DIRECT VISUAL SERVOING USING NETWORK-SYNCHRONIZED CAMERAS

By

DEREK C. SCHUURMAN, M.A.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

DIRECT VISUAL SERVOING USING NETWORK-SYNCHRONIZED CAMERAS

DOCTOR OF PHILOSOPHY (2003) MCMASTER UNIVERSITY

(Electrical and Computer Engineering) Hamilton, Ontario

TITLE: DIRECT VISUAL SERVOING USING NETWORK-SYNCHRONIZED CAMERAS

AUTHOR: Derek C. Schuurman

M.A.Sc.

SUPERVISOR: David W. Capson

NUMBER OF PAGES: xvi, 207

Abstract

A new strategy for direct visual servoing for robotic position control is described. The approach does not rely on any external position or velocity sensors but directly sets motor current using visual feedback alone. The method is novel in that it can be implemented without specialized vision hardware and is capable of processing visual feedback at high frame rates suitable for stable closed-loop position control of practical mechanical systems. A single RS-170 camera has a maximum sampling rate of 60Hz that significantly limits visual servoing performance. This limitation is overcome with multiple RS-170 cameras synchronized over a network in round-robin fashion to capture video fields at different instants in time. "Vision nodes", consisting of a camera and a dedicated computer, continuously process video at field rates to determine robot position. The vision algorithm, which is based on principal component analysis, is demonstrated to be suitable for accurate real-time position determination. Furthermore, the Euclidean distance in eigenspace in the presence of random occlusions is shown to be statistically related to the position measurement error variance. This leads to a novel approach for dealing with occlusions by considering them as "noise", the variance of which can be estimated directly from the Euclidean distance in eigenspace. A Kalman filter is then introduced to provide sensor fusion of the feedback from each vision node by weighting the position estimates from each camera to provide an improved overall position estimate. The Kalman filter also models the vision transport delays to reduce their effects on the visual feedback. Simulation results illustrate improvement in dynamic performance as the number of cameras are increased.

Further simulations predict robustness to simulated occlusions.

An experiment was designed and performed to experimentally verify the strategy for direct visual servoing. A 1 DOF servo drive equipped with a rotating link constituted a simple "planar robot" testbed for demonstrating the distributed vision and control techniques. The testbed was built using "off the shelf components" consisting of a network of four RS-170 cameras and computers connected to a master servo computer over a 100Mbps Ethernet network. An effective visual sampling rate of 240Hz was achieved. Several techniques were developed to achieve deterministic communications over Ethernet. The limitations on the number of cameras was analyzed and it was found that the Ethernet network could theoretically support up to 826 cameras. The main limitation was found to be the processor time on the master servo computer.

Experimental results are shown for the system performing direct visual servoing under various different conditions. A direct visual servo employing four cameras exhibits a step risetime of $190ms$ which closely matches the performance using traditional encoder feedback. Additional experimental results demonstrate the servo-hold performance and step responses with and without occlusions. Both full occlusions in a subset of cameras and partial occlusions in all cameras were investigated. The experiment results validate the simulation results and verify that the strategy is capable of stable direct visual servoing and is robust to occlusions.

Additional experiments are included that demonstrate performance under varying illumination conditions and various tele-robotic extensions.

Acknowledgements

I would like to thank my supervisor, Dr. David Capson, for his generous help, excellent guidance, consistent availability, and for providing me with an opportunity to pursue my research.

I would also like to thank the other members of my thesis committee, Dr. Gary Bone and Dr. Barna Szabados, for their comments and suggestions. I am also grateful for the many helpful discussions with my fellow graduate students in the Machine Vision and Image Analysis Lab.

I would also like to gratefully acknowledge the financial support provided by NSERC.

Last, but not least, I would like to thank my wife Carine for her love, patience, encouragement and support.

Of making many books there is no end,
and much study wearies the body.
Now all has been heard;
here is the conclusion of the matter:
Fear God and keep His commandments,
for this is the whole duty of man.
Ecclesiastes 12:12b-13

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The purpose of this chapter is to provide an introduction to the topic of visual servoing and to provide a snapshot of current research in the field. The chapter ends with an overview of the research that will be described in subsequent chapters.

1.1 Background

Visual Servoing can be defined as the task of using "visual information to control the pose of a robot's end-effector relative to a target object or a set of target features" [28]. Visual servoing is a multi-disciplinary field combining machine vision, robotics, and control theory. A good overview of visual servoing can be found in [28]. Numerous publications on the topic of visual servoing have been published over the past two decades and some books on the topic have also been recently published [67]. Vision provides sensory feedback that enables machines to observe and adapt to their environment. Visual servoing can improve the accuracy and repeatability of a robot, but it remains a challenging engineering problem. Visual servoing is a particularly demanding application because it requires high-speed vision in order to provide feedback at a sufficient rate.

Equipping robots with vision has the advantage of compensating for mechanical tolerances and improving adaptability, accuracy, and repeatability. Potential applications for the control of machines abound, ranging from large space robotics to nano and micro positioners. This has led to a considerable amount of research effort being devoted to the area of robot visual servoing. Despite the compelling advantages and the intuitive appeal of using visual feedback, there remain numerous challenges in the area of visual servoing. However, with the decreasing cost of vision hardware, increasing power of computers, and active research continuing in the field, visual servoing shows much promise for improving existing robot applications and enabling new possibilities.

1.1.1 Advantages of Visual Servoing

There are numerous advantages to employing visual servoing. In general, the use of visual feedback enables a robot to be more flexible by adapting to the world around it. Visual feedback enables a machine to operate on moving parts or on parts that are not precisely positioned thus eliminating the need for costly fixtures for registering the position of objects. Even when an object can be precisely positioned, the absolute position accuracy of the end-effector of a robot is limited by mechanical imperfections and tolerance stacks. The position of a robot is normally controlled by sensors mounted on the rotors of the servomotors which in turn drive mechanical transfer elements such as gears, belts, shafts, or chains. These mechanical transfer elements invariably include imperfections such as torsional compliance, friction, dead-zones, hysteresis, and backlash. The mechanical imperfections reduce accuracy and repeatability of the robot position at the end-effector. The use of visual feedback provides absolute position information that can compensate for mechanical shortcomings correcting the position of the robot end-effector. Ultimately this can provide cost savings by allowing robots to be constructed from cheaper and lighter materials that are more compliant.

1.1.2 Challenges of Visual Servoing

Visual servoing is a multi-disciplinary engineering problem which presents numerous practical and theoretical challenges. The antiquated RS-170 video standard continues to dominate the vision industry with its low 30Hz frame rates and long serial pixel transport delays. The resulting low vision sample rates and feedback latencies are perennial problems in visual servoing. The 30Hz legacy of the RS-170 industry standard has limited the rate at which many vision systems can sample their changing environments unless specialized hardware is employed [46]. These vision sample rates are typically orders of magnitude less than the electrical or mechanical bandwidth of most robot systems. Visual feedback latencies are exacerbated by the additional time required to perform vision computations which must be performed in real-time. As a result, directly closing the position loop of a mechanical system using vision alone is a particularly challenging engineering problem. To overcome these constraints normally requires the use of specialized vision and computer hardware which can dramatically increase costs. Even with high frame-rate cameras, various bottlenecks are encountered when interfacing cameras and transferring visual information into a computer for processing.

The vision sample rate and latencies do not provide the only engineering challenges. Even if the extra cost of adding a vision system can be justified, it can require substantial effort. Vision systems are also notoriously sensitive to slight image variations including variations in illumination, background clutter, and occlusions. These effects reduce the position accuracy and repeatability that can be achieved with vision feedback. Occlusions present a particularly difficult challenge that often occurs during normal operation as a robot or other objects are maneuvered in a workcell. A robot which uses visual feedback to control position must be robust to unexpected changes to ensure safe and reliable operation.

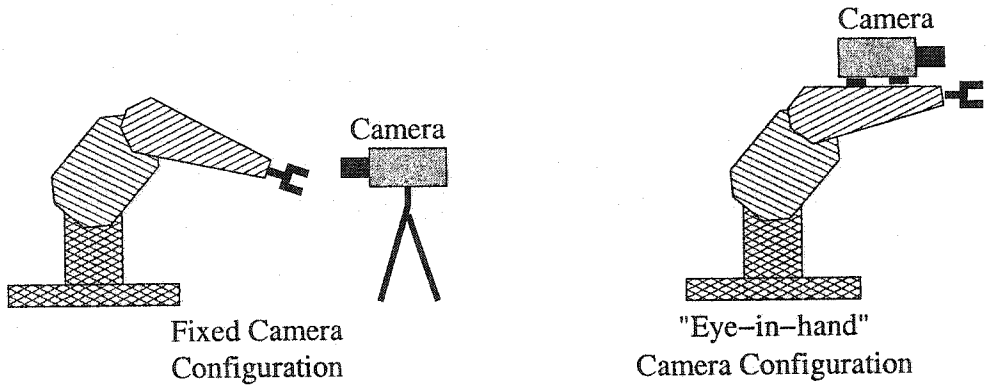


Figure 1.1: Fixed camera vs. "eye-in-hand" configuration.

1.2 Visual Servoing Architectures

The topic of visual servoing can be best described by identifying and categorizing the various approaches to visual servoing. There are numerous ways to categorize visual servoing systems. One way to differentiate visual servo systems is whether the camera is mounted on an end-effector ("eye-in-hand" configuration) or if a fixed camera is used as shown in Figure 1.1.

In [55] image features were used in a control loop thus introducing the distinction between an *image-based* visual servo and a *position-based* visual servo. Another important distinguishing characteristic is whether or not the visual servo is responsible for directly closing the joint position loop. More recently, a formal scheme was described in [28] to categorize all visual servo systems by posing two questions:

1. Is the control *position-based* or *image-based*?

2. Does the vision system provide setpoint inputs into the robot's position controller or does the visual controller *directly* control the joint position loop?

These two questions are discussed in more detail in the following sections and can be used as a basis for creating a taxonomy of visual servoing.

1.2.1 Position-Based versus Image-Based Visual Servoing

The first question above distinguishes *position-based* control from *image-based* control. These two approaches are discussed in detail below.

Position-Based Visual Servoing

Position-based visual servoing (PBVS) uses position in the control loop to guide a robot. Image features are extracted from images and used in combination with a geometric object model to deduce the position of an object. Position measurements may be planar or they may be 3D measurements including orientation. The 3D position and orientation of an object in space is normally specified using a six element vector which includes the 3D Cartesian coordinates of an object along with its roll, pitch, and yaw rotations. The measured position is fed back and the robot setpoints are updated to reduce the position error.

The advantage of PBVS is the convenience of specifying tasks in terms of positions in the real world coordinate frame. In addition, if the state equations are known, a predictor can be optionally added to improve tracking the trajectory of a moving object in Cartesian space.

There are various challenges in position-based approaches. The first challenge is to find a way to determine position from image features which are subject to noise. After the image features are identified, they must be mapped from feature space to position space. This process can be computationally expensive. Also, accurate camera modelling and calibration is critical to accurately compute position from image plane measurements. As a result, any calibration or modelling errors in the vision system will result in position measurement errors. Finally, in addition to requiring a model of the camera, it is also necessary to have a geometric model of the object being worked on.

Image-Based Visual Servoing

In contrast to PBVS, image-based visual servoing (IBVS) uses image features instead of position as a basis for feedback and control. The desired pose of the robot is controlled by specifying the desired image features which can be computed or specified using a "teach by showing" approach. The robot is then controlled by making the image features in the current view equal to the desired image features.

The process of selecting a set of features is critical to the performance of imaged-based visual servoing. Typical image feature sets include image points, lines, or areas [70]. The process of feature selection should take into account feature uniqueness, feature robustness, and the computational complexity of the feature extraction [18].

Controlling the actual robot motion requires a mapping from end-effector velocities to the image feature velocities [70]:

$$\dot{\mathbf{f}} = \mathbf{J}_v \dot{\mathbf{x}} \tag{1.1}$$

where $\mathbf{J}_v$ is the *image Jacobian*, $\dot{\mathbf{x}}$ is the end-effector velocity vector, and $\dot{\mathbf{f}}$ is the rate of change in the image feature vector. The image Jacobian can be determined by solving the image feature projection equations or it can be estimated experimentally.

Once the image Jacobian is determined, the robot can be controlled by setting the velocity setpoints of the robot to drive the image features to the desired setpoint. If the image Jacobian is square and non-singular this can be implemented using a simple proportional control law [28]:

$$\mathbf{u} = \mathbf{K}\mathbf{J}_v^{-1}(\mathbf{f}_d - \mathbf{f}) \tag{1.2}$$

where $(\mathbf{f}_d - \mathbf{f})$ is the difference between the desired image features and the current image features, $\mathbf{K}$ is a gain matrix, and $\mathbf{u}$ is the vector of control inputs for the end-effector velocities. If the image Jacobian is not square, a pseudoinverse $\mathbf{J}_v^+$ can be used.

The image-based approach has several advantages. Image-based control offers the advantages of reduced computation delay because the additional step of determining position

from image features is not required. Also, the accuracy of the image-based approach is less sensitive to camera calibration errors. Also, image-based techniques can provide a useful "teach-by-showing" strategy for specifying tasks.

However, there are also some disadvantages to image-based visual servoing. The image Jacobian $\mathbf{J}_v$ is usually unknown and must somehow be determined. In addition, the relationship between image features and robot motion can be highly non-linear or result in singularities. The result is that small changes in image feature space can map to erratic movements of the robot. Also, straight line trajectories that appear reasonable in feature space can map to inefficient or impractical motions of the actual robot end-effector. There is also no direct control over the Cartesian velocities of the robot end-effector. A phenomenon known as *camera retreat* can also occur in which the robot will first move away from the target direction before returning [13]. In extreme cases, a singularity in the image Jacobian can cause the camera to retreat to a distance of infinity [9].

Hybrid Methods

Problems in image-based visual servoing have led some researchers to investigate hybrid methods. Hybrid methods use IBVS to control some degrees of freedom and PBVS to control the other degrees of freedom. One such approach is described in [42].

1.2.2 "Look-and-Move" versus Direct Visual servoing

The second question posed at the start of this section distinguishes whether or not a visual servo is a *look-and-move* system or a *direct visual servo* system. A *direct* visual servo uses vision feedback *alone* to provide closed-loop position control for a robot joint. Consequently, direct visual servoing requires fast and accurate position determination from images to ensure a stable control loop. *Look-and-move* visual servoing uses other position sensors (such as encoders or resolvers) for feedback to internally stabilize the robot joints.

"Look-and-move" visual servoing can be further categorized as being *static* or *dynamic*.

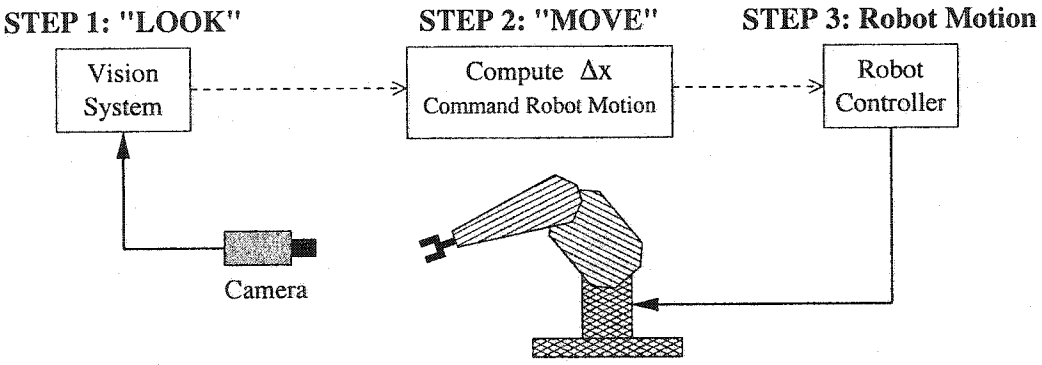


Figure 1.2: Static "look-and-move" visual servoing.

Static "Look-and-Move" Visual Servoing

Static "look-and-move" visual servoing is the most primitive type of visual servoing. Static "look-and-move" visual servoing consists of a simple sequence of three steps [70]:

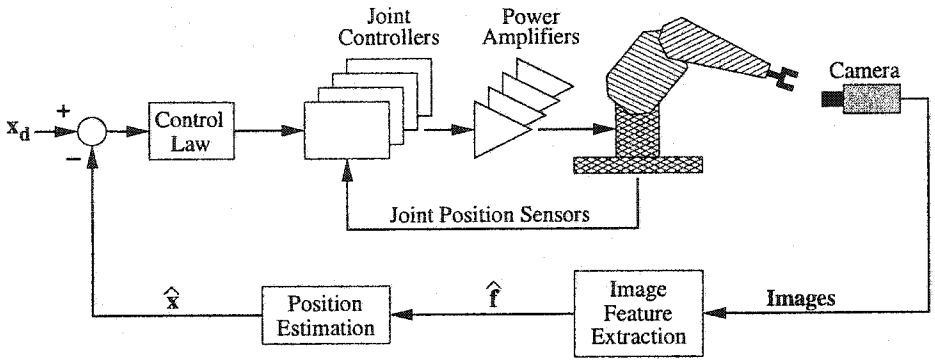
Step 1: "LOOK" The first step is to "look" at the scene and determine the position of the robot or object.

Step 2: "MOVE" The difference $\Delta \mathbf{x}$ between the desired position and the actual position estimate is computed and the robot is commanded to move.

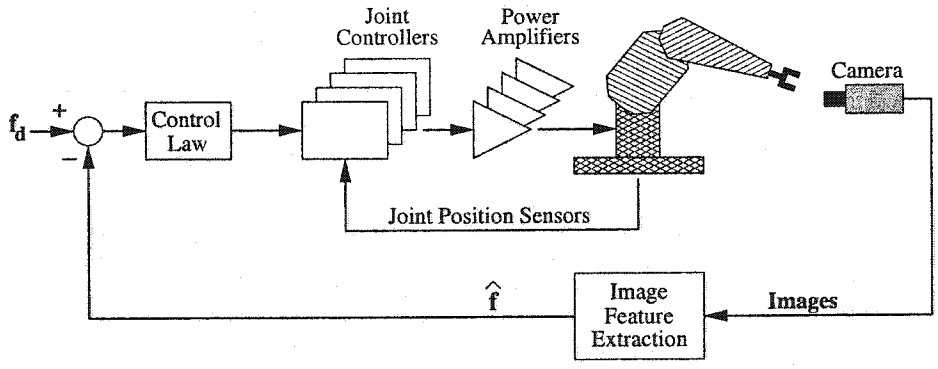
Step 3: "Robot Motion" The robot moves the commanded distance of $\Delta \mathbf{x}$. Step 1 cannot be repeated until the move is completed.

These three steps are illustrated in Figure 1.2 and may be repeated until a specified accuracy is reached. This approach is static in the sense that the "look" and "move" steps do not occur simultaneously but are executed sequentially. For this reason this technique is sometimes referred to as the "look-then-move" approach. The robot must come to a

complete stop before the process can be repeated making this approach unsuitable for applications with moving objects. Although this approach is simple to implement, it can be slow, particularly if the sequence needs to be repeated many times.



(a) Position-based dynamic look-and-move structure



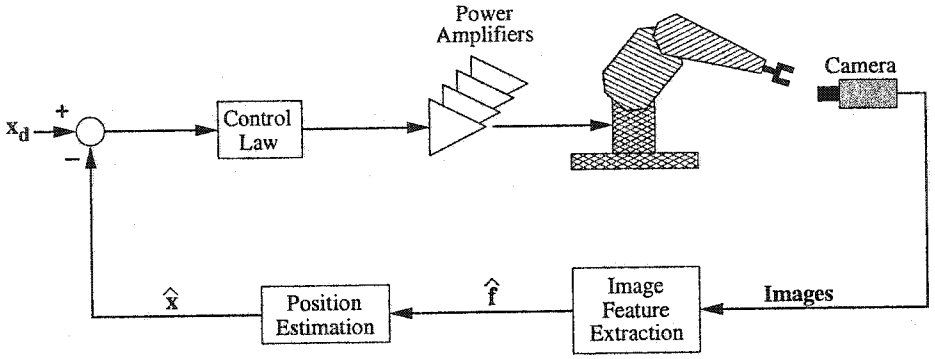
(b) Image-based dynamic look-and-move structure

Figure 1.3: Dynamic "look-and-move" visual servoing structures.

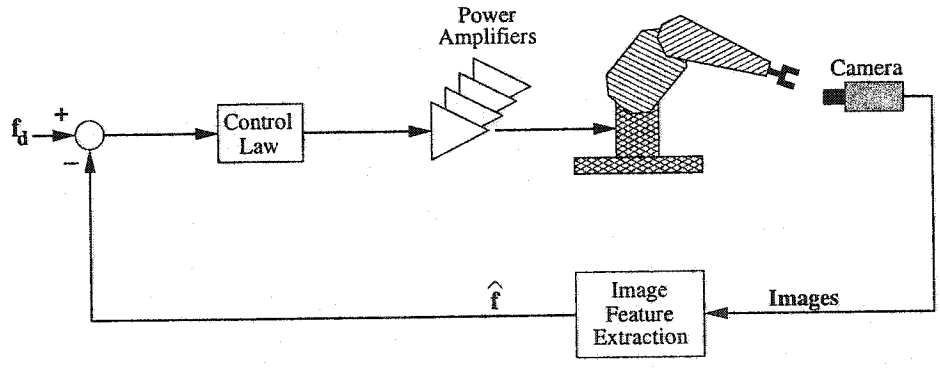
Dynamic "Look-and-Move" Visual Servoing

Dynamic "look-and-move" visual servoing is achieved by performing the three steps described in static "look-and-move" in parallel. Thus the vision system essentially forms

an outer control loop providing position corrections while the robot is moving. This approach can introduce additional system dynamics with interactions between the control loops. However, the underlying position control loop is still closed using independent position sensors such as resolvers or position encoders. Dynamic look-and-move visual servoing can be image-based or position-based as shown in Figure 1.3.



(a) Position-based direct visual servo structure



(b) Image-based direct visual servo structure

Figure 1.4: Direct visual servoing structures.

Direct Visual Servoing

In the past, the term "visual servo" used to imply a *direct* visual servo. However, because direct visual servoing configurations are so rare the term "visual servo" has gradually become accepted to mean any kind of visual robot control. Consequently, the term "direct" is now used to avoid confusion with other types of visual servoing configurations [28]. Direct visual servo systems use vision feedback *alone* to provide closed-loop position control for robot joints. Direct visual servoing can be implemented using fixed cameras or using the "eye-in-hand" configuration. Direct visual servoing can also be either position-based or image-based as illustrated in Figure 1.4.

Direct visual servoing is difficult because it demands adequate vision sample rates and timely feedback to ensure a stable control loop. The 30Hz legacy of the RS-170 industry standard has limited the rate at which many vision systems can sample their changing environments unless specialized hardware is employed. A general rule-of-thumb is to sample a system at a rate that is ten times higher than the closed-loop bandwidth [15]. This would suggest that 30Hz video feedback would only be suitable to control plants where the closed-loop bandwidth is approximately 3Hz or less. This is not adequate for the mechanical time constants and dynamics of most practical robot plants. Typical robot control loops run at several hundred Hertz to over 1kHz. A further problem is that typical image acquisition and processing latencies result in substantial transport delays. The transport delays inherent in vision feedback result in phase delays that cause instability in position loops. As a result, nearly all implementations adopt "look-and-move" visual servoing because low vision sampling rates and image processing latencies make direct visual servoing of a robot difficult [28].

In addition to low vision sample rates and long latencies there are several other practical obstacles to direct visual servoing. Researchers often use industrial robots as a platform for experimentation. Most industrial robots use proprietary interfaces (frequently serial)

which accept position or velocity setpoints but prevent access to the inner control loops and inner state variables. Also, the physical interface to most standard robots are typically low bandwidth serial connections which are unsuitable for high speed closed loop operation. Finally, direct visual servoing also requires the vision system to deal with the robot kinematics.

1.2.3 A Taxonomy of Visual Servoing

The categories described in the previous sections can be used to create a taxonomy of visual servoing. Visual servoing structures can be categorized based on whether they employ "look-and-move" or direct visual servoing and also based on whether they are image-based or position-based. The various structures illustrated in Figures 1.2, 1.3 and 1.4 can be classified based on these categories. The result is a taxonomy of visual servoing as shown in Figure 1.5.

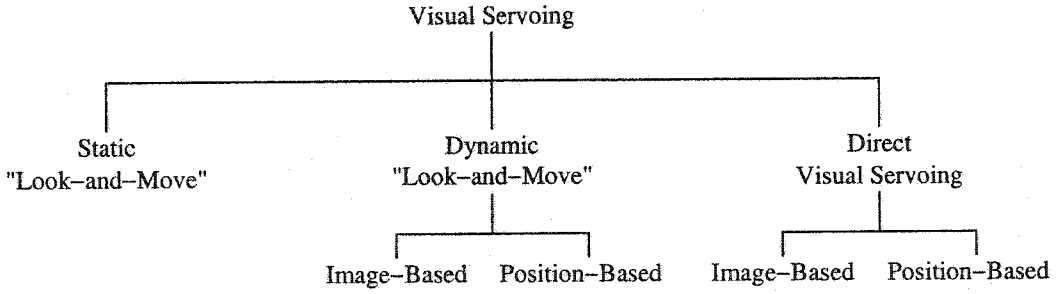


Figure 1.5: A taxonomy of visual servoing.

1.3 Visual Servoing Applications

There is a plethora of potential applications for visual servoing. However, the use of visual servoing is still limited for a variety of reasons. The restrictive cost of vision hardware

and the demanding computational requirements of real-time vision systems have traditionally made visual servoing infeasible in the past. However, the increasing power of personal computers and the decreasing cost of video sensors will make visual servoing a more attractive option. Already there have been numerous publications which describe the successful use of visual servoing for many different applications. Some representative examples of applications are summarized below.

Industrial Applications

Some of the earliest work in visual servoing described the use of visual feedback to guide industrial robots for simple pick-and-place operations [58]. Robot tasks involving stationary objects can be implemented using a simple static look-and-move visual servoing structure and many early vision guided robot systems used this approach. However, many robot tasks must deal with the more challenging problem of working with moving objects. These operations include such diverse tasks as grasping moving objects from a conveyor [75], automating mining machinery [10], or even picking fruit [24]. More recently, visual servoing has aided research in the area of cooperative robots for fixtureless assembly operations [4].

Space Applications

Robotics systems are becoming increasingly more important in space by performing hazardous operations and enabling remote exploration. It is the vision systems which enable a robot to adapt to the environment for exploration of remote planets or to perform hazardous tasks that have traditionally required a human operator.

One area where visual servoing shows promise is in improving the operation of large space manipulators which play a critical role in the operations of the space shuttle and the International Space Station [62]. These robots use joint-angle sensors for position control but the flexible structure of the robot manipulator can lead to position errors or even

oscillations at the actual end-effector. In [61] a visual servo is explored for the precise positioning of payloads with flexible space manipulators. Visual servoing can also be used to automate docking operations or grasp satellites. In these applications, visual systems can also serve to provide redundant feedback for safety and backup purposes. In addition to orbital maintenance and construction, visual sensing is enabling robotic systems to perform remote planetary surface exploration [69].

Game Applications

Over time, the notion of a "seeing" robot has captured the imaginations of many researchers in the field of visual servoing. Visual servoing allows a robot to interact with a changing environment enabling a robot to "play" various games. In [17] a real-time 3D vision system guides a robot for playing one-on-one volleyball with a human player. The robot system is capable of picking up the ball, playing various game sequences, recognizing faces and voice instructions, and even shaking hands with another player. In [7] a robot system is described for catching a toy mouse attached to a moving model train. Others have investigated robot juggling [53] and robot systems capable of playing ping-pong [2]. Robosoccer [8] is another application requiring visual tracking and control which has gained popularity in recent years. Although many of these applications are somewhat esoteric, they also serve to demonstrate the impressive capabilities of vision guided robots.

Tele-robotics Applications

Visual servoing is an important enabling technology for tele-robotic applications. Tele-robotics allows a remote user to operate robotic equipment from a distance in situations where it is not practical or safe for a human to be present. One active research area in tele-robotics is remote medical diagnosis and surgery (tele-medicine).

Visual servoing techniques can be applied to enable the remote control of robots. The long round-trip delays associated with transmitting visual data to a remote site makes it

difficult or impossible to dynamically close a vision loop over a remote link. A task can be specified in terms of visual features by a remote human operator, and visual servoing can be used to execute the task locally [22]. Tele-robotics is an open research area which can benefit greatly from advances in visual servoing.

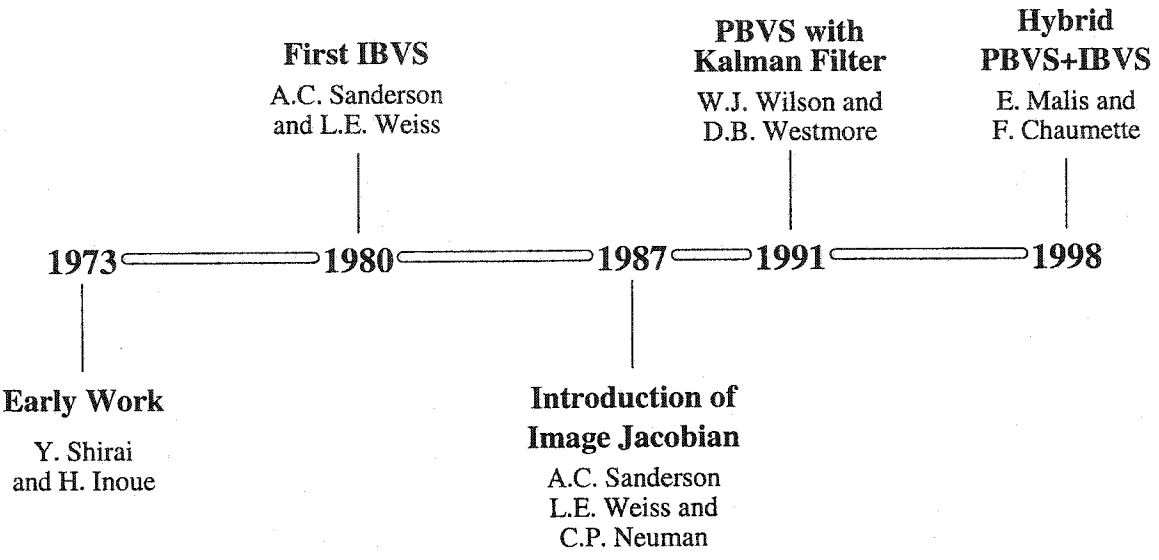


Figure 1.6: Timeline showing some of the major milestones in visual servoing research.

1.4 Previous Work

A timeline marking some of the major publication milestones in visual servoing is shown in Figure 1.6. Some of the first reported work in visual servoing is generally recognized to be that of Shirai and Inoue [58] in 1973. This work described how vision feedback could be used to improve the accuracy of a simple robot assembly operation. The progress of visual servoing systems remained slow for many years due to the demanding computational requirements of real-time computer vision. The early accomplishments in visual servoing are notable since this work was accomplished with only the modest computing power

available at the time. Early general purpose computers did not possess sufficient computational power for robot control. Most work required expensive pixel processing hardware and other custom computing hardware. In the early 1990's, Hulls and Wilson [27] demonstrated the use of multiprocessor transputers to meet the computational demands associated with visual servoing. It has only been in recent years that personal computers have reached a point where the real-time processing of visual information has become practical. An excellent tutorial on visual servoing has been written by Hutchinson, Hager, and Corke in [28]. More recent overviews on the topic have been written by Corke and Hutchinson in [12].

Image-based visual servoing uses image features in the control loop instead of position information and was first described in 1980 by Sanderson and Weiss [55]. Later, the *feature sensitivity matrix* (or *image Jacobian* as it was later called) was identified by Weiss, Sanderson and Neuman [70] in 1987 as a means of mapping end-effector velocities to image features velocities. Subsequent researchers have investigated various image-based approaches using the *image Jacobian* to map image features to robot motion [16] [34].

Other researchers have investigated *position-based* approaches to visual servoing. In 1991, Westmore and Wilson [71] successfully demonstrated 2D position-based visual servoing using a Kalman filter with an end-point mounted camera. In 1992, Wang and Wilson [68] described a vision system capable of tracking the 3D relative pose of an object using a Kalman filter. In 1996, Wilson, Hulls, and Bell [72] described a complete Cartesian position-based visual servo system using a Kalman filter.

Various shortcomings in classic position-based and image-based approaches have led to subsequent research into *hybrid* approaches which seek to combine the best elements from both. In 1999, Malis and Chaumette [42] described the use of a hybrid system called $2\frac{1}{2}D$ visual servoing. This method has advantages over strictly position-based techniques in that it does not require a 3D model nor precise camera calibration. This method also has advantages over strictly image-based techniques in that it ensures convergence over

the entire task space. However, this technique is not as robust as others and is sensitive to noise. Corke and Hutchinson have also suggested a partitioned approach to image-based visual servoing in [13].

The stability problems are magnified in "direct" visual servoing systems where the vision system must not only guide robot motion, but must also close the inner position loop. The need for real-time machine vision has fueled research into custom vision chips [74]. Most direct visual servoing systems that have been described typically employ specialized hardware to overcome the high computational requirements of real-time image processing and to provide faster sample rates [32] [46] [31]. To compensate for the shortcomings of the visual feedback, some systems have employed predictors or observers [26] [25]. Other direct visual servo systems have used simple robots with low-frequency dynamics that can be controlled using a standard 30Hz vision system [64]. Still other implementations that close the position loop using vision have still relied on external velocity sensors to close the velocity loop [21].

The one common theme in all the research has been the practical problems due to low vision sampling rates and latencies. Regardless of their approach, all researchers have had to contend with the low sample rates and high latencies intrinsic to standard vision systems. Low sample rates and high latencies ultimately introduce instabilities in a control loop [11]. Because of low vision sample rates and high latencies almost all visual servoing systems described in the literature employ "look-and-move" rather than "direct" visual servoing [28]. This research describes a novel approach to direct visual servoing that does not require specialized hardware and does not require any external position or velocity sensors.

1.5 Research Overview

This work explores *direct* visual servoing which forms the third branch of the taxonomy shown in Figure 1.5. More specifically, this work describes a direct visual servoing system that is position-based and uses multiple fixed cameras. The following sections describe the motivation behind researching direct visual servoing followed by an outline of the thesis and a list of contributions.

1.5.1 Motivation for Direct Visual Servoing

If the obstacles associated with the vision feedback can be overcome, there are several motivations for employing direct visual servoing. Direct visual servoing can ultimately eliminate the need for traditional position sensors, such as encoders or resolvers, and hence reduce the wiring complexity of a robot. Direct visual servoing can also serve as a "backup" position loop configuration in critical applications in case the primary position sensors fail. Direct visual servoing is of particular interest in applications where vision provides the only practical means of measuring position or where the use of traditional position sensors is not possible. Examples include micro or nano-positioners where the use of accurate position sensors may be costly or impractical. Another example is the case of flexible manipulators, where the position of the end-effector cannot be easily deduced from the joint positions. Other examples include vehicle guidance applications where the position of a vehicle can be measured with respect to other objects or visual landmarks. Examples include underwater station-keeping or the visual docking of space vehicles.

Direct visual servoing has received relatively little attention and most of the previous work describes the "look-and-move" type of visual servoing. This is likely due to the difficult challenges that arise when using machine vision alone for closing a position loop. These challenges include low sample rates, feedback latency, noise, and occlusions. However, many recent developments in both computer hardware, networking and software have

yielded a ripe opportunity for revisiting the problems surrounding direct visual servoing without relying on specialized hardware.

Recent advances in computer technology have produced desktop computers that now have sufficient computational horsepower to be employed for real-time vision applications. Moore's Law is still intact as the power of computers has continued to approximately double every 18 months with recent processors such as the Pentium IV reaching speeds of several GigaFLOPs. The ubiquitous PC (personal computer) is both powerful and cost effective due to the economies of scale. It is no longer advantageous nor necessary to develop custom hardware platforms to solve machine vision problems. In applications where more computing power is required, a Network of Workstations (NOW) can be constructed using multiple computers interconnected with high-speed off-the-shelf networking components [1]. Network technology such as Ethernet are now beginning to find applications in industrial environments [38]. The Internet and network computing have fueled the proliferation of fast and economical network components with the possibility for remote tele-operation.

Recent developments in hardware have been paralleled by developments in software. Many embedded operating systems are now available with a variety of tools. The Linux and open-source community have developed robust and reliable platforms that are finding their way into many embedded applications. Object-orientated software methods have matured and have aided the software development process.

This research leverages many of these recent hardware and software developments by combining new ideas with modern off-the-shelf components as the building blocks for a direct visual servo system.

1.5.2 Outline of the Thesis

This work describes a novel direct visual servo that is robust to occlusions. Throughout the work, the concepts are demonstrated using a 1 DOF (Degree Of Freedom) servo drive equipped with a rotating link which will be referred to as the "planar robot". The reason for working with a simplistic planar robot was to provide a reasonable testbed for the development of the distributed vision and control techniques. This testbed is not intended as a commercial product, but rather to facilitate the investigation of more fundamental problems and to verify a new approach to direct visual servoing.

This first chapter is intended as an introduction to the area of visual servoing along with relevant background information to provide a context for the work that will be described in the following chapters. The various types of visual servoing are identified and the concept of direct visual servoing is introduced.

Chapter 2 is devoted to describing the vision algorithm in detail. The limitation of 60Hz video is overcome with multiple RS-170 cameras synchronized over a network in round-robin fashion to capture video fields at different instants in time. This technique does not suffer from the various bottlenecks present when using high frame-rate cameras. Each RS-170 camera in the network has its own computer that processes video at field rates to determine robot position. The vision algorithm is based on eigenspace methods, otherwise known as Principal Component Analysis. The theoretical background and implementation of the eigenspace vision techniques are described. The feasibility of eigenspace methods for position determination is demonstrated using a simple 1 DOF robot testbed. The issues associated with computational speed, latency and position accuracy are explored along with the effects of occlusions on position measurements.

