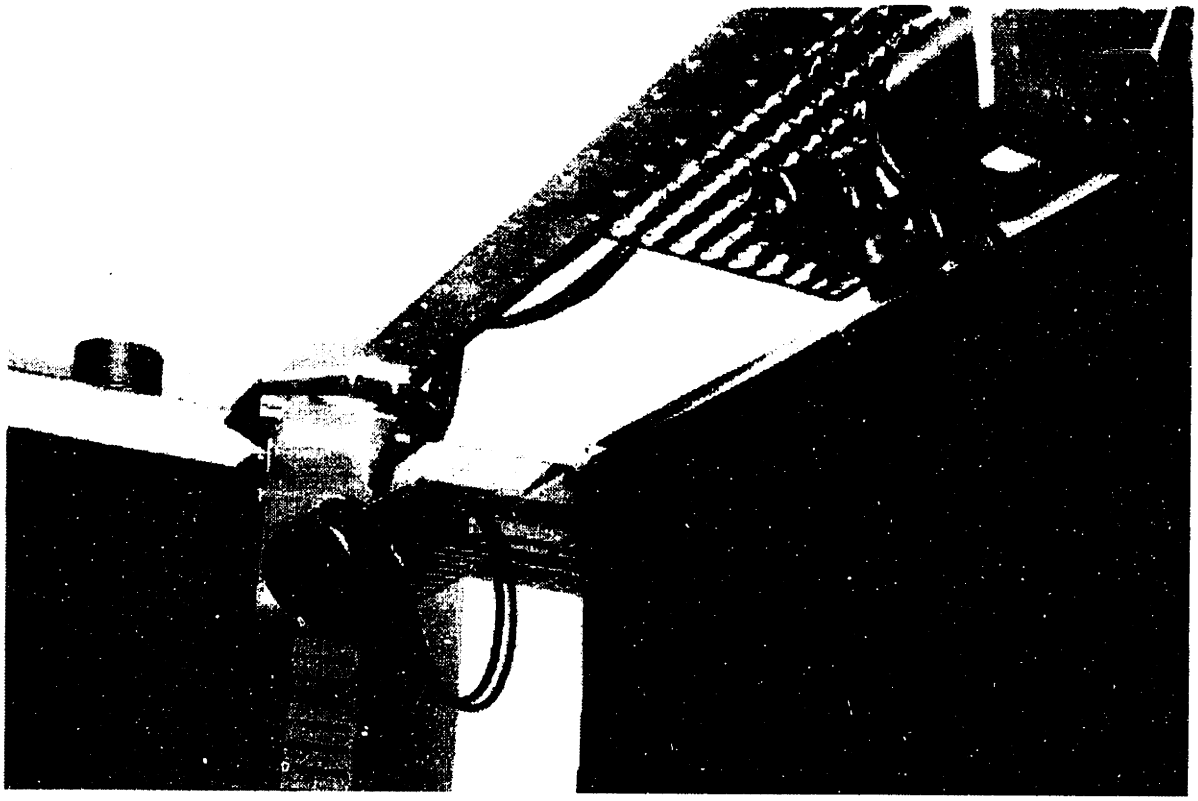


Figure 2-2: A photograph of the FEGs mounted to a ceiling rafter.

future as well as possibly down-sizing the focal length of the lenses.

2.1.3 The Talon

The Talon is a wrist/hand designed by Akhil Madhani as an end effector for the WAM for touch sensing and grasping tasks. The Talon is a cable driven three degree of freedom system. The Talon has a supination/pronation (forearm) joint and two finger joints which each move independently. The degrees of freedom can be thought of as a forearm joint, a "wrist" joint where both fingers move in the same direction, and a "grasping" joint where the fingers move in opposite directions. The forearm joint has a range of ± 90 degrees and can move 180 degrees in less than 0.3 seconds. From the closed position, the fingers can each open approximately 70 degrees, but they each have a full range of approximately 180

(see Appendix A.4).

Figure 2-3: A photograph of the Talon mounted to the end of the WAM.

degrees[2] The Talon requires approximately 0.33 seconds to travel the 70 degrees from the fully open position to the fully closed position. Various fingers have been designed for the Talon, including simple aluminum fingers, spring loaded fingers with hall effect sensors to detect deflection, and highly sensitive fingers which each have four PVDF contact sensors along the outer surfaces and a single strain gauge sensor at the base for measuring net torque. In our catching experiments conducted here, we do not utilize touch sensing or any additional sensors, aside from motor encoders, to detect finger deflection. Instead, we only require durable fingers, so sturdy light aluminum fingers with serrated edges are used for better grasping.

[2]The full 180 degree range of each finger is used to act as a "wrist". With both fingers touching one another, they may both travel 180 degrees, but with a decrease/increase in the amount which they can "open".

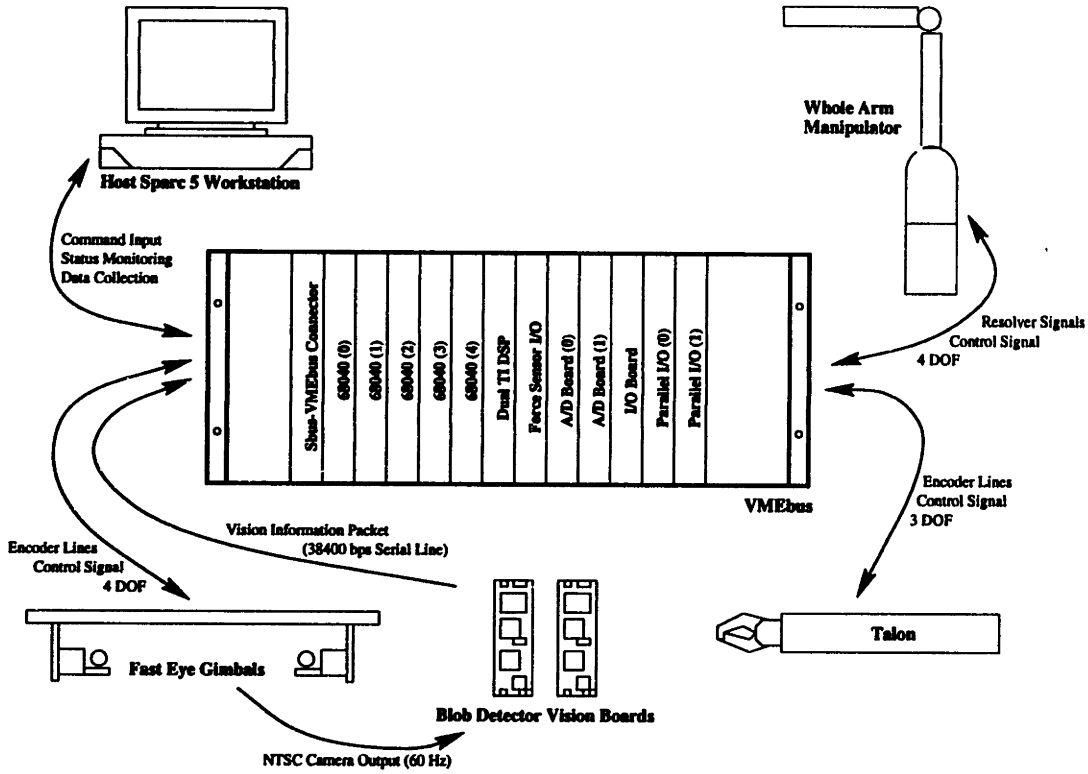


Figure 2-4: Overall system and connections to the VMEbus.

2.2 VMEbus Hardware and Software

The WAM, FEGs, and Talon are controlled by a multiprocessor VMEbus system connected to a host Sun Sparc 5 workstation. The host workstation is used for compiling, downloading, monitoring, and command input. Our VMEbus contains five 68040 processors and a dual DSP board from Ariel and additional I/O boards. The software for this system is written primarily in C. The operating system which is used, called Hummingbird, was developed in-house by Günter Niemeyer specifically for our needs. It is a real-time operating system based on the Condor system developed by Sundar Narasimhan.

The computation and I/O for our system is distributed among the processors in the following manner. The WAM is currently controlled by two 68040 boards, one executing I/O routines at 4000 Hz, and the other executing trajectory planning, adaptation, controller calculations, and kinematics at 200 Hz. For more information about the control of the WAM,

see [Niemeyer, Slotine 88]. The FEGs and Talon are similarly controlled by two 68040s, one executing low level PD control at 2000 Hz, and another executing the FEG adaptive control, trajectory planning for the FEGs and Talon, and any other additional calculations at 400 Hz. The last 68040 is used as an intelligent I/O board, currently handling the serial line transfers from the vision boards.

2.3 Vision Processing Hardware

Previous System versus Current System

A major change in the current system compared to the previous system is the replacement of the stationary vision system with the active vision system and the associated replacement of the previous frame grabber/vision processing hardware with the new blob detector vision boards.

The previous vision system was composed of two black and white cameras which provided information at 20 Hz. The vision processing hardware required a heavily controlled environment. The object of interest was required to be white and the entire background black, which required the surroundings and the arm itself to be covered. The latency in the system from image acquisition until the time when all the vital information is extracted and output was significantly larger than our new system. And lastly, the old vision system provided center of area information only.

In contrast, the current vision processing hardware requires minimal control of the environment. Aside from "marking" objects of interest with the appropriate color, there are no additional requirements on the environment. By choosing an appropriate color for "marking", it is unlikely a regular background will contain distracting items of that color. By using only color keying to detect objects of interest, the speed of the visual processing is greatly increased. This is in accord with the goals for this project, since the objective is not to address the traditional vision problems of edge detection, object recognition, or segmentation which would require more powerful hardware, but rather, to simply have a vision processing system which provides the information required as fast as possible at a low cost.

Blob Detector Vision Boards

The vision boards are based on the 68332 processor and output information via serial line at 38400 bps into one of the 68040 boards in the VMEbus. The vision boards are color keyed and use a histogram lookup table to create binary images from the camera input signal. They were designed and programmed by Anne Wright and Randy Sargent of the MIT Artificial Intelligence Laboratory [Wright 93].

The board receives an NTSC signal from the cameras and outputs a black and white NTSC signal of the processed image to be displayed on a monitor. Since the interlaced signal from the cameras is used, the vision boards have a vertical resolution of half that of the cameras. The horizontal resolution of the vision boards is limited by the speed and memory of the board and is approximately 256. Therefore, excluding the unused border around the outside edge of the screen, the processed image resolution is approximately 230x230. The board communicates via two serial lines operating at 38400 bps. An interface to the board is provided on a Sun Sparc and allows for training, loading new programs, and changing parameters, such as data output format.

The boards operate at frame rate (60 Hz) and require approximately 1/30 th of a second to process each image. Figure 2-5 illustrates a sample binary image from the vision boards and shows the various outputs available. The output format is variable and can be set to include frame count, row number of center, column number of center, number of pixels of the image, major axis angle from horizontal, and the aspect ratio[3] of the image. In the case of multiple non-connected blobs in a single image, the vision boards are capable of outputting full information for up to approximately ten blobs. We currently only compute information for the largest blob. Adding additional blobs increases the required computing, thereby increasing the latency in the system and possibly reducing the rate of information through-put.

The board can be trained to simultaneously detect objects with three distinct color histograms at once. During training, the color of interest is shown to the cameras and a histogram is built, occupying a region of the RGB space. These histograms can be stored

[3]The aspect ratio output by the vision boards are defined here as the square root of the ratio of the second moments of inertia along primary axes.

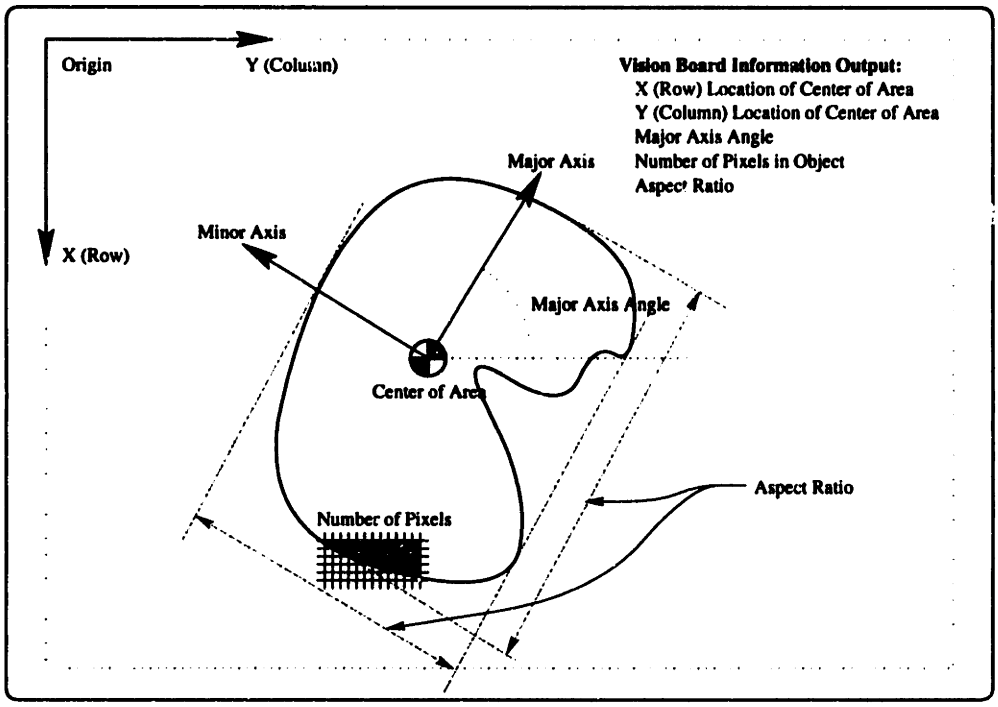


Figure 2-5: Illustration of a sample binary image as produced from the vision boards and the various outputs which are available.

to EEPROM so that the boards do not require re-training on the next boot. Currently we only utilize blob detection with one histogram since we want to limit noise introduced by background and environment and we are only interested in individual objects of one certain color (fluorescent orange). In the future, the WAM itself may be marked in order to study visual servoing (precise positioning using visual feedback).

Chapter 3

Fast Eye Gimbal Control and Use

This Chapter discusses how the FEGs are controlled and various algorithms which are used to apply them to our research goals. Section 3.1 gives the derivation of the dynamic equations of motion for the FEG and discusses various control methods and their performance. Section 3.2 presents triangulation methods for determining the location of objects using stereo vision information. Section 3.3 presents the WAM/FEG calibration methods used to coordinate the vision and the manipulation. Section 3.4 presents various trajectories and behaviors for the FEGs, such as self-calibration, searching, and the application of the WAM/FEG calibration and searching algorithms to the task of locating and grasping objects.

3.1 FEG Dynamics and Control

3.1.1 Dynamics of the FEG

Figure 3-1 shows the FEG with coordinate axes labeled based upon the Denavit-Hartenberg Notation. Using the coordinate frames as defined in the figure, the dynamical equations of the FEG are derived through both the recursive Newton Euler Method and the Lagrange Method (Appendix A.1). They can be expressed as

$$H\ddot{\mathbf{q}} + C\dot{\mathbf{q}} + G = \tau \tag{3.1}$$

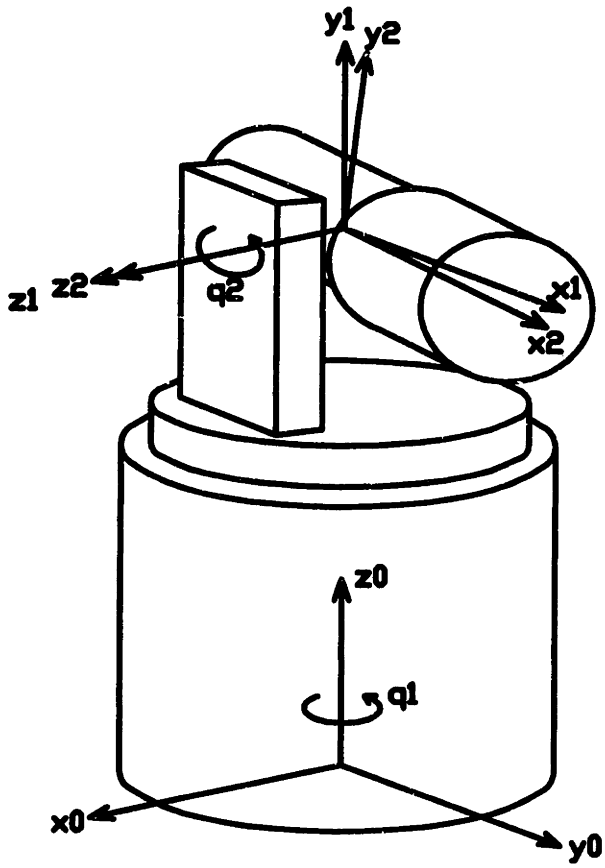


Figure 3-1: FEG with labeled coordinate axes based upon Denavit-Hartenberg Notation.

where H, C, and G are defined as

$$H = \begin{bmatrix} 2J_{12}\cos(q_2)\sin(q_2) + J_{11} + I_{22} + (m_2d^2 + J_{22} - J_{11})\cos^2(q_2) & h_1 \\ h_1 & m_2d^2 + J_{33} \end{bmatrix} \tag{3.2}$$

$$h_1 = J_{13}\sin(q_2) + J_{23}\cos(q_2)$$

$$C = \begin{bmatrix} -c_1\dot{q}_2 & -c_1\dot{q}_1 + (J_{13}\cos(q_2) - J_{23}\sin(q_2))\dot{q}_2 \\ c_1\dot{q}_1 & 0 \end{bmatrix} \tag{3.3}$$

$$c_1 = (1 - 2\cos^2(q_2))J_{12} + (m_2d^2 + J_{22} - J_{11})\cos(q_2)\sin(q_2)$$

$$G = \begin{bmatrix} 0 \\ m_2gd\cos(q_2) \end{bmatrix} \tag{3.4}$$

Here I_{22} is the inertia of the mass 1 (m_1) about the y_1 axis and the J terms are the inertias of the mass 2 (m_2 - including the camera) in the coordinate frame fixed to the camera. Note that since the camera is cylindrical, $J_{22} = J_{33}$. The matrices above are valid for all orientations of the FEGs except for the gravity vector, G. The G vector varies over different orientations, reflecting the change in the direction of gravity and its effects on each of the joint axes. For the configuration used in later sections where the FEGs are mounted horizontally on ceiling rafters, the G vector becomes

$$G = \begin{bmatrix} m_2gd\cos(q_1)\cos(q_2) \\ m_2gd\sin(q_1)\sin(q_2) \end{bmatrix} \tag{3.5}$$

which reflects the change in direction of gravity.

3.1.2 Basic Controllers for the FEGs

A variety of controllers for the FEGs have been implemented (PD, PID, Feed Forward, and Adaptive).

PD The basic PD is implemented with the following torque command on each joint axis.

$$\tau = -K_D\dot{\tilde{\mathbf{q}}} - K_P\tilde{\mathbf{q}} \tag{3.6}$$

where $\dot{\tilde{\mathbf{q}}} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_\mathbf{d}$, $\tilde{\mathbf{q}} = \mathbf{q} - \mathbf{q}_\mathbf{d}$, and $\mathbf{q}$ and $\mathbf{q}_\mathbf{d}$ are the actual and desired joint positions.

PID The PID has an integral term added to the PD which compensates for any steady state error. This was found to be necessary as a result of the video signal cable from the

camera creating large errors. The torque command for this controller on each joint axis is the following.

$$\tau = -K_D\dot{\tilde{\mathbf{q}}} - K_P\tilde{\mathbf{q}} - K_I\int\tilde{\mathbf{q}} \tag{3.7}$$

Feed Forward and Adaptive Control The Feed Forward is an application of the Adaptive Controller with constant parameter estimates. The Adaptive Controller determines a set of parameters which yield the best tracking performance. The Feed Forward controller uses these parameters with no adaptation to attempt to cancel the effects of dynamics of the system. See the next section for more details.

3.1.3 Adaptive Control

Motivation

Adaptive Control for trajectory tracking can achieve close to zero tracking error. PD Control on the other hand does not improve its tracking performance and the tracking error is considerably larger. Adaptive Control can also maintain good tracking error in the presence of changing physical characteristics such as friction and camera cable effects.

There exist other controller options besides PD and Adaptive, but each of these have either prohibitive requirements or undesirable characteristics. For example, a Computed Torque Controller requires knowledge of the physical parameters. But it is difficult to derive a physically accurate model which accounts for all effects, which leads to inaccurate parameter estimates, and therefore decreased performance. Also, a Switching Controller theoretically achieves zero tracking error, but at the cost of chattering and high control activity (which may not be achievable by the actual physical actuators or other hardware).

The Adaptive Controller requires little a priori knowledge of the actual physical parameters of the system and utilizes the same amount of control authority as a simple PD Controller. In addition, the Adaptive Controller has the added benefit that it can in some cases be used for parameter estimation.

Concepts and Controller Design

The Model Reference Adaptive Controller used here is composed of a feed forward dynamical term and a PD term. The controller uses the dynamics of the system to aid in parameter convergence while t PD term adds stability. The basic control law is of the form

$$\tau = Y\hat{\mathbf{a}} - K_D\mathbf{s} \tag{3.8}$$

The variable $\mathbf{s}$ is an intermediate variable which can be used as a measure of the tracking performance and is defined as

$$\mathbf{s} = \dot{\tilde{\mathbf{q}}} + \lambda\tilde{\mathbf{q}} = \dot{\mathbf{q}} - \dot{\mathbf{q}}_\mathbf{r} \tag{3.9}$$

where $\tilde{\mathbf{q}} = \mathbf{q} - \mathbf{q}_\mathbf{d}$, and where λ is the bandwidth of the control system. The reference velocity and acceleration can then be defined as

$$\begin{aligned} \dot{\mathbf{q}}_\mathbf{r} &= \dot{\mathbf{q}}_\mathbf{d} - \lambda\tilde{\mathbf{q}} \\ \ddot{\mathbf{q}}_\mathbf{r} &= \ddot{\mathbf{q}}_\mathbf{d} - \lambda\dot{\tilde{\mathbf{q}}} \end{aligned} \tag{3.10}$$

Using these reference values, the following relation defines the Y matrix and the $\mathbf{a}$ vector.

$$H\ddot{\mathbf{q}}_\mathbf{r} + C\dot{\mathbf{q}}_\mathbf{r} + G = Y\mathbf{a} \tag{3.11}$$

The Y matrix is composed of known terms, and the $\mathbf{a}$ vector of unknown parameters.

The last part of the Adaptive Controller is the parameter estimation update law, which updates the parameters based upon the measure of tracking error, $\mathbf{s}$. The adaptation law is selected so as to formally guarantee global boundedness and tracking convergence [Slotine, Li 91].

$$\dot{\hat{\mathbf{a}}} = -\Gamma Y\mathbf{s} \tag{3.12}$$

Using these equations, it can be shown using Lyapunov-like methods, that the system is stable and that the tracking error should, in ideal cases, converge to zero (see Appendix A.2.1 and [Slotine, Li 91]). And for sufficiently "exciting" trajectories, the parameters should

converge to their actual values (with an accurate model).

Additional terms can also be added to model coulomb and viscous friction. The equation defining Y and $\mathbf{a}$ then becomes

$$H\ddot{\mathbf{q}}_\mathbf{r} + C\dot{\mathbf{q}}_\mathbf{r} + G + D_s sgn(\dot{\mathbf{q}}_\mathbf{r}) + D_v\dot{\mathbf{q}}_\mathbf{r} = Y\mathbf{a} \tag{3.13}$$

where D_s and D_v are diagonal positive definite matrices.

With both of these effects, there are a total of 11 parameters for the system, 7 from the dynamics and 4 from friction. The Y matrix is then a 2x11 matrix and $\mathbf{a}$ is a 11x1 column vector.

Given H, C, and G from equations 3.2, 3.3, and 3.4, equation 3.13 is used to find the Y matrix and the $\mathbf{a}$ vector. For the table mounted FEG (standing vertically), the Y matrix is

$$Y = \begin{bmatrix} Y_{11} & \ddot{q}_{r1} & Y_{13} & Y_{14} & Y_{15} & 0 & 0 & sgn(\dot{q}_{r1}) & 0 & \dot{q}_{r1} & 0 \\ Y_{21} & 0 & Y_{23} & Y_{24} & Y_{25} & \ddot{q}_{r2} & \cos(q_2) & 0 & sgn(\dot{q}_{r2}) & 0 & \dot{q}_{r2} \end{bmatrix} \tag{3.14}$$

where each of the Y_{xy} terms are defined as follows

$$\begin{aligned} Y_{11} &= 2\cos(q_2)\sin(q_2)\ddot{q}_{r1} - (1 - 2\cos(q_2)^2)(\dot{q}_2\dot{q}_{r1} + \dot{q}_1\dot{q}_{r2}) \\ Y_{13} &= \cos(q_2)^2\ddot{q}_{r1} - \cos(q_2)\sin(q_2)(\dot{q}_2\dot{q}_{r1} + \dot{q}_1\dot{q}_{r2}) \\ Y_{14} &= \sin(q_2)\ddot{q}_{r2} + \cos(q_2)\dot{q}_2\dot{q}_{r2} \\ Y_{15} &= \cos(q_2)\ddot{q}_{r2} - \sin(q_2)\dot{q}_2\dot{q}_{r2} \end{aligned}$$

$$\begin{aligned} Y_{21} &= (1 - 2\cos^2(q_2))\dot{q}_1\dot{q}_{r1} \\ Y_{23} &= \cos(q_2)\sin(q_2)(\dot{q}_1\dot{q}_{r1} + \ddot{q}_{r2}) \\ Y_{24} &= \sin(q_2)\ddot{q}_{r1} \\ Y_{25} &= \cos(q_2)\ddot{q}_{r1} \end{aligned}$$

and the $\mathbf{a}$ vector is

$$\mathbf{a} = \begin{bmatrix} J_{12} \\ J_{11} + I_{22} \\ m_2 d^2 + J_{22} - J_{11} \\ J_{13} \\ J_{23} \\ m_2 d^2 + J_{33} \\ m_2 g d \\ D_{s1} \\ D_{s2} \\ D_{v1} \\ D_{v2} \end{bmatrix} \tag{3.15}$$

Conceptually, the Y matrix and the $\mathbf{a}$ vector represent a linear parameterization of the dynamics of the system.

The control law which is implemented slightly differs from that shown in equation 3.8. The K_P gain may be tuned higher than $K_D\lambda$ to achieve better performance in the presence of high frequency unmodelled dynamics. This is done by separating the PD portion of the control law which results in the following expression

$$\tau = Y\hat{\mathbf{a}} - K_D\dot{\tilde{\mathbf{q}}} - K_P\tilde{\mathbf{q}} \tag{3.16}$$

The stability and the convergence of this control law is again shown with Lyapunov methods (see Appendix A.2.1 and [Slotine, Li 91]).