Chapter 3 describes various aspects of the modelling the direct visual servoing system. The various component subsystems in the visual servoing system are modelled and

described in detail. These subsystems include the vision system, the servo motor and amplifier, the position compensator, and a mechanical model of the planar robot. These models serve to aid in the understanding of the system dynamics and for system simulations. The models also form the basis of the discrete-time state equations for a Kalman filter to provide improved state estimates. Simulations show that the Euclidean distance from the manifold in eigenspace in the presence of random occlusions is statistically related to the position measurement error. Occlusions are thus considered as "noise" and the measurement error variance is estimated directly from Euclidean distance. The measurement error variance is used by the Kalman filter to weight position samples which arrive sequentially in a round-robin fashion from each of the cameras. The Kalman filter includes a model of the vision transport delays and provides timely position estimates necessary for stable closed loop servoing. A complete system model is constructed by combining models of the various subsystems. Finally, the system model is used to simulate the direct visual servoing system. Simulations predict the robot performance and illustrate the improvement in dynamic performance as the number of cameras are increased.

The implementation of a direct visual servoing testbed is discussed in Chapter 4. This chapter is significant since there are numerous practical constraints and implementation issues that need to be addressed. Various hardware and software issues are highlighted and the experimental setup is described in detail. System bottlenecks and the upper limits on the number of cameras are also investigated.

Chapter 5 describes the experimental results. Experimental measurements were obtained for a network of four cameras performing direct visual servoing of a simple planar robot. The results demonstrate the step response as well as stable servo-hold operation in the presence of full occlusions in a subset of cameras or with partial occlusions in all cameras. Other experiments demonstrate disturbance rejection, performance under varying illumination, and tele-robotic operation. The experimental results agree with simulations and show that the system performs well for direct visual servoing of the planar robot.

Chapter 6 provides a summary of the results along with the conclusions that were drawn. The contributions of this work to the area of visual servoing are also summarized. This chapter ends with some thoughts on possible directions for future work.

1.5.3 List of Contributions

The direct visual servo that is described is a position-based visual servo (PBVS) and is characterized by the use of multiple fixed cameras to control a planar robot. This is distinctive since the vast majority of the literature in visual servoing describes work based on the dynamic "look-and-move" approach which relies on an inner position loop using traditional position sensors. The work that has been done in direct visual servoing has either relied on specialized hardware [46] or has relied on joint-level velocity sensors for stable operation [21]. The approach described in this work is a pure direct visual servo in that it relies neither on position nor velocity sensors but directly sets motor current based on visual feedback *alone*. In addition, it is distinctive in that it does not rely on specialized hardware but rather uses uses off-the-shelf cameras and components.

Direct visual servoing was selected because of the interesting challenges it presents, particularly in association with the vision feedback. Typical computer vision problems such as low sample rates and long transport delays tend to reduce the performance of "look-and-move" visual servoing, whereas direct visual servoing is not even possible unless these issues are addressed. Low sample rates and delays both lead to instability if vision alone is used to close the position loop.

Several novel techniques were developed to overcome the barriers of low vision sample rates and transport delays to realize stable direct visual servo control. A novel means of increasing the visual sampling rate by using multiple cameras synchronized over a network is described in this work. Using four cameras, an effective visual sampling rate of 240Hz was experimentally demonstrated. This sample rate exceeds the rates reported in other

works that have relied on "off-the-shelf" components. Further analysis indicates that this technique can achieve effective visual sampling rates approaching 1kHz. This technique has several advantages including scalable visual sample rates and increased robustness to occlusions.

The camera synchronization over an Ethernet network also required special techniques to provide deterministic communications. These techniques included steps to avoid packet collisions by suppressing superfluous packets on a dedicated network and managing multiple access through the use of a master/slave round robin polling protocol. Furthermore, the use of UDP (User Datagram Protocol) packets ensured low-overhead and efficient packet exchanges.

Another contribution of this work is the development of a method for handling occlusions when using eigenspace vision methods. Robustness to occlusions is achieved by treating occlusions as "noise" and using a Kalman filter to weight the feedback from each camera based on the measurement error variance. The discovery that the measurement error variance from a camera can be determined directly from Euclidean distance in eigenspace provides an elegant method for quickly computing the measurement error variance. The result is a system which is robust to noise and occlusions.

Finally, it is shown that the ideas underlying the direct visual servo can be extended to realize a tele-robotic system. Issues such as transmitting images for remote monitoring and local task supervision can all be addressed using aspects inherent in the proposed direct visual servoing system. The eigenspace vision techniques used to determine position in the direct visual servo are also shown to be suitable for image compression to enable remote monitoring. Local task supervision is also addressed by using Euclidean distances in eigenspace or data readily available in the Kalman filter.

In the following chapters, the ideas behind these contributions are developed and described in detail. These contributions are demonstrated using both simulations and real-world experiments.

Chapter 2

Distributed Visual Processing in Subspace

2.1 Introduction

In "look-and-move" visual servoing, traditional position sensors are used to stabilize the position loop while the vision system provides additional information to improve accuracy and repeatability. In direct visual servoing, the vision system *alone* is responsible for providing position feedback. To ensure stable operation of a robot, it is critical for a direct visual servo to have a vision system that is both fast and accurate. This chapter describes a novel vision system that comprises off-the-shelf components to provide real-time feedback of the angular position of a planar robot.

The first section outlines the requirements of a vision system suitable for direct visual servoing. A vision system that meets these requirements is then described which employs multiple cameras and computers which are distributed over a network. The effective visual sample rate is increased by synchronizing the cameras over the network to interleave their acquisition times. Machine vision algorithms are then discussed and an algorithm based on Principal Component Analysis (or PCA) is selected. PCA is sometimes referred

to as eigenspace methods, and these two terms will be used interchangeably throughout this chapter. The theoretical background on PCA is briefly summarized along with its practical application to visual position determination. The feasibility of using PCA to determine the angular position of a planar robot from raw brightness images is demonstrated experimentally. The experimental results provide quantitative measures of computational speed, latency and position accuracy.

PCA is not only a useful tool for position determination but it is also known to provide highly compressed representations of images. This has potential uses in tele-robotic applications where PCA could simultaneously be employed to produce efficient image representations suitable for transmitting visual feedback to a remote user over a network. Therefore, additional experiments are performed to investigate the suitability of PCA for compressing images of a robot to enable remote monitoring.

Finally, the effects of occlusions are also explored since these often arise in practical vision applications. An experiment is performed whereby a set of robot images is subjected to numerous simulated occlusions. The results of this experiment indicate that there is a statistical relationship between Euclidean distance in eigenspace and the corresponding position measurement errors. This relationship is the key used in Chapter 3 to fuse data from multiple cameras to ultimately enable robust direct visual servoing.

2.2 Vision System Requirements

The vision system in a direct visual servo serves as the only direct means of feedback in the system. Consequently, it is important that the vision system meet certain requirements. These requirements include:

- Accurate position measurements
- Ability to work with arbitrary backgrounds

- High sample rate
- Low latency feedback
- Robustness to noise and occlusions

Many of these items are *desirable* in a "look-and-move" visual servo, but for a direct visual servo they are absolutely *necessary* in order for the system to be able to servo at all. First, the vision system must provide accurate position feedback that is comparable or better than that provided by traditional position sensors such a encoders. Since visual servoing applications operate in a wide variety of environments, the vision system must be capable of working with general images with arbitrary backgrounds to be practical. Since the vision system alone is responsible for position feedback, the sample rate of the vision system must exceed the mechanical time constants of the robot system in order to ensure a stable control loop. Also, the vision feedback should not include delays which will decrease the phase margin and lead to instability in the control loop. Finally, to prevent erratic motions in the robot position loop, the vision feedback must be robust and have some tolerance to noise and occasional occlusions. Lastly, it may be preferable to avoid costly proprietary components by using "off-the-shelf" components wherever possible.

2.3 Distributed Vision

2.3.1 Achieving High Video Frame Rates

One of the key requirements that have been identified for the vision system is the need for high vision sample rates. Most visual servoing systems operate at standard 30Hz frame rates or even a sub-multiple of the frame rate due to computational limitations. Two basic approaches were considered for addressing the issue of increasing the vision sample rates.

The first approach is to use high-speed digital cameras and the second approach is to synchronize the acquisition times of multiple RS-170 cameras to increase the effective sample rate.

One approach to achieving high video frame rates is to use high-speed digital cameras. Assuming an appropriate high-speed camera can be sourced, this option seems at first to be intuitive and straight-forward. However, there are several drawbacks associated with using a high-speed camera. One issue is that high-speed cameras are specialized devices that often require proprietary software and interfaces in order to operate. In addition, high-speed digital cameras cost significantly more than traditional vision systems. Even if the increased cost can be justified, there are other issues to consider. The use of high-speed cameras may necessitate the use of an appropriate multiprocessor architecture in order to perform all the image computations in real-time. At some point, computational or I/O bottlenecks ultimately limit the frame rates that can be processed by one computer. Even with processor clock speeds of several Gigahertz, the speed of memory in the standard desktop PC remains slow in comparison presenting a bottleneck for image processing. Even a standard RS-170 video signal contains a significant amount of data to process. A standard 60Hz video signal with a resolution of 640×480 pixels presents an effective data rate of over 18.4 Mbytes/second. Although modern computers equipped with a PCI bus are capable of capturing video at this rate, complex image processing of this image data at higher frame rates will present a computational challenge. For instance, capturing 640×480 pixel images at 1kHz results in a data rate of over 300 Mbytes/second which exceeds the bandwidth of a typical 66MHz 32bit PCI bus. At high frame-rates, even the access time of the system memory becomes an issue. On a typical PC using standard 133MHz DRAM, just accessing each individual pixel once could take over 2 milliseconds. This is not nearly fast enough for real-time processing since, at a 1KHz frame-rate, frames are arriving every millisecond. Attempting to process images from a high frame-rate camera in real-time is like "drinking from a firehose".

Another approach to achieving high video frame rates is to use multiple RS-170 cameras and synchronize the cameras to capture at different instants in time. This approach has the disadvantage of increased implementation complexity due to the practical issues of simultaneously controlling the synchronization of multiple cameras. However, assuming this can be overcome, this approach has numerous potential advantages. Some of these advantages include:

- use of off-the-shelf components with industry standard interfaces
- multiple cameras provide redundancy and improved reliability
- multiple cameras may increase robustness to some localized occlusions
- multiple vision computers can perform parallel computations to allow scalability while increasing frame rates

These advantages correspond with many of the vision system requirements identified in the previous section. For this reason, this approach was selected over the option of using a high-speed digital camera.

2.3.2 A Network of Cameras

The limitation of traditional 30Hz video can be overcome by synchronizing multiple vision nodes to capture images at different instants in time to improve the effective vision sampling rate. The use of multiple cameras has also been described for visual servoing to observe different views [43], but here it is used to improve the vision sample rate. The system consists of multiple RS-170 cameras which are each connected to a computer equipped with a frame-grabber to form a "vision node". Each vision node has the ability to control its camera synchronization and to capture and process images. Although each vision node logically consists of separate components, there is no reason why it could not be realized in one physical package to form a "smart camera" [73]. The vision nodes are connected by a

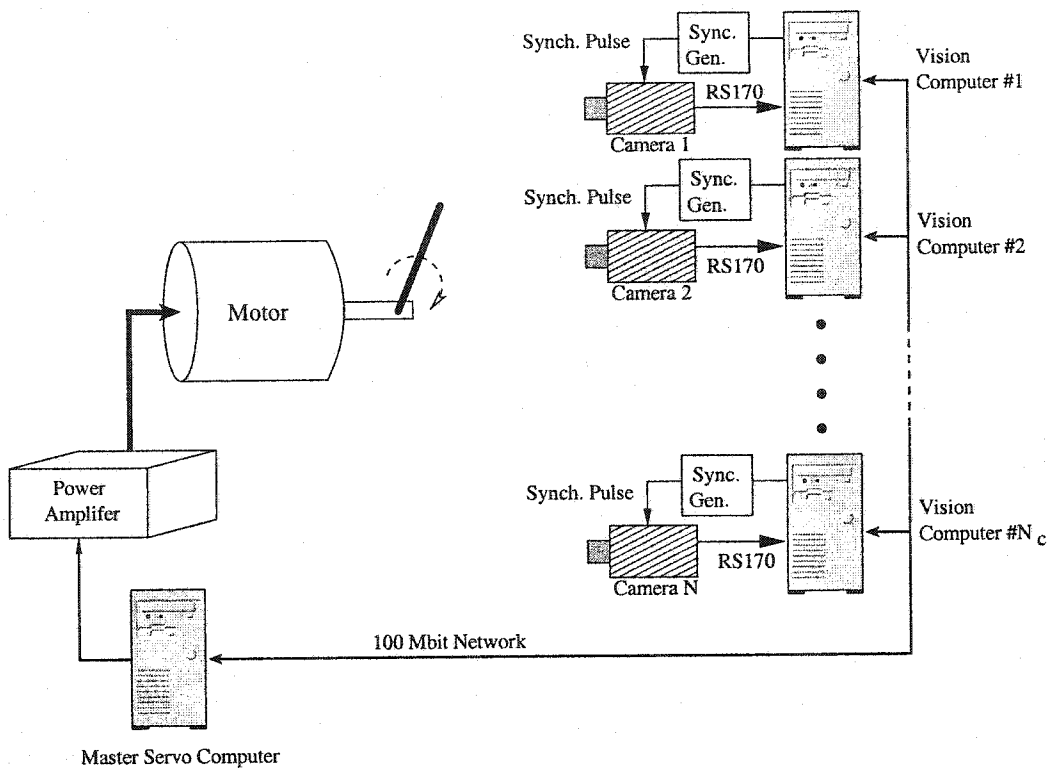


Figure 2.1: Network topology showing distributed vision nodes and the master servo computer.

network to form a distributed vision system controlled by a master computer as illustrated in Figure 2.1. The master computer, in turn, is also connected to the robot which is to be controlled. An image of the actual planar robot that was used is shown in Figure 2.5.

RS-170 video frames are sent at 30Hz and consist of an odd video field and an even video field which are shifted in space by one horizontal scan line and shifted in time by $1/60^{th}$ of a second. Therefore, the vision nodes were configured to process video *fields* rather than video *frames* to double the sample rate to 60Hz and decrease latency. The limitation of traditional video rates is then overcome by synchronizing each vision node to capture video fields at different instants in time to improve the effective vision sampling rate.

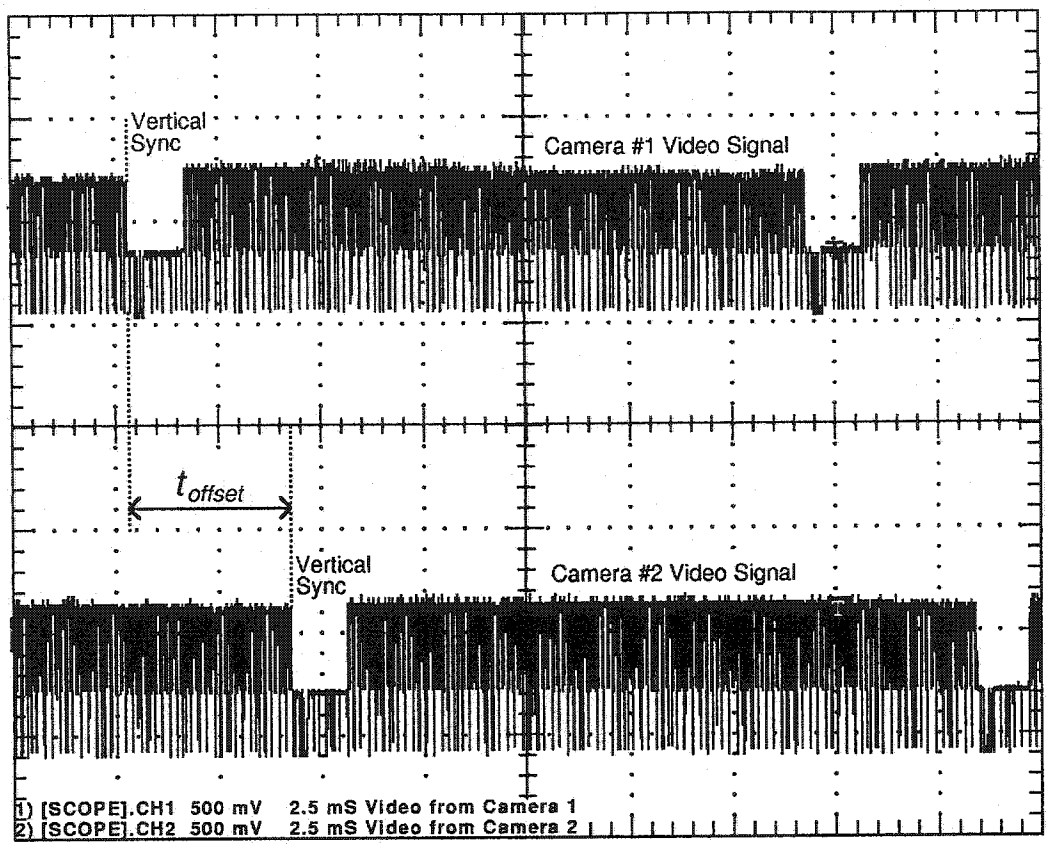


Figure 2.2: Video waveforms showing the relative video synchronization of two cameras for a system where $N_c = 4$.

The master computer communicates video synchronization information over the network to each vision node in a round-robin fashion. The relative timing of video field signals for two cameras measured in the network is illustrated in Figure 2.2. By synchronizing the cameras to interleave their capture times, the effective video sample rate f_s can be increased from 60Hz to:

$$f_s = 60N_c \tag{2.1}$$

where N_c is the number of cameras in the system. Each vision node has a dedicated computer to perform its local image processing so that the computational load is naturally distributed over the network. After the image computations are completed, each vision node sends the results in a small network packet to a master servo computer.

This technique is not bound to the use of standard RS-170 cameras. Even if high speed digital cameras are employed, this approach can be used to multiply the effective sample rate assuming the image acquisition time of the cameras can be controlled. More importantly, this approach serves to distribute the computational load over multiple computers.

This approach has the advantage of using off-the-shelf components and it is scalable by adjusting the number of vision nodes (N_c). Using multiple cameras also presents new possibilities for dealing with occlusions in situations where some cameras are occluded and others have a better view of the target object. This approach can increase the vision sample rate by N_c but it cannot improve the transport delay associated with the transmission and processing of the vision signal. Latency issues are discussed and dealt with in Chapter 3.

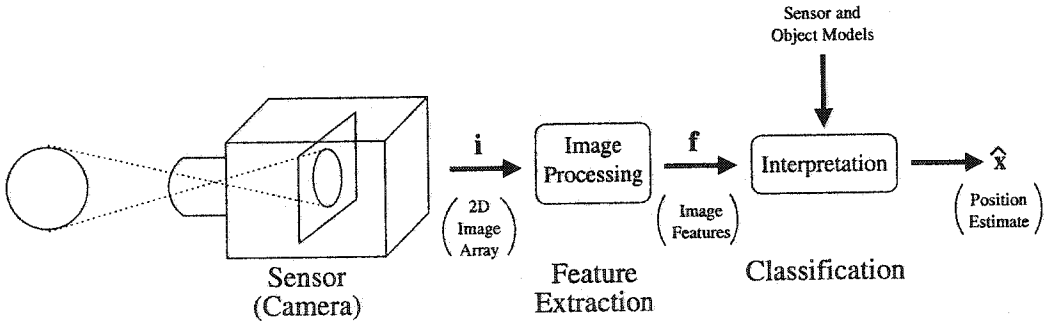


Figure 2.3: Pattern recognition for position determination.

2.4 The Vision Algorithm

Machine vision algorithms rely on pattern recognition techniques to extract information from images. Pattern recognition is the process by which sensory input is sensed, features are extracted, and the data is classified. The use of pattern recognition for machine vision is illustrated in Figure 2.3. Machine vision employs a camera as the sensor and the patterns to be recognized are image features. These image features may include points, lines, or any other measurements suitable for a particular application. The image features extracted from the sensed images form an image feature set that are used to classify the image. Classification proceeds by recognizing features in the feature set which compare with known features based on previous learning or models. In the case of visual servoing, the classifier must provide the pose of an object or of a robot end-effector.

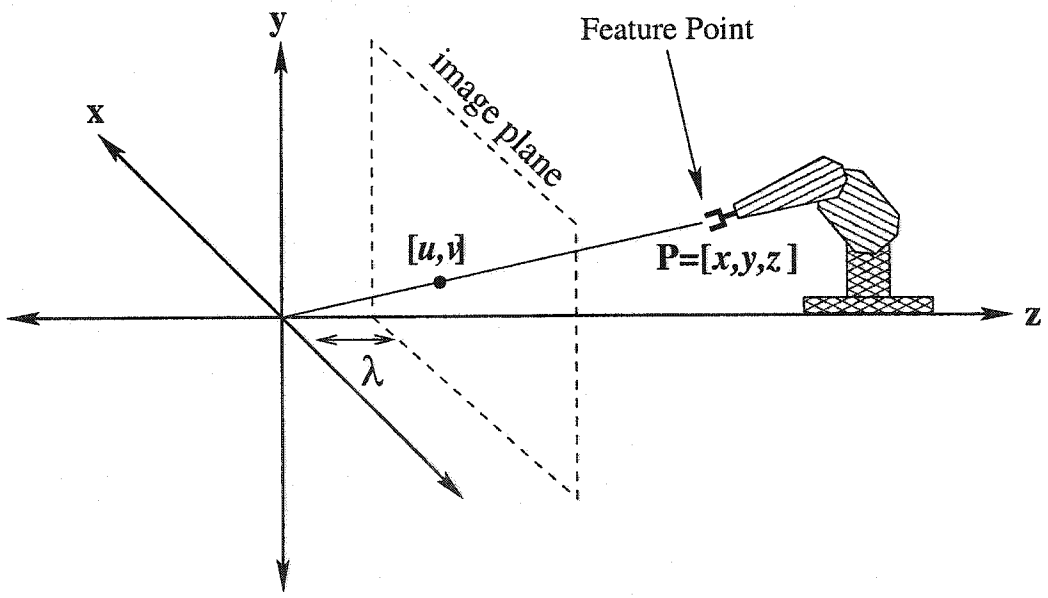


Figure 2.4: The image plane.

2.4.1 Model-Based Vision

Traditional machine vision systems perform pattern recognition on target objects using information from CAD shape models. Physical features of the object in the view of a camera are projected on the image plane and are extracted and classified. These features normally include points, lines, surface areas, or centroids. Figure 2.4 depicts a point feature P in 3D space which is projected onto $[u, v]$ in the 2D image plane of the camera. To classify the image features, a mapping between points in space and pixels in the image array must be determined. This mapping requires knowledge of the extrinsic and intrinsic camera parameters [35]. Extrinsic parameters are determined by the position and orientation of the camera. Intrinsic parameters are determined by the internal geometry of the camera such as the focal length and the principal point. Using perspective projection, one way to express the mapping of a 3D point P into the image plane is described in [28] as:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{AP} + \mathbf{c} \tag{2.2}$$

where $[u, v]$ are the coordinates in the image plane, and A and c are matrices that take into account the extrinsic and intrinsic camera parameters. These matrices must be determined using a camera calibration procedure.

This is a common approach to machine vision but it has some drawbacks. Camera calibration is often a cumbersome process. Small errors in calibration or camera position can lead to poor accuracy in the position measurements. If internal lens distortions are present in the camera, a complex calibration procedure will be necessary. Computational requirements are high and may prohibit the processing of full resolution images at frame rates. Consequently, vision systems must typically operate on small regions of interest which contain key features. Generally, multiple feature points are selected to form an image feature vector which is used to classify the pose of the target object. Features become

increasingly difficult to discern when arbitrary backgrounds are used. If inappropriate features are selected, the overall performance and robustness of this approach is poor. Lastly, this technique is sensitive to occlusions and noise.

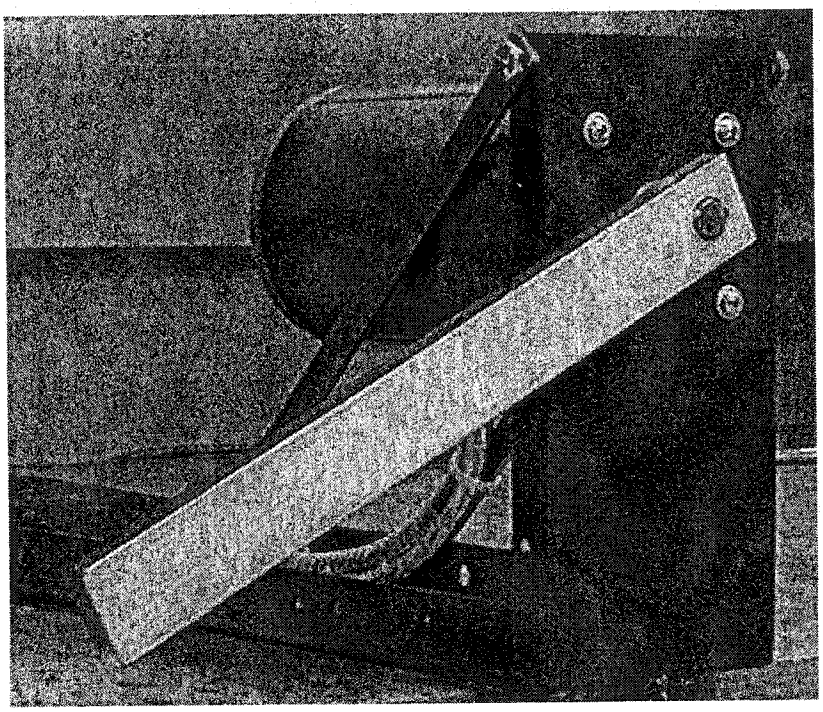


Figure 2.5: The planar robot.

2.4.2 Appearance Based Methods

An alternative approach to machine vision is to use appearance based methods. Appearance based methods rely on matching images with previously acquired images. In the simplest case, matching can be determined by simply taking the sum of square differences (SSD) over each pixel in two images. This will indicate how correlated one image is with another. This approach, also known as template matching, requires numerous computations and becomes unwieldy as the size and number of the images to compare increases. This simple

approach is not suitable for real-time applications where images need to be recognized rapidly to meet critical time deadlines.

However, if the dimensions of the image can be reduced into a suitably compact "subspace", this approach becomes more practical. PCA is one technique for obtaining a subspace called the eigenspace which is useful for reducing the dimensionality of an image down to a size where pattern matching can be efficiently performed. For this reason, PCA is sometimes referred to as eigenspace methods. Besides PCA, there are various other subspace techniques such as Independent Component Analysis (ICA). PCA has been explored for many machine vision applications but ICA has recently shown promise in some situations such as varying illumination conditions [19].

PCA techniques have been extensively explored for use in various vision applications such as face recognition [65], object recognition [45] [48] [47] [66], visual inspection [49], and visual positioning and tracking [49]. As described in section 2.4.1, traditional vision systems require geometric models to identify key features such as points, lines, or areas. In contrast to this approach, PCA uses an optimal set of orthonormal basis vectors to define a low dimensional subspace determined using the Karhunen-Loève Transform (or KLT). Once images are projected into the subspace they can be classified by finding the closest match in the low dimensional subspace [49].

PCA meets many of the vision system requirements listed in section 2.2. This approach works with raw intensity images with arbitrary backgrounds. In addition, this approach is based on appearance and does not require explicit knowledge of the geometry of an object. Because PCA methods do not rely on measuring geometric features, they function well despite the vertical distortion present when using video fields. This is desirable since the distributed vision system described in section 2.3 is based on video fields rather than video frames. Finally, PCA does not require any camera calibrations as long as the camera position remains the same at run-time as during the learning process. Consequently, this approach is tolerant to lens distortions and optical aberrations since these just form part

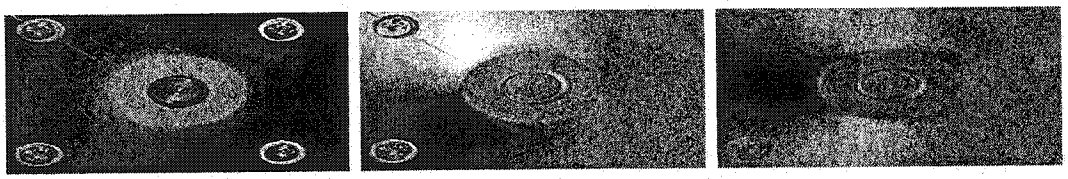
of the "appearance". For these reasons, the machine vision algorithm that was selected is based on PCA. Later in this chapter, the use of PCA will also prove useful for dealing with noise and occlusions.

2.5 Principal Component Analysis

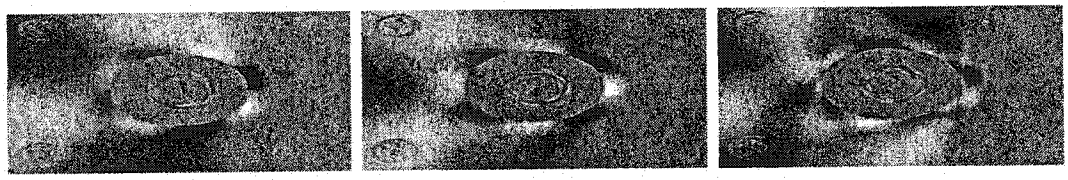
Traditional robot vision systems often use CAD shape models or geometric features. In contrast to this approach, eigenspace methods use a low dimensional subspace for comparing an image with a large set of possible images to determine the position of an object [49]. The KLT can be used to construct an optimal subspace for a set of correlated images. Since the complete range of robot motion is learned a-priori, the subspace provides a compact representation of all possible camera images. This low dimensional subspace can be used to efficiently find the closest match for an image in a predetermined set of images [49]. The compact image representation can also be employed to compress images.

The process begins with a learning phase to determine the eigenvectors and create the eigenspace. Although the learning phase is time-intensive, the run-time position determination uses simple and efficient computations suitable for real-time implementations. However, one constraint of this approach is that the illumination of the scene must remain constant. Difficulties associated with eigenspace methods include sensitivity to both occlusions and variations in illumination conditions. It is assumed that the illumination conditions around a robot can be controlled. However, occlusions are addressed later in section 2.7 since they often occur as a robot or other objects are maneuvered in a workcell.

The following subsections provide background information on the use of eigenspace methods for position determination and for image representation and reconstruction.



(a) Average image (b) Eigenvector #1 (c) Eigenvector #2



(d) Eigenvector #3 (e) Eigenvector #4 (f) Eigenvector #5

Figure 2.6: The average image and the eigen images.

2.5.1 Eigenspace Background Theory

The following section provides a summary review of the eigenspace vision method that was used. This is provided to introduce terminology and techniques that form the background to the vision work.

Creating the Eigenspace

The process begins by first learning the mapping between the appearance and position of the planar robot. This is accomplished off-line by incrementally moving the robot over its full range of motion while obtaining a set of n image vectors:

$$\{\mathbf{i}_1, \mathbf{i}_2, \mathbf{i}_3, \cdots \mathbf{i}_n\}$$

Each image is represented as a $1 \times p$ row vector where p is the number of pixels in an image. Each camera simultaneously captures its own set of training images allowing the learning to proceed in parallel for all N_c cameras. Next, the average $\mathbf{c}$ of each image set is

calculated using:

$$\mathbf{c} = \frac{1}{n}\sum_{j=1}^{n}\mathbf{i}_j \tag{2.3}$$

An example of the average image for the planar robot is shown in Figure 2.6(a). The average image is subtracted from each image in the set to form a matrix $\mathbf{P}$ given by:

$$\mathbf{P} = \begin{bmatrix} \mathbf{i}_1 - \mathbf{c} \\ \mathbf{i}_2 - \mathbf{c} \\ \vdots \\ \mathbf{i}_n - \mathbf{c} \end{bmatrix}^T \tag{2.4}$$

$\mathbf{P}$ has the dimensions $p \times n$ where p is the number of pixels in each image and n is the total number of images in the set. Since all the images in a set for a given camera are of the same robot and background, the set of images are highly correlated. The strong image correlations allow raw brightness images to be projected into a low dimensional subspace determined using the KLT. The KLT proceeds by finding the covariance matrix $\mathbf{R}$ formed from the image set $\mathbf{P}$:

$$\mathbf{R} = \mathbf{P}\mathbf{P}^T \tag{2.5}$$

The eigenvectors are then found using eigenvector decomposition:

$$\mathbf{\Lambda U} = \mathbf{RU} \tag{2.6}$$

where $\mathbf{\Lambda}$ and $\mathbf{U}$ are the eigenvalues and eigenvectors of the covariance matrix $\mathbf{R}$. These $\mathbf{U}$ matrix contains n eigenvectors which are the size of an image and which can be used as orthonormal basis vectors.

However, the covariance matrix $\mathbf{R}$ in Equation 2.5 is a $p \times p$ matrix whose dimensions depend on the number of pixels p. As the number of pixels p increases, the storage requirements for $\mathbf{R}$ increase by $O(p^2)$. For images comprising thousands of pixels, the storage requirements quickly exceed practical memory sizes. Fortunately, it is mathematically possible to also derive the same basis vectors using the $\mathbf{Q}$ covariance matrix [33]. The $\mathbf{Q}$

covariance matrix is given by:

$$\mathbf{Q} = \mathbf{P}^T\mathbf{P} \tag{2.7}$$

which has dimensions $n \times n$ where the typical number of image samples n is substantially smaller than the number of pixels p. This results in a more manageable matrix size that can be manipulated by a computer with a practical memory size. The same basis vectors $\mathbf{U}$ in equation 2.6 can be obtained from $\mathbf{Q}$ by using the singular value decomposition (SVD) of $\mathbf{P}$ given by:

$$\mathbf{P} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{2.8}$$

The SVD of $\mathbf{P}$ can then be substituted into equation 2.5 to give the following:

$$\mathbf{R} = \mathbf{P}\mathbf{P}^T = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T \tag{2.9}$$

Since the matrix $\mathbf{V}$ is orthogonal, the product with its transpose is given by:

$$\mathbf{V}^T\mathbf{V} = \mathbf{I} \tag{2.10}$$

Therefore, equation 2.9 can be simplified to:

$$\mathbf{R} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T \tag{2.11}$$

As expected, this equation is a restatement of equation 2.6 where the matrix $\mathbf{U}$ contains the eigenvectors of $\mathbf{R}$ and $\mathbf{\Sigma}^2$ contains the eigenvalues.

Next, the SVD can be substituted into equation 2.7 to give the following:

$$\mathbf{Q} = \mathbf{P}^T\mathbf{P} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{U}\mathbf{\Sigma}\mathbf{U}^T \tag{2.12}$$

Again, the matrix $\mathbf{U}$ is orthogonal such that:

$$\mathbf{U}^T\mathbf{U} = \mathbf{I} \tag{2.13}$$

Therefore, the equation 2.12 simplifies to:

$$\mathbf{Q} = \mathbf{V}\mathbf{\Sigma}^2\mathbf{V}^T \tag{2.14}$$

where the matrix $\mathbf{V}$ contains the eigenvectors of $\mathbf{Q}$ and $\mathbf{\Sigma}^2$ contains the eigenvalues. The eigenvectors of the $\mathbf{Q}$ covariance matrix can be related to the $\mathbf{U}$ basis vectors by rearranging equation 2.8 as follows:

$$\mathbf{U} = \mathbf{\Sigma}^{-1}\mathbf{P}\mathbf{V} \tag{2.15}$$

Since $\mathbf{\Sigma}$ is equal to the square root of the eigenvalues, the equation for the basis vectors $\mathbf{U}$ can be rewritten as:

$$\mathbf{U} = \mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{P}\mathbf{V} \tag{2.16}$$

where $\mathbf{V}$ and $\mathbf{\Lambda}$ are the eigenvectors and eigenvalues respectively of the $\mathbf{Q}$ covariance matrix, and $\mathbf{P}$ is the original image set matrix.

Because the planar robot sub-images consisted of 34001 pixels, determining the basis vectors using the $\mathbf{R}$ covariance matrix in only 256Mbytes of available memory was not possible. Therefore, equation 2.16 was used to compute the basis vectors from 101 sample images producing a $\mathbf{Q}$ covariance matrix with manageable dimensions. The resulting eigenvalues are plotted in Figure 2.7 and the images corresponding to the first five basis eigenvectors are shown in Figure 2.6(b) to (f). Since the eigenvector computations are done off-line, they have no impact on the final run-time performance.