3.1.4 Performance of PD and Adaptive Controllers

Trajectories for Tracking

Both PD Control and the Adaptive Control have been implemented for the FEG and applied to tracking two trajectories of the form

$$q_{d1} = 0.5\cos(\omega t) \qquad q_{d2} = 0.5\cos(1.7\omega t) \tag{3.17}$$

$$q_{d1} = 0.7\sin(\sin(\cos(\omega t))) \qquad q_{d2} = 0.7\sin(\sin(\cos(1.7\omega t))) \tag{3.18}$$

The first trajectory is non-repetitive over the time of our testing due to the 1.7 term in q_{d2}. The second trajectory is an att pt to increase the "excitation" to encourage parameter convergence. These trajectories were used with ω values of 5, 7, and 10 rad/s.

Implementation

The PD components were tuned as high as possible while minimizing oscillations in trajectory tracking. The figures in this section for PD Control performance were generated with these values.

For the Adaptive Controller, the PD components remain the same, but the bandwidth (λ) and adaptation gains (Γ) need to be specified. Appendix A.2.2 gives some guidelines for selection of these values. The bandwidth is chosen to eliminate undesired higher frequency components by adjusting it's value and examining the performance (keeping in mind the guidelines from equation A.20). The resulting controller bandwidth was chosen to be $\lambda = 50$. Next, the adaptation gains were selected based upon equation A.21 with some additional tuning to improve performance. The final value for the adaptation gains were

$$\Gamma = diag([0.29 \;\; 0.72 \;\; 1.73 \;\; 0.04 \;\; 0.02 \;\; 0.10 \;\; 112.28 \;\; 32.12 \;\; 10.88 \;\; 10.26 \;\; 1.82]) \tag{3.19}$$

Performance Results

Figures 3-2, 3-3, 3-4, and 3-5 show the performance results for both the PD and the Adaptive Control applied to the two trajectories listed earlier in this section.

Figure 3-2 and Figure 3-4 show that the Adaptive Control, with no initial knowledge of the physical parameters, achieves tracking to $\frac{1}{2}^\circ$ after initial transients. This is approximately 4 times better than PD alone. Even during the initial transients of less than 1 second, the tracking error remains 2 times better than with PD.

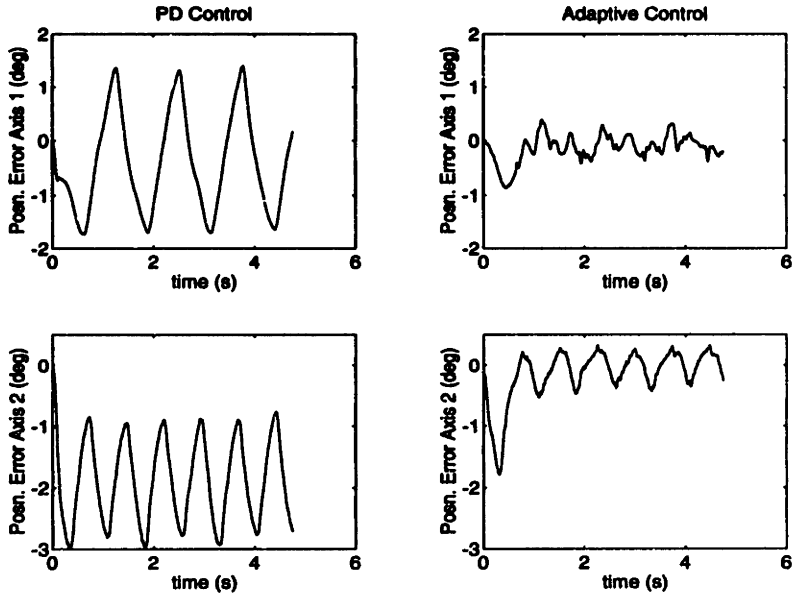


Figure 3-2: Tracking performance for the PD and Adaptive Controllers for Trajectory 1 with $\omega = 5$ rad/s.

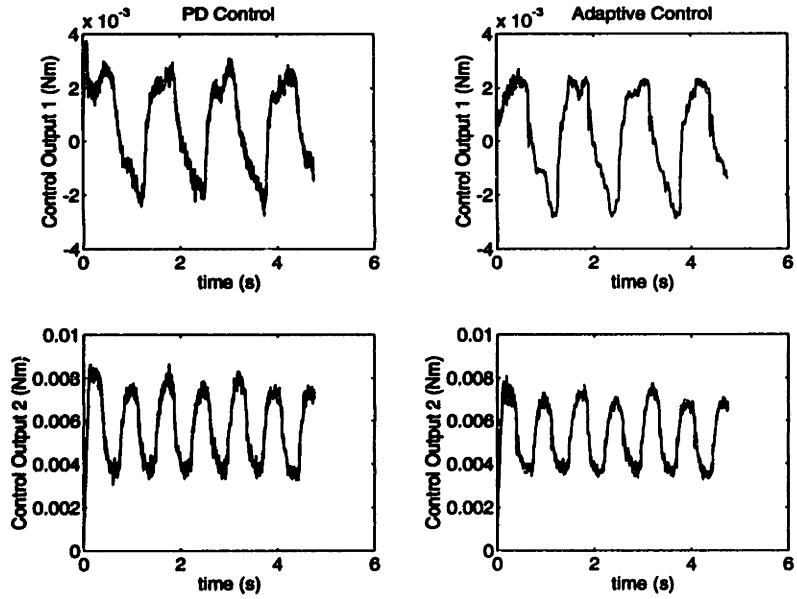


Figure 3-3: Control output for the PD and Adaptive Controllers for Trajectory 1 with $\omega = 5$ rad/s.

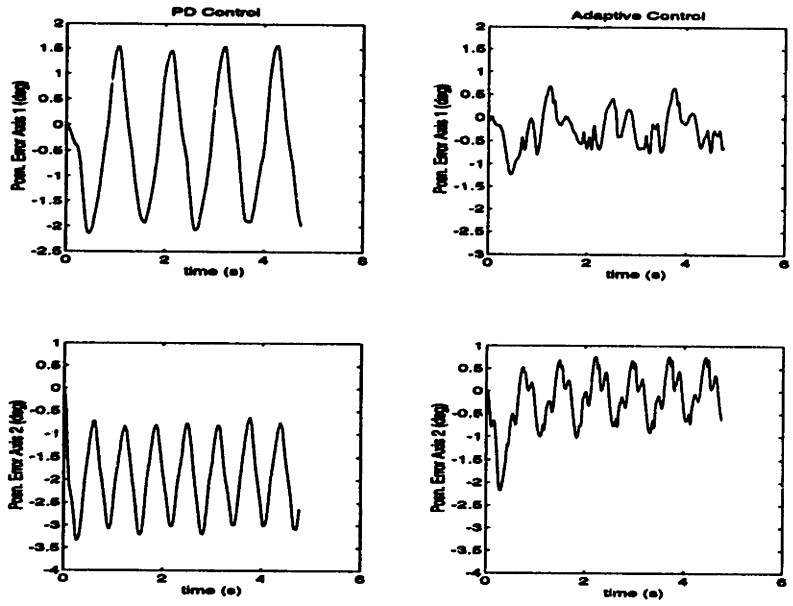


Figure 3-4: Tracking performance for the PD and Adaptive Controllers for Trajectory 2 with $\omega = 5$ rad/s.

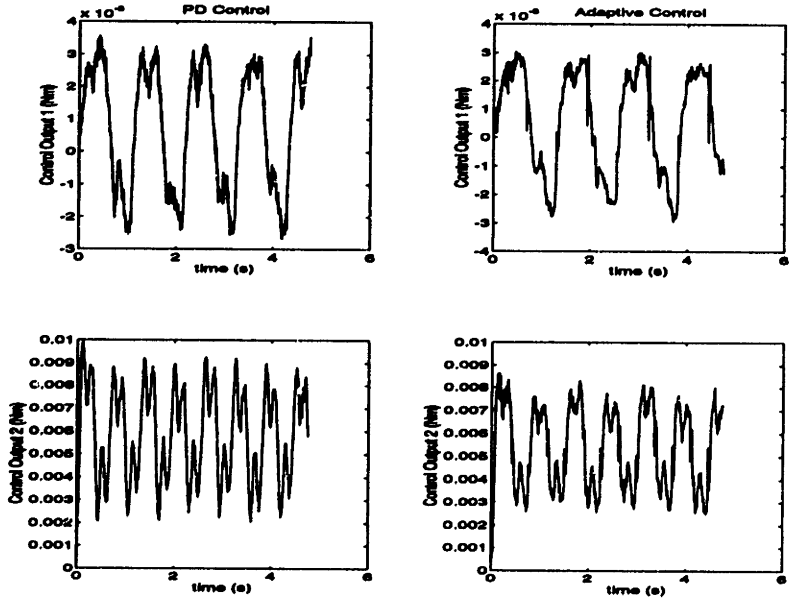


Figure 3-5: Control output for the PD and Adaptive Controllers for Trajectory 2 with $\omega = 5$ rad/s.

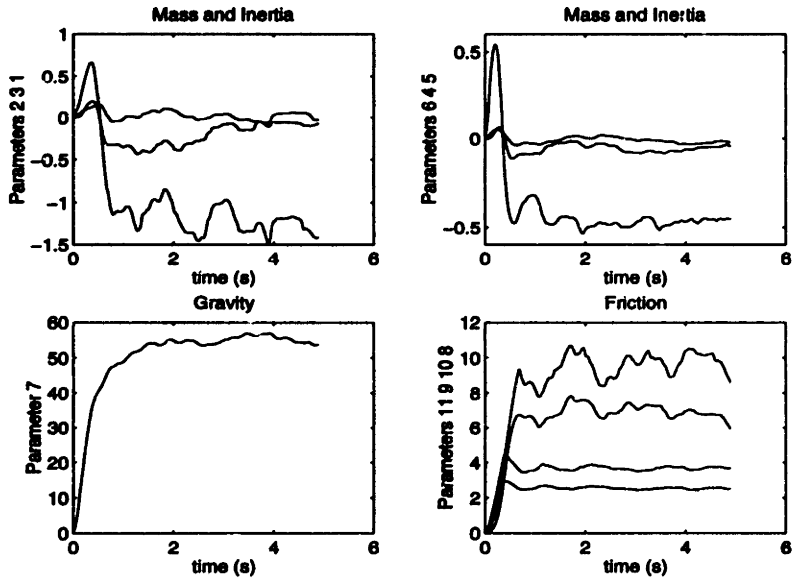


Figure 3-6: FEG parameter estimates from the Adaptive Controller on Trajectory 1 with $\omega = 5$. Labels on the y axes are in order of increasing final parameter value.

Note that the position error on axis 2 for both trajectories under PD Control does not center about zero due to the gravity torque resulting from the mass of the lens. Addition of a compensation terms lead to a shift of the tracking error to center about zero. None the less, the Adaptive Controller still outperforms the PD.

Figure 3-3 and Figure 3-5 show the control torques for each controller applied to the first and second trajectories respectively. Note that the approximate magnitudes are comparable for both Adaptive and PD Control. So thus Adaptive Control does not require more control authority, but rather uses it more effectively.

Additional tests were run with values for $\omega = 7, 10$ rad/s with similar results. The tracking error increases as a result of the faster trajectories, but the relative performance between Adaptive and PD remains the same.

In ideal cases, if our trajectories were "exciting" and our dynamics model accurate, the parameters would converge to their actual values. Figure 3-6 shows the time history of all 11 parameters, most of which converge to a value within 1 second. Note, however, that two of the parameters are are decidedly negative. These two parameters are the primary inertia

terms for each of the axis, and they are multiplied by the $\ddot{q}_r$ terms in the Y matrix. These negative parameters most likely result from the unmodelled dynamics from the video signal cable and the cable transmission. The convergence of these two parameters to negative values adversely effect the other parameters, so the values to which all parameters converge to cannot be used for parameter estimation. In our case, the primary concern is minimizing tracking error rather than parameter estimation. If parameter estimation is necessary, an accurate model for the cable needs to be included, or the effects of the cable minimized.

3.2 Locating Objects in Space

3.2.1 Self Calibration/Homing

Before the task of locating objects in space can be accomplished, the Eyes must be calibrated to some reference frame. As described in Section 3.1.1, the Adaptive Controller for the Eyes need to know the direction of gravity. Therefore, it is useful to calibrate the Eyes with the z axis in the same direction as gravity.

This calibration method defines an independent coordinate system for the FEGs which is defined from a horizontal home position. The encoders on the FEG motors do not output absolute position, therefore the zero position must be initialized at start-up. The FEGs find their home position by moving to each joint limit for each of their axes. Knowing the locations of these joint limits, the FEGs can find their default home position. This method is repeatable to within 0.003 radians. When done calibrating, each FEG pan axis is oriented parallel to one another and perpendicular to the baseline connecting the two cameras and each FEG tilt axis is oriented horizontally with respect to the world. This places each camera horizontal to the floor and looking directly forward. In this orientation, the cameras are able to see specially marked (fluorescent orange) "calibration squares" which are strategically placed in the room to fine tune the calibration of the home position. Using this additional aid, a repeatability of approximately 0.001 radians is achieved.

3.2.2 Locating Spheres

After initial calibration of the FEG coordinate system, the x, y, z coordinates of objects may be determined by solving the triangulation problem shown in Figure 3-7. The cameras are located at the upper corners of the tilted triangle, separated by a baseline distance which we will call $2D$ (equal to 0.8255 meters in our case). The origin of the camera coordinate frame is located at the midpoint of the baseline with axes defined as shown in the inset. The four joint angles, two corresponding to each gimbal, are shown with the sign convention defined in the other inset. The cameras are mounted at the same height and parallel to each other so θ_1 is equal to θ_3. A solution to the triangulation problem is

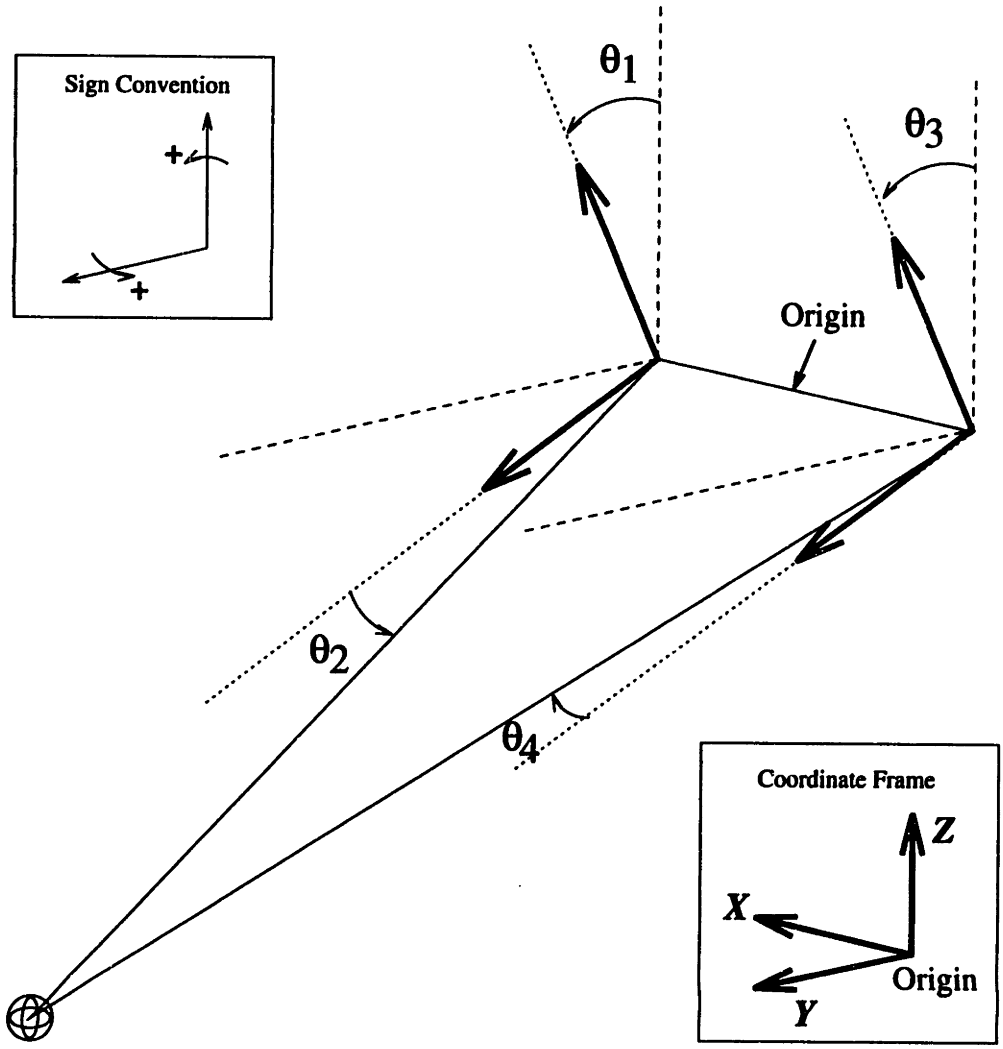


Figure 3-7: Two cameras at the corners of the tilted triangle focus on a ball in space, with joint angles defined as shown. Triangulation is used to locate the coordinates of the ball.

$$x = \frac{D(\tan\theta_2 + \tan\theta_4)}{\tan\theta_4 - \tan\theta_2} \tag{3.20}$$

$$y = \frac{2D\cos\theta_2\cos\theta_4}{\sin(\theta_2 - \theta_4)}\cos\theta_1 \tag{3.21}$$

$$z = \frac{2D\cos\theta_2\cos\theta_4}{\sin(\theta_2 - \theta_4)}\sin\theta_1 \tag{3.22}$$

Experiments in tracking moving objects also require velocity transforms as well. These

can be derived by differentiating the above equations with respect to the θ variables. The derivation is straight forward, with the results shown in Appendix A.3.

The inverse transforms can also be found based upon the relations

$$\tan\theta_1 = \tan\theta_3 = \frac{z}{y} \tag{3.23}$$

$$\tan\theta_2 = -\frac{x-D}{\sqrt{y^2+z^2}} \tag{3.24}$$

$$\tan\theta_4 = -\frac{x+D}{\sqrt{y^2+z^2}} \tag{3.25}$$

The inverse velocity and acceleration transforms can also be found by differentiating these equations, with the results shown in Appendix A.3.

3.2.3 Locating Cylinders

The blob detector vision system outputs the direction of the major axis of the image. The code used by the boards was written and optimized by Anne Wright and Randy Sargent [Wright 93]. The method described here is simply a standard method and not necessarily the one used in their software. A standard method for obtaining the major and minor axes of a two dimensional images consists of minimizing the second moment of inertia. By obtaining I_x, I_y, and I_{xy} by simple summation, you can find the primary directions by using the equation

$$\tan 2\theta = -\frac{2I_{xy}}{I_x - I_y} \tag{3.26}$$

This yields two directions which are 90 degrees apart, indicating the major and minor axes. You can recalculate the new values for the second moments of inertia in the primary directions using the equations

$$I'_x = \frac{1}{2}(I_x + I_y) + \frac{1}{2}(I_x - I_y)\cos(2\theta) - I_{xy}\sin(2\theta) \tag{3.27}$$

$$I'_y = \frac{1}{2}(I_x + I_y) - \frac{1}{2}(I_x - I_y)\cos(2\theta) + I_{xy}\sin(2\theta) \tag{3.28}$$

For an initial approach to the problem of determining the spatial orientation of an

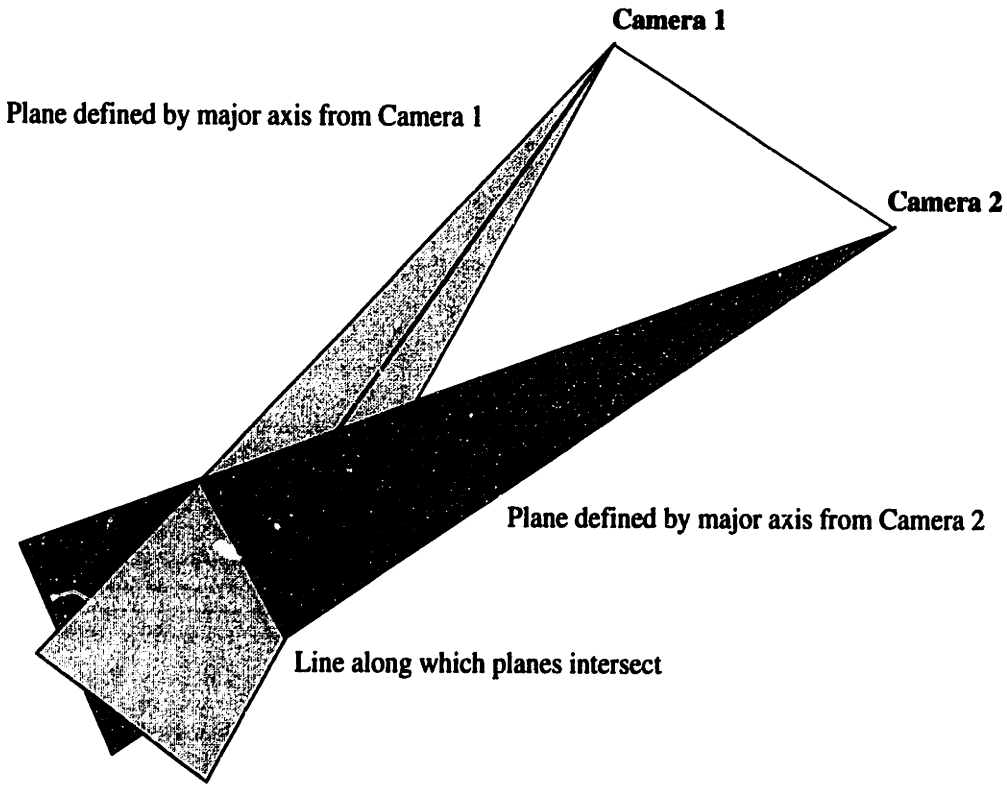


Figure 3-8: Using two cameras and major axes measurements from each, we can examine the intersection of the two planes to determine the orientation of the object.

object, we might use the individual measurements of major axis from each camera to define a plane in which the major axis of the object must lie. Then finding the intersections of the planes defined from each camera, we should be able determine the orientation of the object in space. Figure 3-8 is shown as an illustration of this problem. Note there are also special cases here where the planes might be parallel, or the image might be "deceiving" where incorrect major axis angles are obtained due to our perspective which would lead to incorrect assumptions on orientation. These topics are currently being examined in more depth.

3.2.4 Accuracy of Resulting Coordinates and Sources of Error

The accuracy of this method for determining object location is extremely sensitive to the calibration discussed in Section 3.2. As a test of the vision system, the floor about the

WAM has been mapped using a lattice of points in one foot squares. Variations in x, y, and z of less than 0.5 inches were found across a six foot square region.

Errors of this magnitude equate to approximately 0.003 radians position error in the joints. This error can be attributed to a combination of effects from small errors in gear ratio and small variations in calibration of the zero positions for the joint axes. In addition, the change in viewing perspectives for the same object under varying lighting conditions across the room result in some pixels being detected while others are not. Small errors in the mounting of the cameras to the FEGs cause the focal point to not exactly coincide with the center of rotation. The incorrect mounting can be partially compensated for by using the exact location of the focal points (Appendix A.4) at the cost of increased computation.

Note that the blob detectors only locate the center of the object, so this is ideal for spherical objects and flat objects, because the two vectors would intersect at approximately the center of the object. For other objects of different shape, some error would be introduced because each camera has a different viewpoint and observes different faces of the object, yet considers it flat. A more advanced vision processing system is required to solve problems where this error becomes significant.

3.3 FEG/WAM Calibration

3.3.1 Simplification of Problem

In order to coordinate the FEGs and the WAM, a transformation between the FEGs and the WAM must be determined. Various methods could be used to determine the transformation from FEG joint space to WAM joint or cartesian space. Most of these methods would involve a system of nonlinear equations. In addition, different representations could be used, such as quaternions, euler angles, or rotation matrices. Additional methods and representations are discussed briefly in the Conclusion.

Ideally, we would like to have a simple calibration method which has a straight forward solution and is easy to implement. The method which has been selected places constraints on the physical location and orientation of the FEGs. Methods were examined which attempt to determine a calibration for arbitrary locations and orientations for each FEG. These proved to be difficult and were not guaranteed to converge but are still being explored.

The method which was settled upon greatly simplifies the problem by making assumptions about the mounting and orientation of the FEGs which leads to a one degree of freedom least squares problem. The FEGs must be mounted at the same height from the floor and they must be placed parallel to one another. With this mounting configuration, an independent coordinate frame for the FEGs can be defined relative to the world frame as previously discussed (Section 3.2). Similarly, the WAM has its own coordinate frame relative to the world. Due to the mounting configuration of the FEGs, the z axes for both the WAM and the FEG coordinate frames are parallel. This leaves one rotational degree of freedom and a translation which must be solved to determine the transformation between coordinate frames.

3.3.2 One DOF Least Squares

To determine the transformation from the FEG coordinate frame to the WAM coordinate frame, multiple data points are collected of a ball in the end of the WAM as seen by the cameras and as measured by the WAM. Storing data from 13 points around the workspace of the WAM, and assuming that the horizontal x-y planes are parallel for the FEGs and the

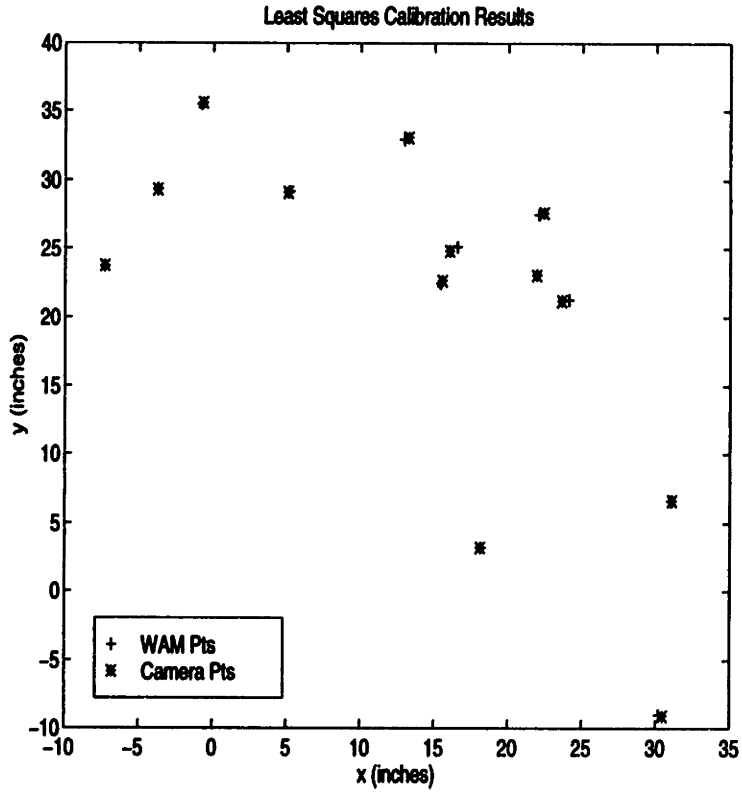


Figure 3-9: Results from Least Squares calibration for transformation from FEG to WAM space.

WAM (as a result of our earlier calibration of the FEG joint axes), the remaining problem involves a one dimensional rotation.

We wish to obtain a rotation matrix, **R**, and a translation vector, **t** which will overlap the two sets of data. First, by using the centroid of each set of data points as the origin, **t** is found by finding the translation that will overlap the centroids. In order to find the rotation matrix, a least squares approach is used [Horn 86]. By referencing each data point relative to their centroids, the problem becomes the minimization of

$$\sum_{i=1}^{n}(\delta x_i^2 + \delta y_i^2) \tag{3.29}$$

where

$$\begin{bmatrix} \delta x_i \\ \delta y_i \end{bmatrix} = \begin{bmatrix} x_i' \\ y_i' \end{bmatrix} - \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3.30}$$

The angle of rotation, θ, which defines $\mathbf{R}$ can then be found by the equation

$$\tan\theta = \frac{\sum \mathbf{r}_\mathbf{i}' \times \mathbf{r}_\mathbf{i} \cdot \hat{\mathbf{z}}}{\sum \mathbf{r}_\mathbf{i}' \cdot \mathbf{r}_\mathbf{i}} \tag{3.31}$$

where $\mathbf{r}_\mathbf{i} = (x_i, y_i, 0)^T$ corresponds to data points from the camera and $\mathbf{r}_\mathbf{i}' = (x_i', y_i', 0)^T$ corresponds to data points from the WAM.

The results from the least squares calibration is shown in Figure 3-9. The sum of the square error was 1.0153 inches^2 over 13 points, with an average error of 0.2795 inches.

It is essential throughout this process that the coordinates from the FEGs and the WAM are consistent. Initially, there were large errors in this calibration, indicating inconsistent coordinate systems, but after extensive calibration of link lengths and gear ratios for both the FEGs and the WAM, the data consistency was much improved.

3.4 FEG Trajectories and Behaviors

3.4.1 Searching/Scanning

The cameras mounted to the FEGs have a small field of view. In order to locate objects, the FEGs must perform a search. The searching/scanning trajectory was designed to find and track objects which are of the correct color.

There are two searching modes. The first is a specific search routine designed to locate objects located on the foam floor within the workspace of the WAM. This search mode incorporates z height consistency checks and handles cases where the view from one camera is possibly obstructed. The second is similar to the first, except the consistency checks and obstructed cases are not included. This more general search mode locates objects anywhere in space and hence, we cannot apply any consistency constraints (besides divergence of the cameras).

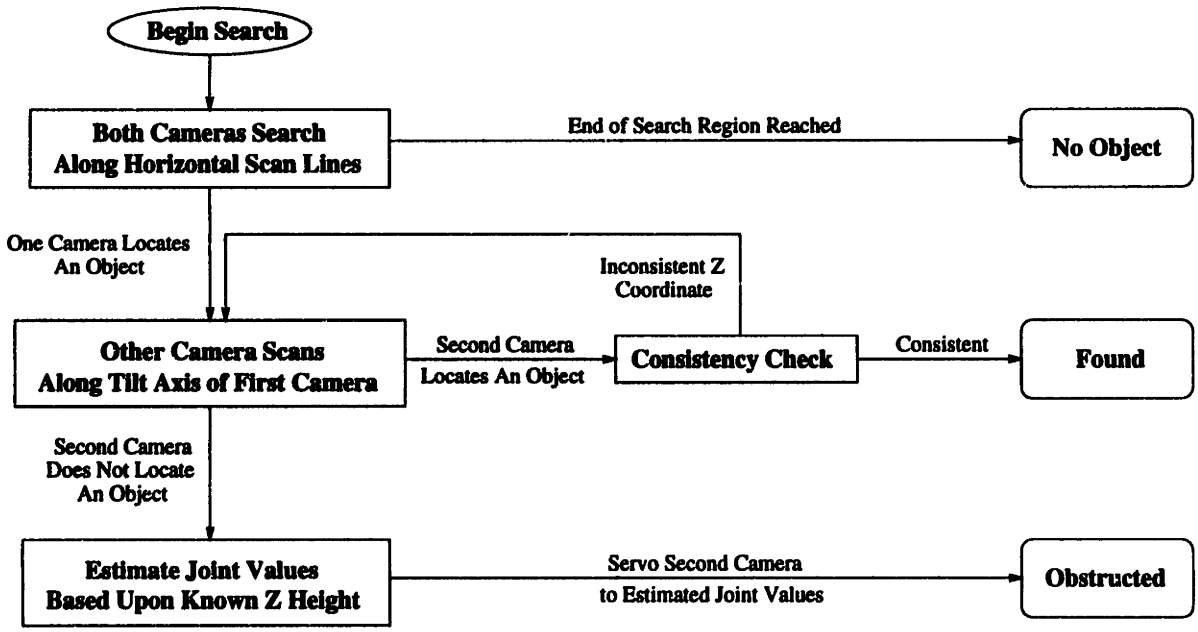


Figure 3-10: Flowchart of SEARCH procedure which scans with the FEGs to locate objects of interest within a user defined region.

Floor Searching Figure 3-10 shows a flowchart of the first searching mode. First, both cameras simultaneously scan over a user defined square region in joint space. The cameras scan in horizontal scan lines by moving their pan axes while simultaneously slowly adjusting their tilt axes. If the search region is completed without either camera locating an object, then the search routine ends, returning "No Object". If one camera locates an object, the second camera moves to the same tilt angle, and then scans across with its pan axis until an object is found. If an object is found by the second, then using the triangulation as discussed previously, the coordinates of the object are obtained and the consistency of the z height with the known height of the foam floor is checked. If the z height is consistent, then the search routine ends, returning "Found". If the cameras have locked on to different objects, then the z height would be inconsistent. In this case, the second camera ignores that object and continues its horizontal motion with the pan axis until another object is encountered. If no consistent object is found by the second camera after the first back and forth horizontal scan motion, then the tilt axis is perturbed by a small amount and the horizontal scanning is repeated. After three perturbations, it is declared that the view from the second camera must be obstructed. But, since the z height of the floor is known,the pan joint angle which corresponds to an object located on the floor at that particular tilt angle can be calculated. The second camera then servos to these calculated joint coordinates and then the search routine ends, returning "Obstructed".

General Searching The second searching mode is exactly the same as the first mode except that it may only return "Found" or "No Object". In addition, if an object is found by the one camera, but not the second, then the horizontal scanning and perturbation will execute indefinitely. No consistency constraints may be applied in this case to stop the search due to obstruction. It is hoped that the object would be moved away from obstruction. During the motion, the first camera will still track the object and the second camera will continue scanning. Once the view from the second camera becomes unobstructed, the search will end with "Found".

3.4.2 Servoing on Stationary Objects

Once an object enters the field of view of the cameras, the Eyes should center the object in the screen. The cameras can visually servo to the color keyed object by using the center of area information provided by the vision boards boards. Visually servoing on static objects or slowly moving objects can simply be accomplished by setting the desired velocity and acceleration to zero and setting the desired position for the Eyes using the following equation

$$\begin{bmatrix} \theta_{pan} \\ \theta_{tilt} \end{bmatrix}_d = \begin{bmatrix} \theta_{pan} \\ \theta_{tilt} \end{bmatrix} + \begin{bmatrix} pixel\_arc_x\,(x - x_{center}) \\ pixel\_arc_y\,(y - y_{center}) \end{bmatrix} \tag{3.32}$$

where x and y are the coordinates of the center of the blob in pixels and x_{center} and y_{center} are the predetermined coordinates of the center of the screen in pixels and finally $pixel\_arc_x$ and $pixel\_arc_y$ represent the angle subtended by one pixel in x and y. The camera lenses have a field of view of ± 5 degrees which allows us to use the approximation $pixel\_arc = \tan^{-1}(1/f) \approx 1/f$ where f is the focal length.

By this method, for stationary or slowing moving objects, the cameras may servo to maintain the object in the center of the screen. Timing delay compensations do not need to be made in this case since the object is stationary.

3.4.3 Generalized Tracking of Moving Objects

Introduction

We would like to have a general purpose tracker which makes as few assumptions on the path of the object as possible. One assumption which is made is that the path can be approximated by a time varying second order polynomial. The acceleration estimate is held constant between vision samples, but the estimate is allowed to vary at each new vision sample.

Difficulties

The primary difficulty in robust tracking of fast moving objects is the compensation for the processing delay in the vision system. Information arriving from the vision boards is

delayed by the amount of time for CCD charging, vision board processing, and transfer over serial lines. Even with accurate estimates of this delay time, estimates of object velocity and acceleration must be used to integrate over the time delay, which amplify any estimation errors. But currently the delay time is not constant, it varies depending upon the content of the image being processed. Therefore the delay time itself is not known accurately.

An average measure of the processing delay time may be obtained. The delay time can be estimated using a sinusoidal trajectory for the cameras while observing the center of area output from the vision boards. By calculating the phase shift in the two sinusoidal signals, the average time delay in the system is determined. For the current configuration, the phase shift is found to be approximately 0.035 seconds. Methods to reduce the delay time and make it invariant to image content are discussed in the Conclusion.

An additional difficulty results from noise in the vision system due to lighting effects. As an object moves about the room, due to non-uniform lighting, the perceived color of portions of the object will change. The change will result in the addition and subtraction of "white" pixels from the processed image which shift the center of area. This effect will introduce oscillations in the system which require greater filtering.

Overview

Vision information from the vision boards arrive at 60 Hz. The new vision information is from a time, $t - t_d$, in the past. Using this vision information, estimates for the position, velocity, and acceleration of the object at the previous time, $t - t_d$, are made. These estimates are in FEG joint space. The primary purpose for conducting tracking in joint space is to allow each camera be able to track independent of the other.

The estimates of the object's state are then used to integrate over the time delay. The integration is done assuming constant acceleration. The result is an estimate of the *current* state of the object. This estimate is then fed to the FEG controller.

Between vision samples, no additional information is available, but a smooth trajectory should be provided to the FEG controller. Therefore, the trajectory generator integrates the position and velocity, assuming constant acceleration during the intervening time.

Implementation

Figure 3-11 shows a block diagram of our observer. The diagram uses x, v, and a to represent position, velocity and acceleration for the joint which is currently being estimated. Each joint for each FEG is computed individually using the exact same method.

The input to the tracker is the measurement for the location of the object. The new information which the tracker receives from the vision boards is actually information from an image which was taken at a previous time. This delay for processing was previously estimated to be $t_d \approx 0.035$ seconds. Past position information for the FEGs is stored continuously so that the position of the FEGs at the time that the image was taken may be known. Using the position of the FEGs t_d seconds in the past and summing this with the offset of the image in the screen, the measurement of the location of the object is obtained.

$$z_k = p(t - t_d) + offset \tag{3.33}$$

From the last vision sample, old estimates for the position, velocity and acceleration exist. Using these position and velocity estimates and the time between vision samples, $t_v = t_k - t_{k-1} \approx 0.0167$ seconds, a prediction of the expected location of the object may be made.

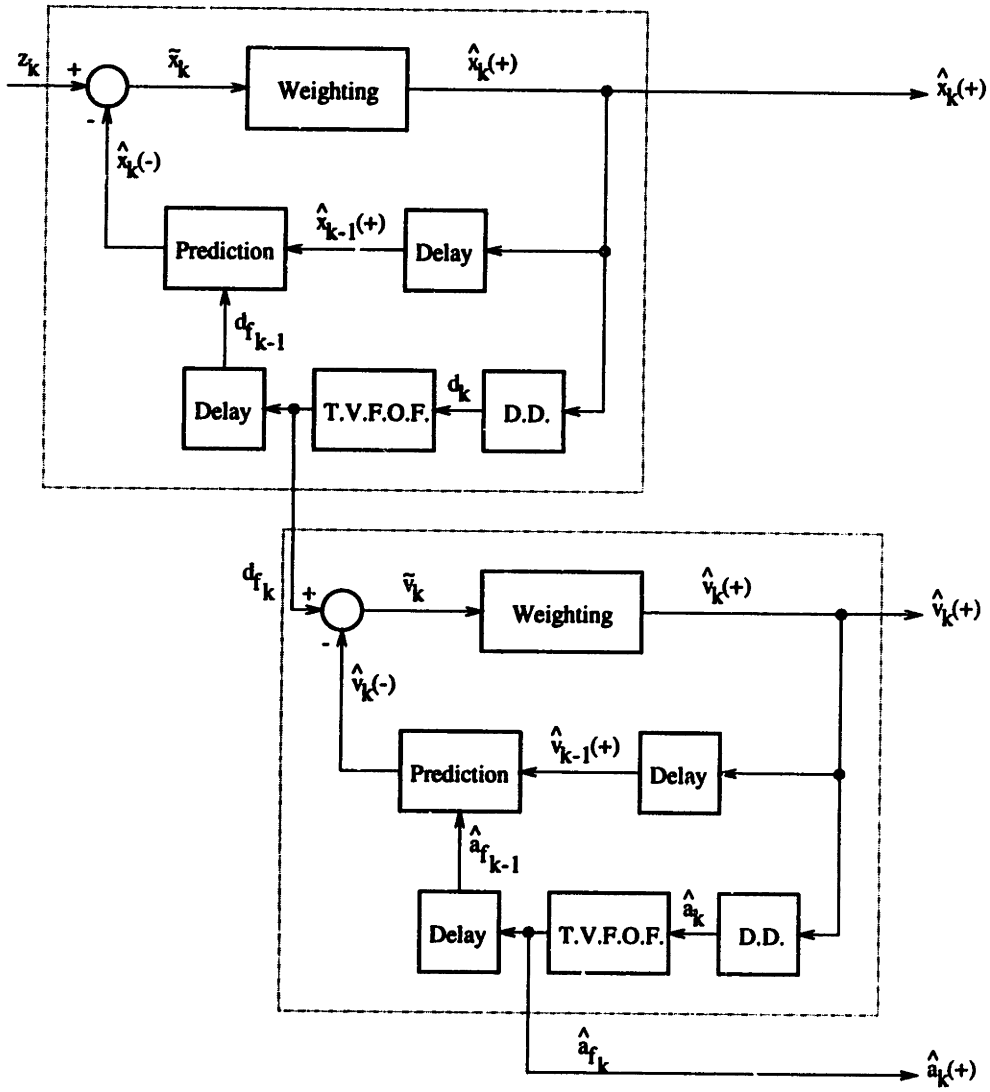
$$\hat{x}_k(-) = \hat{x}_{k-1}(+) + d_{f_{k-1}} t_v \tag{3.34}$$

The $(-)$ and the $(+)$ represent the estimates right before and right after the incorporation of the new measurement and $d_{f_{k-1}}$ is the filtered raw velocity measurement from the previous vision sample, as defined by Equations 3.38 and 3.39.

The error between the measurement and the prediction is then given by

$$\tilde{x}_k = z_k - \hat{x}_k(-) \tag{3.35}$$

This prediction error is used to implement a "Kalman-like" filter which uses a weighted sum. The weighted sum is computed by



Abbreviations: D.D. = Discrete Differentiation, T.V.F.O.F. = Time-Varying First Order Filter

Figure 3-11: Block diagram of our generalized tracker which estimates the current position, velocity, and arceleration of the object from delayed information.

$$\hat{x}_k(+) = w_x \hat{x}_k(-) + (1 - w_x) z_k \tag{3.36}$$

where the weighting, w_x, is determined by the magnitude of the prediction error, $\tilde{x}_k$. Currently the weighting term is given by

$$w_x = \begin{cases} 0 & for \quad \tilde{x}_k = 0, \\ (\tilde{x}_k - E_x)/E_x & for \quad 0 < \tilde{x}_k < E_x, \\ 1 & for \quad \tilde{x}_k > E_x \end{cases} \tag{3.37}$$

The maximum value of the prediction error for which the measurement is still used is E_x. This value is set to reflect the noise content of the system. The weighting function was chosen so that it would remove small magnitude oscillations, yet track large deviations. This may appear backwards. But, in general, the measurements from the vision system do not contain noise in the form of large "spikes" (large magnitude deviations from previous measurements) but rather contain small oscillations due to the changing location of the center of area. The center of area information will often change by small amounts as the object is seen from different perspectives or as the object moves about the room under non-uniform lighting. These small shifts do not reflect the true motion of the object. The contribution of small magnitude noise may be minor for fast moving objects, but this method helps to minimize oscillations when servoing on stationary objects.

The output of the weighted sum is our new position estimate for the object. This new position estimate is next used to obtain estimates for the velocity and acceleration of the object. The position estimate is passed through a discrete differentiator.

$$d_k = \frac{\hat{x}_k(+) - \hat{x}_k(-)}{t_v} \tag{3.38}$$

The output from the differentiator is then passed through a time-varying first order filter to remove noise.

$$d_{f_k} = \lambda d_{f_{k-1}} + (1 - \lambda) d_k \tag{3.39}$$

The bandwidth of the filter is varied depending upon the integrated offset of the object

	Position	
Measurement	:	$z_k = p(t - t_d) + offset$
Prediction	:	$\hat{x}_k(-) = \hat{x}_{k-1}(+) + \hat{d}_{f_{k-1}} t_v$
Prediction Error	:	$\tilde{x}_k = z_k - \hat{x}_k(-)$
New Estimate	:	$\hat{x}_k(+) = w_x \hat{x}_k(-) + (1 - w_x) z_k$
	Velocity	
Differentiation	:	$d_k = \frac{\hat{x}_k(+) - \hat{x}_k(-)}{t_v}$
Filtering	:	$d_{f_k} = \lambda d_{f_{k-1}} + (1 - \lambda) d_k$
Prediction	:	$\hat{v}_k(-) = \hat{v}_{k-1}(+) + \hat{a}_{f_{k-1}} t_v$
Prediction Error	:	$\tilde{v}_k = d_{f_k} - \hat{v}_k(-)$
New Estimate	:	$\hat{v}_k(+) = w_v \hat{v}_k(-) + (1 - w_v) d_{f_k}$
	Acceleration	
Differentiation	:	$\hat{a}_k = \frac{\hat{v}_k(+) - \hat{v}_k(-)}{t_v}$
Filtering	:	$\hat{a}_{f_k} = \lambda \hat{a}_{f_{k-1}} + (1 - \lambda) \hat{a}_k$
New Estimate	:	$\hat{a}_k(+) = \hat{a}_{f_k}$

Table 3.1: A summary of the equations for the tracking algorithm.

from the center of the screen. If the object is constantly offset from the center of the screen, often this indicates that the tracking is lagging behind the object. The integrated offset could be thought of as a measure of the signal content of the motion. If the integrated offset is large, λ is increased. If the integrated offset is small, λ is decreased. Using this method, λ is allowed to vary between a lower and upper bound during tracking. We cannot simply use the upper bound because of the increased oscillations for slower moving objects.

The method for determining the estimated velocity of the object is of exactly the same form as the method outlined above for position. The dotted squares in the block diagram indicate the two regions which are identical in structure. The differences are the input and outputs and the weighting term of the "Kalman-like" filter. The input to the second block, d_{f_k}, can be considered as a "measurement" of the object velocity. The outputs of the second block are the new estimates for velocity and acceleration. The weighting term is of the same form, except a different maximum value is used, E_v.

Table 3.1 presents a summary in equation form, of the complete tracking algorithm. The resulting new estimates of the state of the object are for time $t - t_d$ in the *past*. Therefore, in order to use these values for tracking, they must be integrated over the processing delay. It is assumed that the path of the object may be approximated by second order polynomials.

Thus, the integration over the processing delay is done assuming constant acceleration.

$$x_p = \hat{x}_k(+) + \hat{v}_k(+)t_d + \frac{1}{2}\hat{a}_k(+)t_d^2 \quad (3.40)$$

$$v_p = \hat{v}_k(+) + \hat{a}_k(+)t_d \quad (3.41)$$

$$a_p = \hat{a}_k(+) \quad (3.42)$$

The resulting values are the prediction of where the object *currently* is located. These are fed to the FEG controller for tracking.

Tracking Performance and Difficulties

This tracker performs moderately well but experiences problems related to very fast motions, lighting conditions, and large or odd shaped objects. The filter constants are tuned such that the FEGs do not become unstable, but at the same time, this limits the speed of response of the tracking.

The tracking algorithm designed here is capable of tracking objects which are moving at up to 4 m/s at a distance of 1 m (8 m/s at 2 m, etc.). With very fast moving objects ($> 4 - 5$ m/s towards the FEGs at a distance of 1 m), the FEGs fail to track the object. While the first order filters are tuned for stability and attempt to remove high frequency noise, they also have an associated delay which is sufficient to cause loss of tracking with faster moving objects. The absolute limitation due to hardware (focal length of the lenses, 60 Hz frame rate) is approximately 7 m/s at a distance of 1 meter. This value is for the case where there are no image processing and information transfer delays and if perfect estimation and tracking were possible.

The non-uniform lighting conditions adversely affect the accuracy of the vision hardware. The light reflecting off the ball, or the lack of lighting, changes the color which is seen by the cameras in that region, therefore the center of area changes as the ball moves towards, under, and past light fixtures. These effects limit the maximum bandwidth which may be used while still avoiding unstable oscillations. The weighting function attempts to remove noise resulting from these effects.

Objects which extend past the field of view of the camera will have a randomly moving

center as the camera could possibly servo along the object. Along the same lines, objects whose center of area changes depending upon viewing perspective would introduce noise into our tracking system if the object rotates. In both cases, this tracker would experience problems similar to those for non-uniform lighting.

Problems are also encountered for fast moving objects entering the field of view due to required initialization of the trajectory generator. Initialization of the velocity and acceleration estimates must be done as an object just enters the screen. The tracker currently assumes that the velocity is zero for the first data point and the acceleration zero for the first two data points. This limits the speed with which the tracker can respond to objects entering the screen.

Better methods for tracking are constantly being explored. In the catching algorithms of the next chapter, we use additional information to aid in tracking. During catching, we predict the future path of the ball in cartesian space to determine catch points. We also use this information to aid in tracking by computing weighted sums of parabolic predicted paths and the results from our observer. This increases the performance of the system, but still is not sufficient to track faster moving objects. A new structure for our observer must be designed in order to track these objects. We discuss possible directions of improvement in the Conclusion.

3.5 Grasping Experiments

A grasping sequence has been implemented which uses the methods described in this Chapter and methods developed by a number of persons associated with this system to demonstrate hand/eye coordination. The FEGs are used to locate an object, the WAM and the Talon then use the information from the FEGs to move into position above the object, orient, and grasp the object. Figure 3-12 provides a flowchart for the sequence of events.

The grasp sequence begins with the FEGs searching for objects located on the foam within the workspace of the WAM. When an object is found, the WAM moves above the object and the Talon orients the fingers perpendicular to the major axis of the object. The WAM then moves down to the object and attempts a grasp. The WAM attempts to grasp

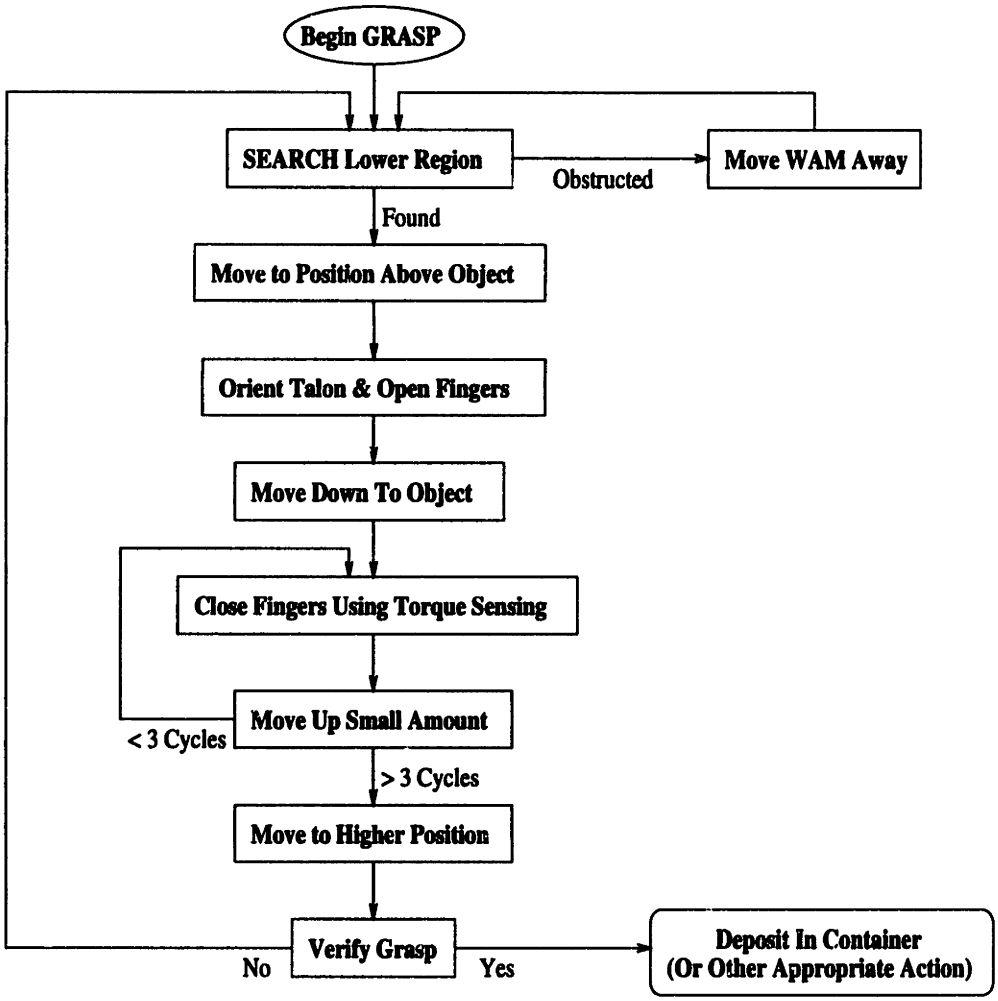


Figure 3-12: Flowchart of GRASP sequence which coordinates the vision and the manipulation in order to locate and grasp objects.

underneath the object by first moving completely down on the object. At this time the fingers close using torque sensing and often touch the foam floor. As the WAM moves up while the Talon closes, the fingers move along the surface of the foam to grasp underneath the object. The repeated upward motion combined with closure ensures a more reliable grasp. After grasping, the WAM moves two feet above the foam and verifies whether it has successfully grasped the object. The motion moves the WAM away from the floor so that if our grasp was unsuccessful, the FEGs have a clearer view of the object. In the case of unsuccessful grasps, the FEGs re-search and the process is repeated.