When using the KLT, the most significant basis vectors are those with the largest eigenvalues. Consequently, the number of dimensions can be reduced by using the k most significant eigenvectors:

$$\{\, \mathbf{u}_i \mid i = 1, 2, \ldots, k \,\} \tag{2.17}$$

where $\mathbf{u}_i$ is the i^{th} basis vector in $\mathbf{U}$ and the the k most significant basis vectors are selected with the largest corresponding eigenvalues such that:

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \tag{2.18}$$

A subspace based on the k most significant eigenvectors forms a set of orthonormal basis vectors that define the *eigenspace*. Typically, eigenspaces of 20 dimensions or less have

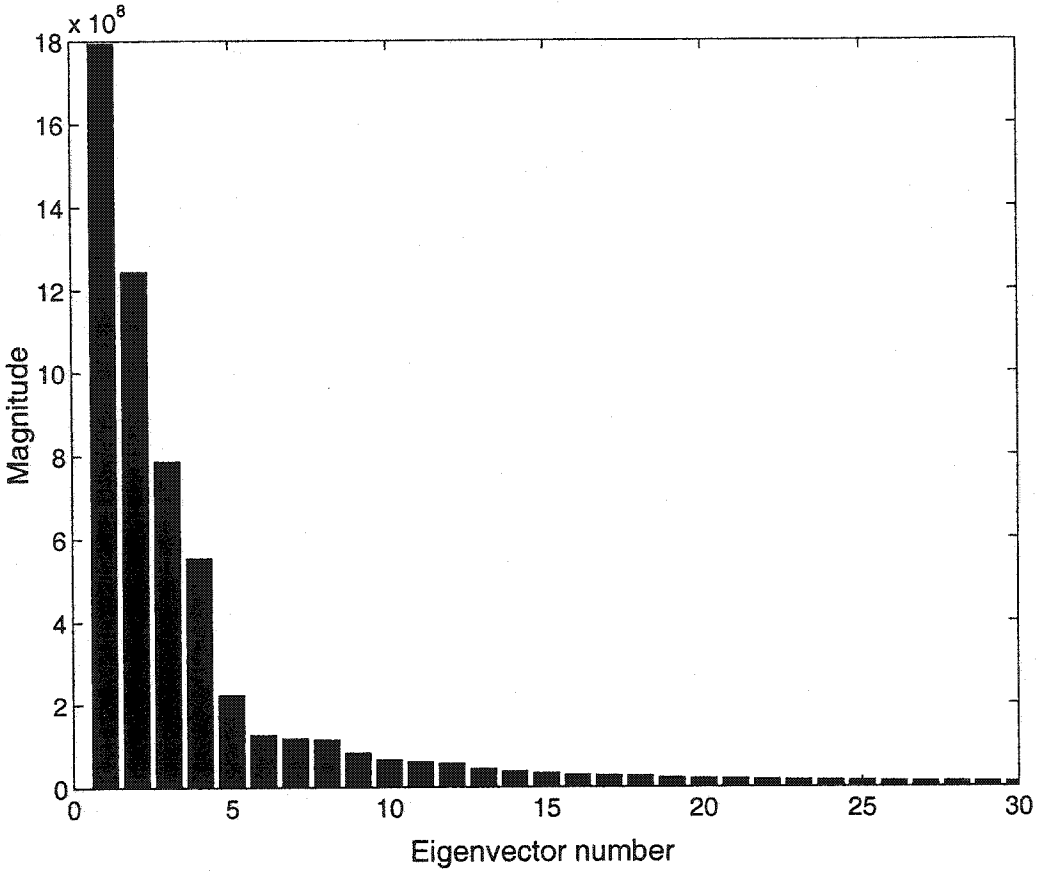
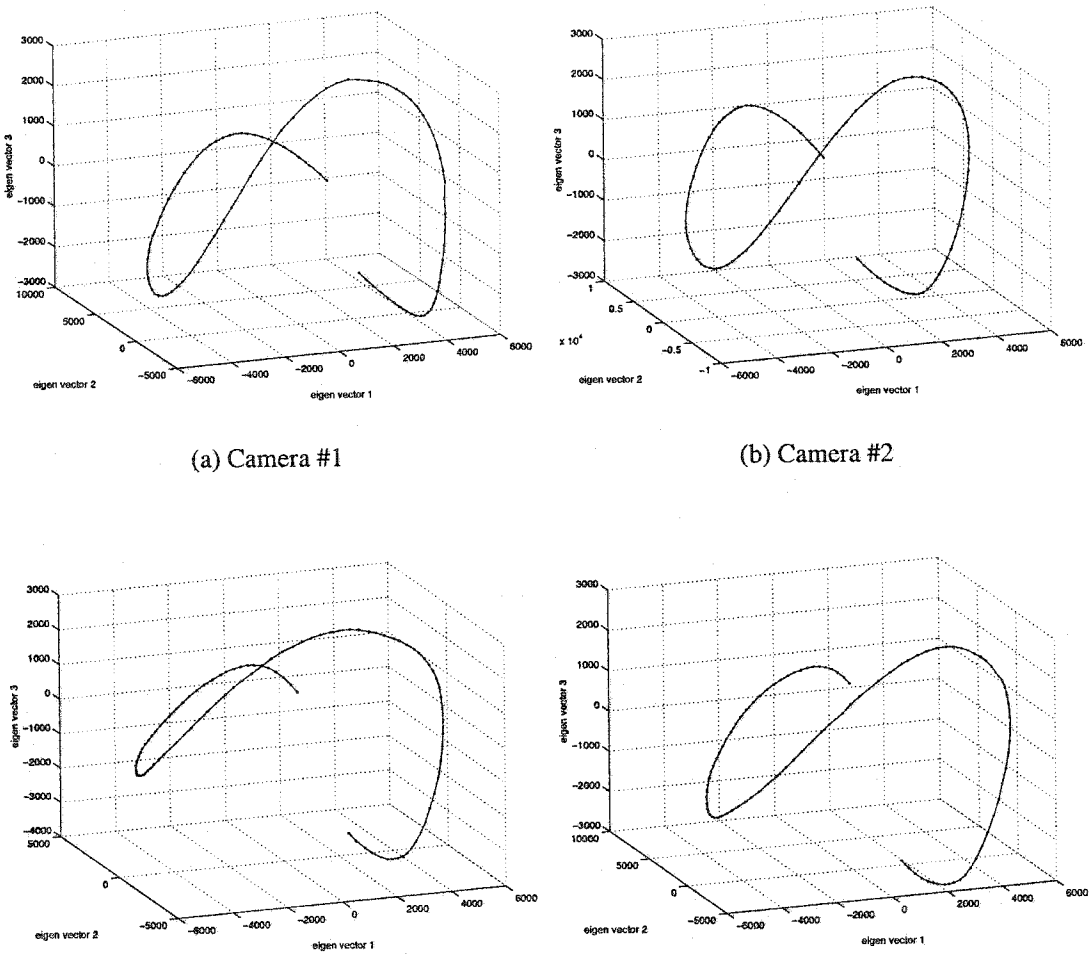


Figure 2.7: Eigenvalue magnitudes.

been shown to be more than adequate for performing recognition in most applications [49]. The exact number of eigenvectors required will depend on the application. In this application, experimentation showed that increasing the number of eigenvectors beyond $k = 5$ yielded little improvement in position accuracy. This is readily apparent from the eigenvalue magnitudes plotted in Figure 2.7 which show that beyond the first five eigenvectors the eigenvalue magnitudes rapidly diminish. Because each camera has a unique view of the robot, a unique eigenspace must be constructed for each camera.



(a) Camera #1

(b) Camera #2

(c) Camera #3

(d) Camera #4

Figure 2.8: The first 3 dimensions of the eigenspace manifolds for 4 cameras.

The Parametric Manifold

Next, each image in the training set is projected to eigenspace using the k most significant eigenvectors. The result of projecting an image $\mathbf{i}_c$ into eigenspace is a point $\mathbf{f}_c$:

$$\mathbf{f}_c = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_k \end{bmatrix} [\mathbf{i}_c - \mathbf{c}]^T \tag{2.19}$$

The complete set of images projected to eigenspace forms a set of points in eigenspace called the *manifold* [45]. The manifold exists in a low dimensional space and is therefore a compact representation of the appearance of an object. Plots of the eigenspace manifolds produced by varying the position of the planar robot through an angular displacement of π radians are shown in Figure 2.8 for four different cameras. Each of the manifolds is parameterized by the angular position of the planar robot that is varied throughout the set of images to produce the *parametric eigenspace* [45]. The workspace manifolds exhibit a smoothly varying plot for each set of images. The number of points in each manifold was effectively increased by interpolating between training points on the manifold to achieve a position quantization interval equivalent to the resolution of a 4000 count/revolution rotary encoder.

Image Classification and Position Determination

Next, images of the robot in arbitrary positions as captured by a given camera can be projected into subspace where they can be classified. The low dimensional subspace is a computationally efficient space to perform matching of an image with a predetermined set of images [49]. It has been established that the similarity between two images can be approximated by the distance between the projected points in eigenspace [45]. By finding

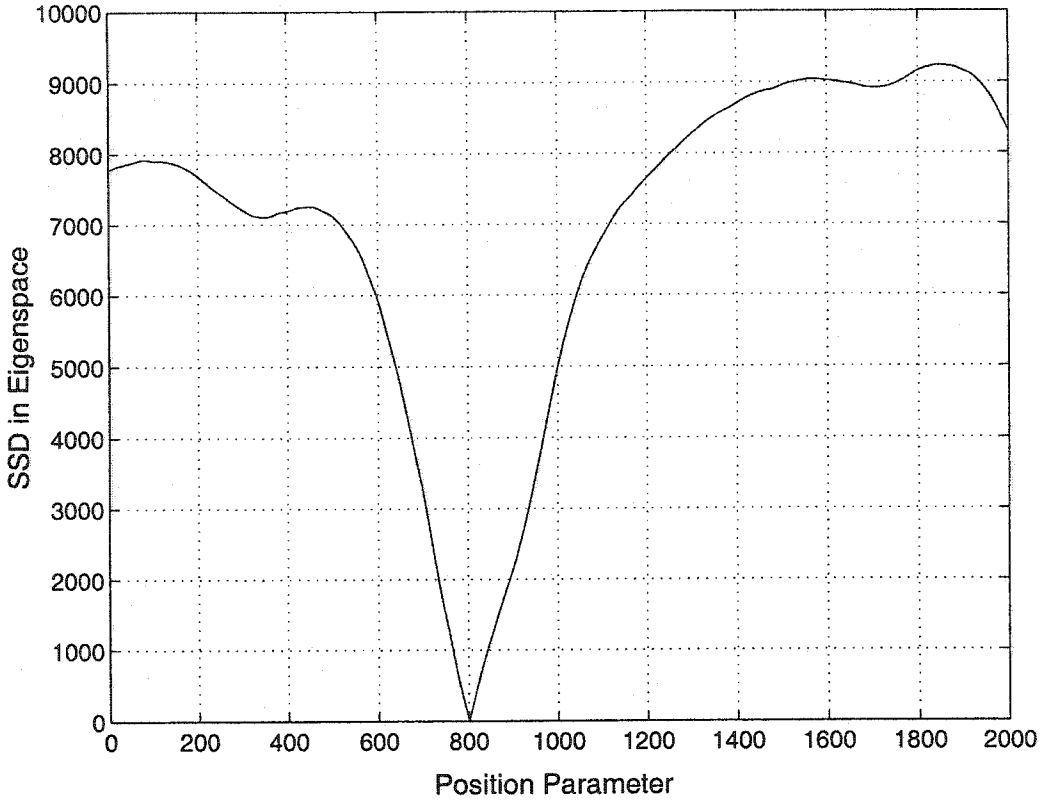


Figure 2.9: Example of a typical SSD Plot.

the closest point on the manifold, the image in the set that most closely matches the unknown image can be determined and the position of the robot can be deduced [49]. The classification of an image proceeds by finding the closest point on the manifold or "nearest-neighbor". The algorithm finds the sum-of-square differences (SSD) to each point on the manifold and selects the smallest SSD as the closest:

$$d = \overset{\min}{i} \|\mathbf{f}_c - \mathbf{f}_i\|^2 \tag{2.20}$$

where $\mathbf{f}_c$ is an image projected into eigenspace and $\mathbf{f}_i$ is a point on the manifold. Figure 2.9 is an example of a SSD vs. position parameter plot for a particular image. The minima in Figure 2.9 corresponds to the closest point on the manifold. Once the manifold point

closest to a projected image point is found, it can be used to obtain an estimate of the position of that image. This approach works well if images match those in the original training set and hence will closely align with a point on the manifold. However, if noise, occlusions or outliers occur, the corresponding projected point will stray from the manifold and classification errors may be introduced.

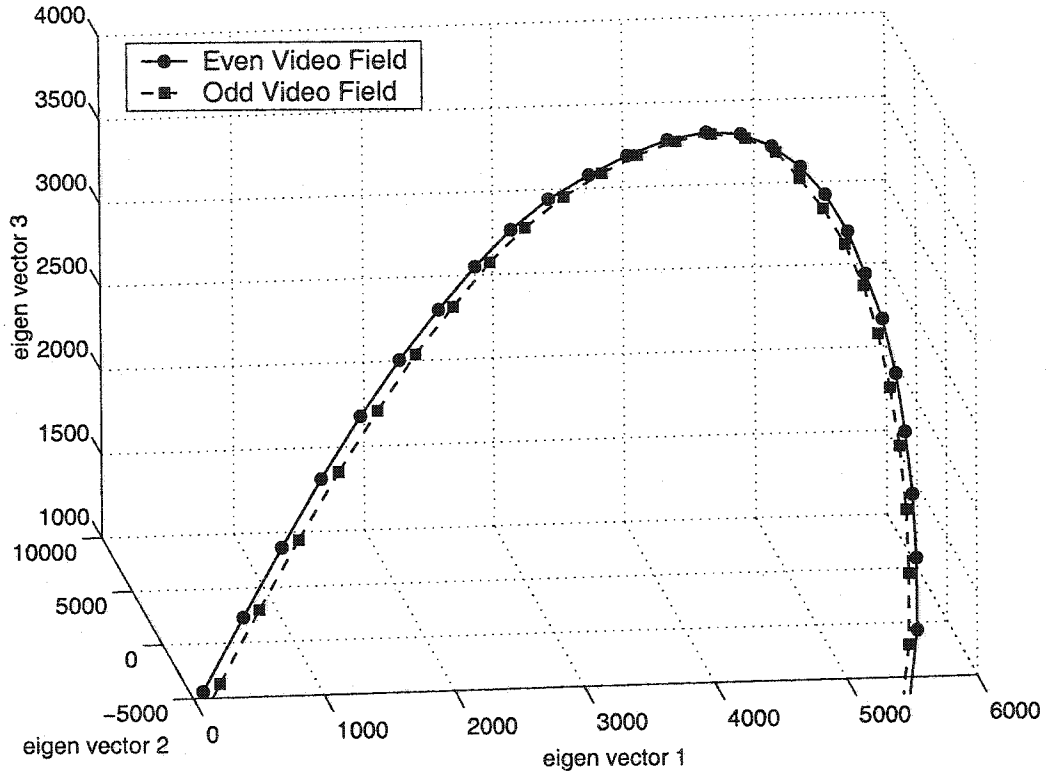


Figure 2.10: Magnified plot showing odd video field images projected into an eigenspace trained using even video field images.

Odd and Even Video Fields in Eigenspace

Eigenspace methods are insensitive to the vertical distortion when using video fields since the position is determined by appearance rather than geometry. However, the appearance

of odd and even video fields will differ slightly since the video fields are offset by one horizontal scan line. Therefore, if the manifold is constructed using even video fields there may be errors when matching with odd video fields or vice versa.

This is illustrated in Figure 2.10 where a magnified portion of the manifold trained using even video field images is shown alongside the projected image points from corresponding odd video field images. Clearly, the odd video field images do not project to points directly on the manifold. The odd video field image points lie adjacent to the manifold following the general shape of the manifold. Unfortunately, the odd video field points appear to be slightly offset from the manifold points in such a way that a minimum Euclidean distance search may result in small position errors.

There are several approaches that could be used to address this undesirable effect. One solution would be to create two separate eigenspaces; one trained using only even video fields and one trained using only odd video fields. This would result in the creation of an odd and even eigenspace each with their own manifold. The vision node would run and alternately project images into the even or odd eigenspace as appropriate and compare them with the corresponding manifold. This would completely eliminate possible errors due to appearance differences in the odd and even video fields. Although this approach would not impact the run-time performance of the vision nodes it would require twice the memory storage requirements. Memory requirements are not insignificant since each eigenvector that must be stored contains as many floating point numbers as there are pixels in an image. Also, the training phase will take almost twice as long if two distinct training sets must be processed. Despite these drawbacks, this approach will be preferred if position measurement accuracy is critical.

A less effective but easier approach is to "average" the odd and even video field images during the learning phase. The raw odd and even video field images can be averaged to form a training set that will yield an eigenspace and manifold based on the average appearance of video field images. The time required to average the odd and even video filed will increase

the time require for the learning phase but it will be faster than that required to find two separate eigenspaces. This manifold will not exactly match odd or even video field image points but will provide a reasonable compromise for both. Small position errors will still occur but the storage requirements will not increase.

Finally, if the errors introduced by the differences in the video fields are small and tolerable, these artifacts can be ignored. This is particularly appropriate if errors introduced due to odd and even video fields are small or comparable to other sources of error such as image noise. This approach has the main advantage that no special considerations need to addressed during the learning phase or during run-time.

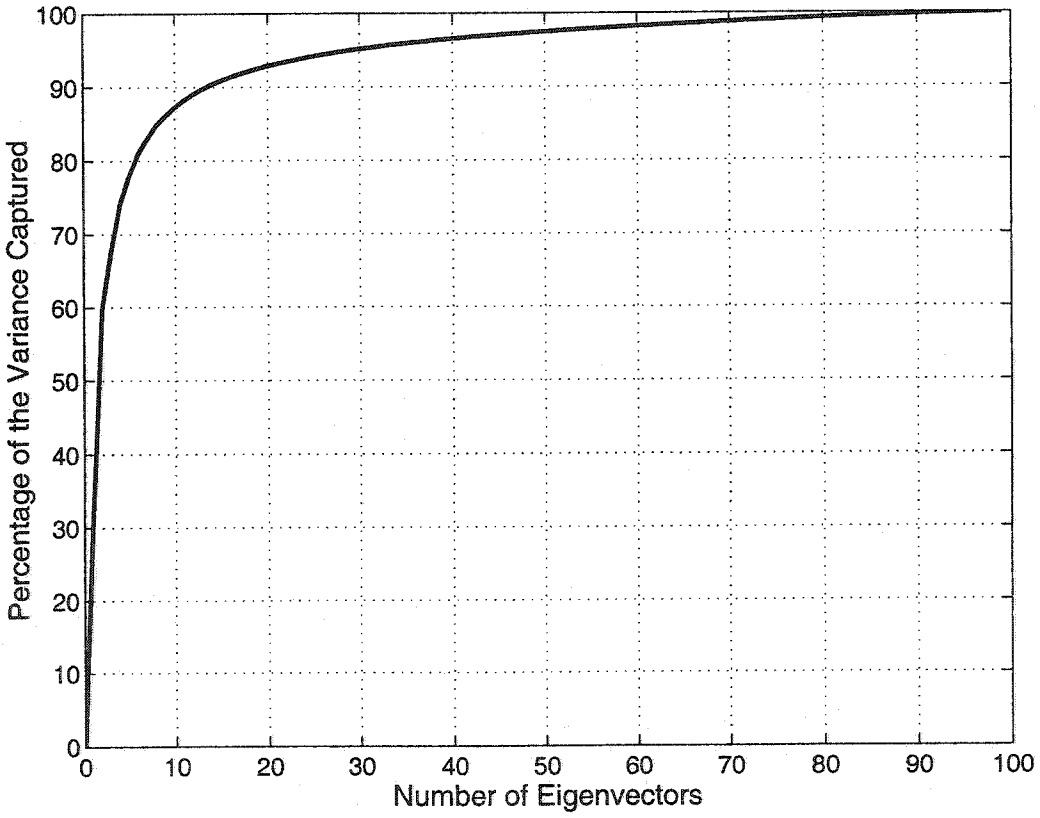


Figure 2.11: Percentage of the variance retained versus the number of eigenvectors used.

2.5.2 Image Representation and Reconstruction

PCA is not only a useful tool for performing image classification but also for providing a highly compressed representation of an image. Eigenspace techniques provide minimum reconstruction error for a given number of basis vectors and have been explored in [50] for image compression. This is a useful tool for tele-robotics applications, where PCA could simultaneously be employed to produce efficient image representations suitable for transmitting visual feedback of a robot operating at a remote worksite. Visual feedback can be used to enable a user to monitor equipment from a distance in situations where it is not practical or safe for a human to directly operate within a work-site. It would be computationally efficient and elegant if a single vision algorithm could be employed to perform both position determination and image compression. This section describes how image eigenspace methods can be simultaneously employed for video-rate position determination *and* for remote visual monitoring [57].

The theory of using PCA for image compression proceeds from the theory already discussed in section 2.5.1. An n dimensional pattern vector $\mathbf{y}$ can be perfectly reconstructed using:

$$\mathbf{y} = \sum_{i=1}^{n} x_i \mathbf{u}_i \tag{2.21}$$

where $\mathbf{u}_i$ is the i^{th} basis eigenvector and x_i is the i^{th} coefficient. If instead we use a subset of k vectors where $k < n$ the resulting approximation is given by:

$$\hat{\mathbf{y}} = \sum_{i=1}^{k} x_i \mathbf{u}_i \tag{2.22}$$

The error can be shown to be [14]:

$$\epsilon = E[(\mathbf{y} - \hat{\mathbf{y}})(\mathbf{y} - \hat{\mathbf{y}})^T] = \sum_{i=k+1}^{n} \lambda_i \tag{2.23}$$

Since all the images are known a-priori the image statistics are well known and an optimal subspace can be constructed to represent compressed images. It can be shown that the basis

vectors that minimize the reconstruction error are in fact the eigenvectors [14] and that the portion of the variance retained by mapping n dimensions down to k dimensions is:

$$r = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{n} \lambda_i} 100\% \tag{2.24}$$

A plot of r versus the number of dimensions k for the planar robot images is shown in Figure 2.11. From this plot, we see that with only 20 eigenvectors, almost 95% of the image variance is retained. Indeed, PCA provides the good image compression since it is optimal in the mean-square error sense and therefore provides better compression when compared to other techniques such as JPEG [50].

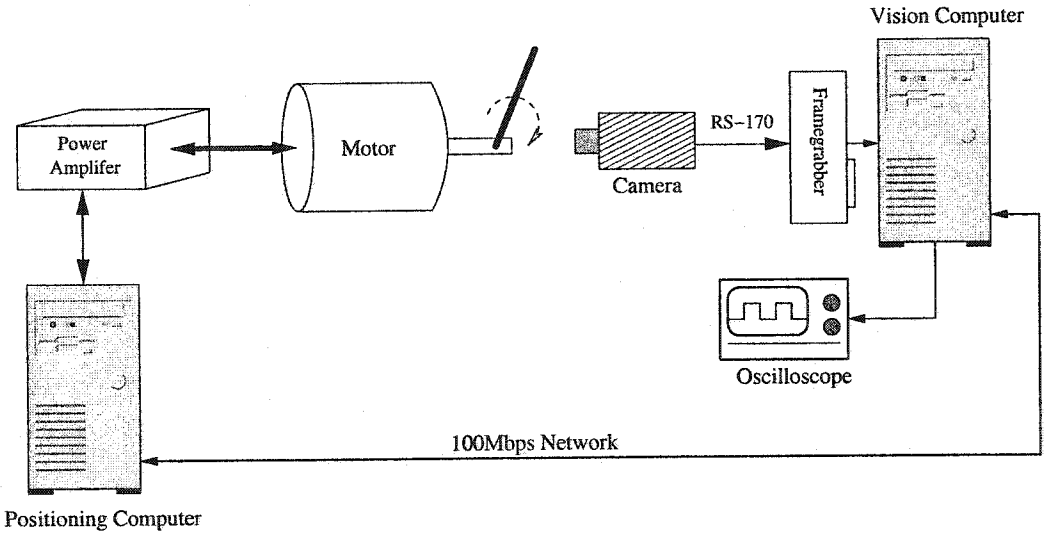


Figure 2.12: Diagram of experimental setup used to determine the performance of the vision system.

2.6 Vision System Performance

Before employing the vision system in a visual servo control loop the basic performance was first analyzed. The key specifications that were experimentally verified included position measurement accuracy, speed of computation, and latency. Speed and latency are

critical factors since they directly affect the stability of the position loop in a direct visual servo. The use of PCA for compact image representation was also evaluated in terms of the computational effort and the quality of the reconstructed images.

2.6.1 Experimental Setup

The suitability of the PCA for video rate position determination was evaluated using the experimental setup shown in Figure 2.12. The simple planar robot illustrated in Figure 2.5 provided a suitable testbed for performing angular position measurements. The setup was constructed using off-the-shelf components including a 550MHz AMD Athalon computer with a Matrox Meteor frame grabber card, a RS-170 grey-scale camera, a servo positioning system, and a digital storage oscilloscope. The oscilloscope was employed for timing measurements by observing parallel port pins which were toggled at key steps in the computation. The off-line calculations, including the determination of the eigenvectors and the eigenspace manifold, was performed using Matlab. The on-line code was written under the Linux operating system using the C programming language. Real-time image processing was ensured by using the POSIX.1b [29] real-time extensions to ensure memory page locking and scheduling priorities. The servo positioning computer and the vision computer were connected using a 100Mbps Ethernet.

Previously, it was determined that the vision system should process video *fields* rather than video *frames* in order to increase the sampling rate and decrease latency. It is not necessary to use an entire video field image since any of the points along the joint are an indication of its position. In order to select a suitable sub-image size, an experiment was performed in which the positions of a series of 50 test images were determined using various sub-image sizes centered around the motor shaft. The results are plotted in Figure 2.13. Clearly, for sub-image sizes less than roughly 10,000 pixels, the position measurement error is substantial. Sub-image sizes above 10,000 pixels are relatively fixed at an average

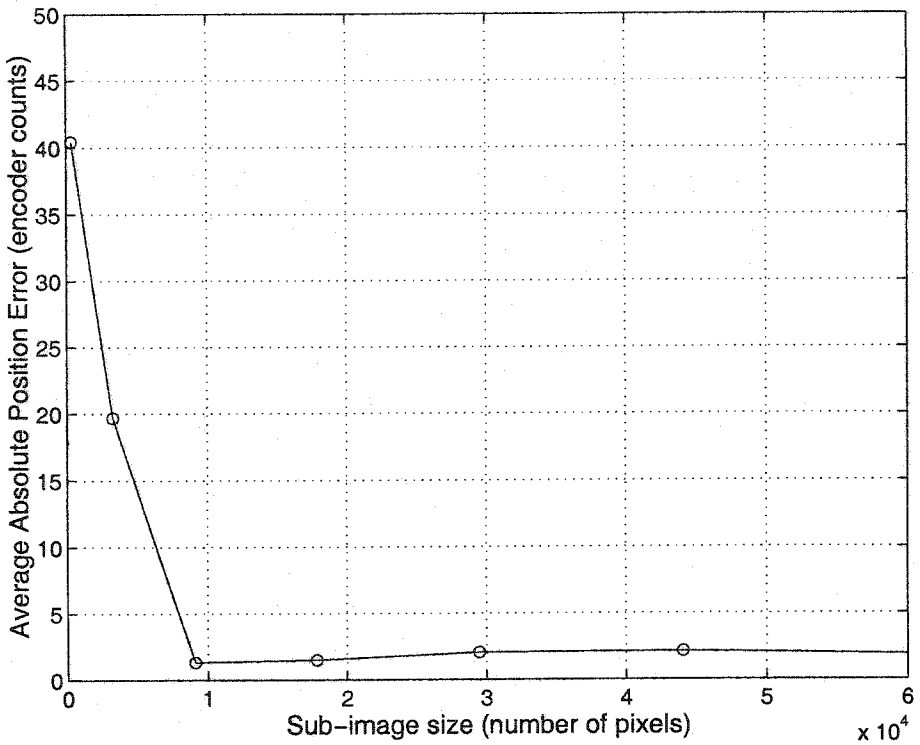


Figure 2.13: Average absolute position error vs. sub-image size.

position error of approximately 2 encoder counts. In order to provide a safe margin, a sub-image comprising 36,811 pixels (281 $\times$ 131 pixels) was used. This sub-image size is well above the "knee" at around 10,000 pixels, yet it is still small enough that it can be computed in real-time by the vision nodes (see section 2.6.4). In addition to increasing computation times, the use of larger sub-images is limited by memory size since the eigenvectors and the P matrix grow by $O(p)$ where p is the number of pixels. Some examples of the resulting video field sub-images are shown in Figure 2.14.

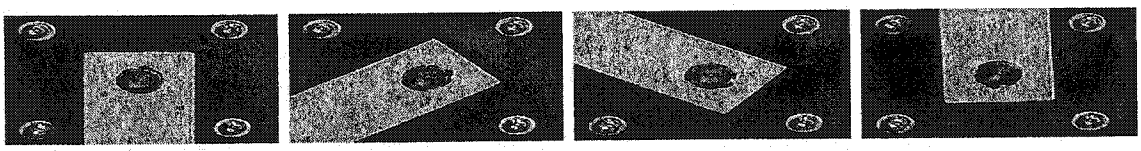


Figure 2.14: Examples of training sub-images.

2.6.2 The Training Phase

The experimental setup in Figure 2.12 was configured for learning by employing a rotary encoder with a resolution of 4000 counts per revolution to measure the actual position of the robot. A set of 101 images like the ones depicted in Figure 2.14 were captured as the joint was moved in 20 encoder count increments throughout a rotation of π radians. Next, the eigenspace was determined using the procedure outlined in section 2.5.1. The images were averaged to obtain the average image intensity of the image set. This average image intensity was subtracted from each of the 101 images in the image set and placed in a single matrix to form the P matrix.

Next, the covariance matrix and its corresponding eigenvalues and eigenvectors were determined. The resulting eigenvalues are plotted in Figure 2.7. It is apparent that the most significant eigenvalues correspond to roughly the first 5 to 10 eigenvectors. The five eigenvectors with the largest eigenvalues are the most significant and were used to construct the eigenspace. The images of the first four eigenvectors are shown in Figure 2.6.

The workspace manifold was created by using the eigenvectors to project each image in the set into eigenspace. A plot of the most significant 3 dimensions of the eigenspace manifold is shown in Figure 2.15. The workspace manifold exhibits a smoothly varying plot for the set of images. Each image sample corresponds to a discrete point in eigenspace. The number of points on the manifold was increased by interpolating between training points using a cubic spline to achieve a position quantization interval equivalent to 4000 points

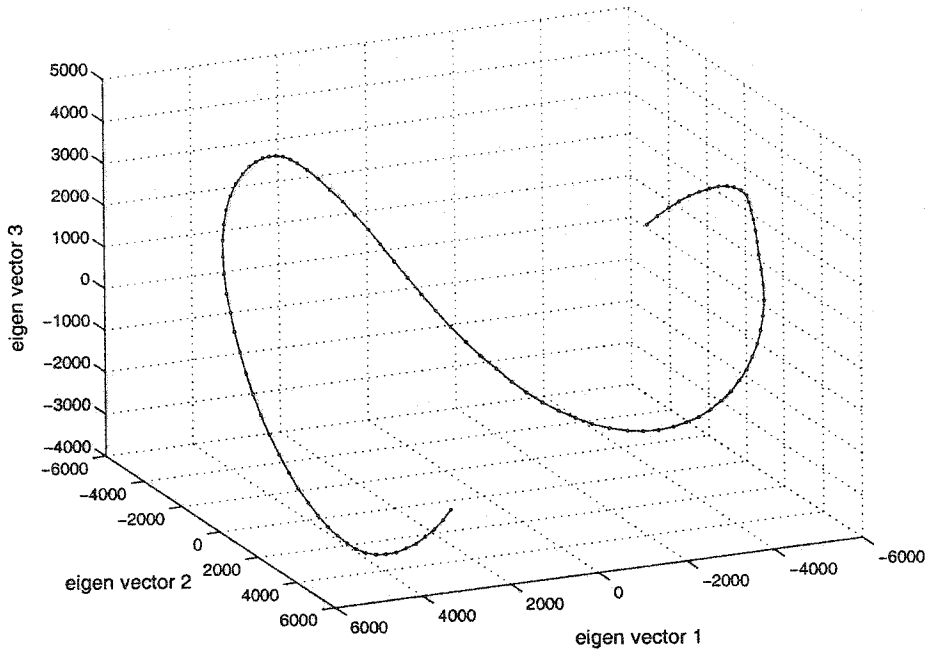


Figure 2.15: The parametric eigenspace manifold (plot of the first 3 dimensions).

per revolution. The final manifold points were stored in an array data structure with each array element corresponding to one encoder count such that the index into the manifold array corresponds to the actual position in units equivalent to the encoder counts used for training. Therefore, once the closest point on the manifold is located, the actual position can be determined from the corresponding index in the array.

The results of the experiments are summarized in the following sections. The two main performance metrics that were investigated were accuracy and speed of computation. The position accuracy was determined by creating a histogram of the measured position errors using 5 eigenvectors. Only even video fields were used in this experiment to decouple the position accuracy measurements using eigenspace methods from odd/even video field artifacts. The speed of computation was measured directly with a digital oscilloscope by toggling port pins at the start and end of each step in the eigenspace computations.

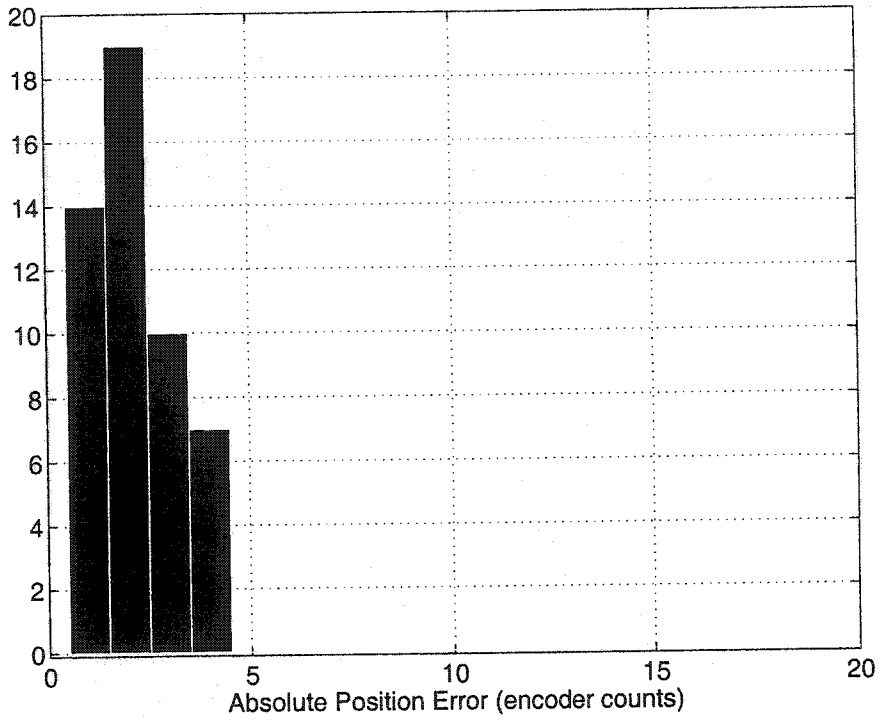


Figure 2.16: Position error histogram results.

2.6.3 Accuracy of Position Feedback

The accuracy of the eigenspace technique to measure the angular position of the joint was tested by taking various images with the joint held in various random positions. These random positions were distributed throughout the possible range of travel of the joint and included positions that were not used for the set of training images. All images were taken under constant illumination conditions and with no occlusions present. These images were then projected into eigenspace to find the point on the manifold with the closest sum-of-squared-difference (SSD). The position estimates were subtracted from the actual position (measured using a 4000 count/revolution rotary encoder) to determine the position error for each of the various test images that were used. A histogram of the absolute position errors using 50 test images was plotted and is shown in Figure 2.16. The experimental results

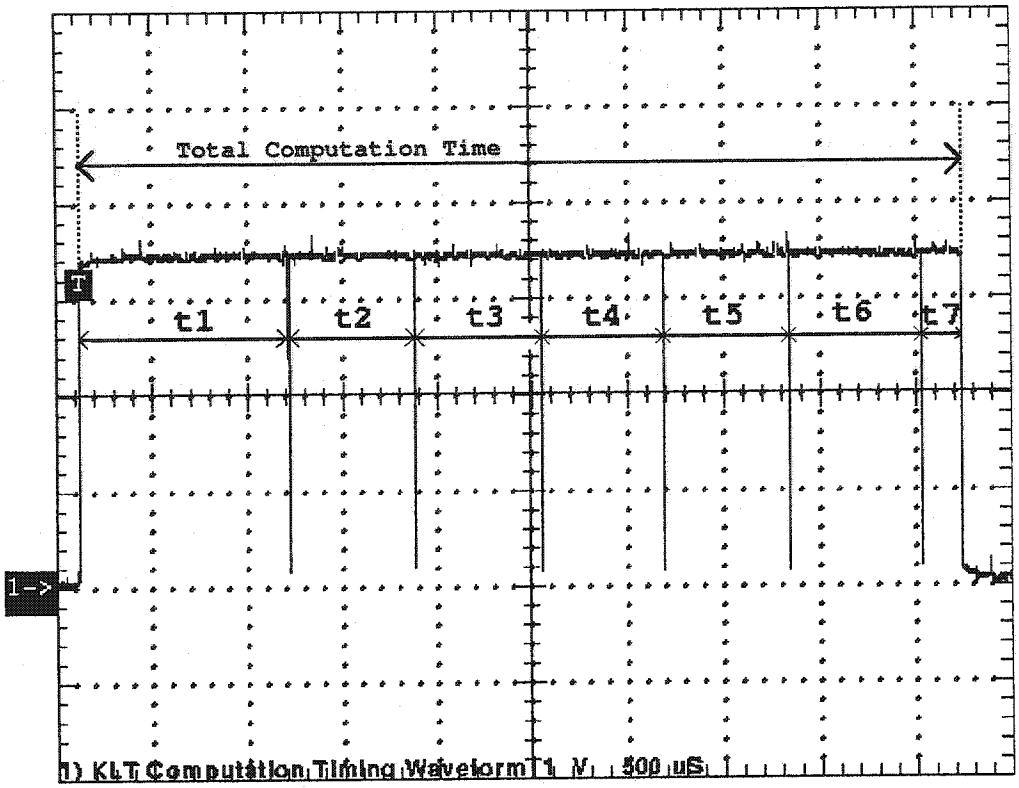


Figure 2.17: Digital Storage Oscilloscope plots of actual computation time for projecting an image into eigenspace running on a 550MHz AMD Athalon processor.

Label	*Computation*	*Time*
t_1	buffer copy and subtract average image	$1.08ms$
t_2	compute eigenvector 1 coefficient	$660\mu s$
t_3	compute eigenvector 2 coefficient	$680\mu s$
t_4	compute eigenvector 3 coefficient	$640\mu s$
t_5	compute eigenvector 4 coefficient	$660\mu s$
t_6	compute eigenvector 5 coefficient	$680\mu s$
t_7	SSD minima search	$220\mu s$
t_c	Total computation time	$4.62ms$

Table 2.1: Times for each step in the computation.

show a position accuracy for every sample of less than 5 encoder counts (equivalent to less than 0.45^o of a rotation) using 5 eigenvectors. Experimentation showed that increasing the number of eigenvectors beyond five yielded little improvement in accuracy. This is to be expected since the eigenvalues in Figure 2.7 diminish very rapidly beyond the first few eigenvectors. Some of the sources of position error include CCD noise, A/D noise, finite wordlength effects, and interpolation errors in the manifold.