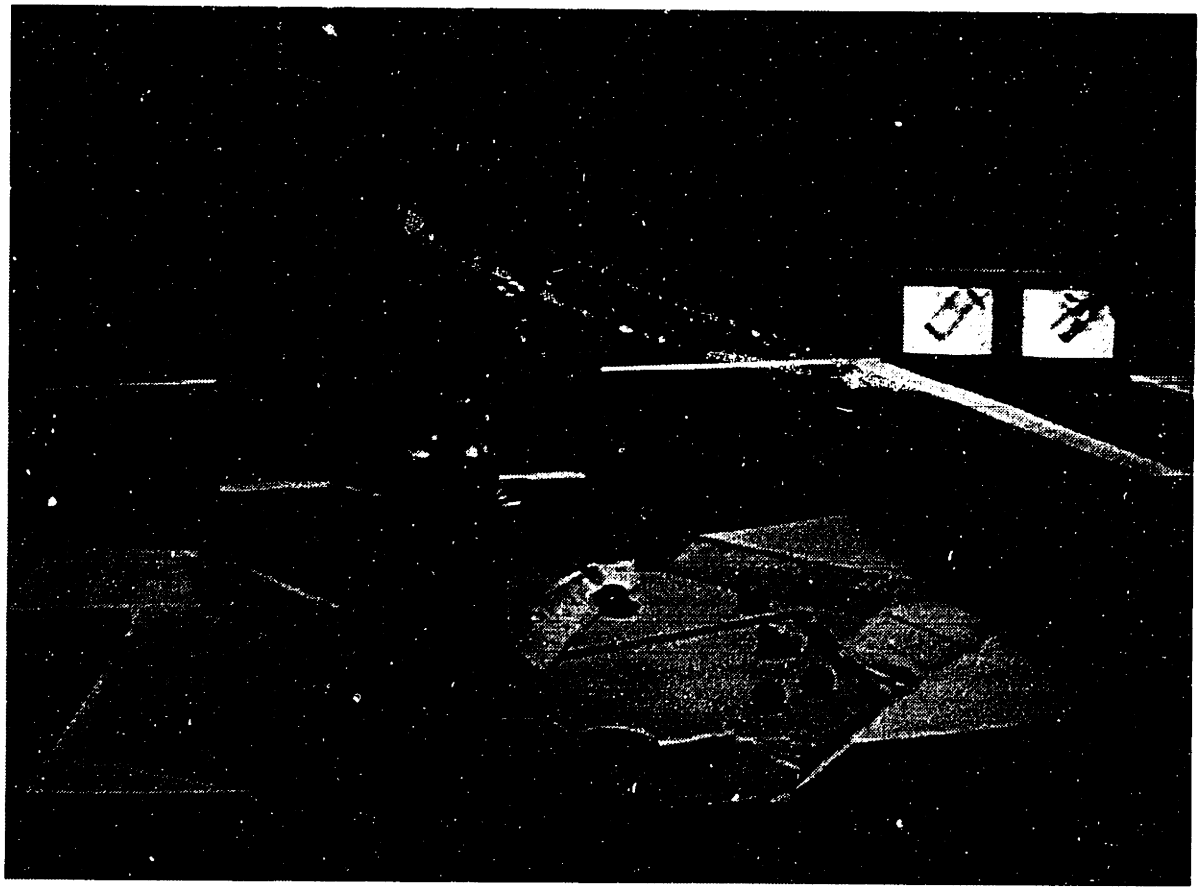


Figure 3-13: A photograph of the WAM in the process of picking and sorting a number of objects placed about its workspace.

Using this procedure, the WAM has successfully picked up to fifteen objects semi-randomly[1] placed about the full workspace of the WAM and deposited them in separate containers. The major axis information from the cameras is used to correctly orient the Talon so that it can grasp long cylinders. The aspect ratio is also used to determine whether objects are spherical or non-spherical. And finally, a dynamic weighing procedure is implemented, which uses the WAM to obtain relative measures of the weight of the objects. Using all of these methods, the WAM has picked up and sorted objects into four containers by aspect and weight. Figure 3-13 shows the WAM in the process of picking and sorting a number of objects.

[1]The objects are semi-randomly placed in the sense that they are not allowed to overlap one another.

The grasping succeeds on the first attempt in most cases, with the exceptions being objects which are very heavy and large, small and thin, or objects which are partially obstructed from view by the WAM itself, yielding incorrect position information.

Chapter 4

Catching Algorithms

4.1 Introduction

This chapter discusses the application of our system to the task of catching free flying tossed objects. The chapter begins by discussing two different approaches to catching. Then Figure 4-1 shows a flowchart of the sequence of events which occur during catching. Each item in the flowchart is discussed in detail in the later sections.

4.1.1 Catching Versus Snatching

There are two fundamental approaches to catching. One approach is to simply calculate an intercept point, move to it before the object arrives, wait, and close at the appropriate time. Humans catch most light objects in this manner, using our visual sense and our sense of touch to determine when to close our hand. Another approach is to match the trajectory of the object in order to grasp the object with less impact and to allow for more time for grasping. Humans do not catch purely in this manner very often. For heavier or faster moving objects, humans often catch using a combination of the two approaches. Humans move towards the path of the object but do not fully match the velocity of the object. The first method will be called "snatching" and the second method where the velocity is matched will be called "catching".

Over the course of our experimentation, it has been found that the "snatching" approach produced poor results for a number of reasons. One of the foremost reasons is the extreme

sensitivity to the timing of closure. Without matching paths, the amount of tolerance for timing errors is minimal. An offset as small as 0.005 seconds produced very few successful "snatches". The presence of a palm could improve the situation by stopping the object temporarily. One more degree of freedom would be required in the Talon in order to orient the fingers in such as manner as to present a "palm" for the ball to impact. One set of fingers cannot be used as a palm because the angle of approach of the object would intersect the other set of fingers as they are in the process of closing (Section 4.6). Therefore snatching must be attempted by orienting the direction of finger closure perpendicular to the objects path and closing based purely on timing predictions from the cameras.

The remaining portion of this chapter deals primarily with "catching". The algorithms are the same for "catching" and "snatching" except for the path generation for the WAM. In the "snatching" approach, a second order filter is used without the velocity matching term so that the arm arrives at the catch point before the object and then times the closure of the fingers appropriately. Some results for the "snatching" approach will be presented in the Experimental Results Chapter.

4.1.2 Sequence of Events During Catching

The flowchart in Figure 4-1 illustrates the different stages of catching. The catching attempt begins with a toss trigger (Section 4.2.1) which starts the initial WAM motion (Section 4.3) and the recursive least square fitting (Section 4.2.2). Within the servo loop, using the parabolic fit information, a satisfactory catch time/point is determined (Section 4.4) and desired path for the WAM is generated (Section 4.5) to intercept the object. At the appropriate time the arm attempts to grasp the object by closing the Talon fingers (Section 4.6). The arm then decelerates (Section 4.7) and returns to the "home" position.

Additional events, not shown in the flowchart, that occur during catching are the activation of WAM and FEG adaptation and data storage. The adaptive controllers for the WAM and the FEG are both activated with large adaptation gains for the short duration of the catching attempt in order to improve dynamic tracking performance by adapting to mass, friction, and inertial parameters.

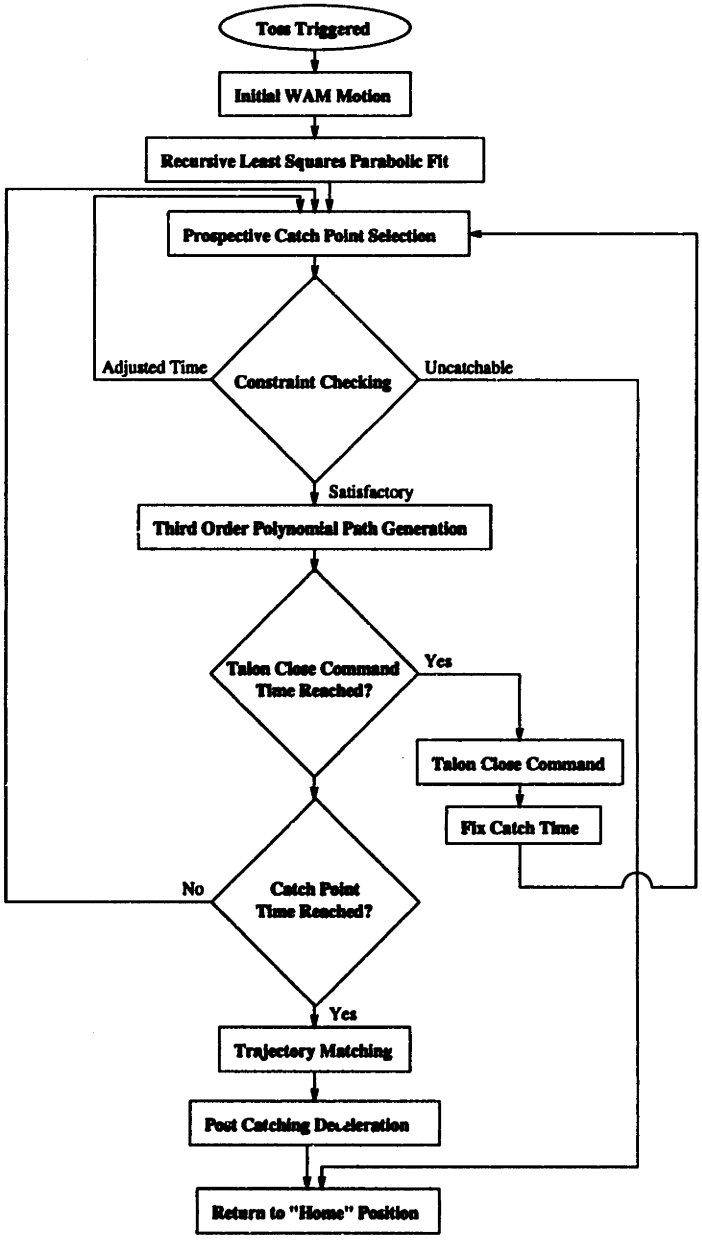


Figure 4-1: Flowchart of events during the catching trajectory beginning with the toss being triggered.

4.2 Toss Triggering and Prediction

In order to be able to catch a free flying ball, the future path must be predicted. In this case, a recursive least squares method is used to fit the ball data to first order polynomials in x and y and a second order polynomial in z. A forgetting term or a windowed fitting approach is not used, therefore, a method for detecting when the toss has begun (i.e. when the ball has left the thrower's hand) must be included. If data from the time when the ball is still in the thrower's hand is incorporated into the fit, then the estimates will be slightly incorrect due to the bad initial data. If the trigger is done late, then some valuable initial information is lost. This section discusses the triggering method which has been implemented and the recursive least squares fitting method.

4.2.1 Triggering

Detecting when a ball has left a thrower's hand and begins free flight is a difficult task. When throwing, different people have different back-swings and follow-through motions. A method is required which throws away minimal amounts of data but which accurately triggers after the ball has left the hand of the thrower.

Two measures are used to detect whether or not a toss has been triggered. One measure is the z height and the second is the x-y radial distance from the base of the WAM. Initially these trigger values are set using offsets from the current location of the ball. The trigger values are updated in cases where the ball moves lower or farther away from the WAM. Thus, the trigger values are ultimately set to be offsets from the lowest and farthest point the ball has reached prior to triggering. Both measures must be satisfied in order for the toss to trigger the parabolic fitting and catching procedures. Figure 4-2 illustrates the trigger offsets.

Relating our triggering methods to how people toss, the low point of the back swing is used as a reference point. It is assumed that the lowest point that the ball reaches is the low point of the back swing. Thus the trigger is set relative to this point in the belief that people will release an object at a consistent distance ahead of this low point when tossing an object.

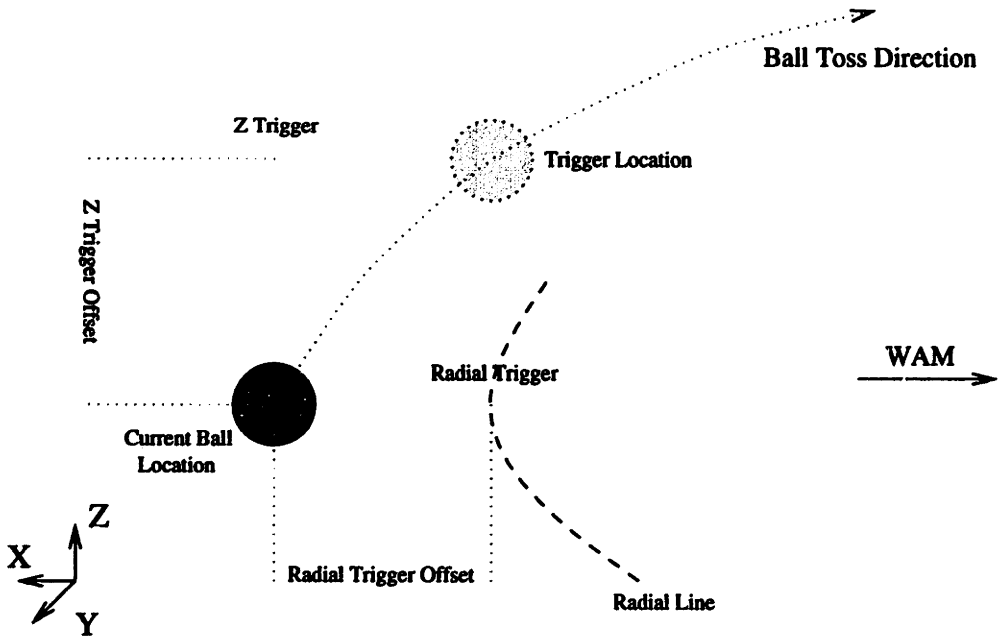


Figure 4-2: Illustration of the trigger distance offsets from the ball location to determine where the toss will be triggered.

There are other possible methods for triggering which have been examined. One possible method is to trigger once the ball has past a preset radial distance from the base of the WAM. This method throws out a large amount of usable data or incorporates bad data from the back-swing or follow-through depending on where the thrower is located. Another possible method is to continuously do a windowed parabolic fit in the z direction until the acceleration is found to be close to -9.81 m/s^2. The problem with this method is that the data from the parabolic fit is quite noisy for approximately the first 10 data points. Therefore a window of at least that size is required. Thus, up to a sixth of a second is lost in which the WAM may have started the catching motion.

One of the primary reasons for choosing the current method of triggering over these other options is the simplicity and the reliability. By setting the offset values appropriately, we can avoid throwing away useful data and also avoid incorporating bad data.

4.2.2 Recursive Least Squares Fitting

Once the toss has been triggered, data storage, fitting, and prediction begins. A recursive least squares parabolic fitting method is utilized for object path prediction [Kimura et al 92, Kimura 92]. A summary of the method is provided in Appendix B.1.

Because the computation is recursive, running summations of intermediate variables are maintained. The variables are used to compute the least squares fit at each time step. The computation required each new data point is independent of how many data points have already been collected. Each data point is weighted equally, the last having as much effect on the predicted parabolic constants.

Using the previous work of [Kimura 92] as a starting point, two modifications are made which improve the results. First, the initial estimates for the z acceleration are constrained because they vary quite wildly for the first several data points due to noise content. If left unconstrained, the z acceleration constants do not converge to correct values until after approximately 10-15 data points. Therefore, for the first 10 data points the predicted Z acceleration is bounded. Initially, the acceleration is set to -9.81 m/s^2. The upper and lower bounds are then set to -8.31 m/s^2 and -11.31 m/s^2 respectively for the first 10 data points. After the first 10 data points, the Z acceleration has usually stabilized and converged to close to -9.81 m/s^2 and the constraints become unnecessary. Secondly, calculations for the x and y acceleration values are not done. Instead, it is assumed that they are zero in these cases. Fitting a polynomial of higher order to a system of lower order decreases the accuracy of resulting estimated constants. Therefore since it is known that the paths of the objects for these experiments are linear in x and y, the acceleration values for these axes are set to zero so that information is not wasted.

The constants which are determined using the least squares fit are

$$\mathbf{C}_{prb} = \begin{bmatrix} 0 & v_x & p_x \\ 0 & v_y & p_y \\ a_z & v_z & p_z \end{bmatrix} \tag{4.1}$$

which can be multiplied by $(\frac{1}{2}t^2\ t\ 1)^T$ to find the state of the object at time t.

The estimates of the parabolic constants are updated every time both cameras receive

new information. Although the cameras are synchronized in hardware, the processing time on each of the vision boards could be different and occasionally the vision boards overrun a frame during computation, so the output drops to 30 Hz. In order to get accurate cartesian position information from the cameras, joint information from the same moment in time is required. Therefore, time stamping must be done for each of the vision packets received from each camera. Using these time stamps and velocity and acceleration estimates from the tracker, the position estimate of one of the two cameras is integrated over the time difference between the two camera time stamps so that joint information from the same moment in time is used.

Once a set of estimated parabolic constants are obtained, the future path of the object may be predicted for any given time. Later sections discuss how a catch point is selected by moving along the predicted path of the object. The later sections require full knowledge of the path of the object so that different points along the path may be examined.

One additional use for the parabolic fit is to aid in FEG tracking performance. In order for the fitting to continue, the FEGs must remain tracking the object being tossed. Since the path of the object is known as a result of the fitting, this information can be used to aid in the FEG tracking of the object. The parabolic fit prediction is done in WAM cartesian space, so it must first be transformed to FEG joint space. The position transformations were presented in Section 3.2.2 and the velocity and acceleration transformations are found in Appendix A.3. After transformation, the fit data is incorporated into the FEG tracker by using a weighted sum of the tracker prediction and the parabolic fit prediction, resulting in better overall tracking performance.

4.3 Initial WAM Catching Motion

Drawing from the previous work on this system [Hove, Slotine 91, Hove 91], the WAM initially moves away from the ball to avoid "spatial backtracking". "Spatial backtracking" is the phenomena where the arm would initially try to move towards the ball and then as the ball progresses along its path past the arm, the arm must move back in the direction it came from in order to intercept the ball. The first several data points are not sufficient

to compute a fit, but it is necessary to start the WAM accelerating so that sufficient time is given to attempt a catch. In these initial stages, gravitational acceleration is assumed in z. The WAM then moves in the direction of the peak of the toss so that the WAM will be moving towards a position a good distance ahead of the ball rather than towards the ball. Avoiding "spatial backtracking" causes the WAM to start moving with minimal data in a generally correct direction and provides greater time for the WAM to accelerate.

4.4 Catch Point Determination

4.4.1 Safety Constraints

The first stage in catching is to determine if the ball is catchable, and if so, where should catching be attempted. These items are determined by checking points along the object path against a series of workspace and deceleration constraints. The path of the object is defined parametrically in time using our parabolic fit. The catch point determination process is actually a catch time determination process. The catch time is varied systematically, with checks based upon the corresponding catch point coordinates. This process is repeated until a satisfactory catch time is determined.

Radial Constraints The first set of constraints are based upon the safe workspace of the WAM. Figure 4-3 illustrates two radial constraints. The physical length of the WAM determines the outer radius limitation which is approximately one meter from the base. The inner radius limitation is to protect the WAM from impacting itself during a catch. In addition, there is a singularity along the line directly above the base of the WAM, so inner radius limitation must be extended upward to avoid crossing through the singularity. These limitations then define a sphere one meter in radius and a cylinder $\frac{1}{4}$ of a meter in radius centered at the base of the WAM within which catching may not be attempted.

Z Height Constraints The next set of constraints are based on z height limitations. Figure 4-4 illustrates two height limitations. The maximum z catch height is to prevent the WAM from attempting to catch objects which are thrown in very high looping arcs over the WAM. The minimum z catch height is to allow enough room for the WAM to decelerate

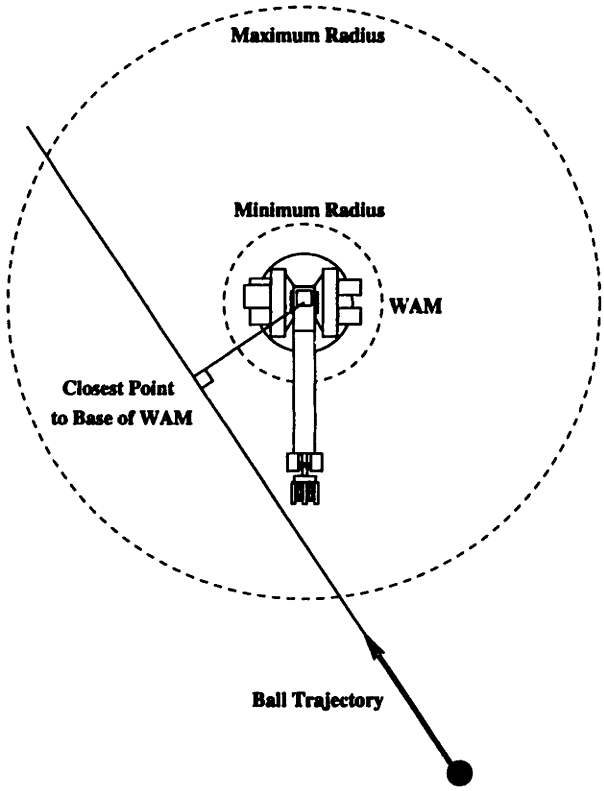


Figure 4-3: The WAM's workspace is constrained by a maximum radial distance and a minimum radial distance.

before impacting the foam floor.

4.4.2 Catch Point Determination Process

The catch point determination process begins by selecting an initial prospective catch time. Coordinates corresponding to this catch time are obtained using the parabolic fit constants. These coordinates are checked against the workspace and deceleration constraints. If the constraints are not satisfied, a new prospective catch time is selected and the process begins again. This process is repeated until a satisfactory catch time/point is determined or the toss is deemed uncatchable. The following paragraphs discuss these steps in more detail.

Starting Point The process of selecting a catch time/point begins by selecting an initial prospective catch point. The parabolic fit constants for the x and y axes have significantly

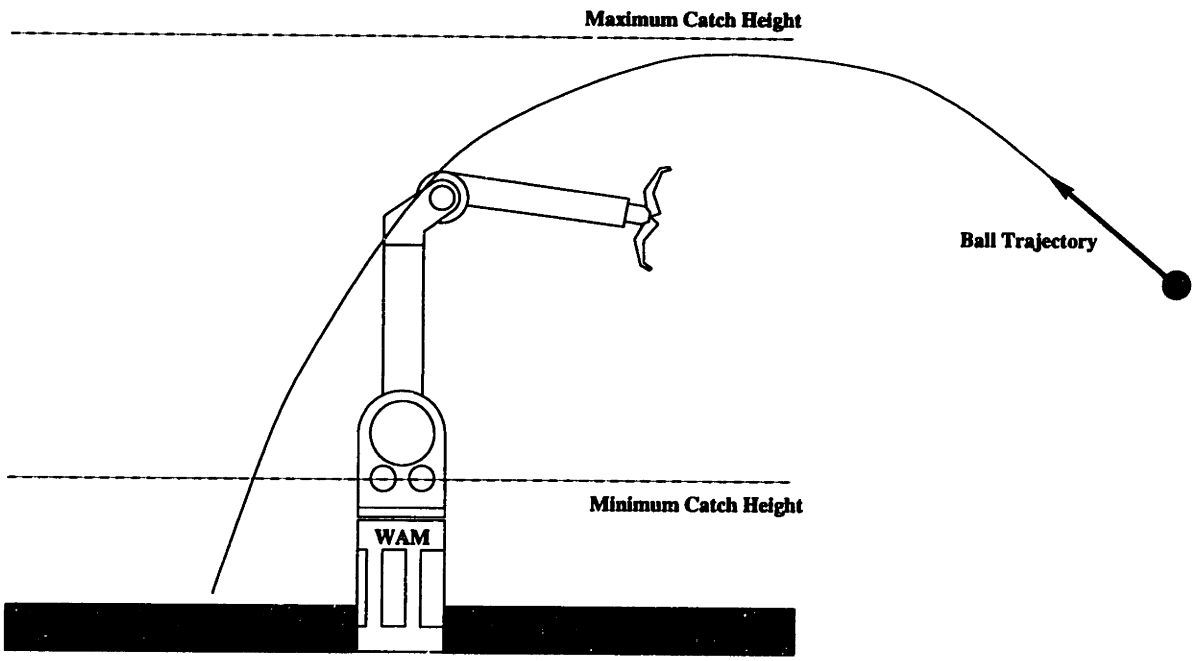


Figure 4-4: The WAM's z catching range is bounded above and below for safety.

less noise and variation during the course of the toss than the z axis. Thus the x and y velocity constants are used to locate the closest point along the path of the ball to the base of the WAM (Figure 4-3). The reason for using the closest point to the base of the WAM, as opposed to possibly using the closest point to the endpoint of the WAM, is to allow for sufficient room for acceleration. The WAM requires room to swing about to approach the catch point with the desired catch velocity.

The time for the closest point is calculated using the initial x y position of the ball and the initial x y velocity from the parabolic fit. Figure 4-5 graphically illustrates the items in the following equation.

$$t_c = \frac{d}{|\mathbf{v_i}|} = \frac{|\mathbf{p_i}|(-\hat{\mathbf{p}}_\mathbf{i} \bullet \hat{\mathbf{v}}_\mathbf{i})}{|\mathbf{v_i}|} \tag{4.2}$$

After determining a catch time, the catch time and the parabolic constants may be used to determine the coordinates of the prospective catch point.

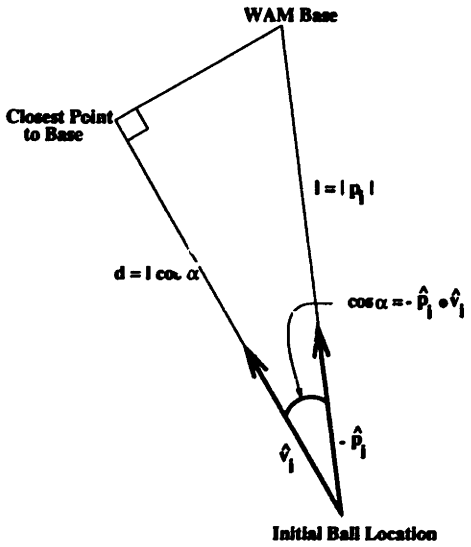


Figure 4-5: Calculations for the closest point to the base of the WAM are done using the initial position and velocity vectors for the ball toss.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} 0 & v_x & p_x \\ 0 & v_y & p_y \\ a_z & v_z & p_z \end{bmatrix} \begin{bmatrix} \frac{1}{2}t_c^2 \\ t_c \\ 1 \end{bmatrix} \tag{4.3}$$

Constraint Checking This catch point is checked against the previously mentioned workspace constraints. If the maximum radius constraint is not satisfied, the ball is not catchable. If the inner radius is not satisfied, then the ball path is passing close to the base of the WAM, so the catch time is decreased to move backwards along the ball path and the catch point is recalculated. The catch time is repeatedly decreased either until the inner radius constraint has been satisfied, or the catch time has been moved to below 0.3 seconds at which time the ball is deemed uncatchable. After satisfying the radial constraints, the z height constraints are checked. If the catch point is above the maximum z height, the ball is deemed uncatchable. If the ball is below the minimum z height, then the catch time is decreased repeatedly until this constraint is satisfied or the time has moved below 0.3 seconds.

Additional Adjustments One additional adjustment was added which gives the WAM more room for deceleration in cases where the catch point is far from the base of the WAM. In these cases the catch time is decreased by a small amount in order to catch the ball higher in its arc. This allows for a greater distance to decelerate to avoid impacting into the ground. At larger radial distances, the inertia of the WAM is greater and larger distances are required to decelerate with the same maximum acceleration.

At this point in time the dynamics of the WAM are not incorporated to determine whether the WAM itself can physically get to the ball in the amount of time given. This requires more information about the dynamics and capabilities of the motors and transmission than is easily available.

Catch Point Update and Timing of Grasp The calculation for the catch time/point is repeated each time the predicted parabolic constants change (approximately 60Hz). This assures that the catch point stays within the safety region of the workspace as the parabolic constants change. As a result, the catch point and time vary as the toss progresses. But, once the command to close the hand has been given the catch time must become fixed. The Talon requires approximately 0.33 seconds to fully close. After the close command is given, the Talon cannot easily stop the fingers. Therefore the catch time must be fixed after the close command is given so that when the Talon is fully closed, the Talon and the ball are coincident. Although the catch *time* is fixed, the catch point is allowed to vary as the parabolic constants vary. If the resulting catch point moves out of the safe workspace of the WAM after the catch time has been fixed, then the ball is uncatchable and the WAM returns to its "home" position.

Result of Determination Process After the sequence of constraint checking above, either a satisfactory catch time/point has been determined or the ball has been deemed uncatchable. If the ball in uncatchable, the WAM decelerates from its current motion and return to its "home" position. If a satisfactory catch time/point has been determined the WAM continues with its path generation to intercept and match trajectories with the object as described below.

4.5 WAM Path Generation

Various methods for desired path generation for the WAM have been examined. In the initial stages of this work, carrying over from past work, a second order filter approach was used. In order to improve the reliability and robustness of the system to a wider variety of tosses and toss locations, polynomial path generation techniques are now applied [Kimura 92].

4.5.1 Second Order Filters

In previous experiments with this system, a second order filter was used to generate a desired path for the WAM. For our initial experiments, a similar method was attempted. After determining a catch point, second order filters with variable filter constants were applied in the x, y, and z directions. The desired acceleration for the x axis ($\ddot{x_d}$) was given by the following equation using the coordinates at the catch point (x_c) and the velocity at the catch point ($\dot{x_c}$)

$$\ddot{x_d} = 2\lambda(\dot{x_c} - \dot{x_d}) + \lambda^2(x_c - x_d) \tag{4.4}$$

with similar equations for the y and z accelerations. The desired velocity and positions were then integrated with this acceleration. The result was a smooth position and velocity profile but with a discontinuous acceleration profile if the catch position and velocity vary.

One problem with this method is that it had very high accelerations initially because of the large driving error. With small driving error, the acceleration becomes small, which translates to slow responsiveness. In an attempt to compensate for this effect, a time varying filter constant was incorporated. The filter constant on the second order filter was increased and decreased between an upper and lower bound based upon the hand's distance from the ball. The variable filter was implemented using a first order filter on λ.

$$\lambda = \lambda + f(\lambda_{high/low} - \lambda) \tag{4.5}$$

Values for f, λ_{high}, and λ_{low} were selected for stability and λ was initially set to λ_{low}. As

the catch progresses, λ approaches λ_{high} as the WAM approaches the ball. Although this allowed for larger accelerations with small driving errors, there were limitations on how large f and λ_{high} could be chosen. With larger filter constants, the filter magnifies noise and becomes unstable.

Another problem with this method is that it never guarantees the time of travel for the path. Although the arm is commanded to move to the catch point, the time at which it will reach the desired point is not known. For similar tosses to the same general region, the system could be tuned to catch successfully, but this method proved to be quite unreliable for a variety of tosses.

4.5.2 Third Order Polynomials

In order to have a better grasp on the timing of the catching as well as the acceleration values for the arm, polynomials are now used to generate intercepting paths for the WAM.

Polynomial Definition Third to fifth order polynomials have been considered for path generation. Third order polynomials allow for matching of position and velocity at both endpoints of the segment. Fifth order polynomials allow for matching position, velocity, and acceleration at both endpoints. The fifth order polynomials are smoother and as a result will usually take a longer path to match the acceleration of the ball at the end of the catch. Fifth order polynomials were found to exceed the workspace of the WAM by a large amount. The third order polynomials cause discontinuities in acceleration, but they are much closer to remaining within the workspace of the WAM. Therefore, third order polynomials in x, y, and z are used for path generation for the WAM.

The third order polynomial are generated between the current location of the WAM and the prospective catch point. The polynomial is of the form

$$\mathbf{p}(t) = \frac{1}{6}\mathbf{j_p}t^3 + \frac{1}{2}\mathbf{a_p}t^2 + \mathbf{v_p}t + \mathbf{p_p} \tag{4.6}$$

where each of the bold faced items are 3x1 vectors with x, y, and z components and $\mathbf{j}$ is used to denote "jerk", the derivative of acceleration.

Concerns One issue which needed to be addressed was what to do when the polynomial path exceeded the workspace of the WAM. Here, a combination of the "catching" and "snatching" approaches are used to resolve this issue. When the path generated by the third order polynomial is found to exceed the workspace of the WAM, the magnitude of the velocity at the catch point which the WAM attempts to match is scaled down. By repeatedly reducing the magnitude of the velocity to be matched, the desired path for the WAM may be maintained within the safe workspace limits. This is equivalent to assuring that the WAM reaches the catch point at the appropriate time but without the full magnitude velocity, thereby reducing the length of the path traveled.

Generation of Path In order to generate the polynomial path, constants in the polynomial equation 4.6 are determined from knowledge of the initial position and velocity ($\mathbf{p}(t_1)$ and $\mathbf{v}(t_1)$), final position and velocity ($\mathbf{p}(t_2)$ and $\mathbf{v}(t_2)$), and the time difference. The jerk and the acceleration constants are determined first

$$\mathbf{j_p} = \frac{-6}{(t_2-t_1)^2}\left\{\frac{2}{(t_2-t_1)}(\mathbf{p}(t_2)-\mathbf{p}(t_1)) - \alpha\mathbf{v}(t_2) - \mathbf{v}(t_1)\right\} \tag{4.7}$$

$$\mathbf{a_p} = \frac{2}{t_2-t_1}\left\{-\frac{1}{6}\mathbf{j_p}(t_2^2+t_1t_2-2t_1^2) + \frac{1}{t_2-t_1}(\mathbf{p}(t_2)-\mathbf{p}(t_1)-\mathbf{v}(t_1)\right\} \tag{4.8}$$

and these are then used in the following two equations to determine the velocity and position constants.

$$\mathbf{v_p} = \mathbf{v}(t_1) - \mathbf{a_p}t_1 - \frac{1}{2}\mathbf{j_p}t_1^2 \tag{4.9}$$

$$\mathbf{p_p} = \mathbf{p}(t_1) - \mathbf{v_p}t_1 - \frac{1}{2}\mathbf{a_p}t_1^2 - \frac{1}{6}\mathbf{j_p}t_1^3 \tag{4.10}$$

The α term multiplying the velocity vector at time t_2 in equation 4.7 is our compensation for the outer radius workspace constraint which was discussed previously. It begins at 1.0 and is decreased as necessary to satisfy workspace constraints during path generation.

These constants are then used in the following equation to generate the path for the WAM for time t.

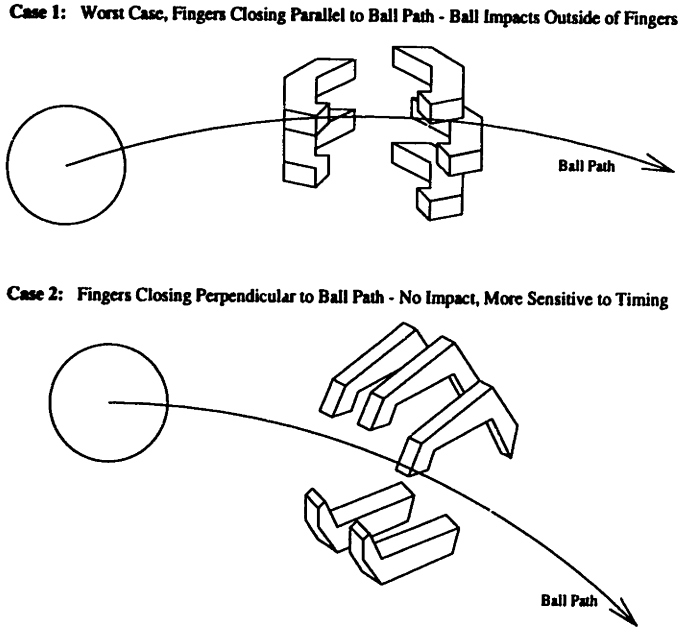


Figure 4-6: Two different cases for Talon finger orientation during closing.

$$\begin{bmatrix} \ddot{x}_d \\ \dot{x}_d \\ x_d \end{bmatrix} = \begin{bmatrix} 0 & 0 & j_{p_x} & a_{p_x} \\ 0 & j_{p_x} & a_{p_x} & v_{p_x} \\ j_{p_x} & a_{p_x} & v_{p_x} & p_{p_x} \end{bmatrix} \begin{bmatrix} \frac{1}{6}t^3 \\ \frac{1}{2}t^2 \\ t \\ 1 \end{bmatrix} \tag{4.11}$$

Similar equations are used to generate paths for y and z. The resulting trajectory is then fed to the WAM adaptive controller.

4.6 Talon Orientation and Timing of Closure

During catching, the Talon supination/pronation (forearm) joint must be oriented such that the fingers close perpendicular to the path of the ball.

At certain orientations the Talon fingers will block the path of the ball entering the hand, so therefore the Talon should orient itself such that the fingers close in a direction which is perpendicular to the path of the object. Figure 4-6 illustrates two cases, one where the fingers are oriented in the worst case, and a second where the fingers are oriented correctly.

In this way, the ball impacting the outer surface of the fingers can be avoided. To determine the correct position for the supination/pronation joint, the forward kinematics of the WAM and the desired velocity vector at the catch point are used. From the joint values for the WAM and the forward kinematics, the transformation from the WAM base coordinate frame (world frame) to the end effector coordinate frame may be determined. The catch point velocity vector is then transformed to the end effector coordinate frame. The transformed velocity vector is projected into the xy plane of the end effector coordinate frame. The projection must be used because an extra degree of freedom would be required in the Talon in order to orient with respect to any arbitrary vector. The desired supination/pronation joint value is then set to the angle between the projected velocity vector and the x axis (see Appendix B.2). The Talon is servo-ed to this joint angle using a second order filter.

Also during catching, the latency between the issuing of the close command the actual finger closure must be considered. The minimum time for the Talon fingers to move from fully open to fully closed is approximately 0.33 seconds. Due to this latency, the command to close the fingers is given before the WAM has reached the catch point. Although the degree of finger closure could be servo-ed based upon the remaining time until catching, the logistics would be much more complicated. Therefore, once the Talon command to close the fingers is given, the motion must be completed. For this reason, once the close command has been issued, the catch time is fixed to ensure that when the fingers are closed, the WAM is in the correct catching location.

4.7 Path Matching and Post Catching Deceleration

Once the catch point has been reached, to increase the probability of successful catching, the WAM matches trajectories with the object for a short period of time before deceleration. For 0.05 seconds after the catch time, the desired WAM path is coincident with the predicted ball path.

After matching paths, care is taken to reduce the jarring experienced by the object and to gradually introduce the change of mass of the end of the WAM for heavy objects. Therefore, the WAM decelerates gracefully along the prior path of the object. A large

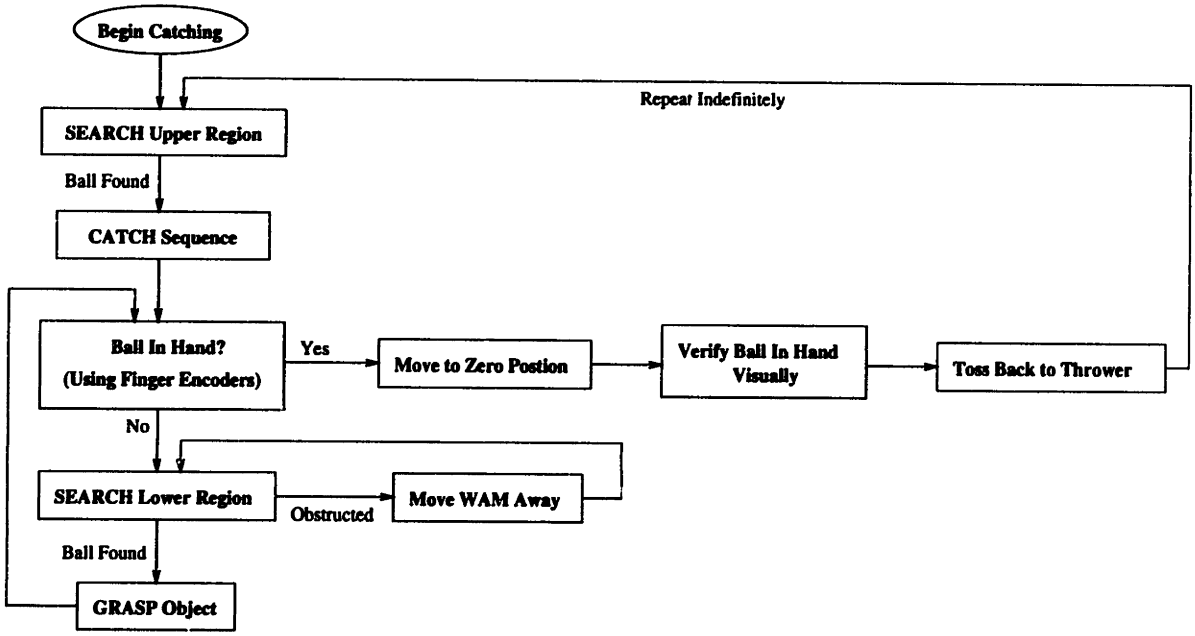


Figure 4-7: Flowchart of events which are used for the WAM to play "catch" with a human thrower.

deceleration is required as a result of the path matching during which the WAM achieves the same -9.81 m/s^2 acceleration as the object. Safety constraints are imposed during deceleration using force fields to prevent impacts with the floor and the WAM itself. A positive z force field in software is placed above the foam and a radially directed force field in software is placed about the base of the WAM. If the WAM desired path intersects the z force field, a positive z acceleration is applied. If the WAM intersects the radial force field, a radial acceleration outward is applied. In extreme cases where the WAM has attempted to catch objects far from the base of the WAM, the maximum torque applied to the motors is insufficient to prevent the WAM from impacting into the foam.

4.8 Testing Environment

To simplify the process of testing changes to the catching algorithm, the WAM has been programmed to play a game of "catch". Figure 4-7 shows a flowchart for the sequence of events in the infinite loop catching state. The game of "catch" begins with the FEGs searching for an object in the thrower's hand. When an object is found, the WAM waits

for the toss to be triggered and then catching sequence in Figure 4-1 is executed. Once the catching sequence is completed, the WAM checks whether the catch was successful. If successful, the WAM moves to its vertical zero position, the Eyes visually verify that the ball is in the hand, and then the WAM tosses the ball back in the direction of the thrower. If the catch was unsuccessful, the ball should have landed on the floor within the WAM workspace. The Eyes search the floor for an object using the consistency checking search routine in Figure 3-10. If the object is obstructed, the WAM moves away and the Eyes re-execute the search. Once the ball is found, the WAM picks up the ball using the grasping sequence in Figure 3-12 and then tosses the ball back to the thrower. Once the thrower has caught the ball, the sequence is begun once again.

Audio output is also incorporated in the system to add some personality to the system and to aid in debugging. Useful pre-recorded and synthesized audio clips are played at various moments during the catching state.

4.9 Applicability of Methods for Non-Spherical Objects or Non-Parabolic Trajectories

The catching algorithms discussed here may be applied to catching non-spherical objects, such as cylinders, with varied success. The problems associated with non-spherical objects primarily relate to the FEG tracking methods. As discussed previously, the FEGs would experience problems with large objects, objects whose center of area varies as the viewing perspective changes, or objects which have different lighting characteristics.

But the problems do not solely lie with tracking. For instance, to catching rotating cylinders, ideally the WAM would like to match rotations to decelerate the rotation as well as the translation in a smooth manner. To attempt to match arbitrary rotations, one additional degree of freedom perpendicular to the direction of closure of the fingers would be required.

For non-parabolic trajectories, new path prediction methods need to be incorporated. Past work [Cannon, Slotine 95] has laid the foundations for using neural networks to predict in real-time the aerodynamic forces on objects and thereby predict their future paths. Cur-

rent work is being explored to combine this work to accomplish successful catching of objects with non-parabolic non-planar trajectories. Aside from the path prediction techniques, the other methods such as catch time/point determination and WAM path generation should be equally applicable with some work on interfacing required. Small modifications to the triggering method might also be necessary due to the change in throwing manner for objects such as paper airplanes.

Chapter 5

Experimental Results

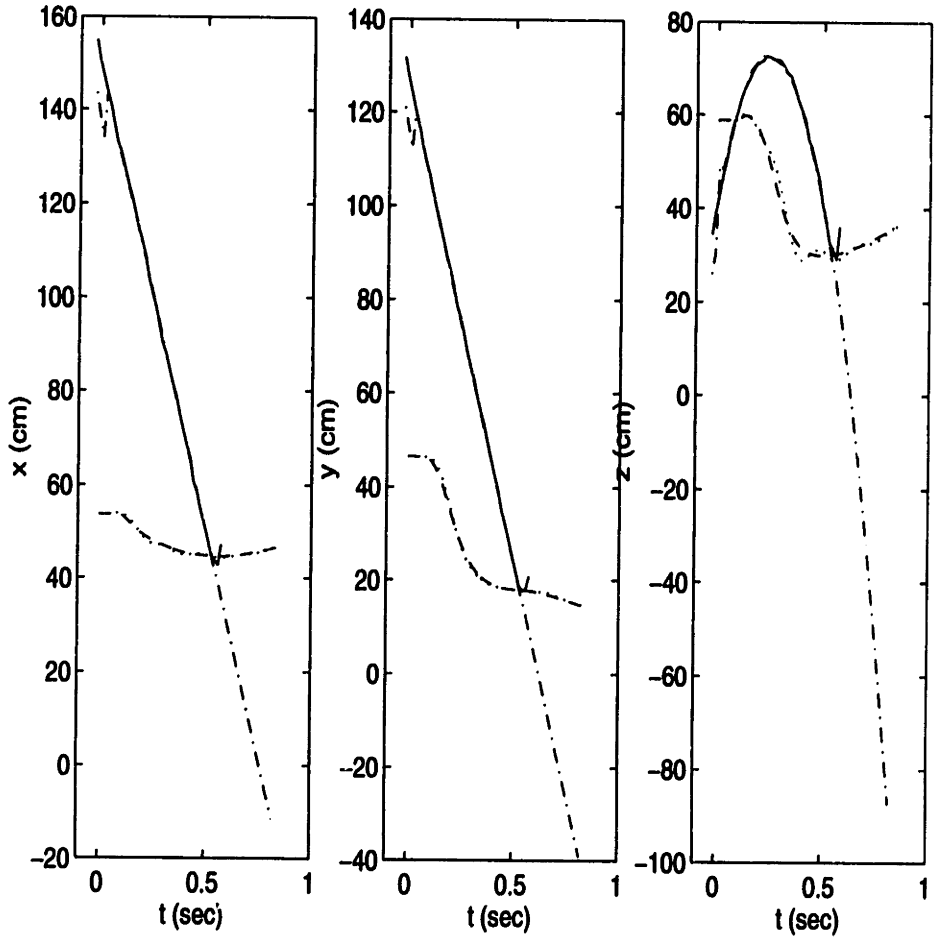
This chapter presents results for the two approaches to catching discussed in the previous chapter. Results for the "snatching" approach are presented first. Then, more in depth results for the "catching" approach are presented. Finally this chapter closes with a discussion of the sources of error which affect the performance of both approaches.

The following experimental results were obtained without special calibration or tuning required. A one time training is required to store color histogram information for the vision boards and a one time FEG/WAM transformation must be computed. Unlike the previous system, which required frequent re-calibration, once completed, these calibrations need not be adjusted each time the system is run. Upon start-up, after the basic home finding procedures are run for both the FEGs and the WAM, the system is capable of catching.

The following sections present results from under-hand tosses of a fluorescent orange ball from random locations approximately 1.5-2.5 meters distant from the base of the WAM.

5.1 Snatching Results

The success rate of the "snatching" approach was found to be less than 10%. This method proved to be too sensitive to timing and noise to produce reliable and robust performance. The WAM would be close, but the fine position tuning required during the last moments of catching were often too quick for the WAM to respond with a second order filter. Often the ball would deflect off the fingers. This section presents data from one successful "snatch".



solid - ball data, dashdot - fit, dash - WAM desired, dotted - WAM actual

Figure 5-1: Plots of xyz vs. time of the Ball and WAM trajectories during a successful "snatching" attempt.

Figure 5-1 shows plots of each of the axes with respect to time. Note how the WAM moves to the desired catch position before the ball and then remains there until the ball and the hand coincide. The small up-curve of the ball data at the end of the toss is a result of the successful grasp where the fingers have obscured portions of the ball. This caused a shift in the calculated coordinates.

In the background of the path generation, there are additional computations which are of interest. Note that the parabolic fit is updated with every new vision sample, therefore the position and time at which the WAM would like to catch changes during the course

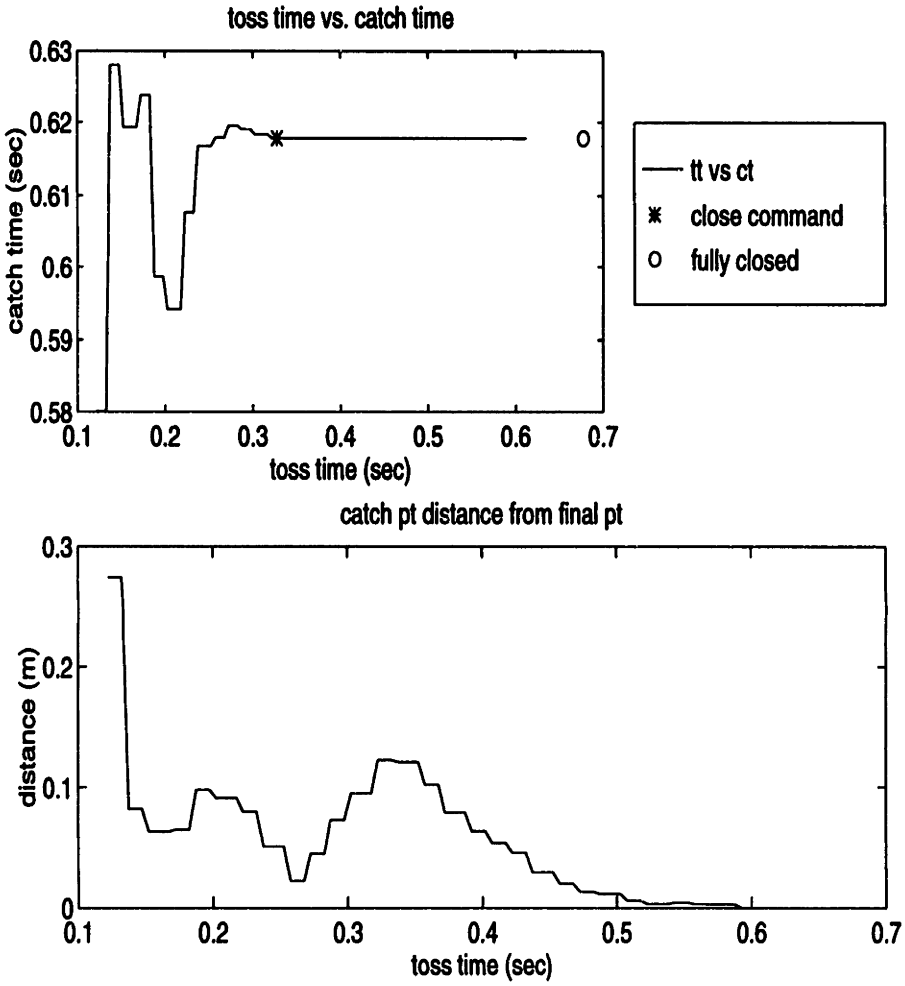


Figure 5-2: Plots of prospective catch times versus toss time (top) and the distance from the final catch coordinates versus toss time for the "snatching" attempt (bottom).

of the toss. Figure 5-2 shows two figures which show the evolution of the catch time and point. The upper plot shows the evolution of the catch time versus the time of the toss. All times are relative to when the toss has been triggered. Note how the catch time becomes constant once the close command for the fingers is given. The lower plot shows the distance of the changing catch point from the final catch point coordinates. Note how the distance decreases after the command to close the hand has been given and the catch time has been fixed. This shows how the parabolic fit converges towards the final catch point. Note that the whole toss and "snatch" attempt is completed in less than 0.6 seconds.

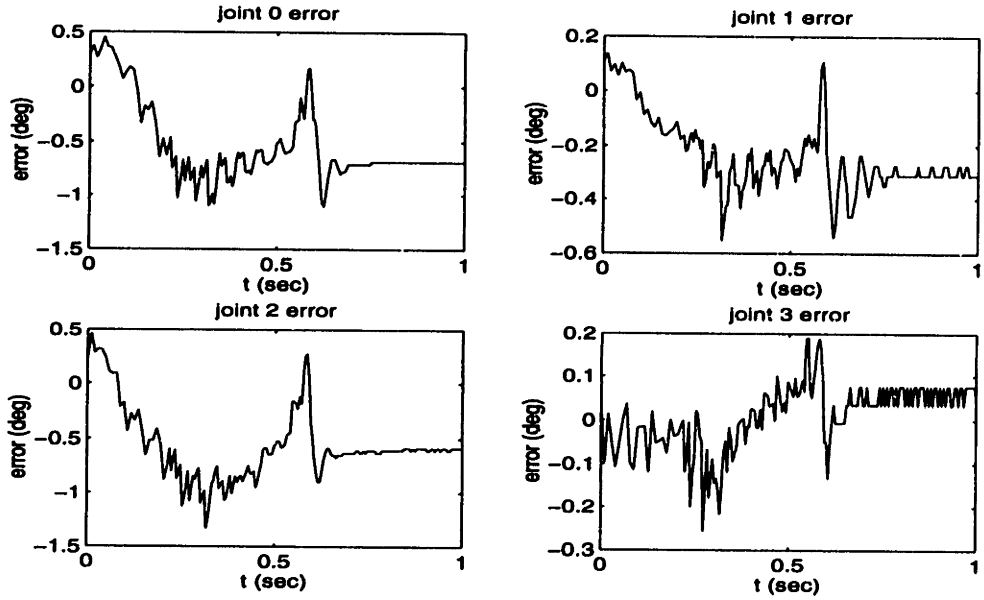


Figure 5-3: Plots of FEG tracking error for all four joints during a successful "snatching" attempt.