2.6.4 Speed of Computation in the Vision Nodes

The projection of an image into eigenspace is a simple computation which is one of its advantages. However, the number of multiply-accumulate operations required per eigenvector is $O(p)$ where p is the number of pixels in an image. If there are k eigenvectors, the total number of operations is $O(kp)$. This can result in substantial computational times for large images with numerous eigenvectors. For a 281×121 pixel sub-image, the number of required multiply-accumulate operations is 34001 per eigenvector. Modern processors with advanced arithmetic units and super-scalar architectures have made arithmetic instructions extremely fast and many provide single cycle multiply-accumulate instructions. However, slow system memory combined with poor cache-hit ratios can result in significant bottlenecks for getting image data and eigenvectors in and out of the processor.

The computations were implemented in C and ran on a 550MHz AMD Athalon processor with a 100MHz memory bus. Timing information was measured using a digital storage oscilloscope on port pins which were toggled by the processor at the start and finish of each computational step. The oscilloscope plots and timing measurements for each computational step are shown in Figure 2.17 and summarized in Table 2.1. These results show a detailed breakdown of the various computational steps that contribute to the overall computation time. The position is determined from a 281×121 pixel raw intensity image using five eigenvectors in approximately $4.62ms$ after the image is received by the frame

grabber. The initial buffer copy and average image subtraction take about $1.08ms$ and each eigenvector coefficient computation takes about $670\mu s$. The eigenvector computations are performed sequentially, but since they are independent calculations, they could be computed in parallel to further improve the computation time. The last step of the computation performs the SSD minima search and takes about $220\mu s$. The computation time is more than adequate to ensure that the computations can be performed at video field rates.

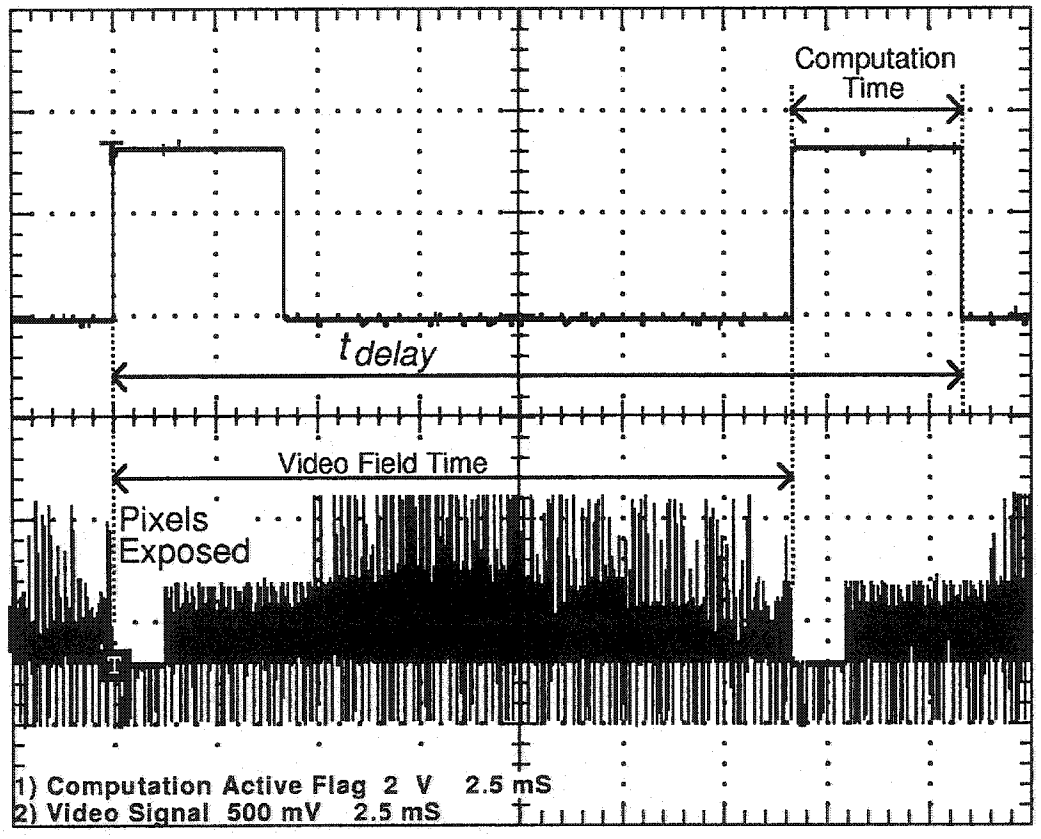


Figure 2.18: Digital Storage Oscilloscope measurement showing vision feedback latency.

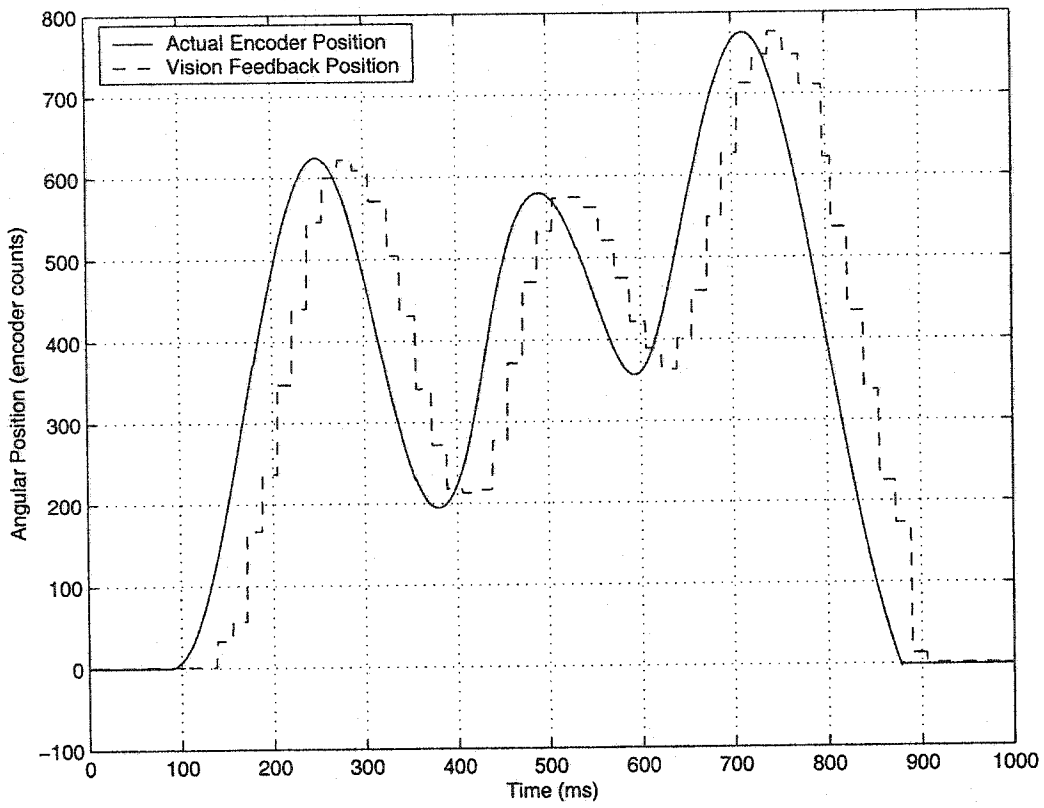


Figure 2.19: Tracking performance showing actual position along with position from vision feedback during a move.

2.6.5 Visual Feedback Latency Due to Transport Delays

When an image is captured, it must be first transmitted from the camera to the framegrabber. Once the framegrabber has received an image, the computations can proceed. Therefore, the total latency t_{delay} from the time the pixels are exposed to the time the position information is determined is as follows:

$$t_{delay} = t_v + t_c \tag{2.25}$$

where t_v is the RS-170 video field transmission time and t_c is the total eigenspace computation time. For RS-170 video signals, the video field transmission time t_v is approximately

$16.67ms$. From section 2.6.4, the total computation time t_c was found to be approximately $4.62ms$. Therefore, using equation 2.25, the total latency t_{delay} is approximately $21ms$.

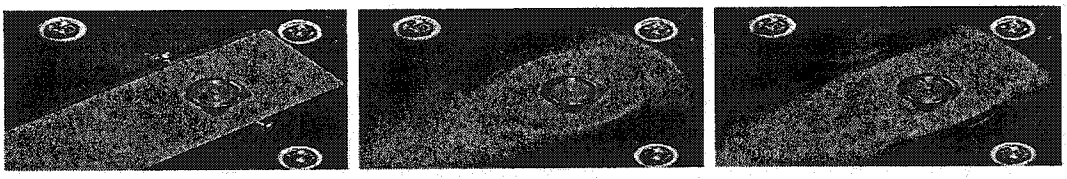
This is illustrated by the waveforms in Figure 2.18, where the latency from the time the pixels are exposed to the time the computations are complete is approximately $21ms$. Most of the delay (about $16.67ms$) is contributed by the serial transmission of pixels in the video signal shown on channel 2 in Figure 2.18. Thus, the computation time is only a small portion of the overall delay in the vision feedback. Figure 2.19 illustrates how the vision feedback tracks the actual position of the robot during a move. The transport delay t_{delay} in the vision feedback is readily apparent along with the discrete sample-and-hold operation of the vision system.

The significant latency in the vision system will cause it to be unstable when used in a closed position loop in direct visual servoing systems. However, the computation time required to project images into eigenspace is constant regardless of image content. This fact will be exploited in Chapter 3 where the image computation delay will be modelled in a Kalman filter by fixed unit delays. The Kalman filter provides a state predictor to reduce the effects of the transport delay in the vision system.

2.6.6 Image Reconstruction Performance

If PCA is being used in a tele-robotic application it can also be used to provide a compact subspace representation of an image as described in section 2.5.2. However, the application of PCA for image reconstruction is quite different from image interpretation. The number of eigenvectors required for accurate image position determination can differ greatly from the number required for reasonable image reconstruction.

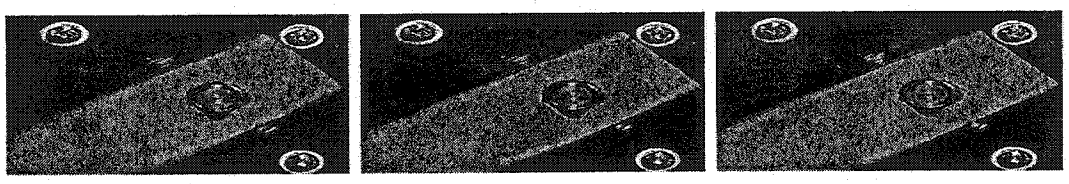
Figure 2.7 shows a plot of the resulting eigenvalues illustrating that there are relatively few significant eigenvectors. Consequently, only the first five eigenvectors were used for position determination. However, it was found that many more eigenvectors were required



(a) Original image.

(b) Reconstructed image using 5 eigenvectors.

(c) Reconstructed image using 10 eigenvectors.



(d) Reconstructed image using 20 eigenvectors.

(e) Reconstructed image using 30 eigenvectors.

(f) Reconstructed image using 40 eigenvectors.

Figure 2.20: Original image and reconstructed images using different numbers of eigenvectors.

to reconstruct an image of reasonable quality. Figure 2.20 illustrates the image quality as the number of eigenvectors are varied from 5 to 40. It was experimentally determined that there was sufficient time to compute coefficients for 25 eigenvectors and still maintain video field rates on the vision nodes. From Figure 2.11 it was determined that with 25 eigenvectors ($k = 25$) roughly 94% of the image variance would be captured.

The timing information shown in Figure 2.21 was measured using the digital storage oscilloscope on port pins which were toggled by the processor at the start and finish of each computational step. The initial buffer copy of the sub-image and average image subtraction occur during t_1 and takes about $1.1ms$. The computation of the first five eigenvector coefficients occurs during t_2 and takes approximately $2.8ms$. The next step of the computation (t_3) takes 300μs and includes the manifold search and the transmission of the resulting position to the servo computer. In order to provide images of sufficient quality for remote

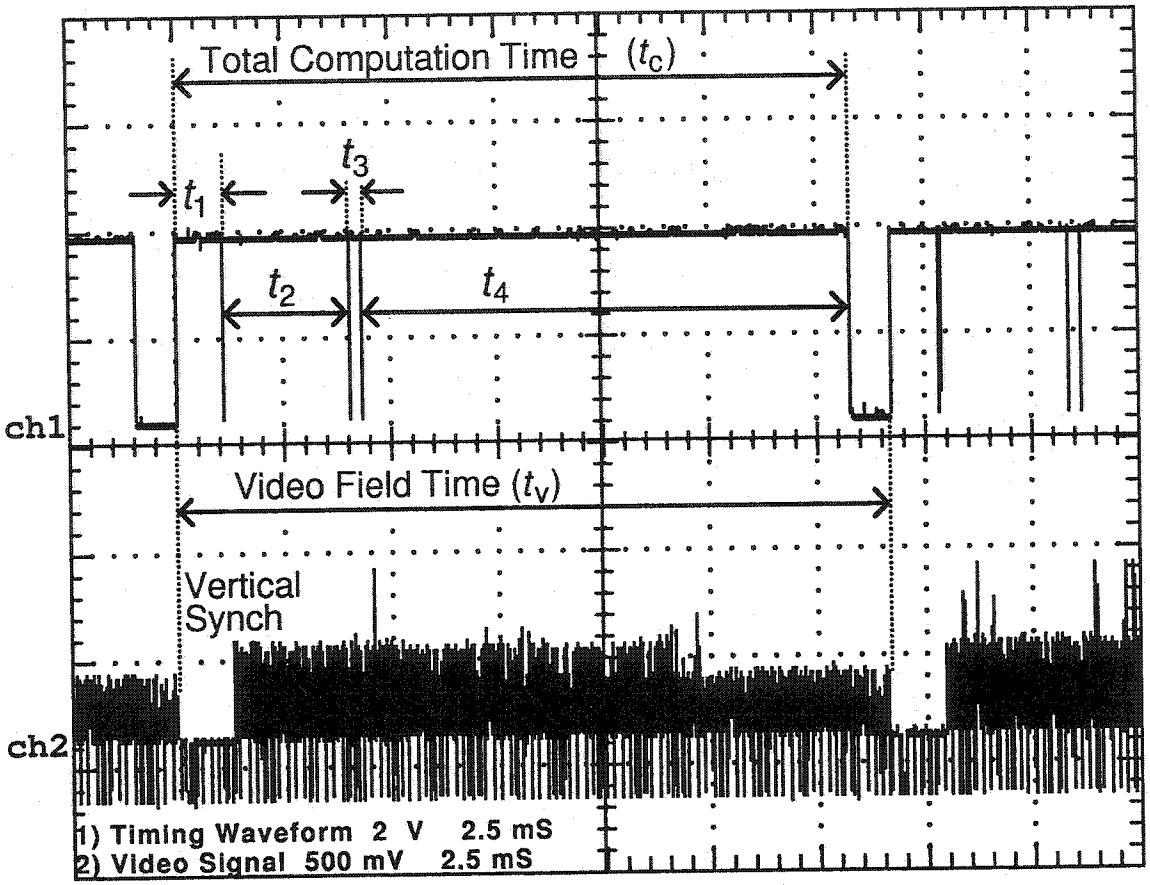


Figure 2.21: Digital Storage Oscilloscope plots of video signal and actual eigenspace computation times.

Label	*Computation*	*Time*
t_1	Buffer copy and subtract average image	$1.1ms$
t_2	Compute eigenvectors 1-5	$2.8ms$
t_3	Manifold search	$0.3ms$
t_4	Compute eigenvectors 6-25	$11.2ms$
t_c	Total computation time	$15.4ms$
t_v	Video Field time	$16.7ms$

Table 2.2: Table of actual eigenspace computation times.

monitoring an additional 20 eigenvector coefficients are then computed during time interval t_4. These are combined with the first five eigenvector coefficients and transmitted to the remote workstation. The net CPU utilization is the ratio of the total computation time (t_c) and the video field time (t_v):

$$\frac{t_c}{t_v} = \frac{15.4ms}{16.7ms} \approx 92\%$$

If PCA is to be used for image representation and assuming 25 eigenvectors are necessary, the computational requirements will increase significantly from 28% to 92% CPU utilization.

2.7 Occlusions

Occlusions are a reality in all real-world applications and must be addressed if the direct visual servoing system is to be robust. Occlusions will vary from camera to camera and will vary over time as work proceeds around a robot. Occlusions can occur when someone walks past a camera, when objects are moved, or when the pose of the robot itself obscures the view of one or more cameras. Measurement errors will increase dramatically when occlusions are introduced rendering eigenspace methods inefficacious unless special steps are taken to compensate. Occlusions present a particular challenge in eigenspace methods since the appearance of an image can change dramatically with occlusions. In [40] occlusions are dealt with by selecting various subsets of the image which are then applied to an algorithm composed of hypothesize, selection, and fitting steps which add computational complexity undesirable for real-time, video-rate, vision applications. In [39] small image patches select several features and are matched using a median statistic which is suitable for applications with only partial occlusions. In contrast to these approaches, this section describes a real-time error estimation approach which treats occlusions as sources of "noise".

Sources of Noise

Typical sources of measurement noise include CCD noise, A/D noise and finite wordlength effects. In addition, eigenspace methods have manifold errors introduced by noise in the original training images or errors due to finite wordlength effects and inaccurate interpolations between training points. Under ideal conditions, the effective noise variance from these sources should remain relatively constant. The position measurement error due to this noise was quantified in section 2.6.3 and are shown in Figure 2.16. This provides an initial estimate of the measurement noise variance which is intrinsic to the camera and system which should normally remain relatively stationary. However, in the presence of occlusions, the measurement error variance can change both suddenly and dramatically.

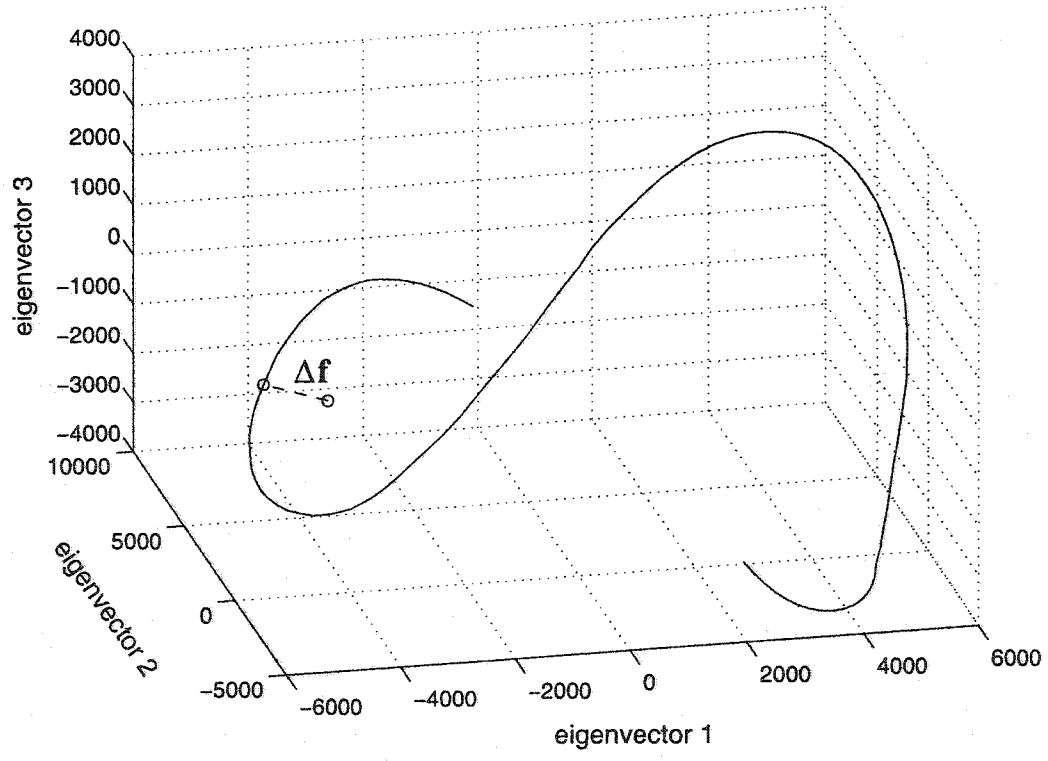


Figure 2.22: Occluded image point in eigenspace and its proximity to the manifold.

The position of a corresponding camera image is determined by projecting an image into eigenspace and finding the closest point on the manifold. The closest point in eigenspace is determined using a minimum Euclidean distance metric. An exact match with an image in the training set provides an exact match with a point on the manifold and yields a Euclidean distance of zero. As the appearance of an image diverges from the training set, its Euclidean distance increases since the distance in eigenspace is a measure of the similarity of appearance of images [45]. Under these conditions the image projected into the subspace can stray far from the parametric manifold making pattern matching unreliable. The magnitude of the Euclidean distance from the manifold can be used as an indicator of how far the sample image is from the set of learned images. As the Euclidean distance grows larger, the degree of uncertainty in the pattern match also grows. Extending this concept suggests there may be information about the degree of classification error and hence the measurement noise based on the size of the Euclidean distance. The process of projecting an image into eigenspace is found by:

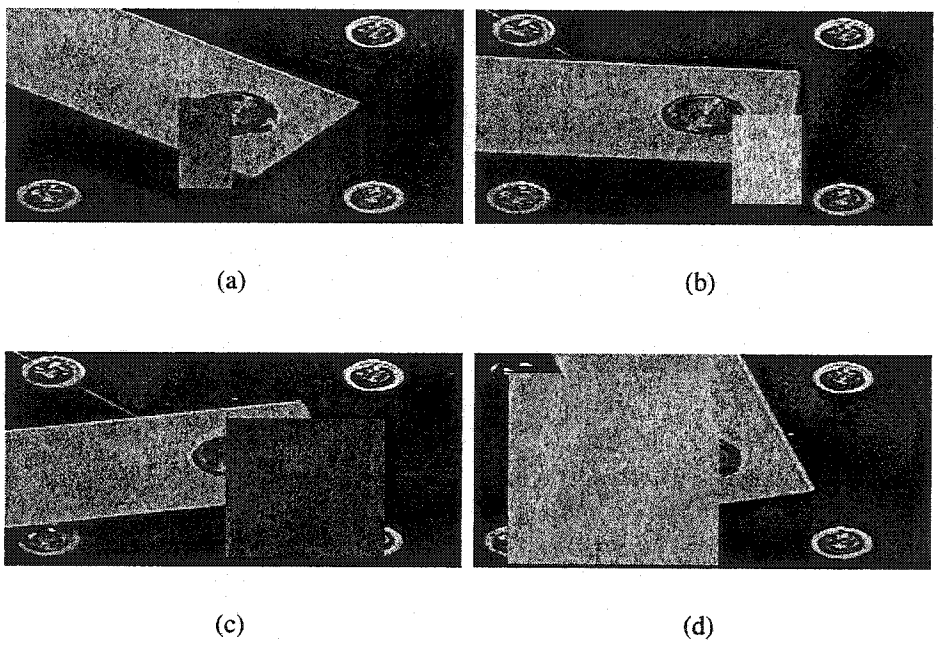
$$\mathbf{f} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_k \end{bmatrix} [\mathbf{i} - \mathbf{c}]^T \tag{2.26}$$

where $\mathbf{i}$ is the raw brightness image and $\mathbf{f}$ is the projected point in eigenspace using the matrix of k eigenvectors. Therefore, it follows that changes in an image will cause corresponding changes to the location of the projected point in subspace:

$$\Delta\mathbf{f} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_k \end{bmatrix} \Delta\mathbf{i}^T \tag{2.27}$$

This effect is illustrated in Figure 2.22 where an occluded image has its corresponding point in subspace stray from the manifold. The changes to a point in subspace $\Delta\mathbf{f}$ due to

corresponding image changes caused by specific occlusions cannot be generalized since it depends entirely on the k eigenvectors which, in turn, are unique to every application. The eigenvectors select features of maximum variance, hence the effect of occlusions will vary depending on how they coincide with important features.



(a) (b)

(c) (d)

Figure 2.23: Images with random rectangular occlusions.

2.7.1 Simulating the Effects of Random Occlusions

A simulation study was performed to determine whether any information about the position error statistics could be derived from Euclidean distance measurements. The experiment selected random images from the training set and random occlusions were introduced into each image. The occlusions were rectangular patches of random size and brightness that were placed at random locations in each image as shown in the examples in Figure 2.23. The brightness of each patch ranged randomly from brightness values of 50 to 200. The

width and height dimensions of the rectangular patches were independently and randomly varied from 2 pixels to the full size of the image. These occluded images were then projected into eigenspace and the minimum Euclidean distance and position error were computed. This process was then repeated 10,000 times. The results are shown in the scatter plot of position error vs. Euclidean distance in Figure 2.24. For Euclidean distances of less than approximately 2500, the majority of the position errors are clustered within a gradually increasing distinct error band. Beyond a Euclidean distance of roughly 2500, the corresponding position errors gradually become more scattered outside of this band.

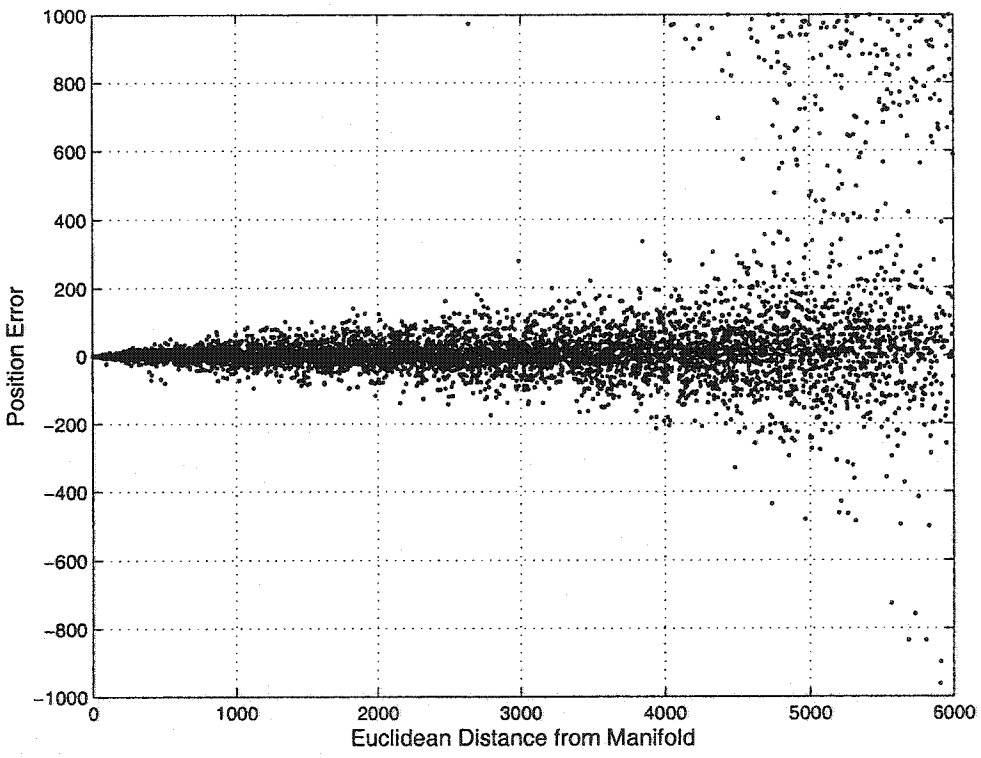


Figure 2.24: Scatter plot of position error vs. Euclidean distance in eigenspace.

The experiment was repeated but the square of the position errors were summed and placed in histogram bins organized into various Euclidean distance ranges. From the data in Figure 2.24, it is evident that the errors are generally centered around zero and appear to fall

within a certain envelop. The resulting histogram bins were then processed to determine the variance of the position errors as a function of Euclidean distance. The final results are plotted in Figure 2.25. It is apparent from these plots that, for a given range of Euclidean distances, *there is a statistical relationship between position error and Euclidean distance.* This relationship will be investigated further in the following section.

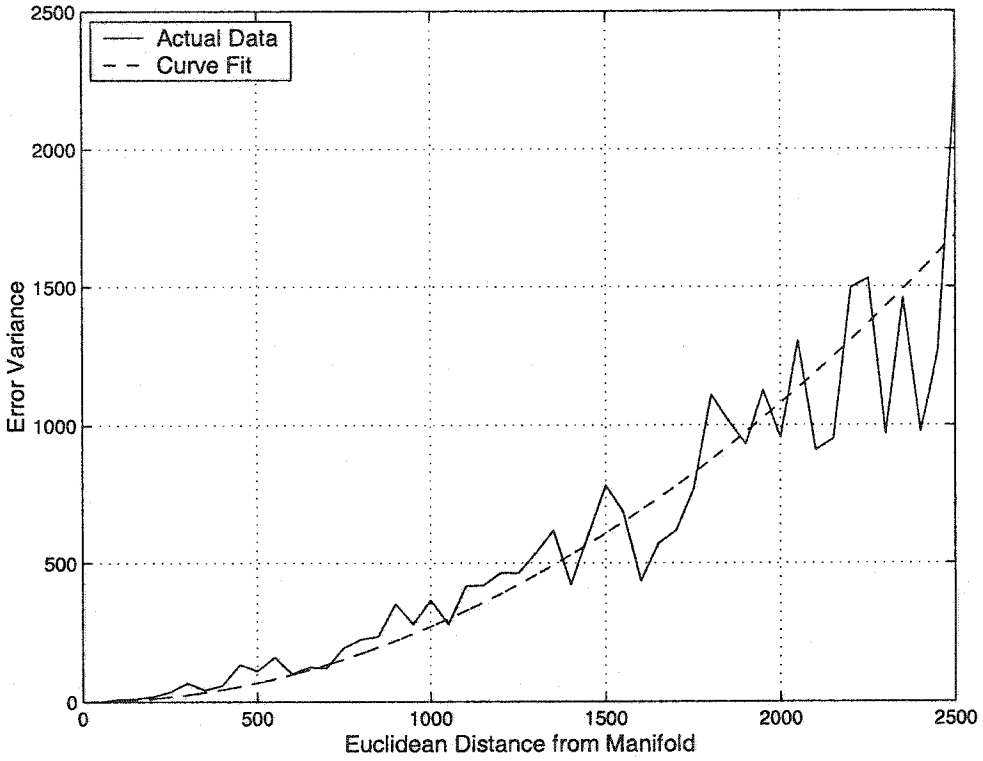


Figure 2.25: Position error variance vs. Euclidean distance in eigenspace.

2.7.2 Measurement Error Variance and Euclidean Distance

A plot of the position error variance vs. Euclidean distance in eigenspace is illustrated in Figure 2.25. A simple square law was used to determine an approximate curve to fit the

actual data as given by:

$$\sigma^2 \approx m \cdot d^2 \tag{2.28}$$

where σ^2 is the estimated position error variance, d is the Euclidean distance from the manifold, and m is a constant. For this particular application, m was found to be approximately equal to 0.00027. Conveniently, d^2 is equal to the sum-of-squared differences (SSD) which is the native value computed in the actual code running in the vision nodes. This enables the measurement error variance to be computed using a single multiplication as follows:

$$\sigma^2 \approx m \cdot SSD \tag{2.29}$$

This simple relationship enables the error variance to be rapidly calculated and is well-suited for use in time-critical real-time control loops. Equation 2.29 will begin to break down for Euclidean distances greater than approximately 2500 at which point the errors in Figure 2.24 appear to become more scattered. The effects of all other sources of noise such as CCD noise and A/D noise result in changes to single pixel values in a manner that is indistinguishable from single pixel occlusions. Hence there is no need to discriminate between occlusions and noise, they may be considered one in the same since they both cause projected points in eigenspace to stray from the manifold. Therefore the net measurement error variance due to occlusions and/or other noise can be generalized using Equation 2.29.

2.7.3 Sensor Fusion of Multiple Cameras with Occlusions

The Euclidean distances in each individual vision node will differ since occlusions may be present in the view of one camera but not in others. The Euclidean distance for each camera will also vary dynamically with time as varying degrees of noise and occlusions occur in each camera. If at least one camera remains free of occlusions, an accurate position estimate can be maintained. This suggests that using multiple cameras not only provides higher vision sample rates, but also has the advantage of being more robust in the presence of occlusions.

It is therefore possible to maintain good position estimates in the presence of partial occlusions by employing appropriate sensor fusion of the visual feedback from all the cameras. One possible approach to sensor fusion would be to reject feedback from vision nodes that report Euclidean distances above a predefined threshold. More complex approaches would seek to weight the feedback from each of the vision nodes based on estimates of the confidence in the measurement. This approach is desirable since feedback from a partially occluded camera still contains some useful information. The measurement error variance σ^2 given in equation 2.29 provides the key for adjusting the weighting of the measurements. The approach to sensor fusion that was used is based around the Kalman filter and is described in detail in section 3.4.6.

2.8 Summary

The experimental results show that the eigenspace methods are suitable for use in real-time position tracking applications without using specialized hardware. The speed of the eigenspace computations using off-the-shelf components and the accuracy of the results obtained indicate that the performance is adequate for use in robot position tracking or as a feedback sensor at video field rates. With 5 eigenvectors, a position accuracy of 0.45^o of a rotation or less can be achieved with a computation time of under $5ms$. By using additional eigenvectors, compact eigenspace image representations can also be used to provide an efficient means for transmitting images for remote monitoring in tele-robotic applications.

Finally, the effects of occlusions on position measurement error was simulated. The results indicate that a statistical relationship exists between Euclidean distance in eigenspace and the position measurement error variance. This discovery will be used with a Kalman filter in Chapter 3 to fuse data from multiple cameras and to provide robustness to occlusions.

Chapter 3

System Modelling for Design and Simulation

3.1 Introduction

This chapter describes the modelling and simulation work which was performed to describe the dynamics of the visual servoing system. The chapter begins by describing the models of each of the individual subsystems. Next, the subsystem models are arranged to provide a complete model of the planar robot plant. The plant model is analyzed and used to design a suitable compensator. A Kalman filter is developed to provide timely estimates of the state variables used in the feedback loop. Finally, the entire system is simulated to verify that stable direct visual servoing can be achieved.

The motivation behind detailed system modelling is threefold. First, the exercise of modelling the system improves the understanding of the system and the issues associated with controlling the system. Second, the modelling enables the system to be simulated and verified *before* it is implemented. Finally, the modelling of the planar robot plant produces the required state equations necessary to implement the Kalman filter predictor in section 3.4. In particular, the performance of the predictor is dependent on accurate and

detailed system models. These models are described in the following section.

3.2 System Modelling

The visual servoing system comprises various subsystems which were each modelled independently. The individual subsystem models include:

- Mechanical load model
- Motor model
- Power amplifier model
- Vision model

These subsystem models are each described in detail in the following sections.

3.2.1 Model of the Mechanical Load

The planar robot is a simple mechanical system consisting of a single fiberglass link coupled directly to the shaft of a servo motor. The use of a simple direct drive mechanism eliminates many of the non-linearities often associated with gears or chain drives such as backlash and dead-zones. However, direct drive systems experience higher load inertias that require more motor torque to accelerate and lead to higher inertia mismatches between the motor and load. Inertia mismatches should be minimized in servo drive systems because they reduce the maximum power transfer to the load and can lead to instability in the control loop. Despite these challenges, direct drive systems have been successfully employed in a wide variety of systems. The direct drive system is simple to model with the major mechanical load components consisting of inertia and viscous damping. Unless the robot joint is operating in a plane that is parallel with the ground, it also experiences a gravitational load that varies with angular position.

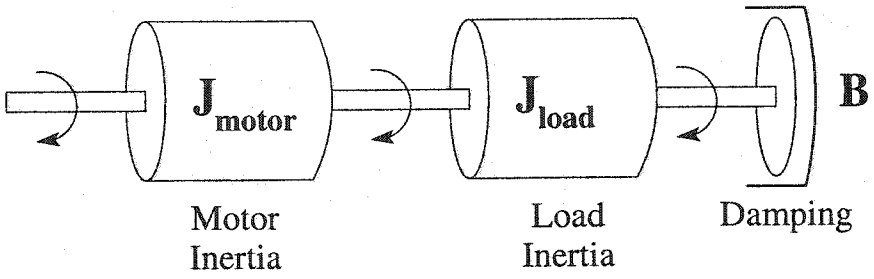


Figure 3.1: Mechanical load with inertia and damping elements.

Inertia and Damping

The servo motors used by the robot will drive a mechanical load which consists of inertia and damping elements as illustrated in Figure 3.1.

The rotational inertia J_{load} is given by [23]:

$$J_{load} = \int r^2 \, dm \tag{3.1}$$

where dm is a particle of mass and r is the distance of the particle from the axis of rotation. The inertia of the single arm of length L on the planar robot can be approximated by a thin rod with the same mass and length. The expression for the mass in terms of the distance dr from the axis of rotation is given by:

$$dm = \frac{M}{L} \, dr \tag{3.2}$$

where M is the total mass of the robot joint and L is the total length of the joint. Substituting equation 3.2 into equation 3.1 to integrate over the full length of the robot arm yields:

$$J_{load} = \frac{M}{L} \int_0^L r^2 \, dr \tag{3.3}$$

Solving the integral in equation 3.3 yields:

$$J_{load} = \frac{M}{L} \frac{r^3}{3} \Big|_0^L \tag{3.4}$$

The result is an expression for the equivalent load inertia in terms of the total mass and length given by:

$$J_{load} = \frac{ML^2}{3} \tag{3.5}$$

The total inertia J is the sum of all the inertias:

$$J = J_{motor} + J_{load} \tag{3.6}$$

where J_{motor} is the rotor inertia of the motor as specified in the manufacturer's datasheet and J_{load} is the inertia of the robot joint using equation 3.5.