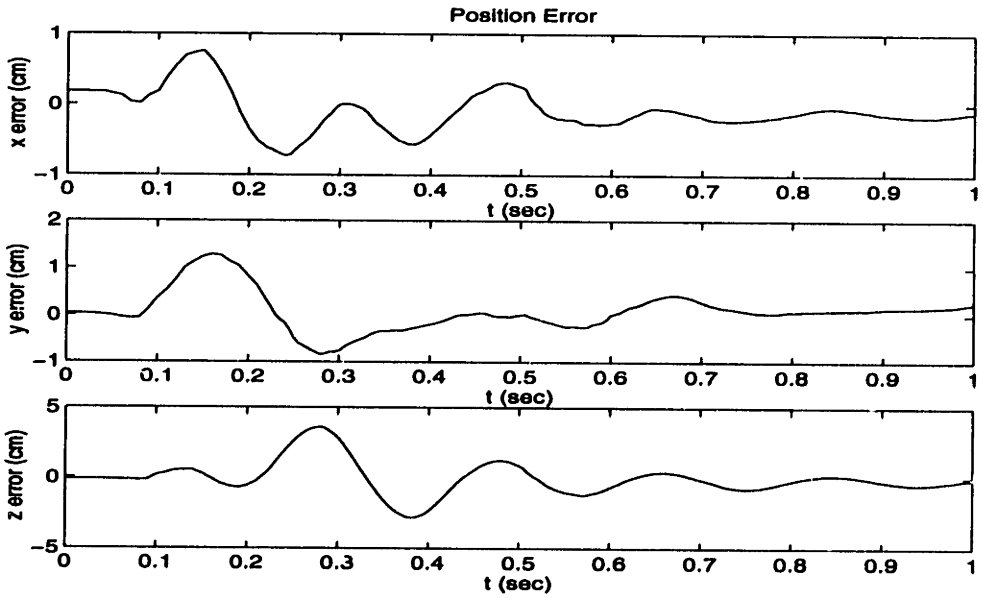


Figure 5-4: WAM cartesian tracking error during a successful "snatching" attempt.

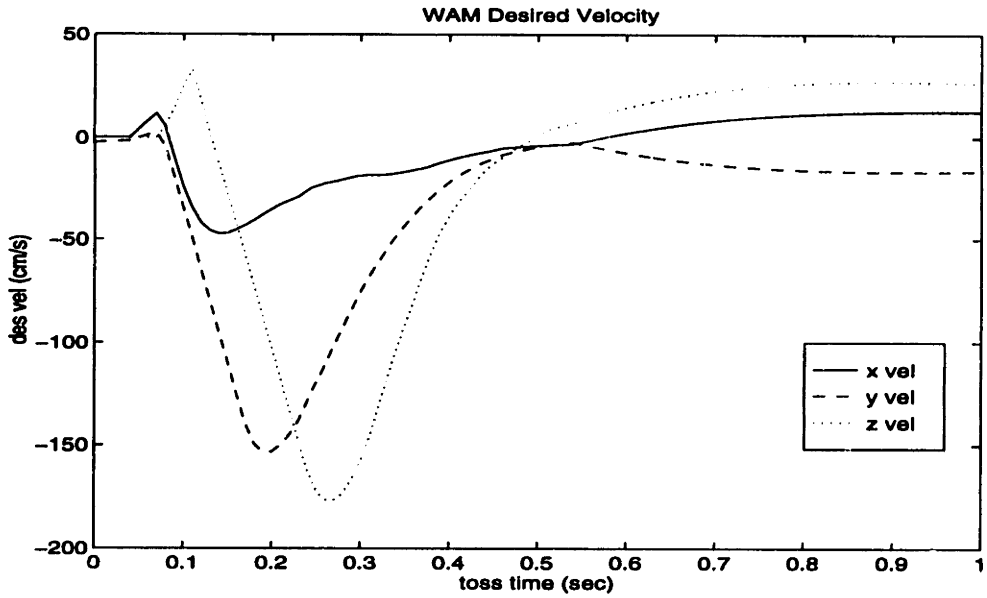


Figure 5-5: WAM cartesian desired velocity during a successful "snatching" attempt.

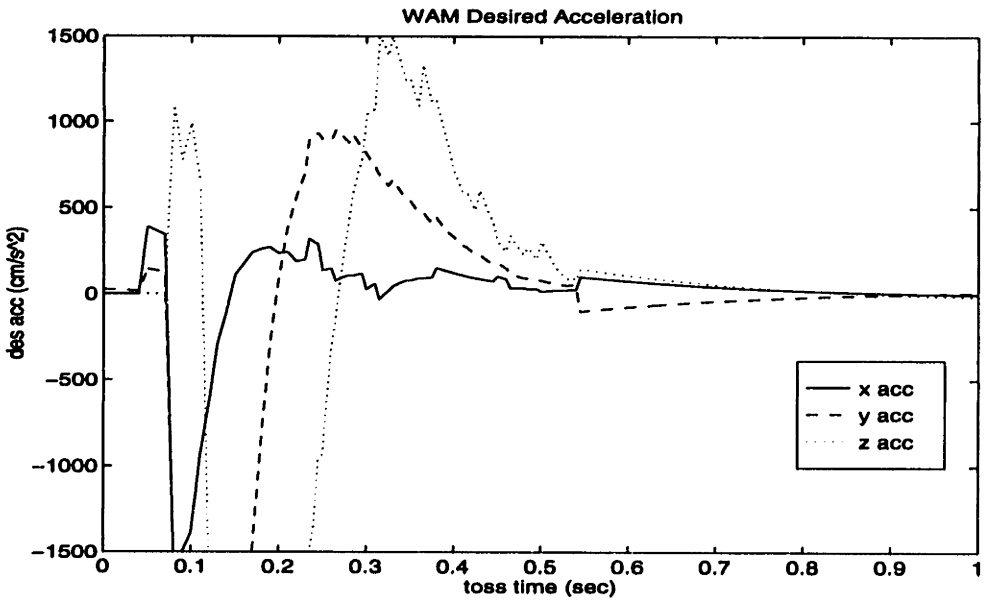


Figure 5-6: WAM cartesian desired acceleration during a successful "snatching" attempt.

In addition to the desired paths for the FEG and the WAM, plots of the FEG and WAM tracking error are shown. Both the FEG and the WAM are running adaptive controllers with large adaptation gains to quickly adapt to parameters during the duration of the toss. Figure 5-3 shows the tracking error for the FEGs for all four joint axes during the toss. Figure 5-4 shows the WAM tracking error during the course of the toss. The maximum error for any one axis was approximately 2.5 cm (1 in) during the initial motion, but note that the error remains within 1 cm during the later half of the "snatching" attempt after time was given for the adaptation to determine good parameters and where positioning accuracy is vital. And finally, to better view the motion of the WAM, Figure 5-5 and Figure 5-6 show the desired velocity and acceleration. The maximum acceleration for the WAM is limited to $\pm$ 15.0 m/s^2, which is a safety limitation based upon calculations of maximum motor torques at full extension.

Results from unsuccessful "snatching" attempts do not appear significantly different. The required precision in timing and positioning for success is so high that it is on the order of best capabilities of the system.

5.2 Catching Results

Our full "catching" approach using third order polynomials for path generation for the WAM proved to be a significantly better method for catching. Testing was done to measure of the repeatability of the system. Tosses were made to the same general area of the WAM's workspace, with all of the final catch points lying within a two foot cube. The success rate for catching was found to be roughly 70% - 80%. On one testing run, there were 53 successful catches from a set of 75 attempts with a lengthy sequence of 14 successful catches. In every failed attempt, the ball impacted the end effector or occasionally passed through the closing fingers. These statistics should be viewed as the current *best* results for the system. Performance decreases for more difficult tosses (faster or farther from the base of the WAM). The sources of error which cause the failures are discussed in more detail in the next section. The results for a sample catching run are presented here, similar to those plots presented for the "snatching" approach previously. In addition, Appendix B.3

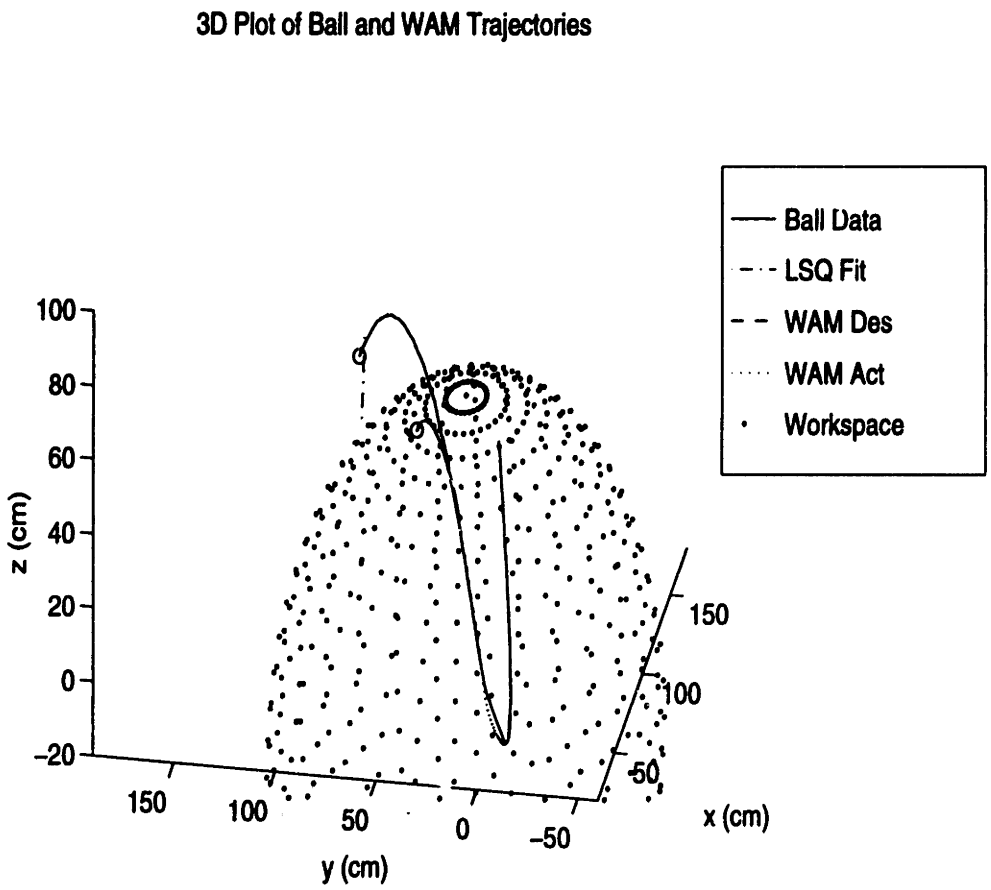


Figure 5-7: 3D View of the Ball and WAM trajectories during a successful "catching" attempt.

contains photographs for two catching attempts.

Figure 5-7 shows a 3D perspective of the catching attempt. Note how the path for the WAM moves parallel to the path of the ball so that the trajectories are matched for a small duration of time.

Figure 5-8 shows the x, y, and z plots versus time. The x data flattens out towards the end of the catch as a result of the force field which is placed about the base of the WAM to help prevent impacts during the deceleration stage. Note how the distance between the WAM and the ball converges to zero and remain zero for the duration of trajectory

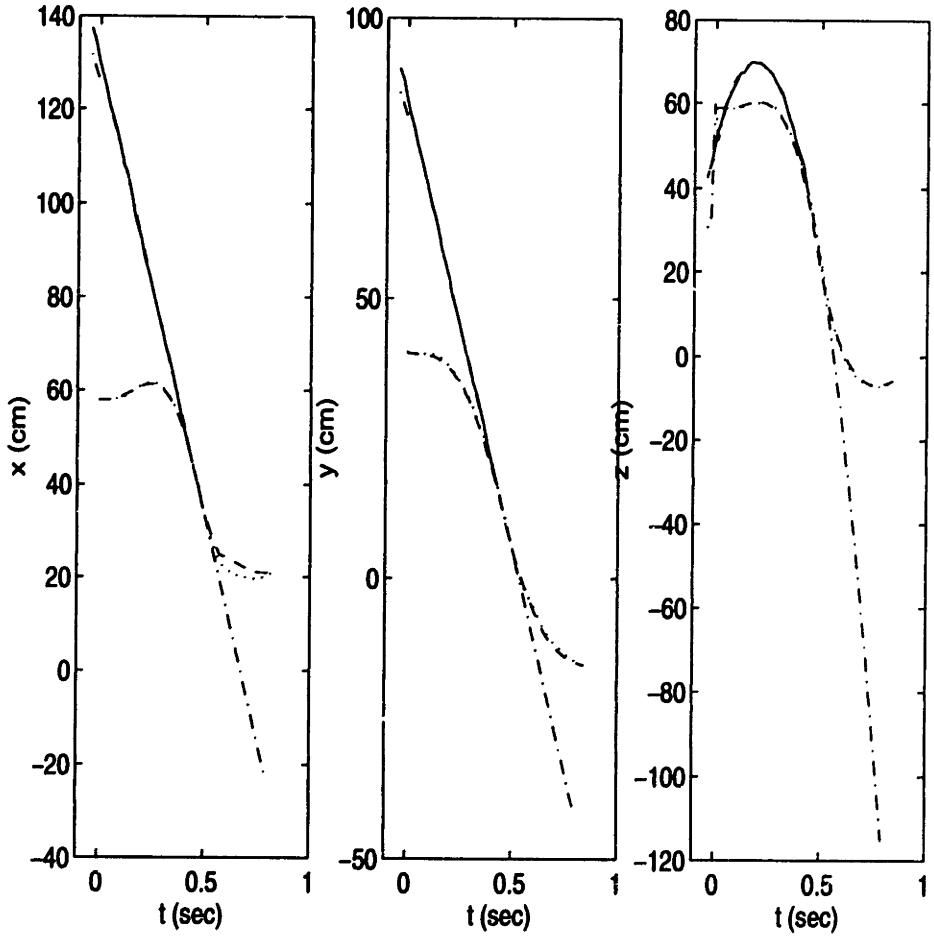


Figure 5-8: Plots of xyz vs. time of the Ball and WAM trajectories during a successful "catching" attempt.

matching. It can also be seen that the parabolic fit matches the actual ball data so well that the lines are overlapping.

Again, because of the changing parabolic fit estimates and the safety constraints which are applied to determine catch time/points, the catch time/point will vary as the toss progresses. Figure 5-9 shows the evolution of the prospective catch time/point versus the time of the toss. The catch point can be seen to steadily decrease and approach the final catch point. Over the course of the toss, except for the initial 0.25 seconds, the catch point varied less than 1 inch, decreasing to almost zero for the last 0.05 seconds.

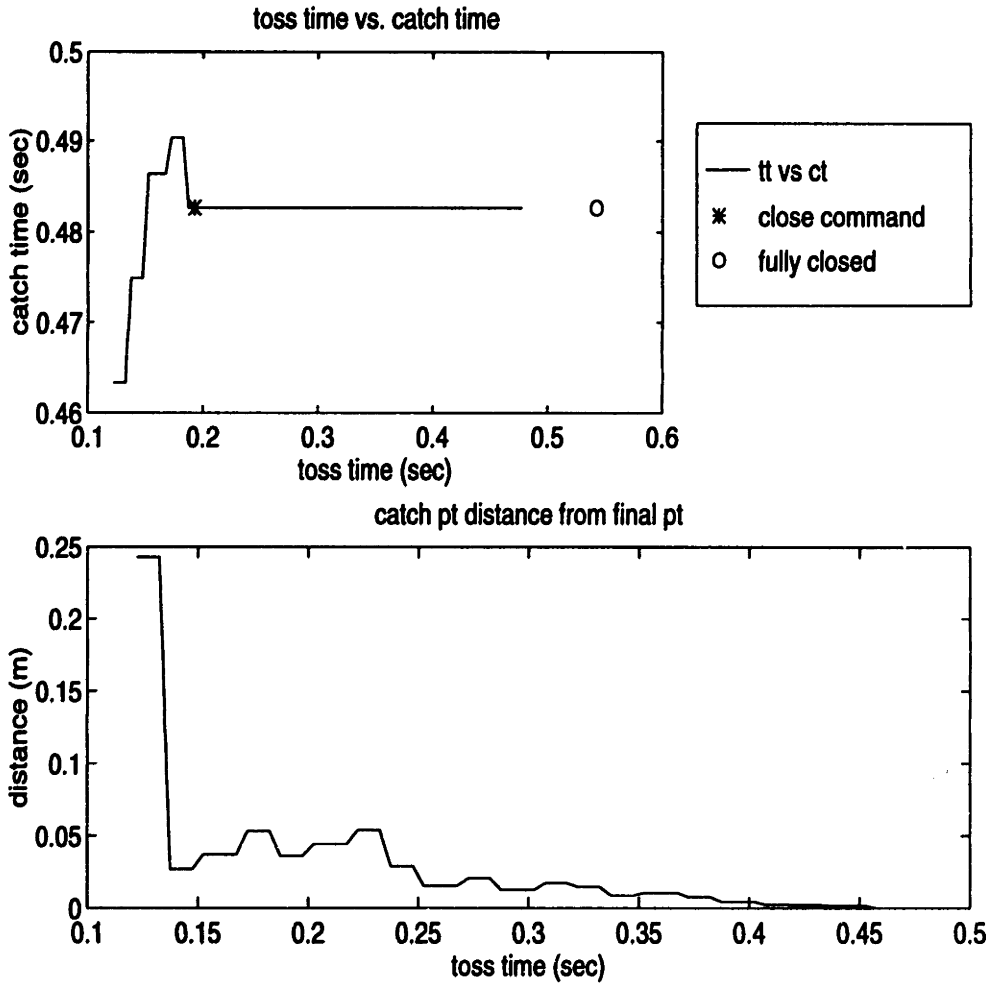


Figure 5-9: Plots of prospective catch times versus toss time (top) and the distance from the final catch coordinates versus toss time for the "catching" attempt (bottom).

The Adaptive Controllers for both FEGs and the WAM are activated with large adaptation gains for the duration of the toss. Figures 5-10 and 5-11 show the tracking error for the FEGs and the WAM. The tracking error for the FEGs is somewhat deceiving because it is the error between the actual and the desired joint values. But there is no guarantee that the desired reflect the actual location of the ball at that time. Similarly, a large error in FEG tracking does not necessarily equate with large error in position determination. As long as the object is in the screen, the offset from the center of the screen is accounted for. But, the farther from the center of the screen, the greater the distortion and the greater

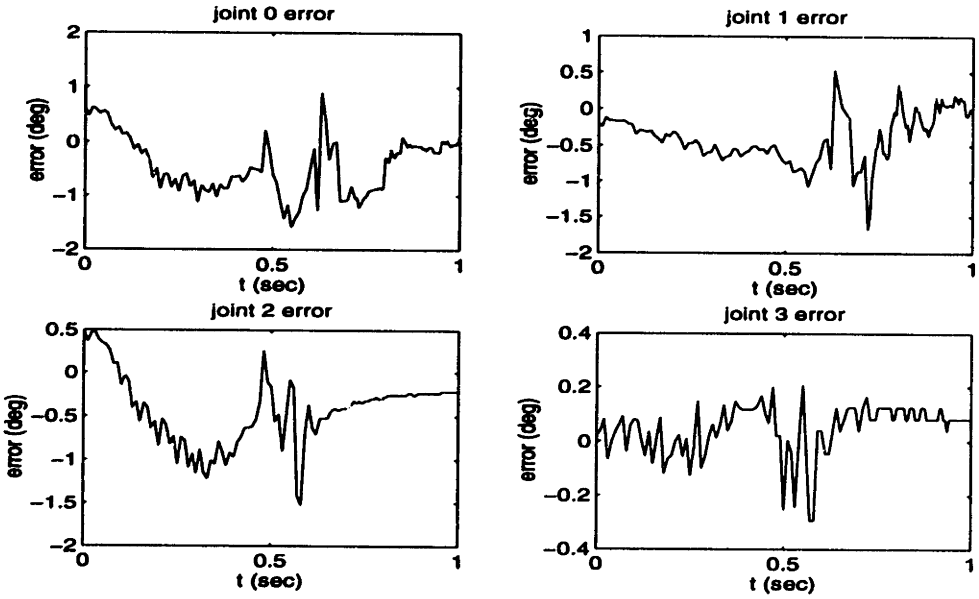


Figure 5-10: Plots of FEG tracking error for all four joints during a succesful "catching" attempt.

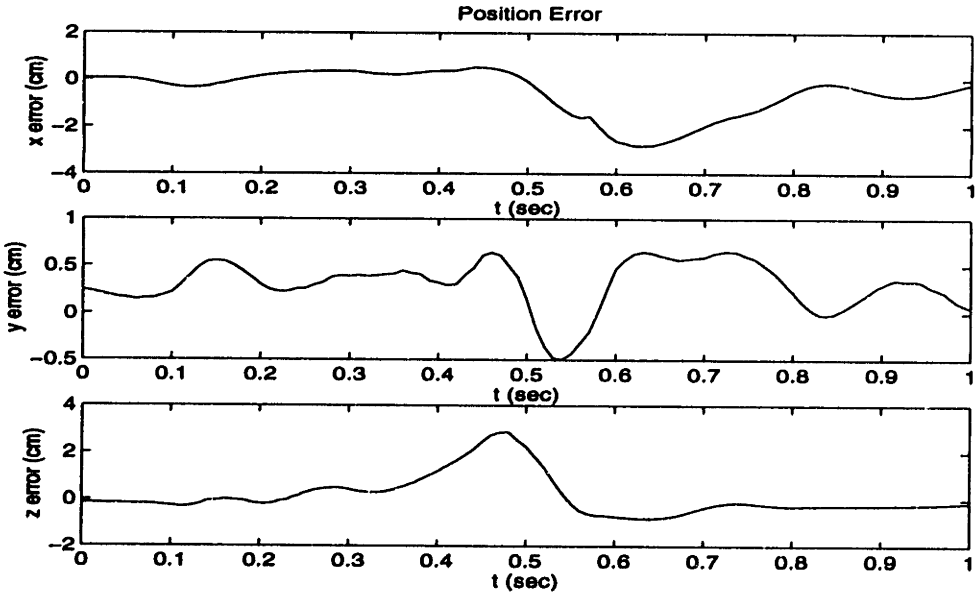


Figure 5-11: WAM cartesian tracking error during a successful "catching" attempt.

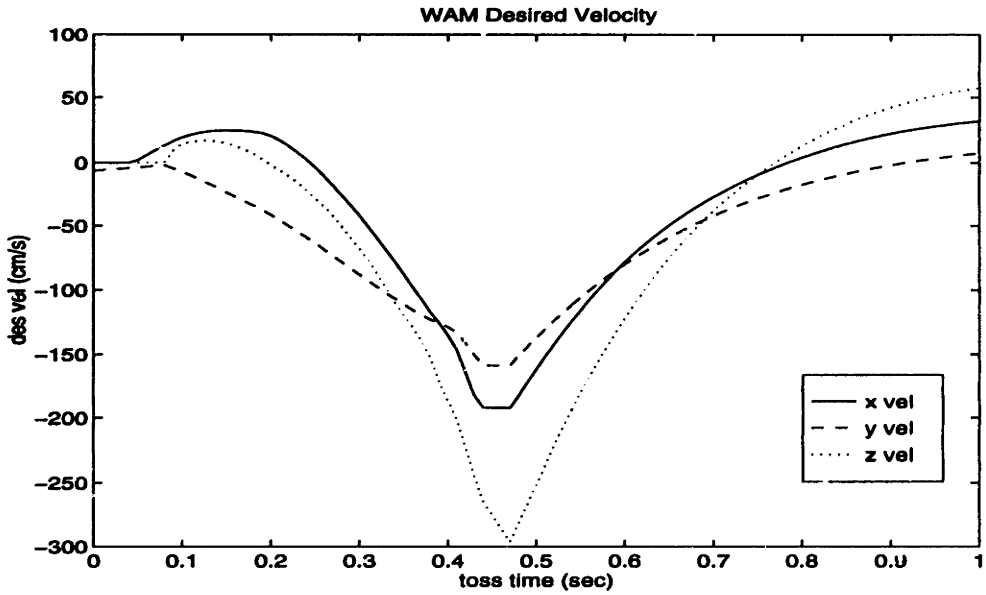


Figure 5-12: WAM cartesian desired velocity during a successful "catching" attempt.

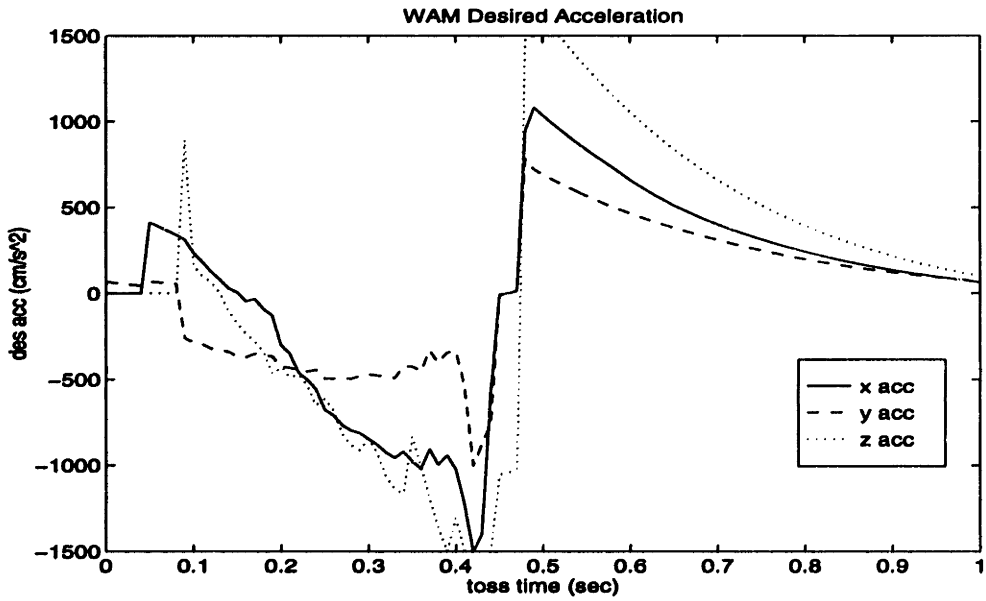


Figure 5-13: WAM cartesian desired acceleration during a successful "catching" attempt.

Number of Samples	Acceleration (m/s^2) X	Y	Z	Velocity (m/s) X	Y	Z	Position (m) X	Y	Z
2	0.000	0.000	-9.810[1]	-1.544	-1.145	1.658	1.317	0.866	0.304
5	0.000	0.000	-8.300[1]	-1.963	-1.596	2.312	1.377	0.911	0.424
10	0.000	0.000	-10.963	-1.890	-1.594	2.447	1.375	0.912	0.424
15	0.000	0.000	-10.745	-1.914	-1.597	2.421	1.377	0.912	0.424
20	0.000	0.000	-10.575	-1.932	-1.601	2.404	1.378	0.912	0.425
25	0.000	0.000	-10.415	-1.918	-1.585	2.382	1.376	0.910	0.426
30	0.000	0.000	-10.146	-1.928	-1.587	2.335	1.378	0.911	0.428

[1] These values are set from our initial constraints for the estimates for z acceleration.

Table 5.1: Evolution of predicted parabolic constants during the "catching" attempt.

the chance for a portion of the object to be outside of the field of view - yielding slightly incorrect data. Towards the end of the toss, as the ball moves closer to the cameras and the tracking task becomes more difficult, the FEGs will often lose track of the ball. But in the majority of cases, the attempted grasp by the WAM occurs first, obstructing the FEGs, causing them to lose track of the ball. In contrast to the FEG tracking performance, the tracking performance for the WAM is much more vital to successful catching. As will be seen in the next section, the tolerance for position error is very small, therefore the WAM tracking error should be as small as possible. The WAM tracking error remains within approximately 1 cm for the vital portion of the catching attempt. The large error near 0.5 seconds is a result of the transition to the deceleration stage where there is a large jump in acceleration.