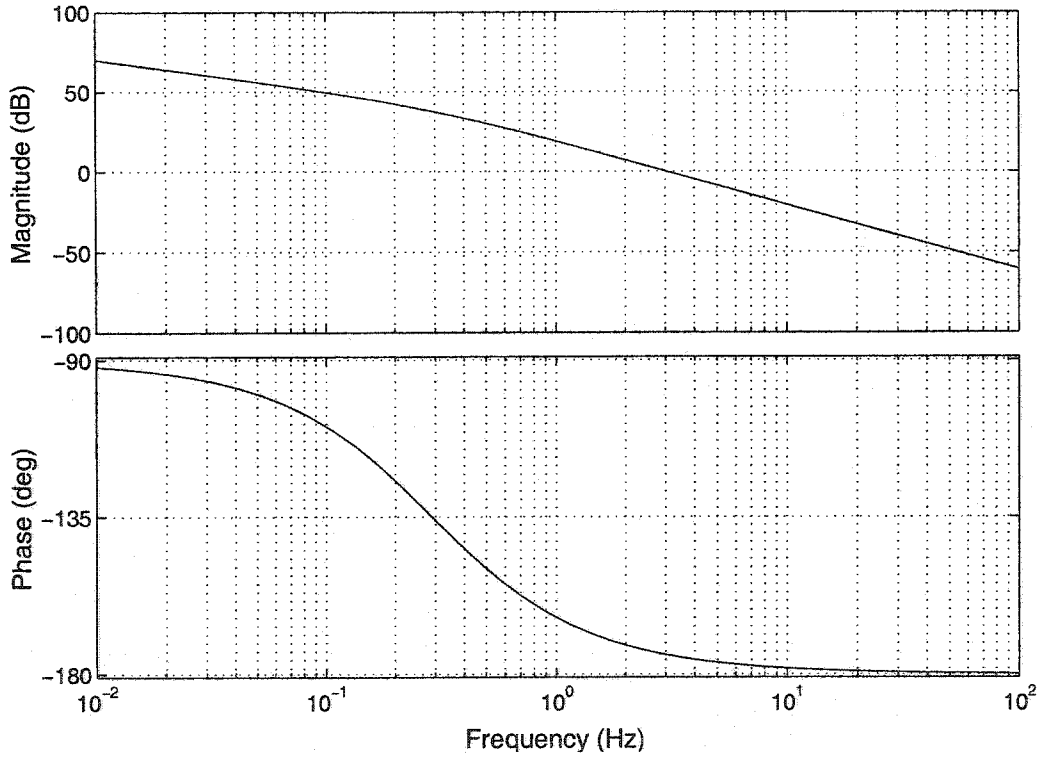


Figure 3.2: Bode plot of the mechanical transfer function.

In additional to the inertial load there is also some viscous damping and friction arising from the motor bearings and windage losses. The total mechanical load torque is due to the

combination of the inertia and damping elements [23]:

$$T = J\frac{d^2\theta}{dt^2} + B\frac{d\theta}{dt} \tag{3.7}$$

Next, this equation can be converted to the Laplace domain:

$$T(s) = Js^2\theta(s) + Bs\theta(s) \tag{3.8}$$

Rearranging equation 3.8 to solve for angular position yields:

$$\theta(s) = \frac{1}{Js^2 + Bs}T(s) \tag{3.9}$$

The mechanical transfer function $G(s)$ can be obtained by rearranging equation 3.9 to express the equation in terms of the position output over the torque input as follows:

$$G(s) = \frac{\theta(s)}{T(s)} = \frac{1}{Js^2 + Bs} \tag{3.10}$$

This is a second order transfer function with two distinct poles. One pole is located at zero and the other pole is dependent on the inertia and damping coefficients and is located at $\frac{B}{J}$ radians. Figure 3.2 shows the Bode plot of the transfer function in equation 3.10 using inertia and damping values based on the planar robot. Clearly, the system behaves like a double integrator at higher frequencies with a phase shift approaching 180^o near the 0dB crossover point. The low phase margin suggests that this system will require a phase-lead compensator design to provide stable operation in a closed-loop configuration.

Gravity

Gravity also applies forces on the joints of a robot. The effects of gravity vary with the pose of the robot. The effects of gravity are greatest when the robot arm is horizontally positioned and zero when the arm is vertically aligned. Between these two extremes, the single joint robot experiences gravity as a continuously varying torque function proportional to the cosine of the joint angle.

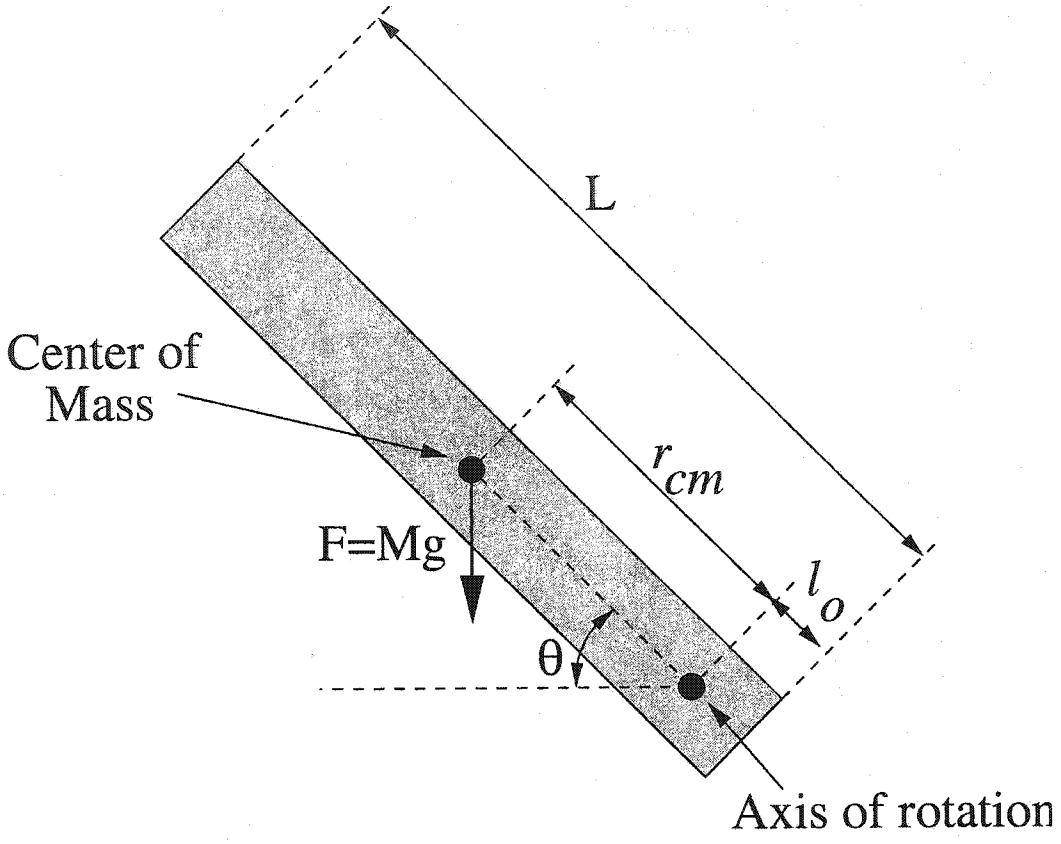


Figure 3.3: Force of gravity acting on the single joint robot.

The torque τ resulting from the force of gravity acting on a point mass m_i in the robot arm is given by [23]:

$$\tau = r \times F \tag{3.11}$$

where F is the force of gravity on the mass and r is the distance from the fulcrum. Therefore, the torque τ_i acting on a point mass can be expressed as:

$$\tau_i = m_i g r_i \cos(\theta) \tag{3.12}$$

where g is the acceleration due to gravity and θ is the angle shown in Figure 3.3. The total torque due to gravity $T_{gravity}$ is given by the summation of all the torques due to all the

particles in the robot arm:

$$T_{gravity} = \sum \tau_i = g\cos(\theta)\sum m_i r_i \tag{3.13}$$

The sum of all the particles and distances can be replaced by the center of mass as follows:

$$r_{cm}M = \sum m_i r_i \tag{3.14}$$

where M is the total mass of the robot arm and where r_{cm} is the distance from the axis of rotation to the center of mass as illustrated in Figure 3.3. Equation 3.14 can be substituted into equation 3.13 to yield the torque due to gravity:

$$T_{gravity}(\theta) = Mgr_{cm}\cos(\theta) \tag{3.15}$$

The net torque delivered to the load T_{load} is the applied motor torque T_e minus the torque due to gravity:

$$T_{load} = T_e - T_{gravity}(\theta) \tag{3.16}$$

The resulting model for the plant, including transfer function from equation 3.10 and the gravitational load from equation 3.15, is illustrated in Figure 3.4.

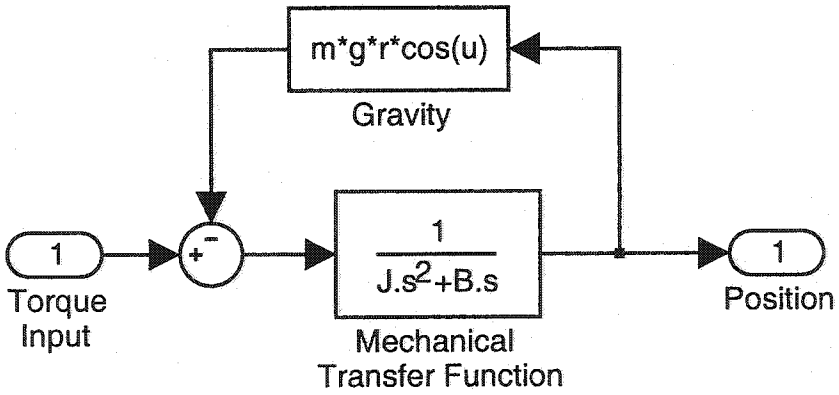


Figure 3.4: Mechanical model of the robot.

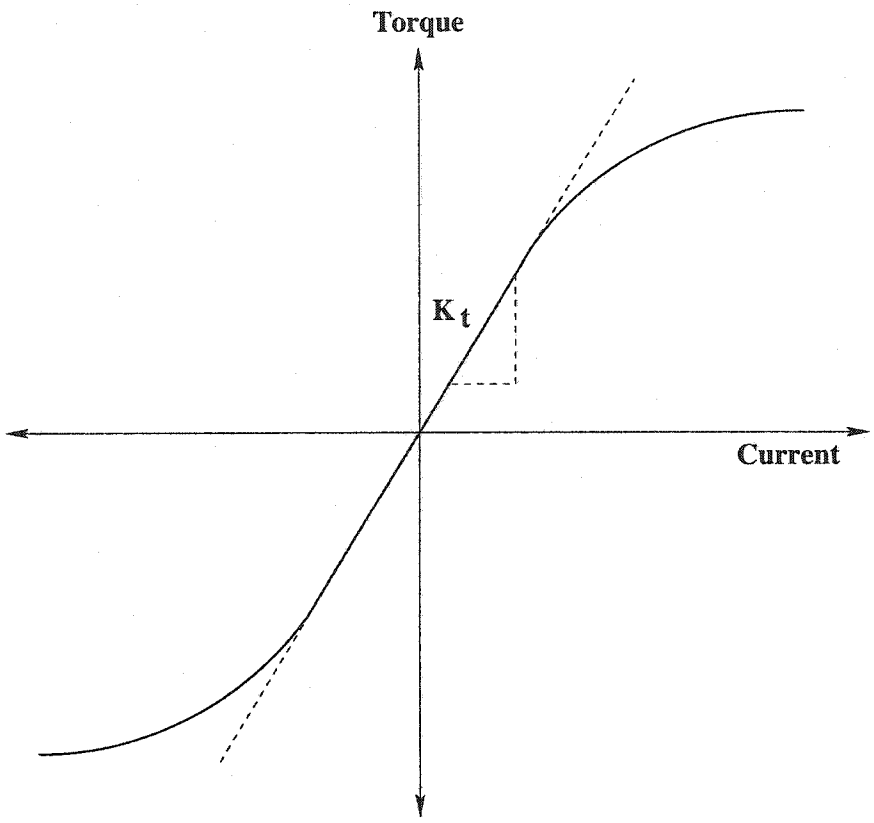


Figure 3.5: Motor transfer function.

3.2.2 The Motor Model

The servo motor that was selected for the planar robot was a brushless DC motor. Brushless DC motors have a permanent magnet rotor and stator windings that rely on hall-effect sensors to provide feedback for phase commutation. A brushless DC motor was selected over a brushed DC motor because it provides higher torque densities and it does not suffer from brush losses and wear. The general relationship between motor current and motor torque is shown in the conceptualized diagram in Figure 3.5. There is a linear relationship between motor current and motor torque up until a certain point at which magnetic saturation occurs. In addition to magnetic saturation, there are various internal losses that reduce

the torque output of the motor. These losses include armature reaction, hysteresis losses, eddy current losses, and stray load losses [52].

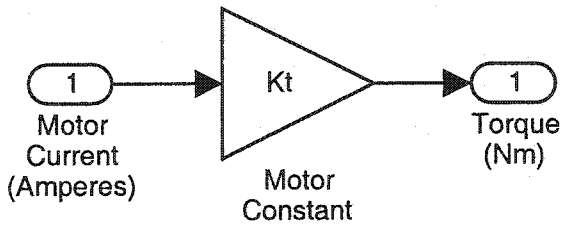


Figure 3.6: The motor model.

Assuming a motor is operated within its ratings, a reasonable estimate of the motor torque described in [52] as:

$$T_e = K_t I_{motor} \tag{3.17}$$

where T_e is the torque produced by the motor in Newton-meters, K_t is the torque constant, and I_{motor} is the motor current in Amperes. The torque constant K_t is a lumped parameter which is determined by various motor design parameters such as the number of pole pairs and the number of stator windings. The resulting motor model is depicted in Figure 3.6

3.2.3 Power Amplifier Model

The power amplifier controls the motor current to correspond with the current setpoint input. The ideal model of the power amplifier is that of a controlled current source. A more realistic model of the power amplifier will take into consideration shortcomings such as limited current handling capability, finite amplifier bandwidth, and output current noise. A practical amplifier has a PWM current control loop that cannot respond instantaneously to input changes and has a finite bandwidth. The inductance of the motor also limits the rate of change in the current. The finite bandwidth of the amplifier can be accounted for in the

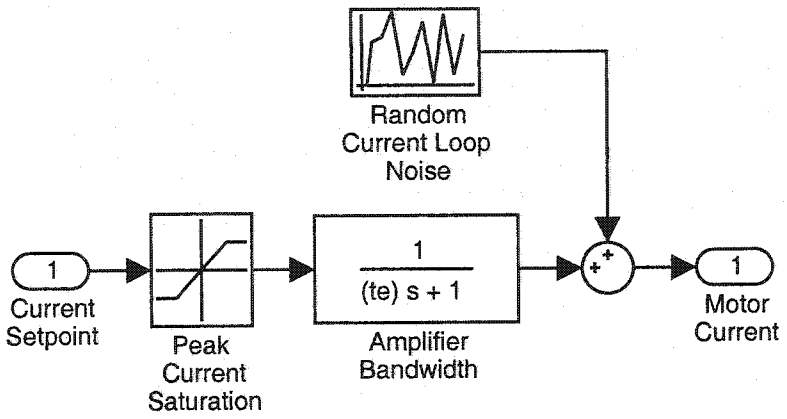


Figure 3.7: Power Amplifier Model.

model by introducing a single pole transfer function:

$$G_{amplifier}(s) = \frac{1}{\tau_e s + 1} \tag{3.18}$$

where τ_e is a dominant time constant that defines the bandwidth of the power amplifier and is normally specified by the manufacturer. The current rating of the power amplifier also limits the maximum current that can be handled by the amplifier. This can be modelled by a non-linear saturation block which limits the current setpoint from exceeding the maximum rated current. Finally, the power amplifier also introduces noise in the output current. This noise includes PWM switching noise, power supply regulation noise, current sensor noise, and commutation noise. A Gaussian noise source is used to approximately model the sum of all the noise sources. The resulting block diagram model of the power amplifier is illustrated in Figure 3.7.

3.2.4 The Vision Model

The vision subsystem model is shown in Figure 3.8. The input to the vision model is the actual robot position and the output is the measured position. The output approximates the

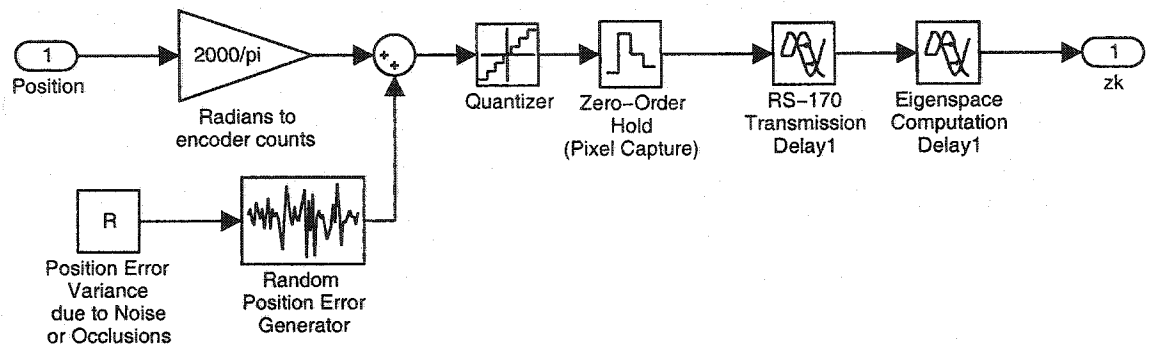


Figure 3.8: Block diagram of the vision subsystem model.

operation of the actual vision system by simulating the pixel capture using a zero-order hold at a sample rate f_s (see equation 2.1). A quantizer block also serves to simulate the position quantization interval used in the parametric manifold. The model also includes representative noise and delays.

A block is used to set the measurement error variance $\mathbf{R}$ for the simulation. This block controls the variance of a Gaussian noise source used to introduce position errors in the model. Thus the value for $\mathbf{R}$ is an adjustable parameter in the simulation. An initial estimate for $\mathbf{R}$ was obtained from the error histogram results in Figure 2.16.

The position measurement z_k also passes through a transport delay. The transport delays in the vision system were set using experimentally determined timing information from section 2.6.5. The total delay t_{delay} from the time the pixels are exposed to the time the position information is determined was determined to be approximately $21ms$. The eigenspace computations require approximately $4.6ms$ once the image is received by the frame-grabber. Most of the delay (about $16.67ms$) is due to the transmission time of the video signal. The delays were incorporated into the vision model by using two separate transport delays representing the video transmission time and the eigenspace computation time.

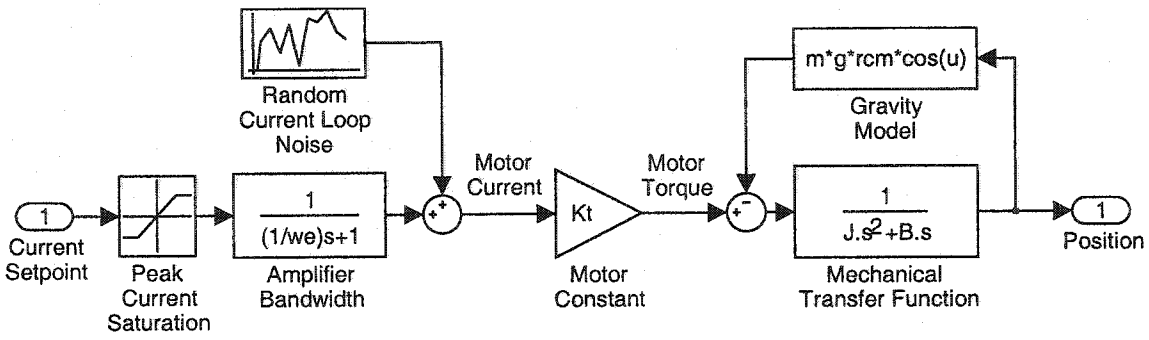


Figure 3.9: Plant model of the planar robot system.

3.3 Position Loop Compensator

Once the model of the planar robot plant is established it can be used to evaluate an appropriate control strategy. Various compensator topologies were explored for stable position control of the robot arm. In order to simplify the initial compensator investigation, the complexities associated with the vision feedback were temporarily set aside and the states variables were assumed to be available. The techniques used to obtain the actual state estimates using a Kalman filter are described in a subsequent section. This approach allows different compensator schemes to be compared based on their relative merits, while deferring the issue of state estimation till later.

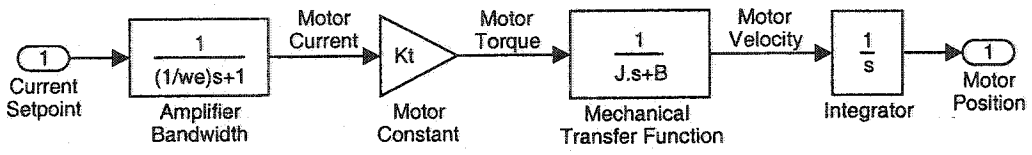


Figure 3.10: Linearized plant model of the planar robot system.

3.3.1 The Linearized Plant Model

A complete model of the plant including the amplifier, servo motor, and mechanical load is shown in Figure 3.9. The effects of noise can be set aside and the model can be linearized around an operating point to produce a plant suitable for classical analysis techniques. The effect of gravity is constant around any operating point, so it is ignored for the transient analysis. The current saturation block can be ignored by limiting the range of the input signals. The linearized plant model can be obtained by cascading the linear subsystem models defined in equations 3.18, 3.17, and 3.10 to form the transfer function:

$$G(s) = \frac{K_t}{s(\tau_e s + 1)(Js + B)} \tag{3.19}$$

The block diagram of this linearized plant model is illustrated in Figure 3.10. By substituting the parameter values for the planar robot into equation 3.19, the Bode plot can be generated and is shown in Figure 3.11. The three poles contribute to a total phase shift of 270^o and the uncompensated phase margin is approximately 11^o.

3.3.2 Compensator Design

In order to provide stable closed-loop position control for the system described by equation 3.19, the phase margin will need to be improved considerably. The first pole is located at zero and the second largest pole is located at $\frac{-B}{J}$ radians. The third pole is due to the amplifier bandwidth and is located at $\frac{-1}{\tau_e}$ radians. This pole will normally be at a sufficiently high frequency that it will not contribute significant phase lag and thus will have no effect on the stability analysis. Therefore, considering just the two dominant poles, the resulting second order plant is given by:

$$G(s) = \frac{K_t}{Js^2 + Bs} \tag{3.20}$$

The damping coefficient B is generally very small because it is primarily due only to the friction in the motor bearings. Therefore, the existing phase margin is tenuous since it very

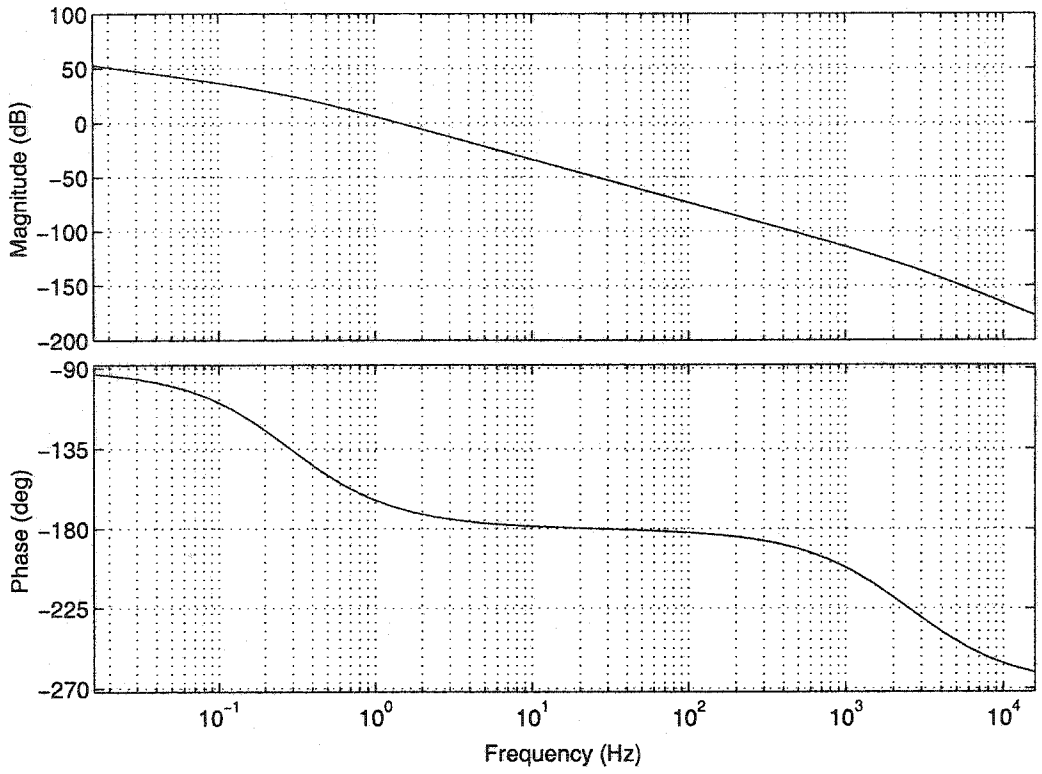


Figure 3.11: Bode plot of the linearized plant.

sensitive to changes in the uncontrolled damping coefficient. If B were of a reasonable magnitude, then one of the poles at zero could be shifted to a higher frequency, but adding mechanical damping to the system is not desirable. A suitable compensator will need to be developed.

As the damping coefficient approaches zero, the second pole slides to zero and the system approaches that of a double integrator. In the worst case, the damping coefficient is equal to zero and the open-loop transfer function becomes:

$$G(s) = \frac{K_t}{Js^2} \tag{3.21}$$

The Bode plot for this transfer function, using the planar robot parameters, is shown in

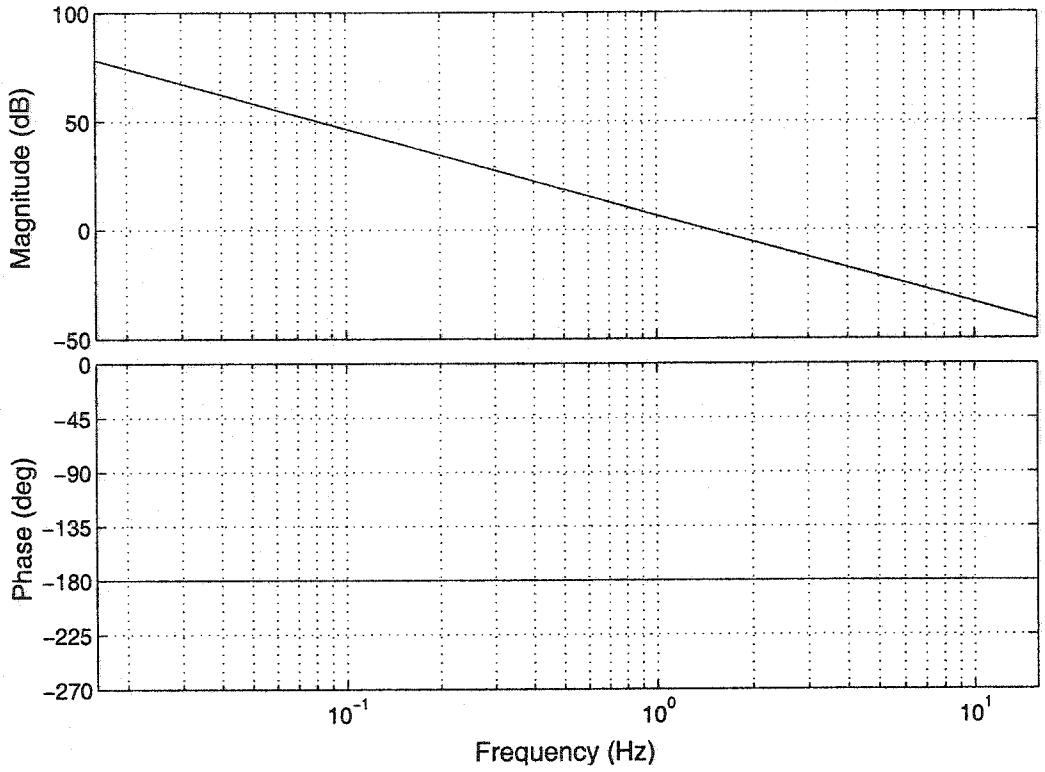


Figure 3.12: Bode plot of uncompensated plant with zero damping.

Figure 3.12. This system has double poles at zero creating a phase shift of 180° at all frequencies which leaves no phase margin. A simple proportional controller will not be adequate because it cannot provide any phase advance. Another possible option is to employ phase-lead compensation but phase-lead compensators are susceptible to noise and the amount of phase margin that can be added is limited. Alternatively, the use of an inner velocity loop was explored to improve the phase margin and it is described in the next section.

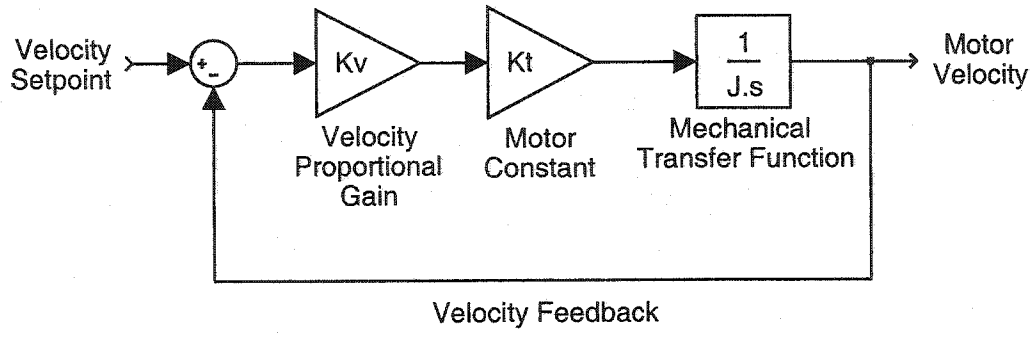


Figure 3.13: Diagram of the inner velocity loop.

3.3.3 Position and Velocity Control

A common approach to servo compensator design is to add "electronic damping" by adding an inner velocity loop within the position loop. The inner velocity loop increases stability by shifting up one of the poles at or near zero so that a reasonable phase margin can be maintained for closing the position loop. By controlling higher order state variables, such as velocity, the overall motion of the robot can also be made smoother.

An inner velocity loop is easily employed when tachometer feedback is available but this is not the case with a direct visual servo. Therefore, for the purposes of the compensator design, the observability of the velocity state variable will be assumed. The issue of extracting the velocity state variable from visual feedback will be addressed later in the chapter.

A velocity loop was formed around the simplified plant model in equation 3.21 and is illustrated in Figure 3.13. The closed-loop transfer function of the velocity loop is given by:

$$G_v(s) = \frac{K_v K_t}{Js + K_v K_t} \tag{3.22}$$

where K_v is the proportional gain for the velocity loop. The velocity loop is stable because there is only one system pole contributing at most a 90^o phase shift. Next, the closed

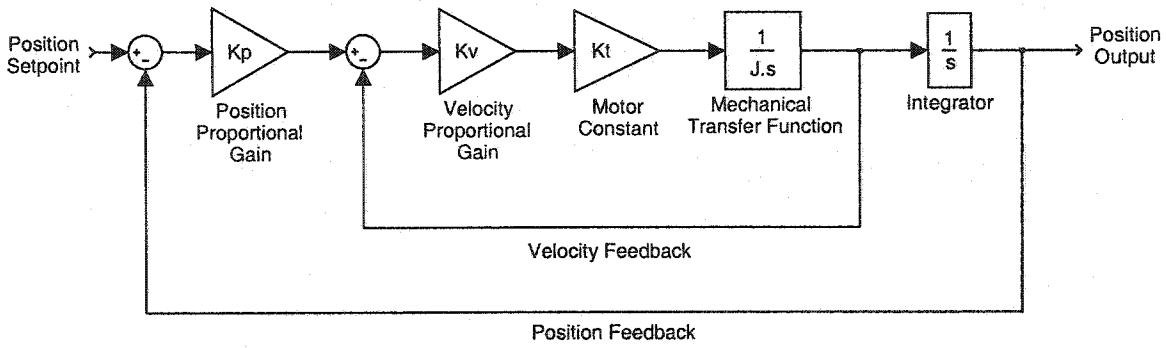


Figure 3.14: Diagram of the position controller with an inner velocity loop.

velocity loop is placed within the position loop as shown in Figure 3.14. The resulting open-loop position transfer function is given by:

$$G_{pv}(s) = \frac{K_p K_v K_t}{Js^2 + K_v K_t s} = \frac{\frac{K_p K_v K_t}{J}}{s(s + \frac{K_v K_t}{J})} \tag{3.23}$$

where K_p is the proportional gain for the position loop.

The physical effect of the $K_v K_t$ term is to provide *electronic damping* which performs the same function as mechanical damping would. The inner velocity loop has the effect of shifting one of the poles from zero up in frequency. As the pole approaches cross-over frequency it will increase the phase margin. The resulting system has one pole at zero and one at a frequency of $\frac{K_v K_t}{J}$ radians/second. The position of the pole is determined in part by K_t and J which are system parameters that cannot be controlled. The pole location also depends on the velocity gain K_v which can be tuned to adjust pole frequency and thereby control the resulting phase margin. However, the K_v term also appears as a gain in the numerator of the transfer function, thereby limiting the amount by which the gain can be increased. This is evident in the root locus plot, shown in Figure 3.15, when the gain is raised and the poles separate from the real-axis becoming underdamped. The K_v term must therefore be judiciously tuned to improve the phase margin without counteracting the improved stability with too much gain.

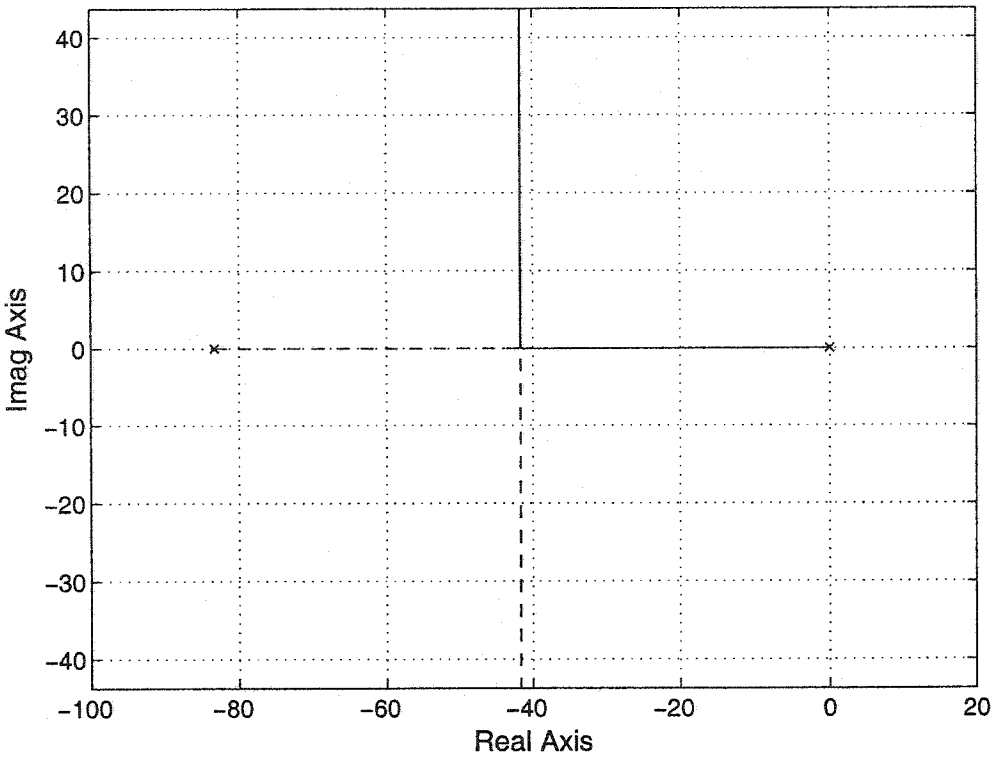


Figure 3.15: Root locus of position loop with an inner velocity loop.

3.3.4 Gravity Feedforward

The effects of gravity on the planar robot also influence the performance of the controller. As the arm rotates, the torque due to gravity varies sinusoidally reaching a maximum when the arm is held parallel to the ground. This torque can contribute to a significant steady state error in the position. This can be countered by adding a feedforward term to counteract the torque due to gravity. The torque due to gravity is described by equation 3.15. The additional current required to cancel the effects due to gravity is given by:

$$I_{gravity}(\theta) = \frac{Mgr_{cm}\cos(\theta)}{K_t} \tag{3.24}$$

where M is the mass of the robot arm, g is the acceleration due to gravity, r_{cm} is the location of the center of mass, θ is the angular position of the robot arm, and K_t is the torque constant. A look-up table is used to evaluate the $\cos(\theta)$ term to speed up the computation since it occurs within the control loop.

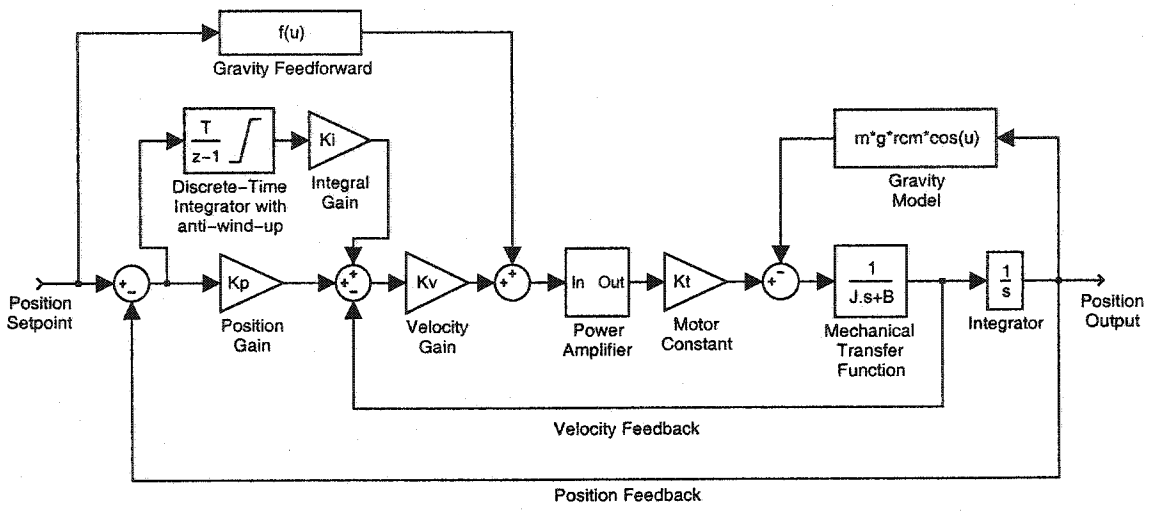


Figure 3.16: Final diagram of plant and compensator.

3.3.5 The Complete Compensator and Plant Model

Both the inner velocity loop and the outer position loop use a simple proportional gain compensator. The addition of a gravity feedforward term will counteract errors due to gravity but a steady state position error is still possible. To eliminate any steady state errors, an integral gain can be added to the outer position loop to form a PI compensator. The integrator includes a saturation limit which limits the magnitude of the integral term to prevent integral wind-up. A block diagram of the entire plant along with the final compensator structure including the proportional velocity loop, the proportional-plus-integral position loop, and the gravity feedforward is shown in Figure 3.16. The final compensator

design was tested by tuning the gains to achieve a reasonable step response as shown in Figure 3.17. This model is a continuous-time model and will eventually be converted to discrete time for implementation in the master servo computer.

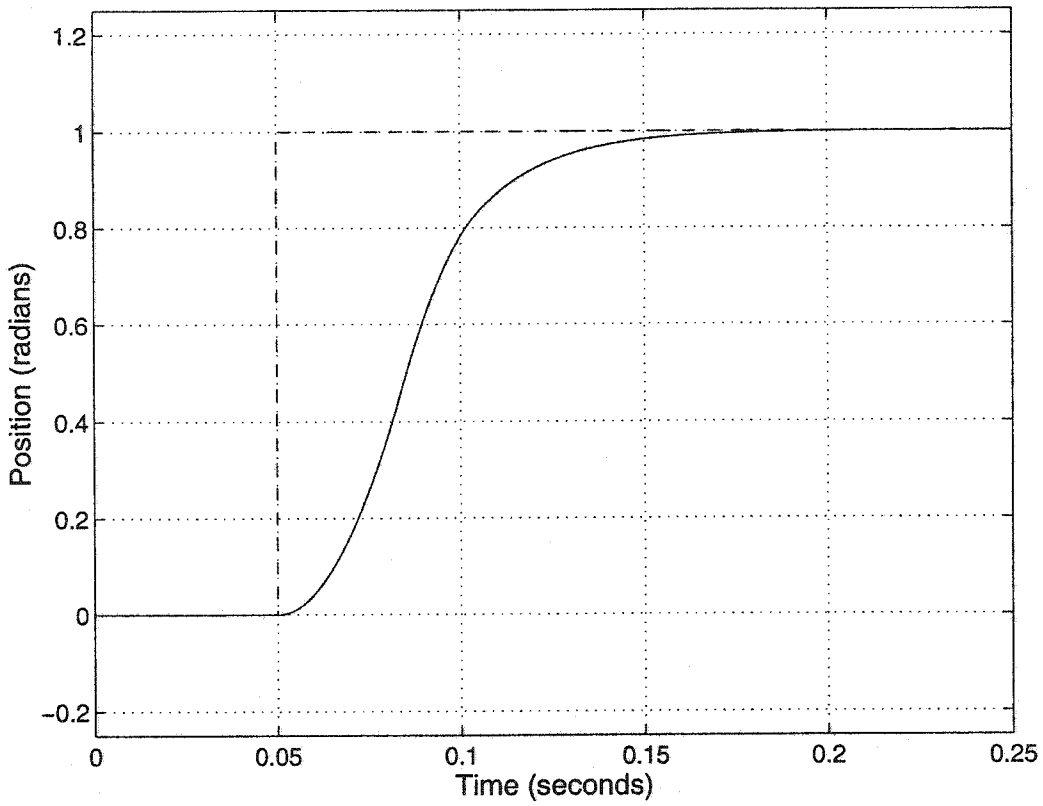


Figure 3.17: Continuous-time step response of the complete plant with the final compensator.

3.3.6 Observability

In the preceding section, a compensator was constructed assuming that both the position and the velocity state variables are observable. In reality, there are no dedicated position or velocity sensors available for feedback. The state variables will need to be obtained from the vision system alone, since it is the only source of feedback in a direct visual

servoing system. However, the vision feedback has numerous shortcomings as described in Chapter 2. There are three major challenges that need to be addressed before the vision system can provide suitable state feedback for the compensator described in the previous section.

The first issue is that there is a large transport delay in the vision feedback. This delay is due primarily to the RS-170 transmission time and the eigenspace computation time. The position measurements provided by the vision system are typically delayed by up to $21ms$ (see section 2.6.5). This delay results in substantial phase delays and a negative phase margin. A stable position loop can only be realized with timely position and velocity feedback.

The second issue is that the vision algorithm does not directly provide a velocity measurement for the inner velocity loop. The velocity can be obtained by differentiating the position measurement but this produces a slightly delayed signal which is also highly sensitive to any noise present in the position signal. Differentiating position to obtain velocity also performs poorly at low motor speeds. A more elegant approach to determining velocity will need to be employed.

The third issue is that the vision feedback is subject to various sources of noise. Sources of noise in the vision feedback include CCD noise, amplifier noise, RS-170 transmission noise, and A/D conversion noise. Other "numerical noise" is also present due to finite wordlength effects and interpolation errors in the eigenspace manifold. As discovered in section 2.7, the overall noise increases substantially in the presence of occlusions.

Each of these issues will need to be addressed to find a solution. The *Kalman filter* was explored as a possible solution to address all of these issues and it is described in the following section.

3.4 The Kalman Filter

The Kalman Filter was published by R.E. Kalman in [36] as a new approach to linear filtering. The filter is essentially a recursive algorithm for the least squares estimation problem. The Kalman filter can optimally estimate in real-time the state variables of a system based on measurements of noisy outputs. The Kalman filter has been successfully employed in numerous applications including space exploration, GPS systems, and navigation systems [37]. The Kalman filter has found many useful applications in vision systems due to its ability to estimate in real-time the state variables of a system based on noisy measurements. The Kalman filter has also been employed in the area of visual servoing. Westmore and Wilson [71] have described the use of a Kalman filter for position-based visual servoing. The Kalman filter algorithm is summarized in Figure 3.18 and includes two main steps: a state prediction step and a measurement update step.

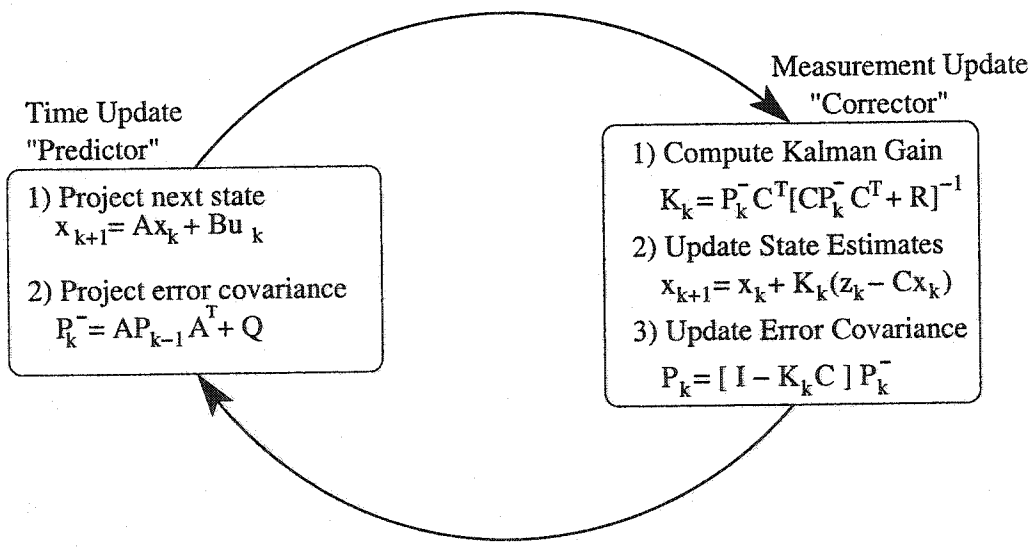


Figure 3.18: Model of the Kalman Filter.

3.4.1 The Continuous State Equations

To perform the state prediction step, the Kalman filter must have a discrete-time state model for the plant. However, the planar robot plant is best described using a continuous-time model. Therefore, the continuous-time model of the robot is first determined and then later converted to discrete time for use in the Kalman filter prediction equations. The continuous-time state-space equations are given by:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{3.25}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{3.26}$$

where $\mathbf{u}$ is the system input, $\mathbf{y}$ is the system output, and $\mathbf{x}$ represents the continuous-time state vector. The continuous-time state equations for the planar robot can be easily derived using the linearized physical plant models that were determined earlier in the chapter. The state vector $\mathbf{x}$ contains the state variables, which were chosen for the planar robot as follows:

x_1 = motor torque

x_2 = motor speed

x_3 = motor position

The continuous-time state matrices are determined using differential equations describing each of the state variables listed above.

Motor Torque State Equation

The state equation for the first state variable (motor torque) can be obtained using equations 3.17 and 3.18:

$$X_1(s) = T_e(s) = \frac{K_t}{\tau_e s + 1} I_{motor}(s) \tag{3.27}$$

where $X_1(s)$ is the Laplace transform of the state variable x_1 and $I_{motor}(s)$ is the Laplace transform of the motor current. This equation can then be rearranged to obtain the following

expression:

$$sX_1(s) = \frac{1}{\tau_e}X_1(s) + \frac{K_t}{\tau_e}I_{motor}(s) \tag{3.28}$$

Finally, the inverse Laplace Transform is applied to produce the first continuous-time state equation:

$$\dot{x}_1 = \frac{1}{\tau_e}x_1 + \frac{K_t}{\tau_e}\mathbf{u} \tag{3.29}$$

where the system input u is defined to be the motor current (I_{motor}).

Motor Speed State Equation

The state equation for the second state variable x_2 (motor speed) can be derived using the Laplace expression from equation 3.10:

$$X_2(s) = \frac{1}{Js + B}T_e(s) \tag{3.30}$$

where $X_2(s)$ is the Laplace transform of the state variable x_2 and $T_e(s)$ is the Laplace transform of the motor torque. Replacing $T_e(s)$ with the corresponding state variable $X_1(s)$ and then rearranging gives:

$$X_2(s)(Js + B) = X_1(s) \tag{3.31}$$

The equation can then be further rearranged to obtain the following expression:

$$sX_2(s) = \frac{1}{J}X_1(s) - \frac{B}{J}X_2(s) \tag{3.32}$$

Finally, the inverse Laplace Transform is applied to produce the state equation:

$$\dot{x}_2 = \frac{1}{J}x_1 - \frac{B}{J}x_2 \tag{3.33}$$

Motor Position State Equation

The state equation for the motor position proceeds by observing that the derivative of position is simply velocity:

$$\dot{x}_3 = x_2 \tag{3.34}$$

The Continuous State Equation for the Planar Robot

The A and B matrices can be formed using equations 3.29, 3.33 and 3.34. Therefore, the continuous state space equations for the planar robot are given by:

$$\dot{\mathbf{x}} = \begin{bmatrix} \frac{-1}{\tau_e} & 0 & 0 \\ \frac{1}{J} & \frac{-B}{J} & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} \frac{K_t}{\tau_e} \\ 0 \\ 0 \end{bmatrix} \mathbf{u} \tag{3.35}$$

$$\mathbf{y} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{x} \tag{3.36}$$

where

$\mathbf{x} = [x_1\ x_2\ x_3]^T$ (the state vector)

$\mathbf{u}$ = motor current setpoint (system input)

$\mathbf{y}$ = motor position (system output)

K_t = motor torque constant

τ_e = electrical time constant

B = load damping

J = load inertia

3.4.2 The Discrete-Time State and Output Equations

The Kalman filter requires a discrete-time representation of the system. The discrete-time state and output equations are given by:

$$\mathbf{x}_{k+1} = \Phi\mathbf{x}_k + \Delta\mathbf{u}_k \tag{3.37}$$

$$\mathbf{y}_{k+1} = \mathbf{C}\mathbf{x}_{k+1} \tag{3.38}$$

where

$\mathbf{x}_k$ = the state vector at time step k

$\mathbf{u}_k$ = motor current setpoint (system input) at time step k

Δ = input matrix

Φ = the state transition matrix

$\mathbf{y}_k$ = system output measurement (position) at time step k

$\mathbf{C}$ = output matrix

The Φ and Δ matrices are a function of the the continuous state equation and the sampling time [15]. Using forward integration on the continuous state equation in Equation 3.25, the Φ and Δ matrices can be found as follows:

$$\Phi \approx \mathbf{I} + \mathbf{A}T_s \tag{3.39}$$

$$\Delta \approx \mathbf{I}T_s\mathbf{B} \tag{3.40}$$

where the $\mathbf{A}$ and $\mathbf{B}$ matrices are from the continuous state equation, $\mathbf{I}$ is the identity matrix, and T_s is the sample time.

Modelling the Transport Delay

These discrete equations must somehow represent the sizable transport delay inherent in the vision feedback. The transport delay is described in equation 2.25 as the sum of the RS-170 transmission time and the vision computation time. One approach for employing a Kalman Filter in systems with a large transport delay involves augmenting the discrete-time state vector with states to represent the delay [3]. Thus, the transport delay can be accounted for in the discrete-time state-space equations by delaying the position state by the number of sampling intervals equivalent to the vision delay. This introduces n new

states as follows:

$$\left.\begin{aligned} x_4 &= z^{-1}x_3 \\ x_5 &= z^{-1}x_4 = z^{-2}x_3 \\ &\vdots \\ x_n &= z^{-1}x_{n-1} = z^{-(n-3)}x_3 \\ &\vdots \\ x_{n+3} &= z^{-1}x_{n+2} = z^{-n}x_3 \end{aligned}\right\} \text{position delay states}$$

where x_{n+3} represents the delayed position measurement from the vision system. The delay states are easily updated at each time step by shifting the values from one state variable to the next. This can be expressed in matrix form as follows:

$$\begin{bmatrix} x_4(k+1) \\ x_5(k+1) \\ x_6(k+1) \\ \vdots \\ x_{n+2}(k+1) \\ x_{n+3}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \ddots & & \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_3(k) \\ x_4(k) \\ x_5(k) \\ \vdots \\ x_{n+1}(k) \\ x_{n+2}(k) \end{bmatrix} \tag{3.41}$$

The value n is the number of discrete delay states in the transport delay rounded up to the nearest integer given by:

$$n = \left\lceil \frac{t_{delay}}{T_s} \right\rceil \tag{3.42}$$

where t_{delay} is the actual vision transport delay and T_s is the sample time equal to $\frac{1}{f_s}$ (see Equation 2.1). One disadvantage of this approach is that the dimensions of the state equations increase by $O(n)$ as the effective sample time decreases.

The Discrete State Equations for the Planar Robot

Next, the complete discrete-time state equations for the planar robot can be determined. As mentioned previously, the Φ and Δ matrices for the discrete state equation can be obtained

from the continuous-time equation using equations 3.39 and 3.40. The number of position delay states n can be determined using Equation 3.42. The discrete state vector is then augmented with n additional states and updated according to equation 3.41. Thus, the complete discrete state equations for the planar robot are given by:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \\ x_5(k+1) \\ \vdots \\ x_{n+3}(k+1) \end{bmatrix} = \begin{bmatrix} 1-\frac{T_s}{\tau_e} & 0 & 0 & 0 & \cdots & & \\ \frac{1}{J}T_s & 1-(\frac{B}{J})T_s & 0 & 0 & \cdots & & \\ 0 & T_s & 1 & 0 & \cdots & & \\ 0 & 0 & 1 & 0 & \cdots & & \\ & \cdots & 0 & 1 & 0 & \cdots & \\ & & & & \ddots & & \\ & & & \cdots & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ x_4(k) \\ x_5(k) \\ \vdots \\ x_{n+3}(k) \end{bmatrix} + \begin{bmatrix} K_t\frac{T_s}{\tau_e} \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \mathbf{u}(k) \tag{3.43}$$

If τ_e approaches the sample time T_s (or if it less than T_s) then the sample rate is not adequate to observe the electrical time constants associated with the power amplifier. In this case, the electrical time constants can be safely ignored since the system response is dominated by the slower mechanical time constants. The model of the electrical time constant can be eliminated by setting the first row of the state transition matrix to zeroes and updating x_1 to $K_t\mathbf{u}$. This will simply update the motor torque based on the motor current setpoint. The last state variable x_{n+3} represents the delayed position measurement output from the vision

system. Hence, the measurement equation becomes:

$$\mathbf{z}(k) = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \\ \vdots \\ x_{n+3}(k) \end{bmatrix} \tag{3.44}$$

where $\mathbf{z}_k$ is the position measurement provided by the vision system.

3.4.3 The Kalman Filter Equations

The discrete state equations stated in equations 3.37 and 3.38 can be used to model a random process by adding process and measurement noise signals as follows:

$$\mathbf{x}_{k+1} = \mathbf{\Phi}\mathbf{x}_k + \mathbf{\Delta}\mathbf{u}_k + \mathbf{\Gamma}\mathbf{w}_k \tag{3.45}$$

$$\mathbf{z}_{k+1} = \mathbf{C}\mathbf{x}_{k+1} + \mathbf{v}_{k+1} \tag{3.46}$$

where

$\mathbf{w}_k$ = system noise input at time step k

$\mathbf{\Gamma}$ = system noise coupling matrix

$\mathbf{z}_k$ = output measurement at time step k

$\mathbf{v}_k$ = measurement noise at time step k

The covariance matrices for the $\mathbf{w}_k$ and the $\mathbf{v}_k$ noise signals are assumed to be known [6] and are given by:

$$E[\mathbf{v}_k\mathbf{v}_i^T] = \begin{cases} \mathbf{R}_k, & i = k \\ 0, & i \neq k \end{cases} \tag{3.47}$$

$$E[\mathbf{w}_k\mathbf{w}_i^T] = \begin{cases} \mathbf{Q}_k, & i = k \\ 0, & i \neq k \end{cases} \tag{3.48}$$

The $\mathbf{w}_k$ and the $\mathbf{v}_k$ signals must also be uncorrelated such that:

$$E[\mathbf{w}_k \mathbf{v}_i^T] = 0 \text{ for all } k \text{ and } i \tag{3.49}$$

The $\mathbf{Q}$ and $\mathbf{R}$ matrices can be constant or they can change with time. In practice, the noise covariance matrices are often difficult to determine.

However, once the discrete state equations and the noise covariance matrices have been determined, the optimal state estimates can be computed using the Kalman filter recursive equations [6]:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{\Phi}\hat{\mathbf{x}}_{k|k} + \mathbf{\Delta}\mathbf{u}_k \tag{3.50}$$

$$\mathbf{P}_{k+1|k} = \mathbf{\Phi}\mathbf{P}_{k|k}\mathbf{\Phi}^T + \mathbf{\Gamma}\mathbf{Q}\mathbf{\Gamma}^T \tag{3.51}$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{C}^T[\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \mathbf{R}]^{-1} \tag{3.52}$$

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}[\mathbf{z}_{k+1} - \mathbf{C}\hat{\mathbf{x}}_{k+1|k}] \tag{3.53}$$

$$\mathbf{P}_{k+1|k+1} = [\mathbf{I} - \mathbf{K}_{k+1}\mathbf{C}]\mathbf{P}_{k+1|k} \tag{3.54}$$

where

$\hat{\mathbf{x}}$ = the state estimate

$\mathbf{R}$ = the covariance matrix of the measurement noise

$\mathbf{Q}$ = the covariance matrix of the system noise

$\mathbf{P}$ = the covariance matrix of the state estimation error

$\mathbf{K}$ = the Kalman gain matrix

At each time step k, a position measurement $\mathbf{z}_k$ arrives from one of the vision nodes which are operating in a round-robin fashion. The Kalman filter equations are thus performed every time a new position measurement arrives from a vision node.

3.4.4 The Stationary Kalman Filter

In some situations the values of the $\mathbf{Q}$ and $\mathbf{R}$ matrices can be assumed to be constant. If the noise is considered constant, a *stationary* Kalman filter can be used to implement a

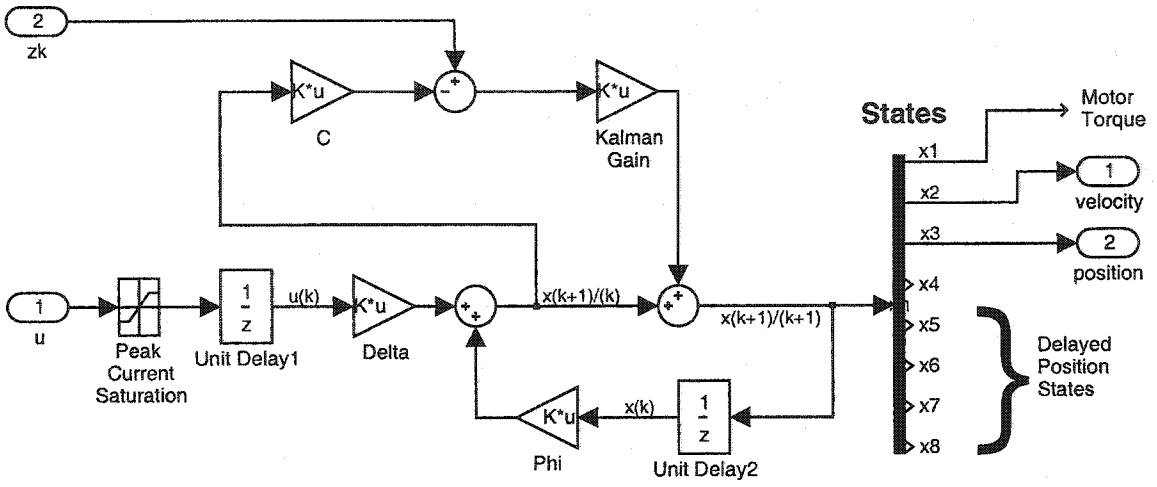


Figure 3.19: Block diagram of a stationary Kalman filter.

direct visual servo controller [56]. The Kalman gains using a stationary noise model of the planar robot system will converge after iterative computations and settle to a constant value. Figure 3.20 shows the rapid convergence of the Kalman gains for the planar robot system. In these situations, the Kalman gain $\mathbf{K}$ may be pre-computed off-line and "hard-coded" to reduce the computational effort at run-time. The precomputed Kalman gains can then be used to construct a stationary Kalman observer as illustrated in Figure 3.19. However, any changes in the system or measurement error variances will result in poor performance. Therefore, this approach is of limited use since it assumes ideal conditions and is not robust in the face unexpected noise sources. If the variance of the measurement noise $\mathbf{R}$ changes over time, such as in the case of occlusions, it will be necessary to use a *non-stationary* Kalman filter and all the iterative Kalman equations will need to be computed at each and every time step.

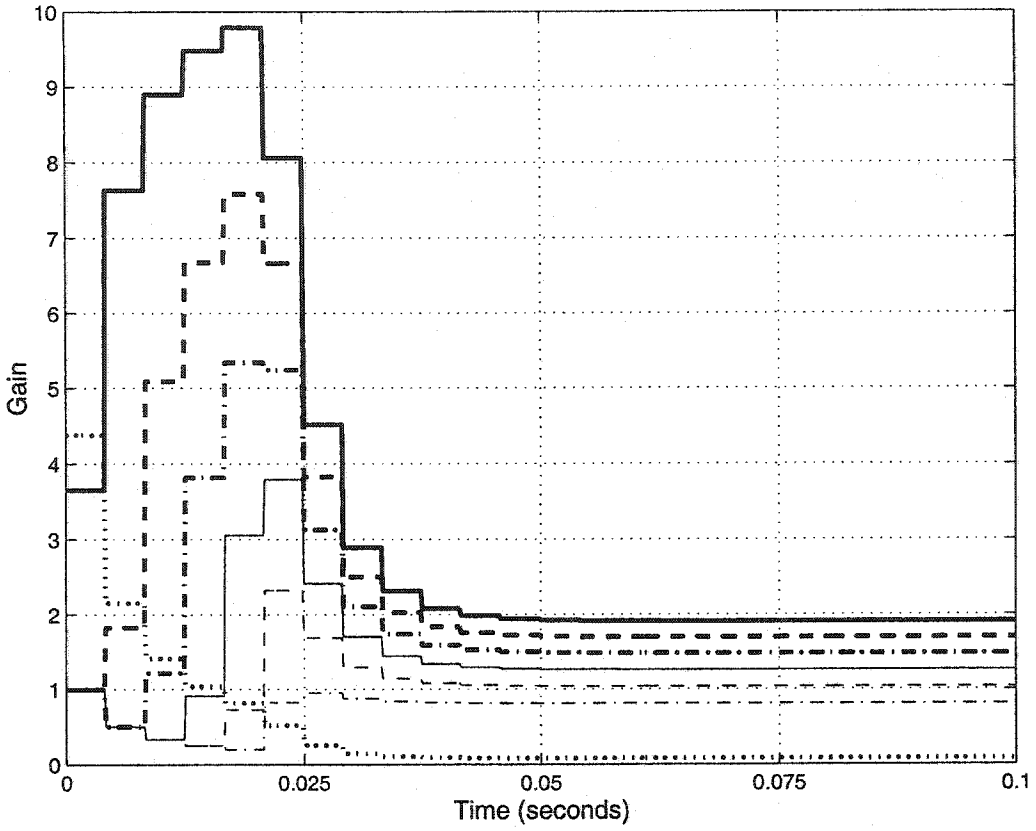


Figure 3.20: Kalman gains vs. time.

3.4.5 The Non-Stationary Kalman Filter

A block diagram depicting the full set of Kalman filter equations is shown in Figure 3.21. If the Q and R matrices are not constant, then the filter is referred to as a *non-stationary* Kalman filter. Determining the Q and R matrices is crucial for good system performance. The relative values of Q and R determine the extent to which the Kalman filter relies on the output measurements as compared to the observer predictions. The efficacy of the Kalman filter for this application in estimating the actual position without the transport delay is highly dependent on the accuracy of the observer state model. Inaccurate models of the Φ and Δ matrices diminish the confidence in the observer output and increase the values in

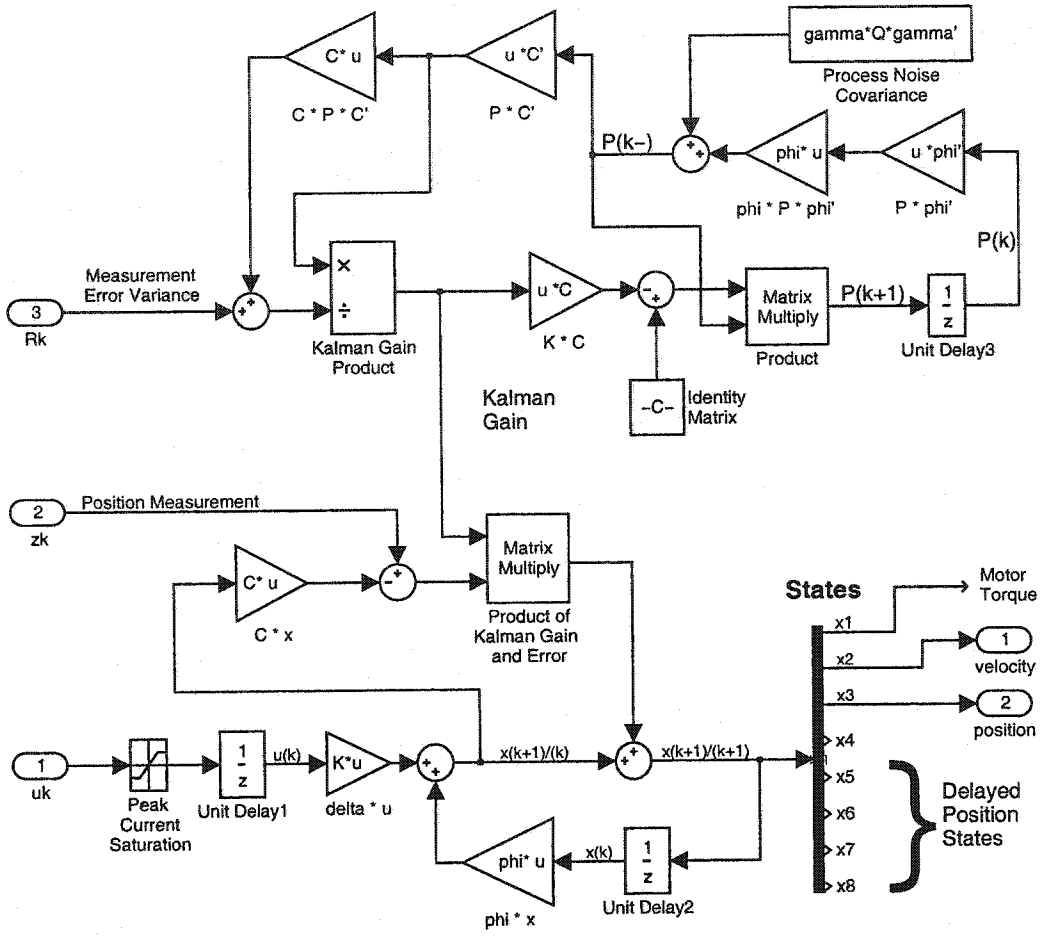


Figure 3.21: Block diagram of non-stationary Kalman filter.

the Q matrix. Large values in the Q matrix have the effect of adjusting the Kalman gains to increase the weighting of the delayed position feedback in the overall position estimate. As the Kalman filter relies more heavily on the delayed position feedback, the effective phase margin of the control system decreases and the dynamic performance of the system is reduced.

3.4.6 Noise and Occlusions

Before the Kalman filter can be implemented, the measurement noise $\mathbf{R}$ must be identified and somehow quantified. In this work, occlusions will be regarded as additional dynamic "noise" in the system. Occlusions therefore render the assumption of constant measurement noise invalid since the measurement noise variance $\mathbf{R}$ will be unique for each camera and may change over time. This implies that a non-stationary Kalman filter will be required. The process noise $\mathbf{Q}$ is independent of the cameras and can therefore still be considered a constant. The Kalman gain computations then become:

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{C}^T[\mathbf{C}\mathbf{P}_{k+1|k}\mathbf{C}^T + \mathbf{R}_{k+1}]^{-1} \tag{3.55}$$

where $\mathbf{R}_{k+1}$ is the time varying measurement noise variance which will also depend on the camera currently being considered. This equation determines the Kalman gain by relating the process noise to the measurement noise for the current camera. Given a finite process noise estimate, as the measurement noise approaches zero, the Kalman gain will tend towards one. As the Kalman gain approaches one, the state outputs will be based on the measured feedback alone and not at all on the state model. Conversely, as the measurement noise approaches infinity, the Kalman gain will tend towards zero. When the Kalman gain is zero, the state outputs are based on the state model alone and not at all on the measured feedback. Thus the state estimates produced by the Kalman filter are based on a blend of the state model and the vision measurements in a manner which is optimal in the least-squares sense. This works well in theory, but the practical problem remains of how to determine the measurement noise variance $\mathbf{R}$ at each time step for each camera.

There are no direct ways of knowing the exact measurement noise variance for any given camera. It is proposed that $\mathbf{R}_k$ can be estimated from the statistical relationship between Euclidean distance and error variance that was identified in Chapter 2. This statistical relationship is illustrated in Figure 2.25 and described in section 2.7. The key idea is that Euclidean distance d_k from the manifold can be used to compute an estimate of the

error variance according to equation 2.29. This equation can be rewritten by substituting the variance σ_k^2 with the measurement error variance $\mathbf{R}_k$ as follows:

$$\mathbf{R}_k = \begin{cases} m \cdot SSD_k, & d_k \leq 2500 \\ \infty, & d_k > 2500 \end{cases} \tag{3.56}$$

where SSD_k is the sum-of-square differences in eigenspace at time step k and m is a constant. This expression is conditional reflecting the threshold beyond which the Euclidean distance measurements no longer fall within a distinct error band and begin to scatter as illustrated in Figure 2.24. The result from equation 3.56 can be fed directly into into the Kalman gain computation in equation 3.55. When none of the cameras are occluded, the Kalman gains should converge to values similar to those precomputed for a stationary Kalman filter. When occlusions occur, $\mathbf{R}$ will increase and the Kalman filter will dynamically adjust the gains to maintain good position estimates. Thus, some useful information can still be gleaned from partially occluded cameras. The weighting of visual feedback from occluded cameras will gradually diminish as the Euclidean distance grows. When the Euclidean distance reaches a threshold distance of 2500, the statistical relationship with error variance breaks down. Therefore, for cameras reporting distances greater than 2500, the measurement will be simply rejected by considering $\mathbf{R}$ to be infinite, which has the effect of setting the Kalman gains to zero. Thus, if the occlusions in a certain camera become too severe (i.e. $d_k > 2500$) then the Kalman filter will proceed to reject the vision feedback from that camera. With multiple cameras all possessing slightly different views of the scene, it is possible that one or more of the cameras will experience differing degrees of occlusion at various times as work proceeds around a robot. Therefore, while some cameras may be occluded, other cameras may still be able to provide accurate feedback measurements. In fact, even if only one camera remains free of occlusions, an accurate position estimate can be maintained. Therefore, multiple cameras not only provide advantages in terms of higher sample rates, they also provide more robustness in the presence of occlusions. The computed measurement error variance $\mathbf{R}$ simply adjusts the weighting of

every measurement from each camera.

If all cameras experience severe occlusions, then the Kalman filter will rely exclusively on the predictor estimates. The predictor is not exact and will ultimately stray from the actual values over time when there is no suitable feedback available to use for correction. However, the Kalman filter will continue to track the overall accuracy of the state estimates in the $\mathbf{P}_k$ covariance matrix. As a safety precaution, there should be a strategy to prevent run-away if all the cameras become severely occluded. For example, the servo computer could shutdown the robot when the values in the $\mathbf{P}_k$ matrix exceed a predetermined threshold.

All noise sources ultimately cause projected points in eigenspace to stray from the manifold. Therefore, other noise sources such as CCD noise and small variations in illumination can be treated in a similar fashion as occlusions. The optimal least-squares operation of the Kalman filter will ensure that all feedback will be weighted accordingly to ensure good position estimates. Even errors due to the differences in odd and even video fields can be dealt with by the Kalman filter albeit with a reduction in the overall accuracy of the measurements.

One major advantage of this approach is that the estimate for $\mathbf{R}_k$ has no delays associated with it since it is not based on past history but relies only on the current measurement data. Equation 3.56 provides an elegant solution to the problem of identifying the non-stationary measurement noise in a computationally efficient manner. This equation requires only *one* multiplication operation making it suitable for use in the time-critical control loops. In addition, this approach provides an effective means of *sensor fusion* by integrating the feedback from multiple camera sensors and weighting each of them appropriately to provide improved state estimates. Finally, this approach provides a degree of robustness to noise and occlusions.

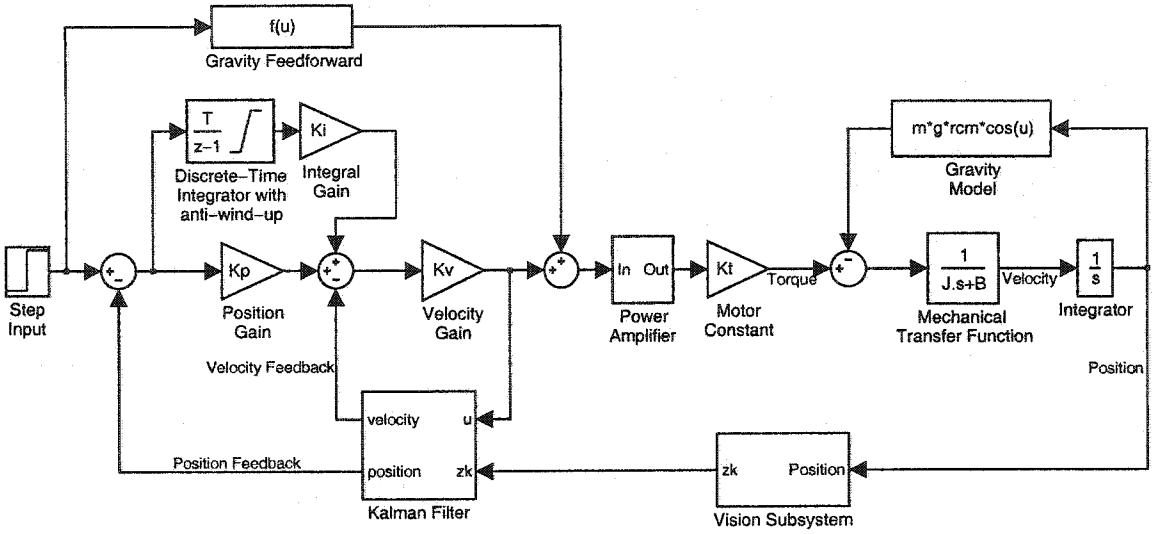


Figure 3.22: Block diagram of the overall system model.

3.5 Direct Visual Servo Simulation

Once the models of the various subsystems are determined they can be combined to build a model of the overall visual servoing system. The different component subsystems models include the planar robot mechanical system, the motor, the servo amplifier, the vision system, and the Kalman filter and controller. The outputs of the Kalman filter provide velocity and position state feedback to the controller. The controller uses a PI position loop with velocity compensation and a feed-forward path to counter the gravitational forces on the planar robot joint. The models of each subsystem can be interconnected to form an overall system model as shown in Figure 3.22.

The system model allow simulations to be performed to predict the performance of this approach for direct visual servoing of a planar robot. Three specific performance issues were explored in simulation and are described in the following subsections. First, the dynamic performance of the planar robot was determined under a varying number of cameras.

Second, the effect of external disturbances on the robot were explored. Finally, the performance of the direct visual servo was evaluated during a simulated dynamic occlusion.