Figures 5-12 and 5-13 show the WAM desired velocity and acceleration respectively. A continuous desired position and velocity, and as much as possible, acceleration should be provided for the WAM. Because third order polynomials are used, a smooth continuous desired velocity is guaranteed. Aside from transients where switching between stages of catching occur, the acceleration should not have large steps as long as the parabolic fit estimates do not vary considerably. The initial jump in acceleration is a result of the start of the polynomial, and the large jump just prior to 0.5 seconds is a result of the transition to the deceleration stage.

Table 5.1 shows the evolution of the predicted parabolic constants during the course of

the toss. Note how after the initial data, the estimated velocity and position constants vary only slightly. Also note how the estimated z acceleration is not very close to gravitational acceleration. Estimating z acceleration provides more accuracy than assuming gravitational acceleration. As evidenced by these results, the assumption of gravitational acceleration would produce incorrect results.

5.3 Sources of Error

5.3.1 Calibration, Accuracy, and Timing

The best positioning accuracy in steady state for the WAM is approximately 1/5 inch primarily due to small inaccuracies in gear ratio measurements. Position determination accuracy for the FEGs across the whole workspace of the WAM is approximately 0.4 inches. But, the effect of the FEG position determination error is considerably reduced because of the parabolic fit over 20-40 points.

In the best case, the hand can afford approximately 1/2 inch position error and less than 0.005 seconds timing error and still succeed in grasping the object. For the average toss, the ball is moving with a velocity near 3 m/s and with gravitational acceleration acting on the z axis. In 0.005 seconds, the ball will travel close to 1/2 inch.

Therefore with these requirements for catching, there is little room for the presence of noise and tracking error for the WAM. The Adaptive Control with the fast adaptation for the WAM is required in order to have the tracking performance which remains within our constraints. Similarly, Adaptive Control for the FEGs is required to maintain the ball within the field of view for successful tracking and prediction.

5.3.2 Vision System Latency and Lighting Effects

The majority of failures can be attributed to noisy data from the vision system which results in more error in the prediction. The bad data results primarily from inaccurate compensation for time delays. It is difficult to completely accurately compensate for the latency in the vision system. Each vision board outputs information once it has finished computation on a frame. Depending upon the number of pixels, blobs, and edges in the

image, the timing of completion will vary. Occasionally processing will take sufficient time to cause the vision boards to overrun the next frame and drop to 30 Hz output.

Therefore there are two places where the code must compensate for timing. First, the delay resulting from vision processing for the current frame for each camera must be estimated. This delay is used to determine where the FEGs were positioned at the time the image was taken. Next, it must be assured that information from both cameras from the same moment in time are being used. The cameras are synchronized in hardware. Therefore, it is known that the images were taken at the same time, but due to processing delays, the controller might receive information from the vision boards at different times. The data from one of the two boards must be integrated using estimates of the object velocity and acceleration to synchronize the time stamps from both cameras. Both of these computations will have small inaccuracies which will result in noise.

For the most part, the cause for increased computation time is the result of white balance and lighting effects. As the ball travels towards the WAM, it passes under lighting fixtures in the room and over different backgrounds. In the bright regions, the light reflecting off the surface of the ball cause white spots in the image which do not register as the appropriate color from training. This creates new edges which cause increased computation. In the dark regions, less of the ball is seen, possibly creating more edges, and more importantly, the center of area is shifted from the true center of the ball. The catching would greatly benefit from more uniform lighting throughout the room.

5.3.3 Additional Sources

There additional possible sources of error which have less noticeable effects. Occasionally it has been found that catching has a better success rate in the evenings than in the mornings. A possible cause is the ambient lighting conditions resulting from the sunlight through the windows. It is also found that performance degrades if the system is run for a very long duration (more than $\approx$ 5 hours). This could possible be a result of temperature effects. In addition, the torque ripple from the PWMs powering the motors for the WAM are occasionally quite noticeable, affecting the tracking performance of the WAM. And finally, the noise in the system might be slightly reduced if the cameras were auto-focus.

Chapter 6

Conclusion and Recommendations

6.1 Summary

This thesis has discussed the experimental investigation of the capabilities of the new active vision system and the robot manipulator for improved robotic catching in unstructured environments. Control and use of the new vision system has been presented. New algorithms for object tracking have been introduced. And structured methods for catch point determination and path generation have been presented.

The incorporation of a new active vision system and the coordination of vision and manipulation has involved the following.

1. The implementation of various controllers for the active vision system, including a Model Reference Adaptive Controller [Slotine, Li 91], has been presented. Performance results for the PD and the Adaptive Controllers, showing the increased performance of the Adaptive Controller for trajectory tracking were shown.

2. A method for determination of object location using stereo vision information was presented.

3. A method for visually tracking moving objects was presented which is capable of tracking objects moving up to approximately 4 m/s at a distance of 1 m from the cameras (8 m/s at 2 m, etc.).

4. A simplified cross calibration method for coordinating the vision system and the manipulator was presented. The resulting calibration had an accuracy of approximately $\frac{1}{2}$ an inch over the full workspace of the manipulator.

Using the new active vision system and the cross calibration, successful results for robotic catching were presented. The elements involved in catching were as follows.

1. The implementation of a recursive least squares fitting of ball data to parabolic paths for toss path prediction was presented.

2. A structured method for utilizing toss path prediction for determining safe catch points was presented.

3. The replacement of the second order filter path generation scheme with a third order polynomial path generation scheme for the WAM was presented.

4. Precise timing for closure and methods for orientation of the end effector were implemented for increased catching success.

Utilizing all of the techniques listed above, this system has achieved successful catching of free-flying spherical balls. The percentage of successful catches for the *best* performance was found to be approximately 70-80 %. It was found that the Adaptive Controllers, which yielded small tracking error, were vital to the success of the catching algorithms.

6.2 Recommendations

This section presents ideas for improvement of the current system. The two most vital areas for improvement are presented first, cross calibration and object tracking. Then, additional improvements for WAM path generation during catching are presented. And finally, additional system improvements are discussed.

6.2.1 WAM/FEG Calibration Methods

In the current calibration method, the problem was simplified by using physical constraints based upon the mounting of the FEGs relative to the WAM. This method relies upon

the accuracy of the mounting. Ideally, a full calibration method should be developed which would account for any inaccuracies, or more generally, for any placement of cameras relative to the WAM.

The process by which data is collected for the cross calibration should be studied in detail and different formulations of the calibration problem should be examined. The logistics of data collection basically should address the problem of how to get accurate position information from both the FEGs and WAM for the same point in space. And, two possible formulations of the cross calibration problem are: (1) the FEGs could define their own cartesian coordinate frame and then the full rotation and translation vector between the WAM and FEG frames could be determined, and (2) a transformation which directly maps the four FEG joint angles to WAM cartesian coordinates could be determined.

Logistics of Data Collection

The method we have used here places the ball at the end of the WAM. After measuring the length to the center of the ball along the last link of the WAM, the forward kinematics can be used to determine the position of the ball as measured by the WAM. At the same time, the cameras may track the ball and thereby have joint or cartesian position information. This method is a bit imprecise, since the ball is large and in certain orientations the WAM itself partially obstructs the ball from the view of the cameras. Thus, it is difficult to mark the WAM in a precise way such that the FEGs can pinpoint a location on the WAM accurately. Possibly the use of a smaller fluorescent orange spherical object placed far enough along the last link to avoid obstruction may yield more accurate results.

Independent FEG Coordinate Frame

In our current method, the FEGs have their own cartesian coordinate frame which is related to the WAM frame through a rotation and translation. There are actually two problems involved here, one is to obtain an accurate coordinate frame for the FEGs, and the second is to determine the transformation from the FEG frame to the WAM frame. The problem of obtaining an accurate coordinate frame for the FEGs is discussed later in Section 6.2.4. The second problem of determining the cross calibration is discussed here.

As discussed in Section 3.3, the problem of cross calibration has been simplified to a one DOF problem. This accuracy of this method depends on the accuracy of the mounting. Since it is known that there are small inaccuracies in mounting, it would be better to solve for the full rotation matrix instead of the simplified one. Therefore, additional calibration methods should be implemented which calibrate for the full rotation and translation. These methods would involve the solution of a nonlinear system of equations. The use of quaternions to represent rotations may be beneficial in obtaining an easier solution to the transformation problem. Rotation matrices would have nine elements (which are not independent), where quaternions will only have four. The simplified method we have been using here was a first iteration. The full transformation calibration should be derived and implemented for greater accuracy.

Direct Mapping from FEG Joint Space to WAM Cartesian Space

There are also other methods which could be pursued which do not define an independent coordinate space for the FEGs, but rather convert FEG joint angles directly to WAM coordinates. This would have the added benefit of allowing incorporation of information from each FEG individually as it arrives. This would eliminate the necessity for some time stamping and compensation which is currently done, reducing the noise. And finally, this method would allow for the placement of each camera arbitrarily relative to the WAM and the other camera. The calibration should then be able to determine the location of each camera relative to the WAM. This problem is more difficult than the transformation problem of the previous section, but would possibly yield better calibration.

6.2.2 Improving Visual Tracking

Currently the range of speed of tosses which the system can catch is limited by the vision system and the visual tracking. In order to be able to catch faster tossed balls, both the vision hardware and the tracking algorithm require improvement. Three approaches for improving the tracking capabilities of the vision system are: use custom built non-NTSC cameras or use smaller focal length lenses (wider field of view), improve the vision boards to speed up the processing and make the latency a constant value, and use statistical

measures and tracking and optimal estimation theory to improve the tracking algorithm used to predict the path of the object.

Custom Cameras and Different Lenses

All NTSC cameras output interlaced frames at 60 Hz. In order to obtain faster output, a custom camera must be built. Along with a custom camera, an associated vision processing system must also be built. For the moment, this is the most difficult and expensive approach and other methods should be examined first.

Using rough calculations, with the current focal length lens, an object moving at 7 m/s at a distance of one meter would move out of the field of view before the next camera frame is taken. By using a smaller focal length lens, providing a wider field of view, this maximum velocity can be increased. This by far is the simplest method for improvement.

Vision Processing Hardware

A major source of error in the current system involves the integration over the latency in the vision system. The integration over the latency time results in amplification of errors in estimates of velocity and acceleration. In addition, the latency time itself is variable, depending upon the number of edges and "white" pixels in the image.

Therefore, the latency should be decreased and made invariant to image content. In order to decrease the latency, a change of hardware is required. The current vision system uses 68332 processors for computation. The more powerful C40 DSP processors could be used for vision processing which would decrease the processing time for each image. In order to make the latency invariant, the largest latency time could be found and the board could be programmed to wait for this length of time for every frame. Another option would be to specify time-stamps for the start and end of processing for each image so that the exact latency time would be known.

There is also the possibility of completely altering our approach to the vision problem. Up to this point, we have used simplified vision using color-keying for speed and simplicity. If more powerful vision processing hardware were obtained, such as DSP based vision boards, then more "traditional" vision algorithms can be applied which would greatly ex-

pand the information we may extract from the vision system. We could develop motion detection/tracking systems, or study image segmentation and recognition. Although this may not improve the speed of object tracking, it would expand the class of objects which may be tracked and possibly provide more information about the object being tracked.

Algorithm for Object Tracking

The method used for tracking fast moving objects could be improved. Other tracking methods have been discussed by other researchers [Bar-Shalom, Fortmann 88, Bar-Shalom, Li 93, Fiala et al 94, Kalata 84]. The existing body of work should be built upon and applied to our task for improved, theoretically sound tracking. In general, the tracking would benefit from a better formulation and understanding of the problem. The signal from the vision boards should be analyzed in more detail. The frequency of the noise should be determined and filtered. More precise models of the behaviors of the object being tracked should be formulated.

6.2.3 Catching Path Generation Possibilities

The WAM can either be given paths in cartesian space or joint space, with associated advantages and disadvantages to each. In both representations, different path generation schemes could be used.

Cartesian Space Path Generation

Additional methods besides third order polynomials for path generation are possible which may better utilize the capabilities of the arm. The third order polynomial often approaches the outer radius workspace limitation as it swings around to approach the catch point with the correct desired velocity. Also, since third order polynomials are being used, accelerations are not matched, therefore there are discontinuities in acceleration at the beginning and at the end of the path. Additional methods which are being considered are the use of chained splines or the combining of sections with predetermined acceleration profiles. These methods may provide a means for achieving smooth acceleration profiles while remaining within the workspace limitations.

Joint Space Path Generation

There is also the possibility of conducting catching in joint space rather than cartesian space. Currently the catching is done in cartesian space, with the desired path for the WAM being specified by cartesian position, velocity, and acceleration values. These values are then converted within the controller using path dependent inverse kinematic methods [Niemeyer 90]. As another possible option, the catching may be conducted by specifying desired joint position, velocities, and accelerations. The advantages of using joint space are: there are no singularities in joint space; you have better control over the desired torque on each of the individual motors; and this allows for better utilization of the acceleration capabilities of each of the joints for catching and deceleration. The disadvantages of using joint space are: the path of the endpoint is not controlled explicitly, so safety constraints to prevent the arm from impacting itself or the floor are more difficult to implement; the position and velocity at the catch point need to be computed through the use of inverse kinematics; and the matching of trajectories of the arm with the ball path for a length of time for grasping becomes more difficult.

6.2.4 General System Improvements

There are a number of items relating to hardware, software, and environment which may be changed to increase performance. The Talon may be strengthened and improved. The code may be revised to consider the WAM and Talon as a single manipulator, rather than two separate ones. The accuracy of resulting coordinates from the FEGs could be improved by more precise mounting. And lastly, the lighting could be better controlled.

The Talon

The Talon is currently speed limited due to the large gear reduction in the fingers. By using stronger motors with smaller gear reductions, we can reduce the time required for the fingers to close as well as getting firmer grasps on objects.

In addition to time and strength considerations, the design of the Talon could be modified. Occasionally during catching, the object impacts the fingers or end of the Talon, bouncing out of reach. The end of the Talon could be padded to create a softer surface to

impact upon. Lastly, one more degree of freedom in the Talon would allow for orientation to any direction in space. This would allow the for the use of a "palm" in catching as well as catching rotating cylinders.

Seven DOF WAM/Talon

Currently, the kinematics for the WAM and Talon are handled separately. In software, the WAM and the Talon are treated independently. In the future, the software should do the full seven DOF kinematics. This would ease orientation of the Talon with respect to the world. It would also provide more flexibility in control method and ease planning of currently difficult motions.

FEG Mounting Methods

The current vision system is accurate to approximately 0.5 inches. Although this value is large, the effect is greatly reduced during catching due to the least squares fit over 20-40 points. Nevertheless, the system would greatly benefit from greater accuracy.

More rigorous calibration and mounting is required to improve the accuracy of the vision system. Internal calibration for the cameras parameters should be done and mounting misalignments with the center of rotation of the FEG should be determined. The exact location of the focal point of the cameras should be determined and then the offsets from the center of rotation of the FEGs should be calculated. These may be used with the methods in Appendix A.4 to obtain more accurate coordinates. In addition, the FEGs may be more precisely mounted relative to one another, possibly by mounting to a single base. Currently each FEG is mounted to a ceiling rafter by means of bolts through holes which have some clearance. By mounting both FEGs to a single piece with threaded holes, the relative position of one FEG to the other can be known to machining precision. Once the mounting is completed, a more complete calibration should be implemented which accounts for any remaining mounting inaccuracies.

Shortening the baseline between the two FEGs might also be a consideration. This would make accurate mounting easier, make the system more portable, and would reduce the effects of vertical misalignments and horizontal misalignments in the direction perpendicular

to the baseline. But by reducing the baseline, the effects of errors in pan angles are increased. For instance, if the object were located by one FEG at a pan angle of $\pi/4$, for a pan angle error of 0.01 radians, decreasing the baseline by half would increase the position error by approximately 25%.

Lighting

Currently, the lighting in the room is very concentrated under the light fixtures placed in the room. Creating uniform lighting about the workspace would serve to reduce the noise in the vision data by stabilizing the binary image seen by the vision boards as the object moves about the room.

6.3 Conclusion and Future Work

This thesis has presented the methods and results for our experiments in 3-D robotic catching of free-flying objects. Sources of error and recommendations for improvements to the system have been discussed.

Current research is being done to examine catching of objects with different aerodynamic characteristics. Objects of interest are light-weight foam balls, paper airplanes, and other items with non-parabolic trajectories. The current least squares approach to path prediction for the tossed object is replaced by a neural network based prediction method based upon [Cannon, Slotine 95]. Interfacing this with the catching algorithms documented here have yielded good preliminary results.

Future work will try to incorporate learning in tracking, prediction, and catching. The system may also be applied to additional dynamic tasks such as playing baseball or multiple object juggling.

Appendix A

Fast Eye Gimbal Appendices

A.1 Derivation of the Dynamic Equations for the FEGs

The Recursive Newton Euler Method originally developed by Luh, Walker, and Paul was first used to derive the dynamical equations of motion. Then the Lagrange Method was used as a check to verify the dynamics. Here we briefly summarize the fundamental equations for both methods. For more information on both methods, see [Asada, Slotine 86].

A.1.1 Recursive Newton Euler

First we apply kinematic equations to each link

$$\begin{aligned} \omega_{i+1} &= \omega_i + \dot{q}_{i+1} b_i \\ \dot{\omega}_{i+1} &= \dot{\omega}_i + \ddot{q}_i b_i + \omega_i \times \dot{q}_i b_i \\ v_{i+1} &= v_i + \omega_{i+1} \times r_{i,i+1} \\ a_{i+1} &= a_i + \dot{\omega}_{i+1} \times r_{i,i+1} + \omega_{i+1} \times (\omega_{i+1} \times r_{i,i+1}) \end{aligned} \tag{A.1}$$

where all the terms are vectors except for the q_i terms which are scalar, and b_i is the direction of the joint axis. Next we use these equations to find the motion of the center of mass for each link

$$\begin{aligned} v_{ci} &= v_i + \omega_i \times r_{i,ci} \\ a_{ci} &= a_i + \dot{\omega}_i \times r_{i,ci} + \omega_i \times (\omega_i \times r_{i,ci}) \end{aligned} \tag{A.2}$$

Now, we use dynamic equations to find the joint torques

$$\begin{aligned} f_{i-1,i} &= f_{i,i+1} - m_i g + m_i a_{ci} \\ N_{i-1,i} &= N_{i,i+1} - r_{i,ci} \times f_{i,i+1} + r_{i-1,ci} \times f_{i-1,i} + I_i \dot{\omega}_i + \omega_i \times (I_i \omega_i) \end{aligned} \tag{A.3}$$

We are only interested in the component of the torque in the direction of the joint axis, so

$$\tau_i = N_{i-1,i} \cdot b_i \tag{A.4}$$

Through all these equations, it is important to keep track of which coordinate system you are using. In order to use a configuration independent inertia matrix, I_i, we need to use link coordinates. Thus in the above equations, when a variable with respect to frame i is involved in an equation with respect to frame $i+1$, it should first be pre-multiplied by the rotation matrix, R_i^{i+1}.

A.1.2 Lagrange's Equations of Motion

The primary equation for the Lagrangian Formulation is

$$\tau_i = \sum_{j=1}^{n} H_{ij} \ddot{q}_j + \sum_{j=1}^{n} \sum_{k=1}^{n} h_{ijk} \dot{q}_j \dot{q}_k + G_i \qquad i = 1, 2 \tag{A.5}$$

First let us show how the expression for the H matrix is found. Given the following relations which define $J_L^{(i)}$ and $J_A^{(i)}$

$$\begin{aligned} v_{ci} &= J_L^{(i)} \dot{q} \\ \omega_i &= J_A^{(i)} \dot{q} \end{aligned} \tag{A.6}$$

and also using the configuration invariant inertia tensor $\bar{I}_i$ to get the inertia tensor in the base frame

$$I_i = R_i^0 \bar{I}_i (R_i^0)^T \tag{A.7}$$

we get an expression for the H matrix of the form

$$H = \sum_{i=1}^{n} (m_i J_L^{(i)^T} J_L^{(i)} + J_A^{(i)^T} I_i J_A^{(i)}) \tag{A.8}$$

To see a more thorough investigation of this equation, see [Asada, Slotine 86].

Next, we find expressions for the other terms in equation A.5.

$$h_{ijk} = \frac{\partial H_{ij}}{\partial q_k} - \frac{1}{2}\frac{\partial H_{jk}}{\partial q_i} \tag{A.9}$$

and

$$G_i = \sum_{j=1}^{n} m_j g^T J_{Li}^{(j)} \tag{A.10}$$

where $J_{Li}^{(j)}$ is the ith column vector of $J_L^{(j)}$.

Thus from equation A.5 we find expressions for the joint torques.

A.2 Adaptive Control

A.2.1 Lyapunov Proof of Stability

The following is a proof for the case with separate K_D and K_P gains as in equation 3.16. The proof for K_D and $\mathbf{s}$ is very similar and simpler.

First, let us restate the dynamics here

$$H\ddot{\mathbf{q}} + C\dot{\mathbf{q}} + G = \tau \tag{A.11}$$

where τ is our control input defined by the equation

$$\tau = Y\hat{\mathbf{a}} - K_D\dot{\tilde{\mathbf{q}}} - K_P\tilde{\mathbf{q}} \tag{A.12}$$

and Y $\mathbf{a}$ is defined by the following

$$H\ddot{\mathbf{q}}_\mathbf{r} + C\dot{\mathbf{q}}_\mathbf{r} + G = Y\mathbf{a} \tag{A.13}$$

Now we select a Lyapunov function of the form

$$V = \frac{1}{2}\mathbf{s}^T H\mathbf{s} + \frac{1}{2}\tilde{\mathbf{a}}^T\Gamma^{-1}\tilde{\mathbf{a}} + \frac{1}{2}\tilde{\mathbf{q}}^T(K_P + K_D\lambda)\tilde{\mathbf{q}} \geq 0 \tag{A.14}$$

Note that V is positive definite since H, Γ, K_D, and K_P are all symmetric positive definite

matrices and λ is positive.

We take the first derivative of V and we have

$$\dot{V} = \mathbf{s}^T H \dot{\mathbf{s}} + \frac{1}{2}\mathbf{s}^T \dot{H}\mathbf{s} + \dot{\hat{\mathbf{a}}}^T \Gamma^{-1}\tilde{\mathbf{a}} + \dot{\tilde{\mathbf{q}}}(K_P + K_D\lambda)\tilde{\mathbf{q}} \tag{A.15}$$

substituting, we get

$$\dot{V} = \mathbf{s}^T(\tau - G - C\dot{\mathbf{q}} - H\ddot{\mathbf{q}}_\mathbf{r} + \frac{1}{2}\dot{H}\mathbf{s}) + \dot{\hat{\mathbf{a}}}^T \Gamma^{-1}\tilde{\mathbf{a}} + \dot{\tilde{\mathbf{q}}}(K_P + K_D\lambda)\tilde{\mathbf{q}} \tag{A.16}$$

We know that $\dot{H} - 2C$ is skew-symmetric based upon energy conservation. Thus removing these terms and substituting τ we get

$$\dot{V} = \mathbf{s}^T(Y\tilde{\mathbf{a}} - K_D\dot{\tilde{\mathbf{q}}} - K_P\tilde{\mathbf{q}}) + \dot{\hat{\mathbf{a}}}^T \Gamma^{-1}\tilde{\mathbf{a}} + \dot{\tilde{\mathbf{q}}}(K_P + K_D\lambda)\tilde{\mathbf{q}} \tag{A.17}$$

Finally, selecting the adaptation law $\dot{a} = -\Gamma Y^T\mathbf{s}$ we get

$$\dot{V} = -\dot{\tilde{\mathbf{q}}}K_D\dot{\tilde{\mathbf{q}}} - \lambda\tilde{\mathbf{q}}^T K_P\tilde{\mathbf{q}} \leq 0 \tag{A.18}$$

where K_D and K_P are symmetric positive definite.

Now, we note that V is lower bounded and $\dot{V}$ is negative semi-definite. From $\dot{V} \leq 0$ we also see that $\mathbf{s}$, $\tilde{\mathbf{a}}$, $\tilde{\mathbf{q}}$, and $\dot{\tilde{\mathbf{q}}}$ are all bounded if $\mathbf{q_d}$, $\dot{\mathbf{q}}_\mathbf{d}$, and $\ddot{\mathbf{q}}_\mathbf{d}\mathbf{d}$ are all bounded. Therefore τ is bounded and thus $\ddot{\mathbf{q}}$ is bounded.

We finally need to look at $\ddot{V}$ and check if it is bounded.

$$\ddot{V} = -2\ddot{\tilde{\mathbf{q}}}^T K_D\dot{\tilde{\mathbf{q}}} - 2\lambda\dot{\tilde{\mathbf{q}}}^T K_P\tilde{\mathbf{q}} \tag{A.19}$$

Thus we see that $\ddot{V}$ is bounded as well. Using Barbalat's Lemma [Slotine, Li 91], we conclude that $\dot{V}$ converges to zero. This then implies that $\tilde{\mathbf{q}}$ and $\dot{\tilde{\mathbf{q}}}$ also converge to zero. In other words, the tracking error ideally converges to zero using this control law.

A.2.2 Implementation Issues

In order to implement the Adaptive Controller, we need to specify the values for the bandwidth, λ, and also for the adaptation gain matrix, Γ.

The bandwidth should be tuned so that higher frequency elements are not considered by the controller. Some basic guides for selecting the bandwidth from [Slotine, Li 91] are

$$\begin{aligned} \lambda \leq \lambda_R \approx \tfrac{2\pi}{3}\nu_R \\ \lambda \leq \lambda_A \approx \tfrac{1}{3T_A} \\ \lambda \leq \lambda_S \approx \tfrac{1}{5}\nu_{sampling} \end{aligned} \tag{A.20}$$

where λ is selected to be the minimum of the three bounds above. These bounds relate to unmodelled structural modes, time delays from actuator dynamics or other delays, and sampling rate limitations. Ideally we would like to have $\lambda \approx \lambda_R \approx \lambda_A \approx \lambda_S$, thereby limiting wasted resources. The slowest structural resonant mode of the system is a physical characteristic of the plant and is a "hard" limitation, where the other two can be altered by changing various portions of our experimental apparatus.

In selecting Γ, we desire each of the parameters to converge at approximately the same rate. Thus from equation 3.12 we see that the convergence is related to the Y matrix. From [Niemeyer, Slotine 88], we find a method for selecting initial base values for the Γ matrix from which we can tune for better performance.

$$\Gamma \propto diag\left(\int_0^T Y^T Y dt\right)^{-1} \tag{A.21}$$

where Y is defined as before from equation 3.14. This relation can be seen as related to least-squares identification rules. It was found that some additional tuning from these base values was required to achieve better performance.