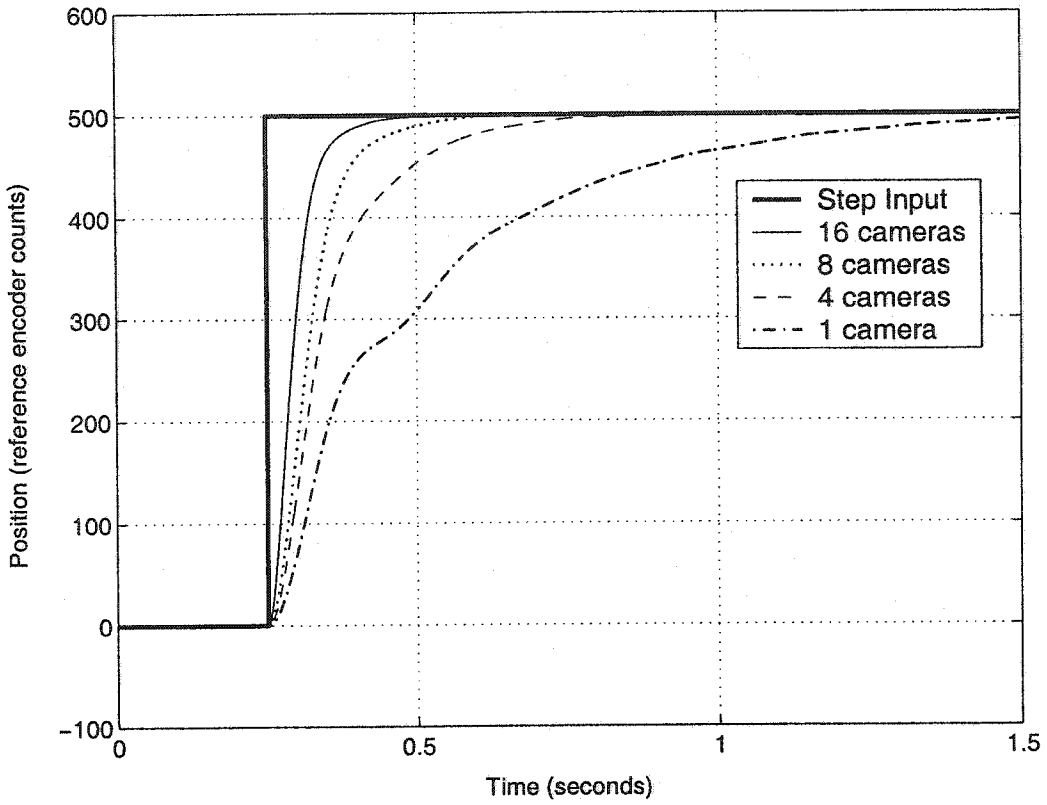
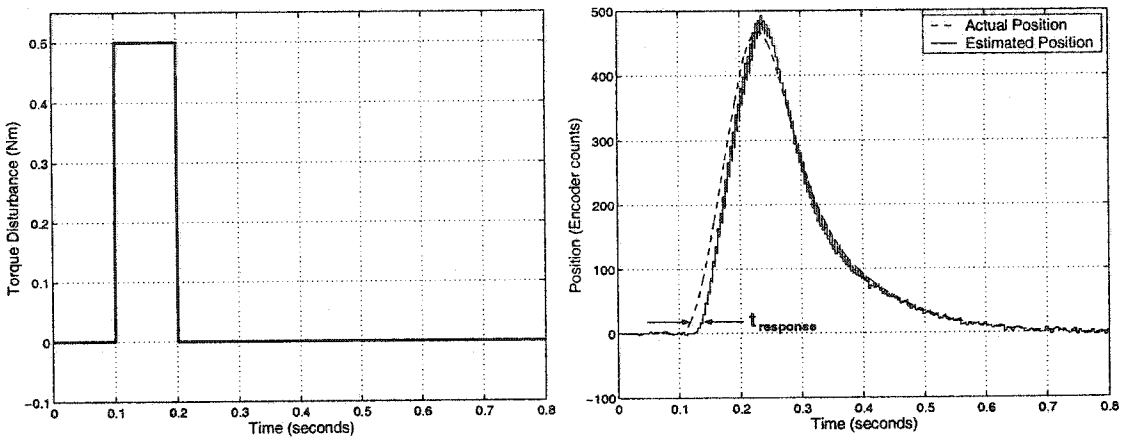


Figure 3.23: Simulated response to a step input for systems employing different numbers of cameras. (Position units use 4000 counts/revolution).

3.5.1 Step Response Using Multiple Cameras

Simulations were performed using Matlab and Simulink for systems employing different numbers of cameras for direct visual servoing of the planar robot. The simulations were performed using stationary noise sources allowing the Kalman gains to be pre-computed and fixed using the structure shown in Figure 3.19. The compensator gains were adjusted for critical damping in the step response. The simulation results for systems with 1, 4, 8

and 16 cameras are plotted in Figure 3.23. Clearly, the use of multiple cameras improves the transient step response time since more cameras increase the effective sample rate. A system with 16 cameras has an effective sample time of about $1ms$ and exhibits a step risetime of approximately $75ms$ which is comparable to the time required for two standard RS-170 video frames. The risetime using 16 cameras approaches the risetime achieved using continuous-time state-feedback shown in Figure 3.17. Clearly, the use of more cameras improves the dynamic performance of the visual servo system.



(a) Torque Disturbance vs. time.

(b) Position vs. time.

Figure 3.24: Torque disturbance simulation.

3.5.2 Disturbance Rejection

The predictor in the Kalman filter is able to accurately predict the response of the system to any changes in the input. This allows the Kalman filter to provide timely state estimates despite the transport delay in the vision feedback. A shortcoming of this approach is that the predictor will not work for unmodelled target motion since it cannot anticipate external disturbance inputs. Therefore, a direct visual servoing system cannot sense an external

disturbance until it propagates through the vision system. This implies that the system response to disturbance inputs is dominated by a substantial vision transport delay.

The system response to an external disturbance input can be simulated to predict how the system will perform. A system model was configured assuming four cameras and the planar robot system. A external torque disturbance with a magnitude of 0.5Nm was temporarily applied to the robot model as shown in Figure 3.24(a). The simulated transient position response of the direct visual servo and the estimated position from the Kalman filter are plotted in Figure 3.24(b). Because the torque disturbance is not modelled, it cannot be anticipated by the Kalman filter predictor. Consequently, the estimated position momentarily remains unchanged while the actual position changes dramatically in response to the external torque pulse. The latency in the estimated position measurement is labeled as $t_{response}$ in Figure 3.24(b). The time $t_{response}$ is the time necessary for the disturbance to be captured by a camera and for the measurement to propagate through the various transport delays described in section 2.6.5. It is not until the Kalman filter "sees" the external disturbance that it begins to converge on the new state of the system. It is only after this time that the controller can begin to respond to the external disturbance. The minimum delay for $t_{response}$ occurs when the torque disturbance occurs just prior to the acquisition time of a camera. The maximum delay for $t_{response}$ occurs when a torque disturbance occurs just after an image is acquired by a camera so that one additional sample time must pass before the disturbance can be captured by the next camera. Therefore, the range of the $t_{response}$ is given by:

$$t_{delay} < t_{response} < t_{delay} + T_s \tag{3.57}$$

where T_s is the sample time ($\frac{1}{60N_c}$), and t_{delay} is the transport delay from the time the pixels are captured until the position measurement is available (see Figure 2.18). Therefore, as the number of cameras N_c increases, T_s decreases, and the maximum response time to an external disturbance also decreases.

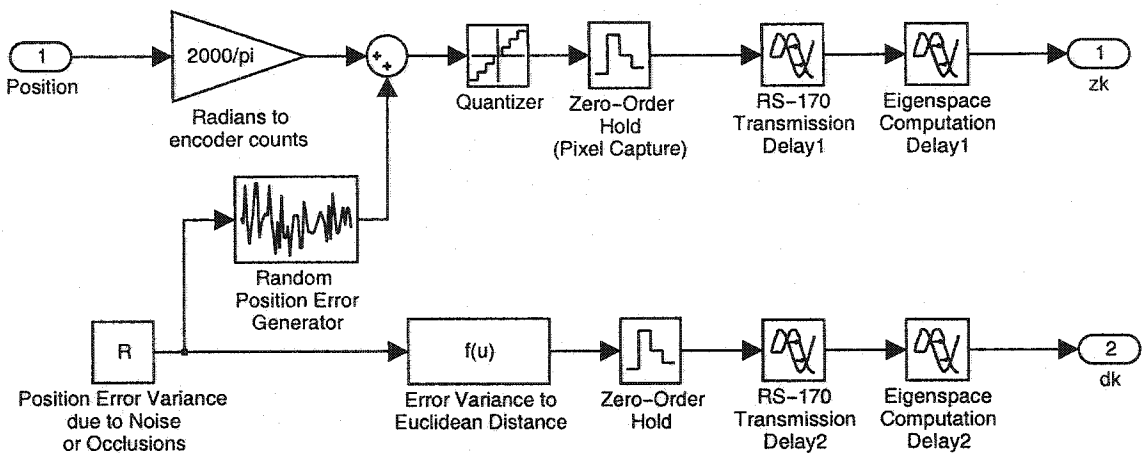


Figure 3.25: Block diagram of the vision model augmented with a Euclidean distance output.

3.5.3 Simulated Occlusions

The performance of the direct visual servo in the presence of dynamic occlusions can also be simulated with some small enhancements to the system model. Since occlusions are treated as "noise", the error variance in the vision model can be made time-varying to simulate dynamic occlusions. The vision model in Figure 3.8 can be augmented with an additional output to provide Euclidean distance information. The Euclidean distance output can be simulated using the inverse of equation 3.56 as follows:

$$d_k = \sqrt{\frac{\mathbf{R}_k}{m}} \tag{3.58}$$

where $\mathbf{R}_k$ is the value of the variance used in the vision model and m is the constant found in section 2.7. The Euclidean distance output from the vision model must also be delayed with a transport delay equivalent to the output position measurement. The augmented vision model with the delayed Euclidean distance output is illustrated in Figure 3.25. The Euclidean distance information can then be used to determine the measurement error variance and then fed into a non-stationary Kalman filter as shown in Figure 3.21. The final

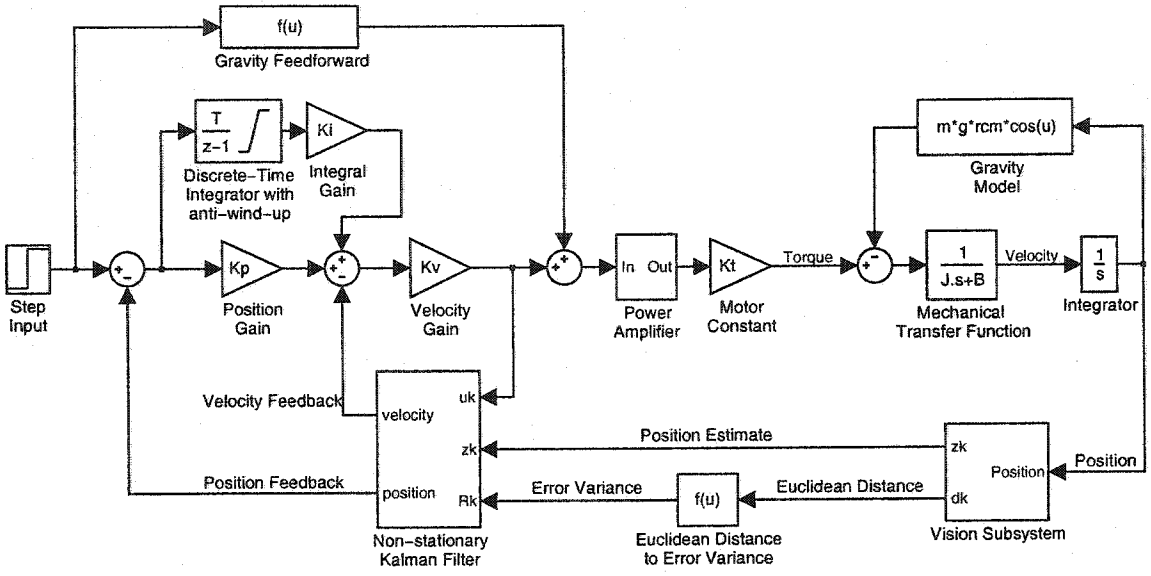
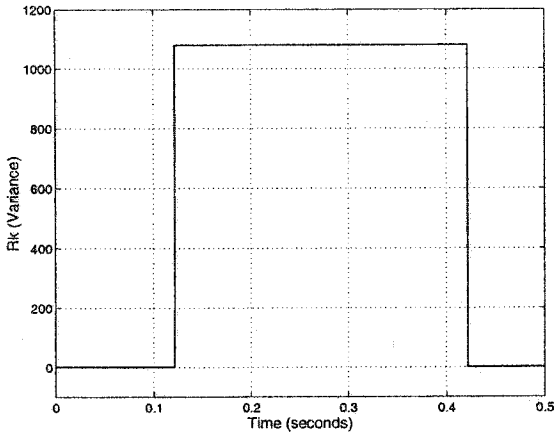


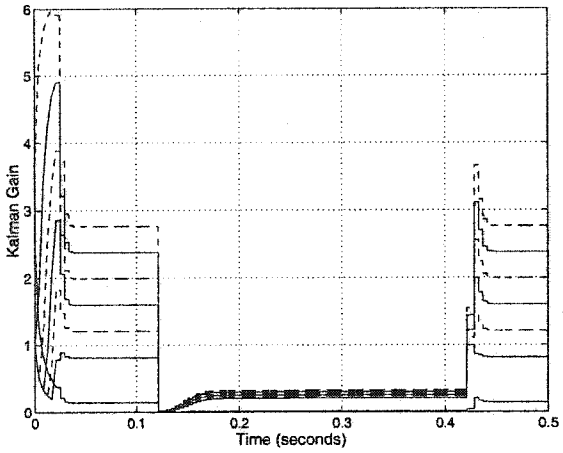
Figure 3.26: Non-stationary system block diagram.

system model suitable for simulating the effects of occlusions is shown in Figure 3.26.

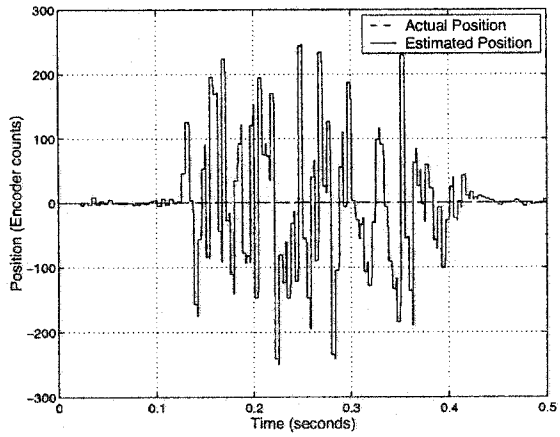
The model parameters were adjusted to simulate 4 cameras performing a servo hold on the planar robot. A simulation was run with a step change in the variance to simulate a large sudden occlusion. The variance vs. time profile is shown in Figure 3.27(a). The peak variance corresponds to an occlusion producing a Euclidean distance of 2000. This distance represents a significant occlusion that is still less than the maximum Euclidean distance threshold of 2500. The simulation was run first with a stationary Kalman filter and then with a non-stationary Kalman filter. Figure 3.27(c) shows the estimated position using a stationary Kalman filter which assumes a fixed measurement error variance. The estimated position is seen to vary wildly from the actual position when the step change occurs in the actual measurement variance. These variations have a magnitude of several hundred encoder counts and would result in erratic robot motion. Figure 3.27(d) shows the estimated position using a non-stationary Kalman filter which determines measurement error



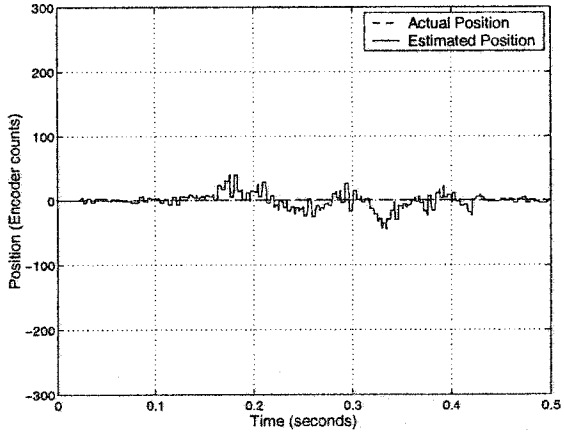
(a) Variance vs. time.



(b) Non-stationary Kalman filter gains vs. time.



(c) Estimated position using stationary Kalman filter



(d) Estimated position using non-stationary Kalman filter

Figure 3.27: Simulated occlusion plots.

variance from Euclidean distance. The resulting position estimation error is substantially reduced as compared with the stationary Kalman filter. The simulated Kalman gains for the non-stationary Kalman filter are shown in Figure 3.27(b). The beginning of the Kalman gain plot shows a transient as the Kalman gains initially converges at the beginning of the simulation. During the time that the simulated occlusions occur, the Kalman gains are reduced to lower the weighting of the measurement feedback. The Kalman gains return to their previous values once the simulated occlusion is removed. Clearly, the non-stationary Kalman filter is capable of providing reasonable position estimates even in the presence of sizable measurement noise.

3.6 Summary

This chapter has provided a detailed description of the modelling and simulation of the direct visual servo system. The various subsystems in the robot were individually modelled. The system model provided insight and understanding of the system dynamics. The system model was linearized so that classical control analysis could be performed using Bode plots. A suitable compensator was designed which required both position and velocity state feedback. The systems models were then used to construct the state equations for a Kalman filter to provide the position and velocity state feedback. The Kalman filter works well in the presence of noise and serves to overcome the latency of the vision feedback by modelling the transport delay and providing timely state estimates.

Through numerous simulations, a statistical relationship between Euclidean distance in eigenspace and measurement error variance was discovered. This insight provided the key to the sensor fusion of the multiple cameras. The measurement error variance was fed into a non-stationary Kalman filter to provide robustness to occlusions.

Finally, several simulations of the entire system verified various aspects of the direct visual servoing system. Simulations confirmed that the dynamic step response could be

improved by employing more cameras. Further simulations demonstrated the response time to an external torque disturbance can also be improved by using multiple cameras. Lastly, simulations of the direct visual servo showed that providing the error variance estimate to a non-stationary Kalman filter can dramatically improve the position state estimates in the presence of dynamic occlusions.

Chapter 4

Testbed Implementation

4.1 Introduction

Simulations can provide useful insights into the feasibility and performance of the proposed direct visual servoing system. However, the accuracy and dependability of simulations only extends as far as the accuracy of the models and the underlying assumptions. For this reason, a working testbed was built and various experiments were performed to verify the practicality of the overall concept.

The implementation of an experimental testbed uncovered many thorny practical issues. This chapter is organized into the hardware and software issues and concludes with a section on how the implementation could be extended to support tele-robotic applications.

4.2 Hardware

An experimental testbed system consisting of four RS-170 grey-scale cameras, four vision computers equipped with frame-grabbers, and a master servo computer was assembled using off-the-shelf components. Figure 4.1 shows a picture of the experimental setup wherein the planar robot, cameras, along with the four vision node computers are all visible. The

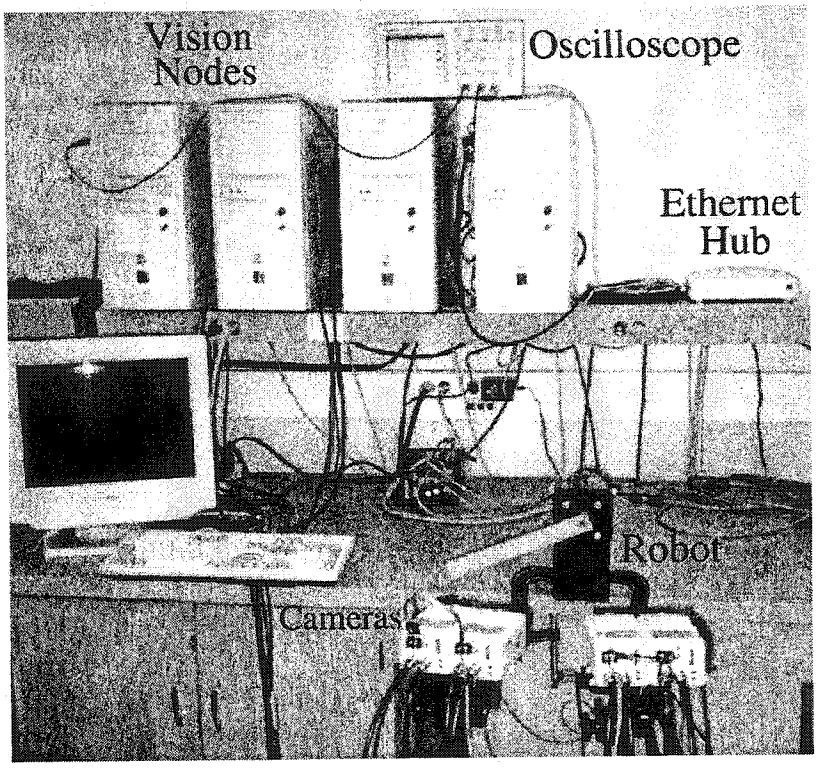


Figure 4.1: Picture of the experimental setup.

components were connected according to the scheme illustrated in Figure 2.1. The robot itself was built to allow access to all inner control loops.

The various components that were used to implement the system can be organized into the following categories:

- Mechanical components
- Electrical components
- Network hardware
- Vision nodes
- Master servo computer

The components in these categories are described in further detail in the following subsections.

4.2.1 The Mechanical Components

The servo motor was mounted in a metal frame with the rotor parallel to the ground. A single robot joint was constructed out of fiberglass and attached to the motor shaft. The robot did not employ any reduction gears and provides direct drive control of the planar robot joint. As a result, the load inertia and angular velocity are reflected directly back to the motor shaft. The inertia and friction associated with this mechanical configuration present some control challenges that were addressed in section 3.3.

4.2.2 The Electrical Components

The main electrical components include the servo motor and the power amplifier. The torque produced by the servo motor is regulated by the current provided by the power amplifier. The power amplifier, in turn, is controlled by the master servo computer. Several cables serve to connect various signals between the electrical components. The final wiring diagram showing the interconnection of the computer, servo motor, and power amplifier is shown in Figure 4.2. Each of the electrical components are described in the following subsections.

The Servo Motor

In the past, many robots were built using DC motors because of the relatively low cost of both motors and controllers. The DC motor has a wound rotor that is switched mechanically with a commutator and brushes. One disadvantage of brushed DC motors is that the commutator and brushes increase the motor volume and tend to wear over time. This has led to an increasing use of *brushless* DC motors which provide high performance with high

reliability. Brushless DC motors have permanent magnet rotors and a wound stator that is electronically switched to produce a rotating flux. Hall effect sensors provide absolute position feedback to control the commutation of the motor phases.

The motor selected for the experimental setup was a Reliance Electric Brushless DC Motor (model #1842419031). The motor was chosen because it had a rated torque that was more than capable of providing a reasonable acceleration under peak mechanical load without exceeding the rated motor current. The motor was selected such that:

$$T_{rated} > J_{max}\alpha_{max} + T_{gravity}(\theta = 90^o) \tag{4.1}$$

where T_{rated} is the rated torque of the motor, J_{max} is the maximum load inertia, α_{max} is the maximum angular acceleration, and $\theta = 90^o$ is the angle at which the force of gravity is maximum on the robot joint. Also, the rated speed and voltage were selected to be more than adequate.

A Dynapar (model M15) modular encoder was attached to one end of the motor shaft and was read by an I/O card in the master servo computer. The encoder provides angular position measurements with a resolution of 4000 counts/revolution. Because the system uses direct visual servoing, the encoder was not used in the position feedback loop. The encoder was only required during the eigenspace training phase to provide the mapping from appearance to position. The encoder was also employed during various experiments to independently log the actual motor position.

The Power Amplifier

The power amplifier regulates the motor current in proportion to a small-signal reference input. The servo amplifier uses power electronic switches to produce a high frequency pulse-width modulation (PWM) signal which regulates the current in the motor windings. The power amplifier is also responsible for electronically commutating the motor phases

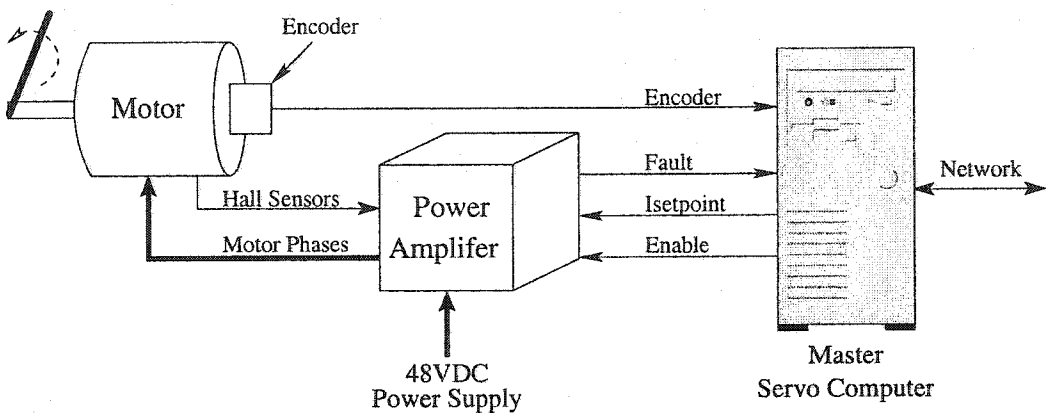


Figure 4.2: Servo motor connection diagram.

according to the hall sensor feedback. The power amplifier also normally provides over-current, over-voltage, and over-temperature protection.

The current and voltage ratings of the power amplifier were selected based on the motor ratings and the expected duty cycle of the robot. The amplifier that was chosen was an AMC brushless PWM servo amplifier (model #B15A8F) with a rating of 15 Amperes at a maximum of 80 volts. The specified bandwidth of the current loop is 2.5kHz. The power to the amplifier was supplied by an external 48Volt DC power supply rated at 300 Watts. This power supply was selected on the basis of its power rating and its voltage rating. The voltage was selected to provide sufficient voltage headroom to ensure that the current can be properly regulated above the back emf of the motor when running at top speed.

The average operating duty cycle for the robot resulted in a relatively small power dissipation so no special thermal mounting considerations were necessary. The amplifier did require some initial tuning to set the current loop gain and offset. Once the loop gain was set, the motor and amplifier were configured to perform the various experiments. The inputs to the power amplifier from the master servo computer include a digital enable input and an analog setpoint input. The outputs from the power amplifier to the computer include

an analog signal proportional to the actual motor current and a digital output to indicate when a fault occurs. These signals are shown on the wiring diagram in Figure 4.2.

4.2.3 The Network Components

Ethernet has become ubiquitous in computer networks and is increasing in popularity for industrial automation applications [38]. Ethernet was selected because it provides high-speed communications using off-the-shelf components. The issues associated with providing deterministic communications over Ethernet are described in section 4.3.2. The vision nodes and master servo computer were all connected via 100Mb/s Ethernet through a common hub. The hub is a dedicated hub so it is isolated from any outside network traffic. The use of an Ethernet *hub* is preferred because it is more cost-effective and the use of an Ethernet *switch* or *bridge* can introduce short routing delays. Ethernet switches buffer all or part of incoming packets before routing them to the proper destination port [63]. The advantage of an Ethernet switch is that it provides multiple collision domains but this is unnecessary for this application since collisions are prevented by coordinating the traffic with the master servo computer. The protocol and collision prevention details are described in greater detail in section 4.3.2.

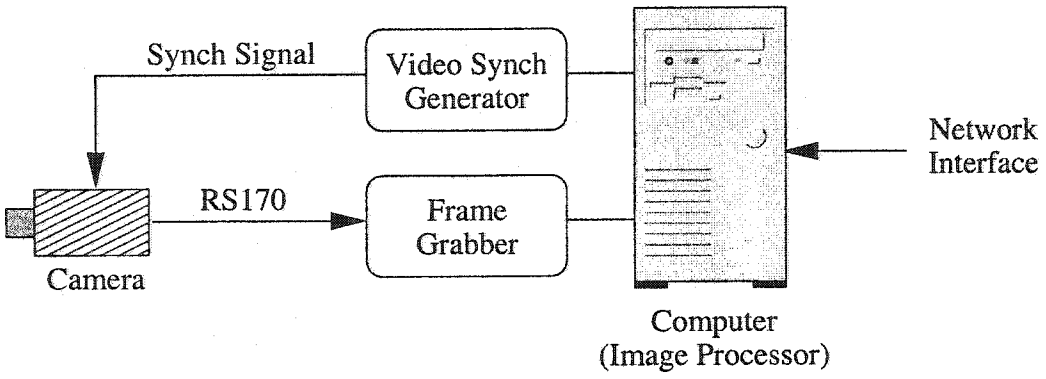


Figure 4.3: Block diagram of a vision node.

4.2.4 The Vision Nodes

The vision nodes are identical and logically consist of the following components:

- Camera
- Frame-grabber
- Video synchronization generator
- Computer connected to the network

Although each vision node was constructed using discrete components, there is no reason why the components could not be consolidated into one "smart camera" package. A smart camera would provide a useful building block for many vision applications beyond the one described here. A block diagram of a vision node is illustrated in Figure 4.3 and each of the components are described in detail in the following subsections.

The Cameras

The cameras are grey-scale CCD RS-170 cameras with a resolution of 640×480 pixels. The video signals are fed over 50Ω co-axial cable allowing the computers to be located remotely from the cameras. Each camera has two BNC connectors for the video signal output and the video synch input. The video synch inputs on the cameras allow the video output to be synchronized by an external source. The video output is an interlaced video signal with 60Hz alternating odd and even video fields which can be combined to provide 30 frames per second.

RS-170 video frames consist of an odd video field and an even video field which are shifted in space by one horizontal scan line and shifted in time by $1/60^{th}$ of a second. Therefore, the vision nodes were configured to process video *fields* rather than video *frames* to increase the sampling rate to 60Hz and to decrease the latency. The eigenspace vision

methods that were employed are insensitive to the vertical distortion present in video fields because they are based on appearance rather than geometry. Multiple vision nodes are used to overcome the limitation of traditional video rates by synchronizing each vision node to capture video fields at different instants in time to improve the effective vision sampling rate.

Significant interlacing artifacts can occur during motion in a scene when two video fields are combined into a single video frame. This is due to the fact that a video frame contains two fields which are taken at two different instants in time. Therefore, half the rows of a video frame will show an object at one point in time and the other half will show the object $1/60^{th}$ of a second later. By processing video fields instead of video frames, these artifacts during motion are removed.

Motion blur is another issue that can arise during periods of fast motion. Most standard cameras have relatively large CCD integration times that are commensurate with their frame-rates. Motion blur can be dealt with in the same way as ordinary image noise or occlusions as described in section 3.4.6. As the effects of motion blur increase, the corresponding projected point in subspace will tend to stray further from the manifold and the position measurement error variance will increase. However, the Kalman filter will continue to provide estimates of the actual position. Motion blur will be most pronounced during the midpoint of a robot motion when joint velocities are at a maximum becoming negligible near the end of a motion when the joint velocities are approaching zero. This may be tolerable for applications which only require precise positioning when a robot approaches its destination position. If accurate position measurements are critical during fast motion, a camera with an electronic shutter control should be used.

There were no special considerations taken for the selection of camera lenses. The aperture was set to allow adequate light and provide a reasonable depth of field. Characterizing lens distortions and camera calibration is not necessary with the eigenspace approach as long as the camera parameters remain the same at run-time as during the learning process.

The placement of the cameras was based on several considerations. Primarily, the cameras were placed so the region of interest around the moving joint was visible at a sufficient resolution. Other factors for camera placement took into consideration such factors as field-of-view and anticipating views that would be less affected by possible occlusions.

The Frame-Grabber Card

The vision nodes were equipped with a PCI frame-grabber card. Video is captured and converted by the frame-grabber card and the information is then transferred using DMA directly into a reserved area in RAM. The DMA operation requires that the destination memory block be a contiguous portion of memory. Consequently, a patch was applied to the Linux kernel to reserve contiguous pages of physical memory space at boot time. The kernel patch was necessary to ensure that a reserved contiguous block of memory of sufficient size can always be allocated for the frame-grabber even when the memory is highly fragmented.

A driver provided an interface for configuring and controlling the operation of the video frame-grabber. Like all devices in Linux[54], the frame-grabber device is accessed logically as a file which appears in the device directory. Once the video driver was installed, it was configured to continuously acquire video fields. The driver asserts a signal (SIGUSR2) when the DMA transfer of each video field is completed. The vision nodes were programmed to respond to the signal by performing a video buffer copy followed by the eigenspace computations.

The Synchronization Circuit

The schematic diagram for the video synchronization circuit is shown in Figure 4.4. Each vision node was equipped with one video sync generator to control the acquisition time of its camera. The circuit is based on the 74ACT715 programmable video sync generator integrated circuit which is available in a single 20 pin package. The 74ACT715 contains

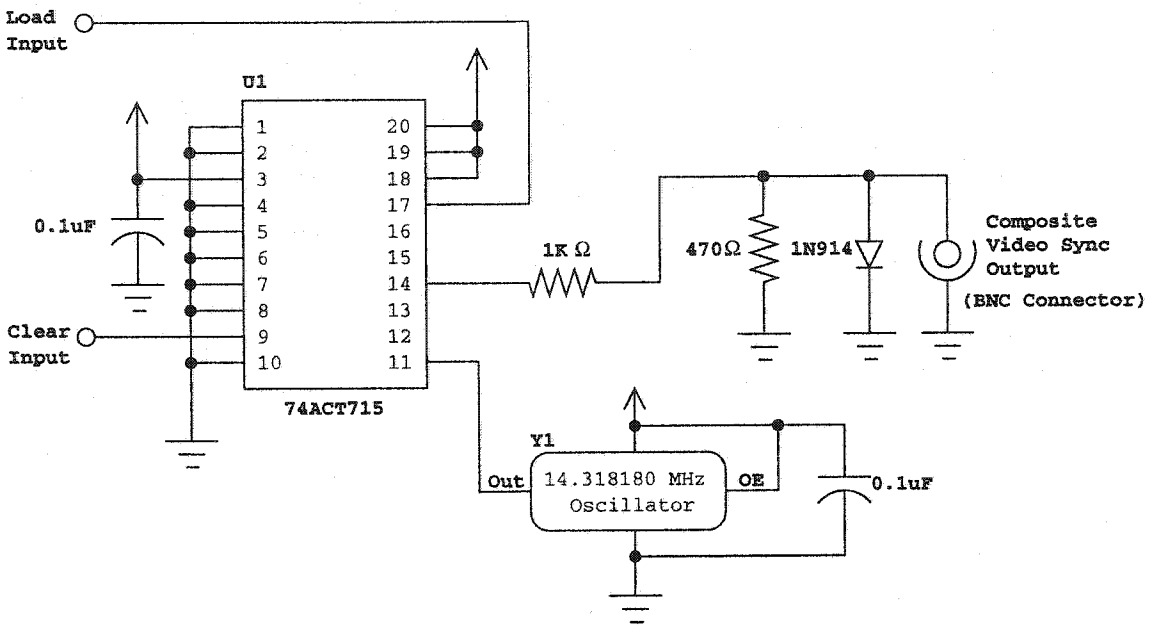


Figure 4.4: Vertical synch generator circuit.

various registers, counters, and comparators which can be programmed to generate the desired timing intervals for both the horizontal and vertical sync pulses. The timing for the circuit is derived from a single clock input pin which is fed with a 14.318180 MHz oscillator to produce the necessary timing for RS-170 video. A single composite video output signal is fed from the sync generator into an external resistor divider to attenuate the output TTL signal to a level appropriate for a video input. This video output signal was then fed through 75Ω coaxial cable to the video sync inputs on each of the cameras. The two control inputs labeled "Load" and "Clear" in Figure 4.4 control and initialize the 74ACT715 sync generator. These control inputs were interfaced to parallel port pins on each of the vision nodes. By toggling these control lines, the timing of the video sync signal and hence the acquisition time of the cameras could be controlled.

The Vision Computers

The vision nodes were built around a standard desktop PC with a 550MHz AMD Athalon processor and 128 MBytes of RAM running the Linux operating system. The synch generator circuit described in section 4.2.4 was mounted in a small cabinet and attached to the side of each of the vision nodes. The control inputs were connected via a signal cable to the parallel port and the video synch output was connected with a co-axial cable to the cameras. Finally, each computer was equipped with a 100Mbps PCI Ethernet adapter to facilitate the communications with the master servo computer.

4.2.5 The Master Servo Computer

The master servo computer was implemented using a 400MHz Intel Pentium II with 128 MBytes of RAM and was connected to the vision nodes via a 100Mbps Ethernet card. The computer was also equipped with an I/O card to provide the interface to the external power amplifier. The I/O card also provided inputs for an incremental encoder that was used for vision training and to provide an independent measure of actual position during various experiments. The I/O card also included a programmable timer chip which provided the time base for the entire system.

4.3 Software

The software is responsible for controlling a real-world mechanical system and must therefore operate in *real-time*. A real-time system is one where the correctness of the system depends not only on the logical result of a computation, but also on the time at which the results are produced [59]. Both the vision nodes and the master servo computer need to run periodic real-time tasks.

Not all processes associated with the direct visual servo require real-time performance.

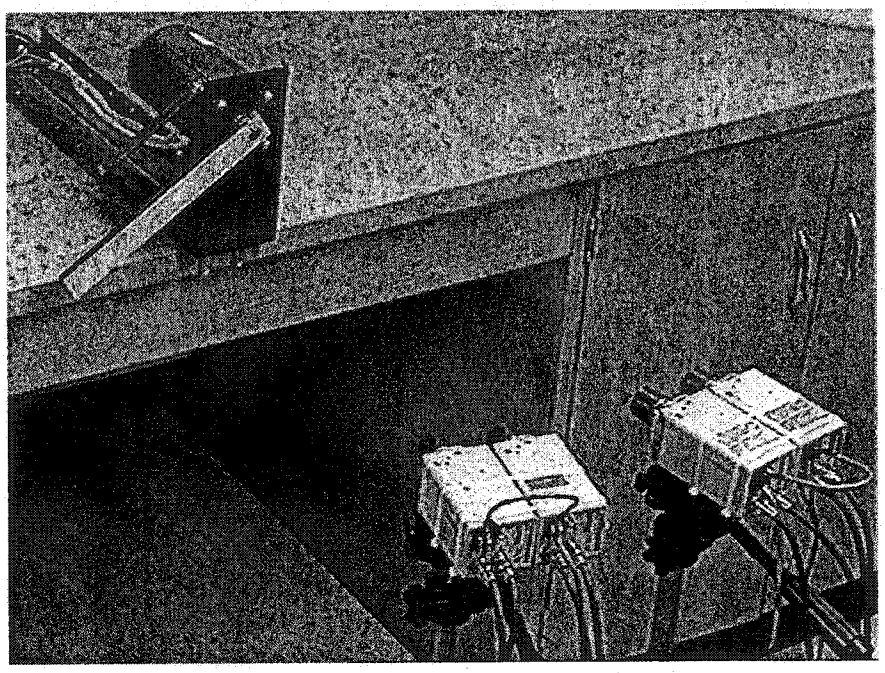


Figure 4.5: Experimental setup with 4 cameras.