A.3 Velocity and Acceleration Transformations

The transformation from joint space velocity to cartesian velocity can be obtained by differentiating the x, y, and z equations, which lead to

$$\dot{x} = \frac{D}{\theta_2 - \theta_4}(-\sin(2\theta_2)\,\dot{\theta}_4 + \sin(2\theta_4)\,\dot{\theta}_2) \tag{A.22}$$

$$\dot{y} = \frac{2D\cos\theta_1}{\sin^2(\theta_2 - \theta_4)}(-\cos^2\theta_4\,\dot{\theta}_2 + \cos^2\theta_2\,\dot{\theta}_4) - \frac{2D\sin\theta_1\cos\theta_2\cos\theta_4}{\sin(\theta_2 - \theta_4)} \tag{A.23}$$

$$\dot{z} = \frac{2D\sin\theta_1}{\sin^2(\theta_2-\theta_4)}(-\cos^2\theta_4\,\dot{\theta}_2+\cos^2\theta_2\,\dot{\theta}_4)+\frac{2D\cos\theta_1\cos\theta_2\cos\theta_4}{\sin(\theta_2-\theta_4)} \quad (A.24)$$

The transformation from joint space acceleration is not required since the only times we will be transforming acceleration values is from cartesian space into joint space for specifying desired accelerations.

In cartesian space, the velocity transformation into joint space is given by the following equations, again obtained through differentiation.

$$\dot{\theta}_1 = \dot{\theta}_3 = \frac{y\,\dot{z}-z\,\dot{y}}{y^2+z^2} \quad (A.25)$$

$$\dot{\theta}_2 = \frac{(x-D)(y\,\dot{y}+z\,\dot{z})-(y^2+z^2)\,\dot{x}}{\sqrt{y^2+z^2}((x-D)^2+y^2+z^2)} \quad (A.26)$$

$$\dot{\theta}_4 = \frac{(x+D)(y\,\dot{y}+z\,\dot{z})-(y^2+z^2)\,\dot{x}}{\sqrt{y^2+z^2}((x+D)^2+y^2+z^2)} \quad (A.27)$$

The acceleration conversion is a bit more lengthy, and results in the following

$$\ddot{\theta}_1 = \ddot{\theta}_3 = \dot{y}\left(-\frac{z\,\ddot{y}-\dot{z}}{y^2+z^2}+\frac{2y(z\,\dot{y}-y\,\dot{z})}{(y^2+z^2)^2}\right)+\dot{z}\left(\frac{y\,\ddot{z}-\dot{y}}{y^2+z^2}+\frac{2z(z\,\dot{y}-y\,\dot{z})}{(y^2+z^2)^2}\right) \quad (A.28)$$

$$\begin{aligned}\ddot{\theta}_2 = {} & \frac{1}{\sqrt{y^2+z^2}((1))}\Bigg(\\ & \dot{x}\left((y\dot{y}+z\dot{z}-(y^2+z^2)\ddot{x})-2(x-D)(\frac{(x-D)(y\dot{y}+z\dot{z})-(y^2+z^2)\dot{x})}{((1))}\right) \\ & +\dot{y}\left((x-D)(y\ddot{y}+\dot{y})-2y\dot{x}+2y\frac{((2))}{((1))}+y\frac{((2))}{(y^2+z^2)((1))}\right) \\ & +\dot{z}\left((x-D)(z\ddot{z}+\dot{z})-2z\dot{x}+2z\frac{((2))}{((1))}+z\frac{((2))}{(y^2+z^2)((1))}\right)\Bigg)\end{aligned} \quad (A.29)$$

where $((1)) = (x-D)^2+y^2+z^2$ and $((2)) = (y^2+z^2)\dot{x}-(x-D)(y\dot{y}+z\dot{z})$. The expression for $\ddot{\theta}_4$ is exactly the same as $\ddot{\theta}_2$ with $(x+D)$ instead of $(x-D)$.

A.4 Compensation for Focal Point Location

In this section, we present a process through which we can compensate for misalignment of the focal point and the center of rotation of the FEG. This section assumes you know the location of the focal point of the camera which can be done through camera calibration

methods.

We begin by giving a summary of the procedure. First, we determine the location of each of the focal points in the FEG cartesian coordinate frame. We also determine the orientation of the vector along each of the cameras. Using the location of each focal point and the orientation of each FEG, we determine new tilt and pan angles in a rotated frame. With these new angle values, we can use the same triangulation procedure as discussed in Section 3.2.2 to obtain cartesian coordinates in the rotated frame. We can then rotate and translate these coordinates back to the original FEG cartesian coordinate frame to get our resulting corrected coordinates.

Next, we give a more detailed explanation of the procedure.

First, we find the location of the focal points in the base coordinate frame for each of the FEGs.

$$\mathbf{p_1} = \mathbf{R}(-\theta_1, -\frac{\pi}{2})\,\mathbf{R}(\theta_2, 0)\,\mathbf{d_1} \tag{A.30}$$

$$\mathbf{p_2} = \mathbf{R}(\theta_3, -\frac{\pi}{2})\,\mathbf{R}(\theta_4, 0)\,\mathbf{d_2} \tag{A.31}$$

In the equations above, $\mathbf{p_1}$ and $\mathbf{p_2}$ are the locations of the focal points and $\mathbf{d_1}$ and $\mathbf{d_2}$ are the mounting offsets from the center of rotation[1] for FEG 1 and 2 respectively. The θ terms are the joint values for the FEGs, as defined in Figure 3-7. The rotation matrices $\mathbf{R}(\theta, \alpha)$ are defined as

$$\mathbf{R}(\theta, \alpha) = \begin{bmatrix} \cos\theta & -\sin\theta\cos\alpha & \sin\theta\sin\alpha \\ \sin\theta & \cos\theta\cos\alpha & -\cos\theta\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{A.32}$$

Next, we transform these focal point locations from each FEG base coordinate frame to the FEG cartesian coordinate frame located between the two FEGs (as shown in Figure

[1]These are 3x1 vectors which are the distances to the focal points of each camera from the center of rotation of the FEGs. The first element is the distance along the axis of the camera, the second is the offset from the pan axis of rotation, and the third is the offset from the tilt axis of rotation.

3-7).

$$\mathbf{c_1} = \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{p_1} + \begin{bmatrix} D \\ 0 \\ 0 \end{bmatrix} \tag{A.33}$$

$$\mathbf{c_2} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{p_2} + \begin{bmatrix} -D \\ 0 \\ 0 \end{bmatrix} \tag{A.34}$$

where D is defined as before as the distance from each FEG to the origin of the coordinate frame.

In addition to the location of each of the focal points, we also need to determine the vector along which the camera is currently oriented. These directions are then

$$\mathbf{q_1} = \mathbf{R}(0, \theta_1)\, \mathbf{R}(\theta_2, 0)\, \hat{\mathbf{y}} \tag{A.35}$$

$$\mathbf{q_2} = \mathbf{R}(0, \theta_3)\, \mathbf{R}(\theta_4, 0)\, \hat{\mathbf{y}} \tag{A.36}$$

where $\hat{\mathbf{y}} = (0\ 1\ 0)^T$.

Now, we define a rotated coordinate frame based upon the line connecting the locations of the two focal points. The location of the object will then first be calculated in this rotated frame. The resulting coordinates will then be transformed to the FEG cartesian coordinate frame. The rotation matrix and translation vector are required to do this transformation. We define the x axis for the rotated frame to be along the line connecting the two focal points. Then we find a rotation matrix which maps the original x axis from the FEG cartesian coordinate frame to this rotated frame. The rotation matrix is given by

$$\mathbf{R}_{rot}^{feg} = \mathbf{R}(\alpha, -\frac{\pi}{2})\, \mathbf{R}(\beta, 0)\, \mathbf{R}(0, \frac{\pi}{2}) \tag{A.37}$$

where $\alpha = \tan^{-1}((c_{y2} - c_{y1})/(c_{x2} - c_{x1}))$ and $\beta = \tan^{-1}((c_{z2} - c_{z1})/(c_{x2} - c_{x1}))$.
The translation between the FEG cartesian coordinate frame and the rotated frame is given by

$$\mathbf{t}_{rot}^{feg} = \frac{1}{2}(\mathbf{c_1} + \mathbf{c_2}) \tag{A.38}$$

Now that we have the rotation between the coordinate frames, we can represent the orientation of each camera in the rotated frame.

$$\mathbf{r_1} = \mathbf{R}_{rot}^{feg}\,\mathbf{c_1} \tag{A.39}$$

$$\mathbf{r_2} = \mathbf{R}_{rot}^{feg}\,\mathbf{c_2} \tag{A.40}$$

Using these orientations in the new rotated frame, we can find the tilt and pan angles which are equivalent to these directions. The tilt angles would be given by $\tan^{-1}(z/y)$ and the pan angles would be given by $\pm\tan^{-1}((y^2+z^2)/(\sqrt{y^2+z^2}))$.

With these new tilt and pan angles in the rotated frame, we can use the triangulation method given in Section 3.2.2 to obtain cartesian coordinates for the object. The resulting coordinates would be in the rotated frame. We then rotate and translate these back to the FEG cartesian coordinate frame to yield the resulting corrected coordinates.

$$\mathbf{x}_{corrected} = \mathbf{R}_{rot}^{feg^T}\,\mathbf{x}_{rotated} + \mathbf{t}_{rot}^{feg} \tag{A.41}$$

where $\mathbf{x}_{rotated}$ are the coordinates found from the triangulation method applied in the rotated frame.

Appendix B

Catching Appendices

B.1 Recursive Least Squares Parabolic Fit

The Recursive Least Squares Parabolic Fit was originally documented in [Kimura 92]. We summarize the method here.

We require a minimum of four data points in order to do the full least squares parabolic fit. With two and three data points, we assume an x and y acceleration of zero and use linear fits and for z, we calculate the assume gravitational acceleration for two data points and we calculate the acceleration directly for three data points. For four or more data points, we compute the full recursive least squares fit.

Let the parabolic constants be defined in matrix form as

$$\mathbf{C}_{prb} = \begin{bmatrix} a_x & v_x & p_x \\ a_y & v_y & p_y \\ a_z & v_z & p_z \end{bmatrix} \tag{B.1}$$

The cost function can then be defined as

$$\begin{aligned} \mathbf{J} &= \sum_{t=1}^{N} \mathbf{X}_e[t_i]^2 \\ &= \sum_{i=1}^{N} (\mathbf{X}[t_i] - \mathbf{C}_{prb} \bullet \mathbf{T}[t_i])^T (\mathbf{X}[t_i] - \mathbf{C}_{prb} \bullet \mathbf{T}[t_i]) \end{aligned} \tag{B.2}$$

where $\mathbf{X}_e[t_i]$ is the estimation error at time t_i, N is the current number of samples and $\mathbf{T}[t_i] = (\frac{1}{2}t_i^2\ t_i\ 1)^T$. We then minimize $\mathbf{J}$ with respect to $\mathbf{C}_{prb}$ with the following results for the X axis parabolic constants.

$$a_x = \frac{2(D_{4x} \cdot D_5 - D_{1x} \cdot D_2)}{(D_6 \cdot D_5 - D_3 \cdot D_2)} \tag{B.3}$$

$$v_x = \frac{1}{D_5}(D_{1x} - \frac{1}{2}a_x D_3) \tag{B.4}$$

$$p_x = \frac{1}{N}(\sum_{i=1}^{N} x[t_i] - v_x \sum_{i=1}^{N} t_i - \frac{1}{2}a_x \sum_{i=1}^{N} t_i^2) \tag{B.5}$$

where

$$D_{1x} = N\sum_{i=1}^{N}(x[t_i] \cdot t_i) - \sum_{i=1}^{N} x[t_i] \sum_{i=1}^{N} t_i \tag{B.6}$$

$$D_2 = \sum_{i=1}^{N} t_i^3 \sum_{i=1}^{N} t_i - (\sum_{i=1}^{N} t_i^2)^2 \tag{B.7}$$

$$D_3 = = N\sum_{i=1}^{N} t_i^3 - \sum_{i=1}^{N} t_i^2 \sum_{i=1}^{N} t_i \tag{B.8}$$

$$D_{4x} = \sum_{i=1}^{N}(x[t_i] \cdot t_i^2) \sum_{i=1}^{N} t_i - \sum_{i=1}^{N}(x[t_i] \cdot t_i) \sum_{i=1}^{N} t_i^2 \tag{B.9}$$

$$D_5 = N\sum_{i=1}^{N} t_i^2 - (\sum_{i=1}^{N} t_i)^2 \tag{B.10}$$

$$D_6 = \sum_{i=1}^{N} t_i^4 \sum_{i=1}^{N} t_i - \sum_{i=1}^{N} t_i^3 \sum_{i=1}^{N} t_i^2 \tag{B.11}$$

The y and z directions can be calculated similarly by computing D_{1y}, D_{1z}, D_{4y}, and D_{4z}. These computations can also be done recursively in the sense that the whole summations are not required to be computed each time around. The summations may be maintained from the last loop with the new data being added to the summation. The result is a process which does not require more computation as time progresses, each servo loop, the same amount of computation is required.

B.2 WAM Forward Kinematics for Talon Orientation

The transformation from the world frame at the base of the WAM to the coordinate frame at the last link of the WAM can be calculated using Denavit Hartenberg Notation. Table B.1 show a list of the constants for the four joints of the WAM.

Joint	θ	d	a	α
1	θ_1	0	0	$\frac{\pi}{2}$
2	θ_2	0	0	$-\frac{\pi}{2}$
3	θ_3	h_3	a_3	$\frac{\pi}{2}$
4	θ_4	0	a_{17}	$-\frac{\pi}{2}$

Table B.1: Forward kinematics for the WAM expressed in Denavit Hartenberg constants.

The constants in the table are $h_3 = 0.5588$ meters, $a_3 = 0.04064$ meters, and $a_{17} =$ -0.02794 meters. The actual endpoint of the WAM is located $h_{17} = 0.46196$ meters along the z axis of the last coordinate frame.

Using the θ and α values from the table, we can take the velocity vector for the ball at the catch point and transform it to the endpoint coordinate frame for the WAM. Then using only the x and y components, we can determine the joint value for the forearm joint of the Talon to correctly orient the fingers perpendicular to the path of the ball. The x and y components can then be found by the following equations.

$$R_1^4 = \begin{bmatrix} (c1c2c3 - s1s3)c4 - c1s2s4 & (c1s3 + s1c2c3)c4 - s1s2s4 & c2s4 + s2c3c4 \\ c1c2s3 + s1c3 & c1c3 - s1c2s3 & -s2s3 \end{bmatrix} \tag{B.12}$$

$$\begin{bmatrix} v_{x4} \\ v_{y4} \end{bmatrix} = R_1^4 \begin{bmatrix} v_{x1} \\ v_{y1} \end{bmatrix} \tag{B.13}$$

Then the joint angle for the Talon forearm is then given by the following.

$$\theta_f = \tan^{-1}(\frac{v_{y4}}{v_{x4}}) \tag{B.14}$$

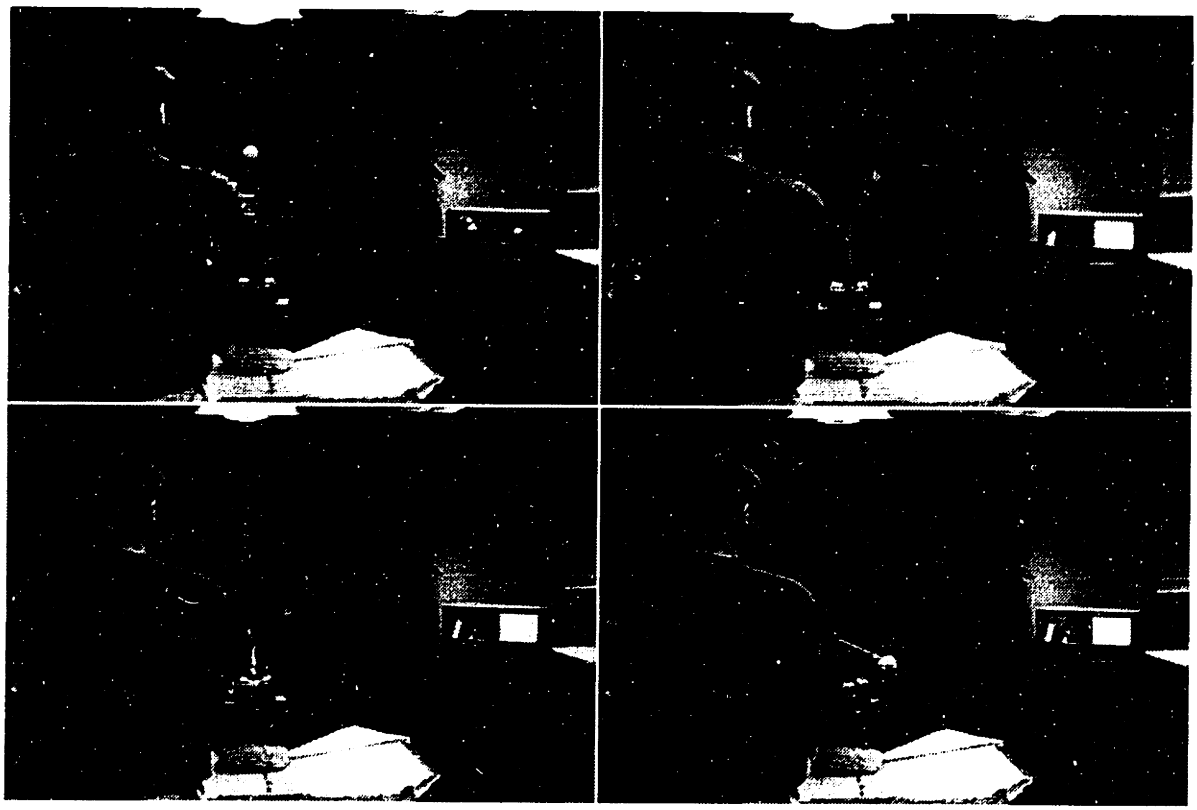


Figure B-1: A sequence of images of a catching attempt with the toss originating from the left side (right to left, top to bottom).

B.3 Photographs of Catching Attempts

Here we present a sequence of images for two catching attempts. In the first, the toss originates from the right side. In the second, the toss originates from the left side.

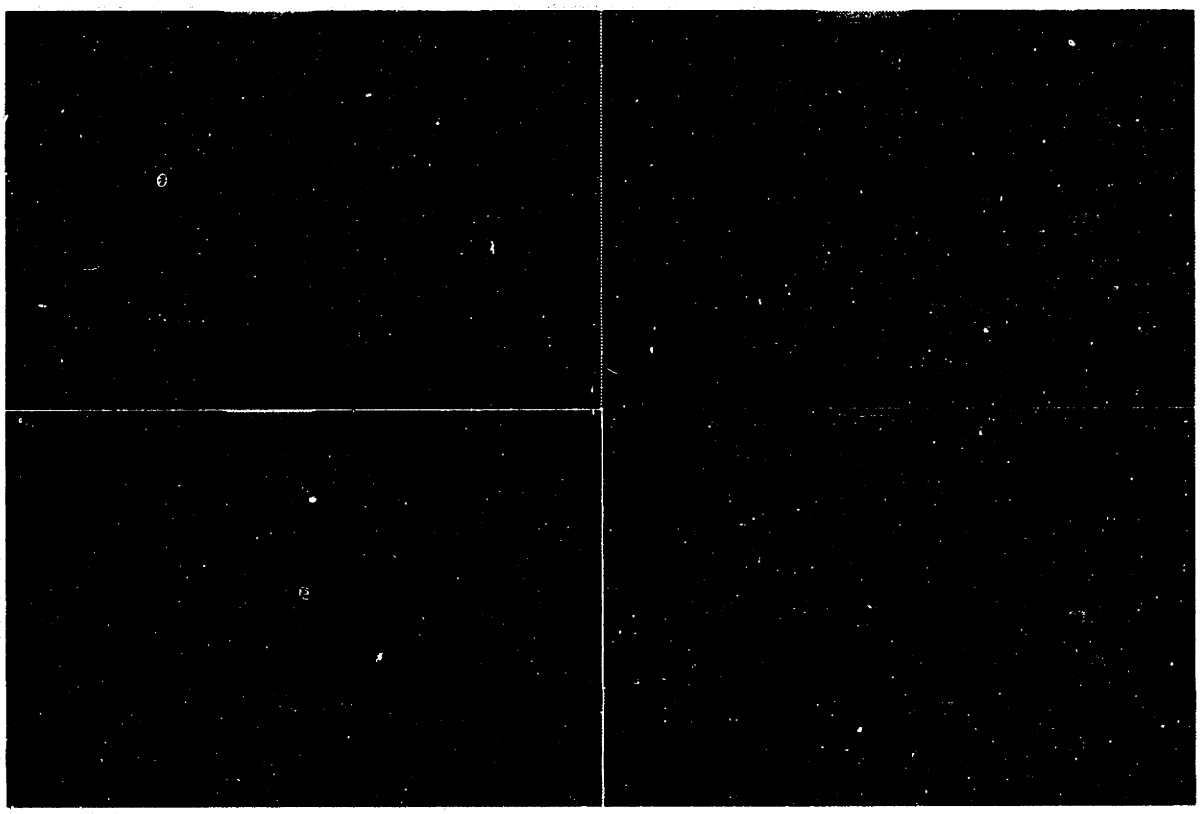


Figure B-2: A sequence of images of a catching attempt with the toss originating from the right side (right to left, top to bottom).

Bibliography

[Andersson 87] R. L. Andersson, *A robot ping-pong player: Experiment in real time control*, MIT Press, Cambridge, MA, 1987.

[Asada, Slotine 86] H. Asada and J.-J. E. Slotine, *Robot Analysis and Control*, John Wiley and Sons, New York, 1986.

[Bar-Shalom, Fortmann 88] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*, Academic Press, Inc., Boston, 1988.

[Bar-Shalom, Li 93] Y. Bar-Shalom and X.-R. Li, *Estimation and Tracking, Principles, Techniques, and Software*, Artech House, MA, 1993.

[Billingsley 83] J. Billingsley, *Robot ping pong*, Practical Computing, May 1983.

[Castaño, Hutchinson 94] A. Castaño and S. Hutchinson, *Visual Compliance: Task-Directed Visual Servo Control*, IEEE Transactions on Robotics and Automation, Vol. 10, No. 3, June 1994.

[Cannon, Slotine 95] M. Cannon and J.-J.E. Slotine, *Space Frequency Localized Basis Function Networks for Nonlinear System Estimation and Control*, Neurocomputing, 9(3), 1995.

[Fässler et al 90] H. Fässler, H. A. Beyer, and J. Wen, *A robot ping pong player: optimized mechanics, high performance 3D vision, and intelligent sensor control*, Robotersysteme 6, Springer-Verlag, pp. 161-170, 1990.

[Faugeras 93] O. Faugeras, *Three-Dimensional Computer Vision*, MIT Press, Cambridge, MA, 1993.

[Fiala et al 94] J. C. Fiala, R. Lumia, K. J. Roberts, and A. J. Wavering, National Institute of Standards and Technology, *TRICLOPs: A Tool for Studying Active Vision*, International Journal of Computer Vision, Vol. 12:2/3, pp. 231-250, 1994.

[Gelb 74] Technical Staff, The Analytic Sciences Corporation, ed. A. Gelb, *Applied Optimal Estimation*, MIT Press, Cambridge, MA, printed 1994.

[Hager et al 95] G. D. Hager, W.-C. Chang, and A. S. Morse, *Robot Hand-Eye Coordination Based on Stereo Vision*, IEEE Control Systems, 0272-1708/95, February, 1995.

[Hashimoto et al 87] H. Hashimoto, F. Ozaki, K. Asano, K. Osuka, *Development of a ping pong robot system using 7 degrees of freedom direct drive arm.*, Industrial Applications of Robotics and Machine Vision (IECON 87), Cambridge, MA, November 1987.

[Horn 86] B. K. P. Horn, *Robot Vision*, MIT Press, Cambridge, MA, 1986.

[Hove, Slotine 91] B. M. Hove and J.-J. E. Slotine, *Experiments in Robotic Catching*, Proceedings of the 1991 American Control Conference Vol. 1, Boston, MA, pp. 380-385, June 1991.

[Hove 91] B. M. Hove, *A Study of 3-D Robotic Catching*, M.S. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 1991.

[Kalata 84] P. R. Kalata, *The Tracking Index: A Generalized Parameter for α - β and α - β - γ Target Trackers*, IEEE Transactions on Aerospace and Electronic Systems, Volume AES-20, No. 2, March 1984.

[Kimura et al 92] H. Kimura, N. Mukai, and J.-J. E. Slotine, *Adaptive Visual Tracking and Gaussian Network Algorithms for Robotic Catching*, DSC-Vol. 43, Advances in Robust and Nonlinear Control Systems, Winter Annual Meeting of the ASME, Anaheim, CA, pp. 67-74, November 1992.

[Kimura 92] H. Kimura, *Adaptive Visual Tracking Algorithms for 3-D Robotic Catching*, M.S. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 1992.

[Niemeyer, Slotine 88] G. Niemeyer and J.-J. E. Slotine, *Performance in Adaptive Manipulator Control*, International Journal of Robotics Research, December 1988.

[Niemeyer 90] G. Niemeyer, *Computational Algorithms for Adaptive Robot Control*, M.S. Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, February, 1990.

[Nishihara, Thomas 94] H. K. Nishihara, H. J. Thomas, *Real-Time Tracking of People Using Stereo and Motion*, IS&T/SPIE Symposium on Electronic Imaging: Science & Technology, San Jose, California, February 6-10, 1994.

[Salisbury 87] J. K. Salisbury, *Whole Arm Manipulation*, Proceedings 4th International Symposium on Robotics Research, Santa Cruz, CA, August, 1987.

[Salisbury et al 88] J. K. Salisbury, W. T. Townsend, B. S. Eberman, D. M. DiPietro, *Preliminary Design of a Whole-Arm Manipulation System (WAM)*, Proceedings 1988 IEEE International Conference on Robotics and Automation, Philadelphia, PA, April 1988.

[Slotine, Li 91] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*, Prentice Hall, Englewood Cliffs, New Jersey, 1991.

[Swarup 93] N. Swarup, *Design and Control of a Two-Axis Gimbal System for Use in Active Vision*, S.B. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 1993.

[Townsend 88] W. T. Townsend, *The Effect of Transmission Design on Force-Controlled Manipulator Performance*, PhD Thesis, Department of Mechanical Engineering, MIT, April 1988. (See also MIT AI Lab Technical Report 1054).

[Watanabe et al 95] H. Watanabe, Y. Yoshii, Y. Masutani, and F. Miyazaki, *Motion Control for A Hitting Task: A Learning Approach to Inverse Mapping*, International Symposium on Experimental Robotics (ISER), Stanford, California, June 30-July 2, 1995.

[Wavering, Lumia 93] A. J. Wavering and R. Lumia, *Predictive Visual Tracking*, SPIE Volume 2056, Intelligent Robots and Computer Vision XII, 1993.

[Woodfill et al 94] J. Woodfill, R. Zabih, and O. Khatib, *Real-time Motion Vision for Robot Control*, Robotics for Challenging Environments, New York, pp. 10-18, Proc. ASCE 1994.

[Wright 93] A. Wright, *A High Speed Low-Latency Portable Vision Sensing System*, SPIE, September 1993.