The learning phase, which included the calculations of the eigenvectors and parametric manifold, does not require real-time operation and was performed off-line using Matlab. On the contrary, the run-time control loop computations do require real-time performance. Thus the run-time code was implemented using the C programming language running under the Linux operating system using various real-time extensions. The performance of the software relies on the operating system to provide real-time and deterministic service. This is particularly important in the following areas:

- Real-time process scheduling
- Deterministic network communications

First, the scheduler is an absolutely critical component of real-time systems since it is responsible for ensuring that the execution of urgent tasks meets specified deadlines. Second,

the network forms part of the closed-loop feedback path from the vision nodes so it must operate in a fast and deterministic manner. Both of these issues are addressed in the following sections.

4.3.1 Real-Time Process Scheduling

Standard desktop operating systems typically schedule processes to provide fairness. This is normally accomplished by decreasing the average response time which in turn causes the worst-case response time to increase. In addition, user processes can be routinely pre-empted at any time by the operating system or by other processes. Other factors such as demand paging and swapping of memory can further delay the execution of a process. In order to provide deterministic operation, the underlying operating system must provide "real-time" scheduling and control over memory swapping.

Real-time operating systems provide scheduling guarantees instead of fairness to ensure that high priority tasks are not pre-empted by lower priority tasks. Real-time operating systems also ensure that real-time tasks are not swapped from memory. There are numerous different types of real-time operating systems which can be grouped into two categories: *hard* real-time systems and *soft* real-time systems. Hard real-time systems have strict time deadlines which must be met whereas soft real-time systems can tolerate an occasional minor delay.

The master servo computer and the vision nodes both require an operating system that can provide real-time services. However, the vision nodes and the master servo computer require different degrees of determinism. The vision nodes run at a fixed rate and must perform eigenspace computations on each video field every $16.67ms$. Consequently, the vision nodes use a *soft* real-time system since an occasional timing jitter on the order of a fraction of a millisecond is tolerable. In contrast, the master servo computer has a control loop that runs at a rate that is N_c times faster than that of the vision nodes and could

conceivably run at frequencies on the order of 1kHz. The effects of occasional delays worsen as the frequency of the control loop increases. Any jitter in the control loop on the master servo computer affects the synchronization of all the vision nodes. Jitter in the control loop computations also leads to noise in the current loop which introduces torque pulsations that cause excess heating in the motor windings. Therefore, a *hard* real-time operating system was used on the master servo computer.

The Vision Nodes

The operating system selected for the vision nodes was Linux using POSIX.1b extensions. The POSIX (Portable Operating System Interface) standard defines a common interface to improve the portability of source code. The POSIX standard defines numerous operating system functions with the POSIX.1b standard [29] focusing on real-time operations. The key features in POSIX.1b include [20]:

- Memory locking
- Soft real-time scheduling
- Improved signals
- Improved Inter-process communications

The first two features, memory locking and soft real-time scheduling, are required for the vision nodes. The standard provides for soft real-time processes and related data to be locked in memory to prevent them from being swapped to the hard disk. Pages of memory that are swapped to a hard disk can cause paging delays on the order of tens of milliseconds to retrieve. This can be avoided by locking memory pages associated with the video driver and the eigenspace computations. In addition, the eigenvector and parametric manifold arrays located in data memory must also be locked to ensure that the eigenspace calculations can proceed quickly when a video field is captured.

The POSIX.1b scheduling enhancements ensure a predictable order of execution. The standard defines three basic scheduling classes [20]:

- SCHED_FIFO
- SCHED_RR
- SCHED_OTHER

The SCHED_FIFO policy provides preemptive priority-based scheduling without time-slicing. A SCHED_FIFO process will always preempt a lower priority process and will continue to run until it is preempted by a higher priority process or until it voluntarily yields the processor. The SCHED_RR policy provides preemptive priority-based round-robin scheduling whereby priority processes are run for a specified time quantum. In Linux, the SCHED_OTHER policy implements the standard Linux scheduler [5].

The SCHED_FIFO policy was selected to provide simple priority scheduling for the real-time vision computations in the vision nodes and for all network communications. The SCHED_FIFO policy was used to set the process performing the eigenspace computations to run at the highest priority level. Thus, when the capture of a video field is complete, the eigenspace computations can preempt all other running process and then run to completion. However, there will still be a finite dispatch latency before a SCHED_FIFO process can run resulting in a small amount of timing jitter which depends on the performance of the computer hardware and the amount of other system activity [20]. The amount of timing jitter when running on a 550MHz AMD Athalon is negligible for the vision computations which run at a modest rate of 60Hz.

The master servo computer has tighter timing constraints and was implemented using a hard real-time operating system. A hardware timer interrupt running on IRQ 5 was placed at the highest priority. The hardware interrupt was configured to perform the control loop computations and drive the network timing. All other processes running on the master servo computer were set to lower priorities using rate monotonic scheduling [41].

All real-time processes must also be *schedulable*. A real-time process is schedulable if [59]:

$$\sum_{i=1}^{m} \frac{P_i}{T_i} \leq 1 \tag{4.2}$$

where m is the number of periodic processes, and where process i occurs with period T_i and requires P_i seconds of processor time. The eigenspace computations in the vision nodes are performed by one main process ($m = 1$) which runs at video field rates with a period $T_1 = 16.67ms$. From section 2.6.4, the process time for the computations P_1 was measured at $4.62ms$. Therefore, the process is schedulable since:

$$\frac{P_1}{T_1} = \frac{4.62ms}{16.7ms} \approx 28\% \tag{4.3}$$

The low CPU utilization ensures that the eigenspace computations are easily schedulable at normal video field rates.

However, the CPU utilization for the controller computations on the master servo computer is more complicated. The frequency of the control loop depends on the number of cameras N_c. For this reason, the number of cameras presents a potential limitation to the schedulability of processes running on the master servo computer. This issue is explored further in section 4.5.

4.3.2 Network Communications

The network is implemented using the Internet Protocol (IP) over 100Mbps Ethernet. The motivations for using Ethernet are numerous. Ethernet is faster than most proprietary networks with current speeds reaching up to 1Gbps. Also, Ethernet has become ubiquitous in local area networks and uses "off-the-shelf" cards and cables that are cost-effective due to the economy of scales. Finally, the use of Ethernet for manufacturing applications enables automation machinery to interface with ease to existing networks and even the Internet.

However, there are some challenges associated with using Ethernet. Traditionally, one

of the biggest drawbacks to using Ethernet for real-time applications is the lack of determinism. Access contention in Ethernet networks is handled using CSMA/CD (Carrier Sense Multiple Access with Collision Detection) [30]. Although each station listens before transmitting, a packet "collision" can occur if another station attempts to transmit at the same time. When a packet collision occurs, each station will retry the transmission after a random period of time determined by a binary exponential back-off algorithm. This results in a probability that a message sent over Ethernet will be delayed for a nondeterministic period of time. In the extreme, if each retransmission results in another collision, a message can be lost entirely. If Ethernet is to provide the feedback path for a direct visual servo, it must be deterministic.

A multifaceted approach was developed to provide deterministic communications over Ethernet. This was necessary before proceeding with the direct vision servoing application. The strategy hinges on the ability to *prevent* collisions and providing low-overhead packet exchanges. The techniques that were employed to ensure deterministic communications include the following:

- Suppressing superfluous packets
- Master/Slave round-robin network polling
- Use of the UDP protocol

These items are discussed in detail in the following subsections.

Suppressing Superfluous Packets

The first step to improving network performance and determinism is to eliminate all superfluous network traffic. Typical local area networks provide numerous services that result in increased network "chatter" which increases the probability of collisions. Consequently, a private dedicated network was constructed that was completely isolated from other networks to eliminate the possibility of remote traffic interfering with local traffic.

Next, all network services such as e-mail, file sharing, and web servers were turned off. However, there is still traffic that occurs periodically as part of the normal operation of an Ethernet network. This traffic includes DHCP (Dynamic Host Configuration Protocol) requests, DNS (Domain Name Server) requests, and ARP (Address Resolution Protocol) requests. Special considerations need to be made for each of these services to ensure that they do not interfere with real-time traffic.

DHCP traffic is common on many networks to manage IP addresses for network clients and configure various network settings. Shared IP addresses are "leased" to client computers by a DHCP server for a specified period of time after which they can be renewed. Initial DHCP requests take the form of broadcast packets that pass throughout a network in search of a DHCP server. After a reply is obtained from a DHCP server each client can configure its network settings. However, whenever the network address expires new traffic will be generated to renew the address lease. This traffic can potentially collide with real-time traffic. Consequently, all DHCP traffic was eliminated by assigning fixed IP addresses to each vision node and to the master servo computer.

DNS traffic routinely occurs over networks to resolve hostnames to IP addresses [51]. Before the master servo computer can connect with any of the vision nodes it must resolve each of the vision node hostnames to IP addresses. Normally, the master servo computer would perform name resolution by sending a query to a DNS server. By assigning static IP numbers to each vision node on the network, the hostname to IP number resolution can be determined a-priori. In Linux, the name resolution can performed using a local "/etc/hosts" file which contains a table that associates hostnames to IP addresses. The use of a local look-up file eliminates the need for any DNS traffic on the network. The DNS network service can therefore be safely disabled to eliminate the possibility of DNS traffic causing collisions when the vision nodes are running.

Even with DNS traffic disabled, there will still be traffic due to ARP requests. Before

any packet can be sent by the link-layer over an Ethernet network to a destination IP address, the corresponding Ethernet hardware MAC (Media Access Control) address must be known. The hardware MAC addresses are normally obtained using the Address Resolution Protocol (ARP) [44]. An ARP request packet is broadcast on the local network and an ARP reply is sent containing the destination MAC address being requested. Every time an ARP request/reply sequence completes, a temporary entry is made in an ARP *cache* so that ARP requests do not need to be repeated for every packet exchange. The ARP cache provides a table which maps the IP address of a station with its corresponding Ethernet MAC address. The ARP cache entries are normally considered temporary and will trigger new ARP requests after a certain timeout period. The ARP timeout period defaults to 60 seconds for the Linux kernel that was used. Although this timeout period is relatively infrequent compared to the vision node traffic, the periodic ARP requests generate Ethernet frames that have a finite probability of colliding with regular network traffic. ARP requests can be eliminated by making *permanent* entries in the ARP cache. Permanent entries in the ARP cache do not require ARP requests and are not subject to timeouts. Since all the computers are part of an isolated network, all the MAC addresses can be collected a-priori and used to create permanent ARP cache entries in all the vision nodes and in the master servo computer. With permanent ARP cache entries established, there will be no ARP requests or replies generated. At this point the ARP protocol may optionally be disabled. With ARP requests and replies eliminated, there is no possibility of collisions with normal traffic.

Master/Slave Round-Robin Network Polling

With all superfluous traffic eliminated from the network, the problem remains of how to manage the flow of normal traffic to prevent collisions. All the lower protocol layers in Figure 4.7 from the physical layer up to the transport layer are unable to provide guarantees of no collisions. Therefore, collisions on the physical layer will need to be prevented at the application layer. This is accomplished by employing a master/slave round-robin polling

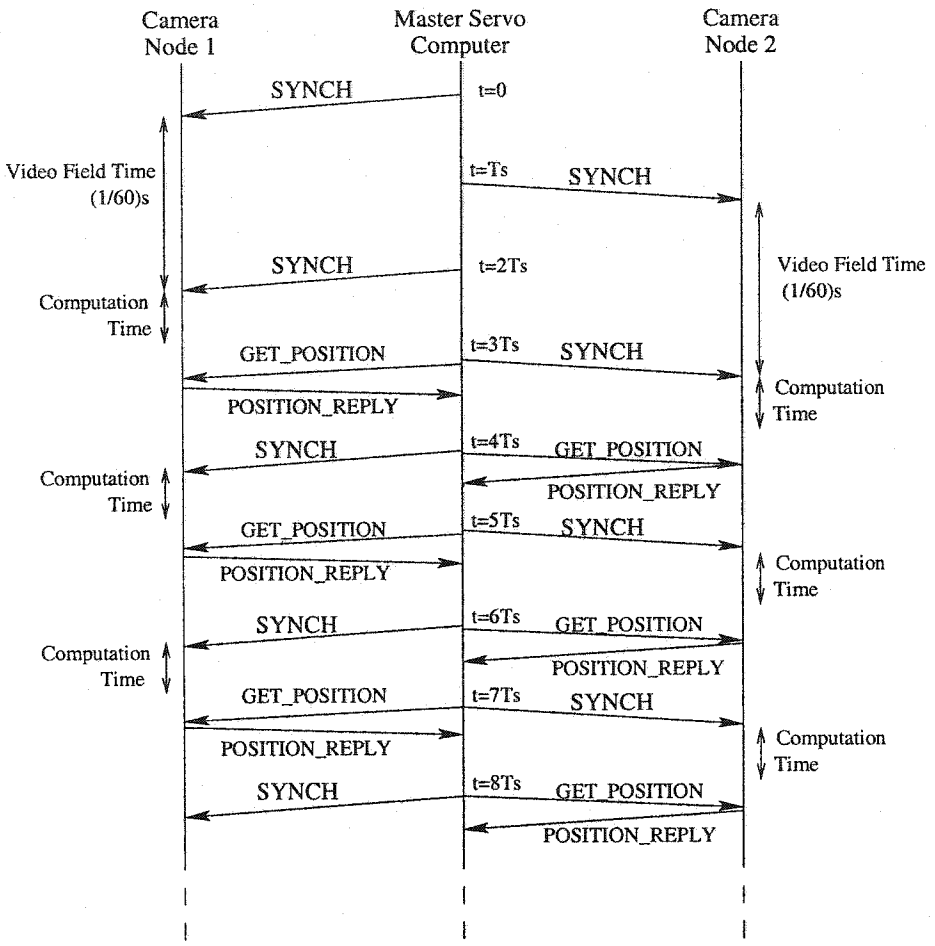


Figure 4.6: Camera network packet diagram for the case of 2 cameras.

protocol. The master servo computer serves as a *network master* responsible for polling each of the vision nodes which act as *network slaves* that "speak" only when "spoken" to. The polling proceeds in a round-robin fashion at times governed by a real-time hardware clock on the master servo computer.

The master servo computer acts as the network master so all network transmissions and sample times are derived from its clock. Thus the polling scheme serves two purposes: to

prevent collisions and to to synchronize the vision nodes. Polling allows the master computer to control network access thus eliminating the possibility of collisions. The polling also allows the master servo computer to drive the overall system timing. Alongside the polling packets, the master servo computer also sends synchronization packets at video rates to each vision node in a round-robin fashion. Upon receipt of a synchronization packet, each vision node will adjust its camera synchronization to ensure that the camera remains properly synchronized and does not drift with respect to other cameras. This is done by toggling a port pin connected to the video synchronization circuit shown in Figure 4.4. At the end of every video field, each vision node computes robot position using the eigenspace methods described in Chapter 2. When a vision node is polled, it replies with the most recent results which are sent over the Ethernet network to the master servo computer. This ensures that all transmissions remain synchronous with the master servo computer and that the feedback delays remain an integer multiple of the sampling time T_s. This is important for the predictor in the Kalman filter since the transport delays were modelled using a series of unit delays.

The master servo computer thus forms the time-base upon which all vision nodes synchronize their acquisition times and remain in lock-step relative to one another. Consequently, the master servo computer is equipped with a real-time operating system to minimize timing jitter, particularly as the number of cameras N_c becomes large. Each of the vision nodes acquire video fields at sample times that are offset from each other to increase the effective sample rate as described in section 2.3. Each vision node thus completes their respective vision computations at times offset by T_s seconds from each other (where $T_s = \frac{1}{60N_c}$). Conveniently, this provides natural time windows of T_s seconds where each vision node can be polled for results while other vision nodes are in different stages of image acquisition or image computation. Collisions are thus avoided since vision nodes no longer compete for network access but are granted an opportunity to transmit at a rate equal to the video field time.

Therefore, the network has three distinct types of packets associated with video synchronization, polling, and replies. The three different packet types defined in the application layer are as follows:

SYNCH Packet a packet sent from the master servo computer to each vision node to synchronize the video signals (1 byte payload)

GET_POSITION Packet a packet originating from the master servo computer requesting a vision node to send its latest position feedback (1 byte payload)

POSITION_REPLY Packet a reply packet from a vision node containing a position measurement *and* the corresponding Euclidean distance (8 byte payload)

The SYNCH packets and GET_POSITION packets are both small packets which contain a payload of one byte to differentiate the packet type. The POSITION_REPLY packet payloads consist of one 32-bit integer for the position measurement and one 32-bit floating point number representing the square of the Euclidean distance. The square of the Euclidean distance is required to determine measurement error variance for the Kalman filter according to Equation 2.29. The master servo computer receives all the results from each vision node as interleaved packets arriving in the same relative order as the cameras are synchronized. Upon receipt of the vision feedback at each time step, the master servo computer performs the Kalman filter and controller computations and updates the servo outputs.

A diagram illustrating the sequence of packets exchanged between a system comprising two vision nodes and a master servo computer is shown in Figure 4.6. The diagram starts at time $t = 0$ when the first SYNCH packet is sent to the first camera node followed by a SYNCH packet to the second vision node T_s seconds later. The SYNCH packets are continuously sent at rate of 60Hz to each vision node with the sample time T_s forming the timebase upon which all network traffic is based. Position measurement results are

available in the vision nodes after a time equal to the transport delay (video field time plus computation time) after the SYNCH packets are received. After an initial synchronization startup time, the results from the vision nodes become available and the network begins continuously polling and synchronizing the vision nodes.

Although this scheme works well, it has some shortcomings. One major disadvantage is that the master servo computer forms a single point of failure. The system could potentially tolerate the loss of one or more vision nodes but not the loss of the master servo computer which is responsible for for both the control of the robot and the network. Although polling solves network contention problems and provides synchronization, it also requires significant network overhead. The effect of network polling overhead on the maximum number of vision nodes is discussed in section 4.5. For this reason, this scheme may not be appropriate for general automation applications.

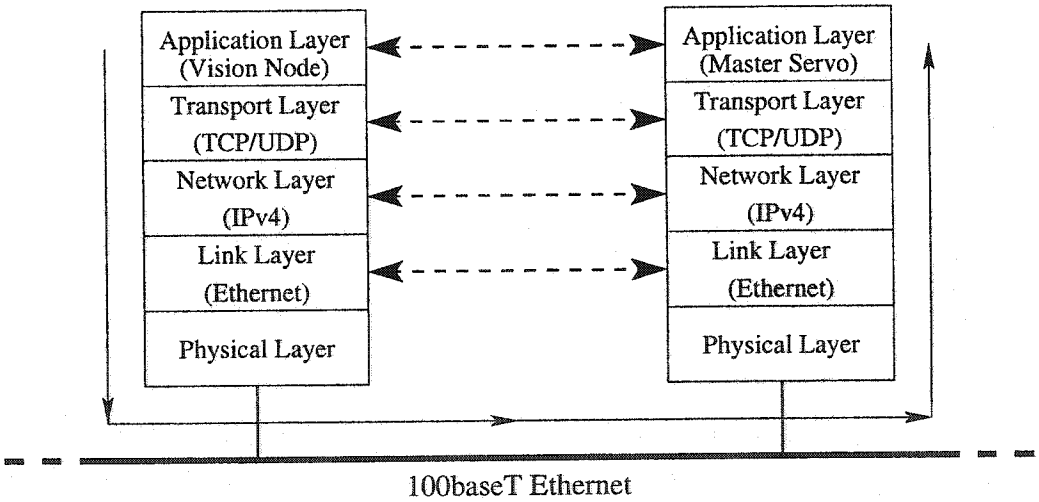


Figure 4.7: Network layers present in the distributed camera system.

Using the UDP Protocol

The exchange of packets must not only be free of collisions, it must be fast and efficient. Networks normally use different layers to provide different services that facilitate communication between a sender and a receiver. The standard OSI (Open Systems Interconnect) reference model defines 7 different protocol layers [63]. A message sent from one application program must pass down through each of the layers in the protocol stack then pass over a physical medium before ascending back up the protocol stack to a remote application. The time required to pass a message through the network layers is determined in part by the network protocols being used. Because the network forms part of the feedback path in a closed control loop it *must* provide fast and deterministic service.

The layers typically found in networks employing the Internet Protocol (IP) are illustrated in Figure 4.7. The two main transport layer protocols used in conjunction with the Internet Protocol are UDP (User Datagram Protocol) and TCP (Transmission Control Protocol). UDP is a simple "send and forget" protocol, whereas TCP provides numerous services including [60]:

- windowed flow control
- slow-start and congestion avoidance
- acknowledgments and retransmission of lost packets

However, these services provided by TCP are unnecessary for this application and would only serve to increase the overhead and delays in the network. The windowed flow control provided by TCP is used to prevent a sender from exceeding the buffer space available at the receiver. This feature is not required since the packet size and rate of data exchange between the vision nodes and the master servo computer is extremely low. In fact, the real-time operation and latency model in the Kalman filter would be invalidated if any packets in the network were to be subjected to unpredictable delays that could be imposed by flow

control mechanisms. The packet sizes are 1 byte for SYNCH and GET_POSITION packets and 8 bytes for POSITION_REPLY packets which are only sent at a rate of 60Hz for each vision node. The slow-start feature in TCP is also problematic since it would prevent real-time and deterministic packet exchanges in the network. Congestion avoidance is also unnecessary since the packet exchanges are controlled to ensure there are no collisions and there are no routing queues in the path since a direct connection exists between all nodes in the network. Finally, acknowledgments and packet retransmission are not needed. Acknowledgments increase overhead and are not necessary since there can be no collisions and the probability of dropping a packet is almost zero. In the unlikely event of packet loss due to a large noise spike or hardware "glitch", the controller can continue to operate. If a synchronization packet is missed by a vision node, the video synch generator continues to run with its own built-in oscillator. Synchronization packets are currently sent every video field time to simplify the implementation of the software state machine. Theoretically, synchronization packets are only necessary occasionally to ensure that the internal oscillators do not drift too far from the master servo computer clock. If a position reply packet is lost, the Kalman filter can simply estimate the current position until the next position measurement is received. In this application, quality of service (QoS) for real-time performance takes precedence over reliable communications. Therefore, the advantages normally provided by TCP connections are liabilities for this particular real-time application.

The advantages of UDP include the following [60]:

- support for broadcasting and multi-casting
- requires no connection setup or tear-down
- requires less code to implement

Broadcasting and multi-casting are not required for this application but low-overhead and deterministic communications are critical. UDP provides fewer services than TCP but experiments comparing latency and throughput show that UDP performs better than TCP [51].

Unlike normal I/O devices in Linux, which are controlled with a device file, the network interface receives data from the network directly into kernel memory [5]. Thus network I/O is not performed with the regular device file system calls but rather using special *socket* communication system calls. The communications were implemented using UDP socket communications [60] with connections established between the master servo and each vision node on a preassigned port number and using a special "connect" packet. Once the connections are established, UDP packets can be exchanged between the vision nodes and the master servo computer without any additional overhead. The use of the UDP protocol ensures that the propagation delay of a UDP packet through the protocol stack to the remote computer is as fast and efficient as possible.

4.4 Tele-Robotics Extensions

Tele-robotics allows a remote user to operate robotic equipment from a distance in situations where it is not practical or safe for a human to be present. Tele-robotics applications are becoming increasingly important and span applications ranging from remote surgery to space exploration. For this reason this section was included to demonstrate how the direct visual servoing concepts can be readily extended to support tele-robotic applications.

The long round-trip delays associated with transmitting visual data to a remote site makes it difficult or impossible to dynamically close a vision loop over a remote link. Communication delays and a lack of adequate QoS (Quality of Service) guarantees imply that the network connection to the remote site should not form part of a closed-loop feedback path. The transport delays also mean that the remote operator may not have adequate time to respond to unexpected events that may occur at the remote worksite. This is particularly a concern for space exploration where the round-trip delay time can become quite large. Visual servoing is an important enabling technology for tele-robotics. Visual servoing allows a task to be specified in terms of visual features by a remote human operator

and visual servoing can be used to execute the task locally [22]. Tele-robotics is an open research area which can benefit greatly from advances in visual servoing.

The implementation described in the preceding sections has several attractive features that enable it to be easily adapted for tele-robotic applications. First, the network implementation is based on the standard IP protocol which allows interconnection with most existing communication infrastructures including the Internet. Second, the image processing is performed using eigenspace techniques which provide a compact representation suitable not only for analysis but also for image compression. Eigenspace representations can be used to compress images for transmission with minimum reconstruction errors as described in section 2.5.2.

The tele-robot extensions that were implemented allow a user working from a remote client computer to specify robot tasks to be performed by the visual servo. Images compressed using eigenspace techniques are simultaneously streamed back to the client computer to enable the remote user to monitor and observe the operation of the of the visual servo. In the following section, the additional hardware and software necessary to to support these tele-robotic operations over the Internet will be described.

4.4.1 Hardware to Support Tele-Robotic Operation

The tele-robotic setup was based on the same distributed network of vision nodes shown in Figure 2.1. The setup uses the same off-the-shelf components including 550MHz AMD Athalon computers coupled with RS-170 grey-scale cameras for the vision nodes. The master servo computer hardware remained the same except for the addition of a second network interface card. The extra network interface provides a connection to the Internet to allow communication with a remote client computer. The internal network of vision nodes remains isolated from the outside world because packets are not permitted to be forwarded from the Internet to the internal network. This ensures that the internal network continues

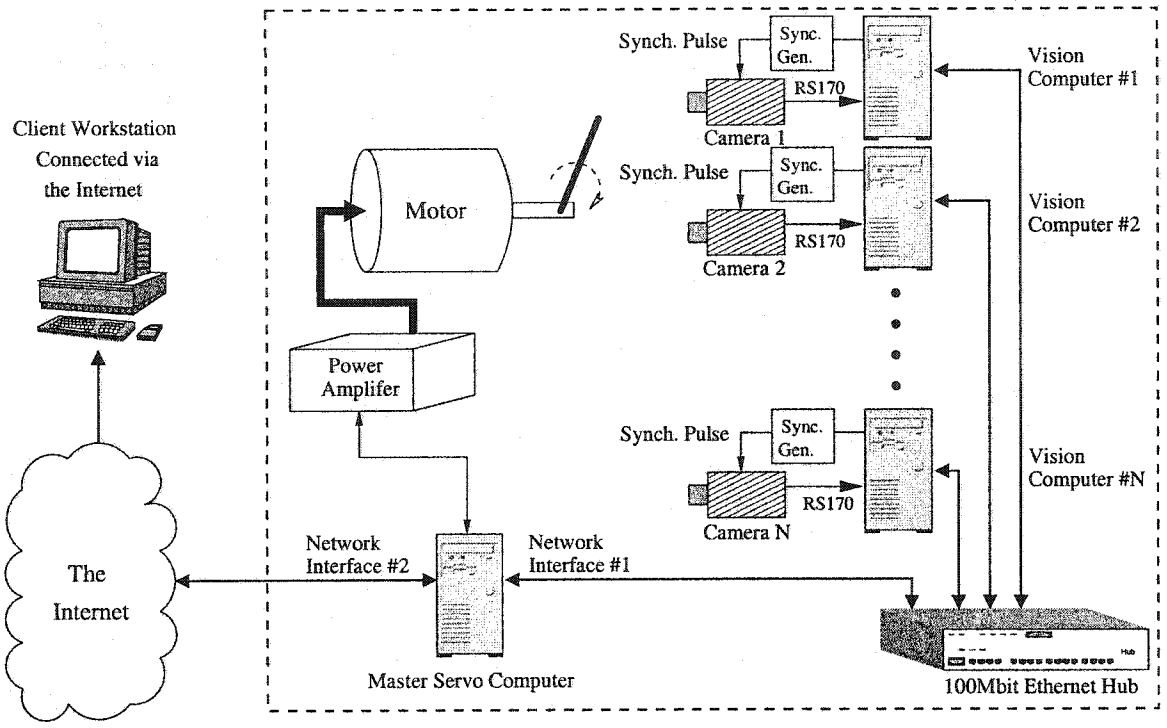


Figure 4.8: Tele-robotic System Diagram.

to function without the threat of packet collisions even while packets are exchanged with the outside world. A diagram of the internal network with the external connection to a remote client through the Internet is shown in Figure 4.8.

4.4.2 Software to Support Tele-Robotic Operation

Although the additional hardware requirements for tele-robotic operation are minimal, there is a substantial amount of additional software required. The extra software includes communication and control software for the master servo computer and software for the remote client computer. Priority scheduling is used in the master servo computer to ensure that the additional overhead of tele-robotic functions do not effect the determinism of the

underlying visual servo loop. The two main functions provided by the tele-robotics extensions are facilities for sending remote commands to the robot and the ability to send visual information for remote monitoring.

Software for Remote Monitoring

Eigenspace methods offer a convenient subspace for image compression to efficiently transmit images to a remote computer for visual monitoring. As described in section 2.5.2, an image can be reconstructed from an n dimensional pattern vector with minimum reconstruction error using a subset of k vectors where $k < n$ as follows:

$$\hat{\mathbf{y}} = \sum_{i=1}^{k} x_i \mathbf{u}_i \tag{4.4}$$

where $\hat{\mathbf{y}}$ is the reconstructed image and x_i and $\mathbf{u}_i$ are the i^{th} eigenvector coefficient and eigenvector respectively. Although 5 eigenvectors are used for position determination, in section 2.6.6 it was shown that more eigenvectors ($k \gg 5$) are required to reconstruct an image of good quality.

The k eigenvectors must be sent to the remote client computer before any image reconstruction can proceed. Once the eigenvectors are received, the master servo computer can begin to stream images of the robot operation by sending only the eigenvector coefficients. However, before this can be done, the master servo computer must obtain the eigenvector coefficients from the vision nodes. Up to this point, the vision nodes have been discarding the eigenvector coefficients after sending the position and Euclidean distance information in the POSITION_REPLY packets. Therefore, to support the image transmission to the remote computer, the POSITION_REPLY packets described in section 4.3.2 will need to be expanded to include the eigenvector coefficients. This will require an additional packet overhead of 80 bytes assuming 20 eigenvector coefficients represented with single precision floating-point numbers. The images are then forwarded to the remote client workstation using only 80 bytes to represent a 34001 pixel image resulting in a compression ratio

of approximately $425:1$. This high compression ratio comes at the cost of increased processor utilization for the subspace computations in both the vision nodes and the remote client computer.

The number of eigenvectors required for image reconstruction will be more than the 5 eigenvectors required for accurate position determination. The extra computation time required for the additional eigenvectors can significantly increase the latency in the position feedback. Consequently, the computation of the first 5 eigenvector coefficients and the position are given first priority in the vision nodes since the latency of the position feedback directly affects the performance of the visual servo. In contrast, the streaming of eigenvector coefficients for remote monitoring can be performed on a best-effort basis. Thus *after* the POSITION_REPLY packet is sent the remaining eigenvector coefficients can be computed using whatever spare processor time is available before the next video field is acquired. As a result, the eigenvector coefficients are delayed by one video field time before they are sent with the POSITION_REPLY packet. Therefore, the position and Euclidean distance information in the POSITION_REPLY packets are always the most recent values, whereas the eigenvector coefficients are stale by one video field time.

Remote Client Computer Software

Before a remote client computer can begin any tele-robotic operations it must first initiate a connection with the master servo computer over the Internet. A TCP socket connection is made to a predefined port on the master servo computer to download the image eigenvectors. A TCP connection is used to ensure that the initial eigenvectors are reliably received before image reconstruction begins. Once the eigenvectors have been received the TCP connection is closed and the images begin by simply transmitting the coefficients for each image. A UDP connection is established to "stream" the coefficients since UDP packets provide a low-overhead protocol suitable for streaming images. If a UDP packet gets lost or dropped in transit, a video frame is simply dropped with little or no consequences to the

remote user. A second TCP connection is established to allow the client to send command and control information to the robot. These commands allow the remote user to enable or disable the servo loop, to set destination position setpoints, and to set loop gains. The TCP protocol is used to provide reliable transport for user commands and messages since loss of command packets can result in serious consequences.

The remote client was implemented using Java because it has strong built-in support for networking and graphics and it is platform independent. The Java Native Interface (JNI) was used to unmarshall the contents of binary packets into native Java data types. The remote workstation used various built-in Java classes to render the resulting images on a graphics display. Because the images are based on video fields rather than video frames, the aspect ratio of the reconstructed images is distorted in the vertical dimension. This can be corrected by either reconstructing video frames from two consecutive video fields or by vertically interpolating each video field to restore the aspect ratio. Besides the overhead required to render the images, the computational effort for image reconstruction is similar to projecting an image into eigenspace. Consequently, the sustainable video rate at the remote monitoring workstation is limited by available computing power and is also dependent on network congestion and available bandwidth. Finally, although only images from one camera were used, the master servo computer could be configured to allow the remote user to select the coefficients that are sent to switch views between different cameras.

Local Task Supervision

In cases where the round-trip communication delays are substantial, the remote user can monitor the robot but may not have time to intervene if something unexpected occurs while the robot is performing a task. Unexpected events that can occur while performing a task may include unanticipated obstacles, occlusions, or the malfunction of the robot itself. In situations such as space exploration with substantial round-trip delays, the actual visual

feedback from a robot can arrive long *after* a robot task has actually completed. In such cases, it is impossible for a remote user to supervise the robot. Consequently, there is a strong motivation to enable the robot itself to automatically respond to unexpected events while performing a task.

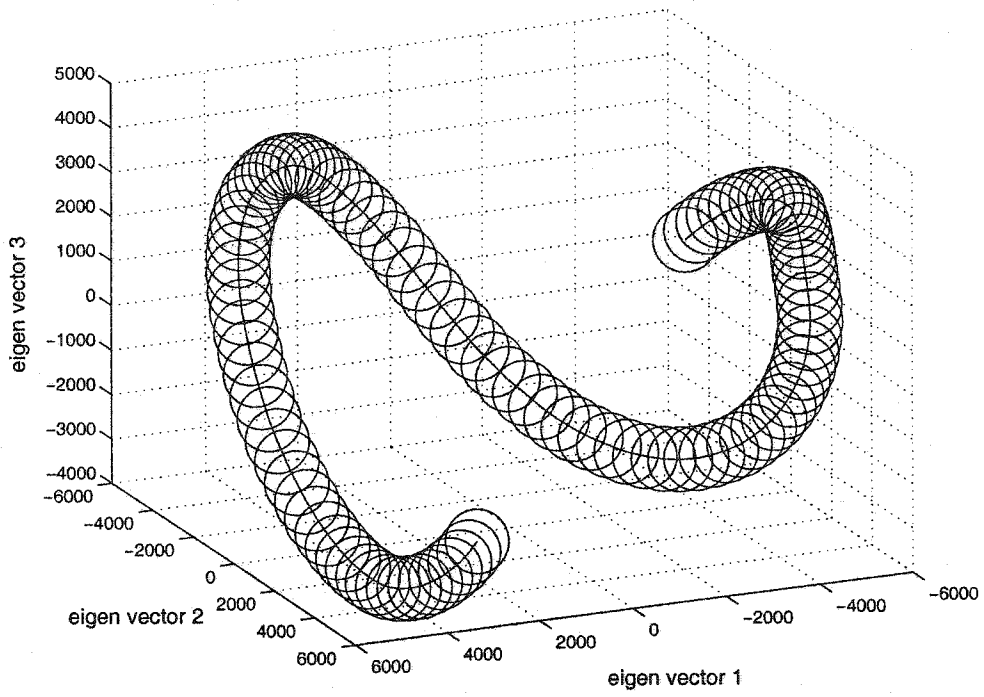


Figure 4.9: Plot of the corresponding "expected appearance tube" around the manifold.

The appearance based methods used in the vision system can be employed to supervise a task while it is being performed. Normally, a specified task will involve a position trajectory which will have a corresponding trajectory along the manifold in subspace. Ideally, each image captured during a motion should project directly to a point on the manifold. However, various sources of image noise will cause the projected image points to stray from the manifold. Unexpected events, such as unanticipated objects entering the scene or occlusions, will result in projected points that will generally stray further from the manifold. Consequently, a Euclidean distance threshold can be established to detect when the

images captured during a task stray too far from the manifold. This corresponds to ensuring that each step in the task does not stray too far from an "expected appearance" based on the training images. The expected appearance essentially forms a "tube" around the manifold where the radius of the tube is equal to the Euclidean distance threshold. An example of an "expected appearance tube" for the planar robot is illustrated in Figure 4.9.

If a projected point falls outside the "expected appearance tube" the robot can be stopped or disabled until the remote user decides if it is safe to proceed. If the appearance of the task changes due to unexpected motion of the robot or due to stray objects entering the workcell the robot can be shut down. This technique can be further enhanced by reducing the Euclidean distance threshold around critical regions of the manifold where more precise positioning is required resulting in a tube of varying thickness. However, this technique also has some shortcomings, namely that the robot could be disabled due to a single large noise spike in the image.

Another option is to perform local task supervision based on state information available in the Kalman filter. As described in section 3.4.6, the Kalman filter tracks the overall accuracy of the state estimates in the $\mathbf{P}_k$ covariance matrix. Therefore, the servo computer could supervise the operation of the robot by monitoring the values in the $\mathbf{P}_k$ and taking appropriate steps if the matrix values exceed a predetermined threshold.

4.5 Limitations on the Number of Cameras

The theoretical upper limit on the number of cameras N_c that could be used in this scheme has not been tested but it can be calculated. The concept of network-synchronized cameras is not bound to one particular type of network but the limitations are a function of the network bandwidth and the available processor time.

The limitation on the number of vision nodes depends on two fundamental constraints:

- Network bandwidth

- Processor time for the main control loop in the master servo computer

The actual limit will be determined by which of these two limitations are encountered first.

4.5.1 Network Limitations

As the number of vision nodes increase the demands on the network also increase. The system depends on real-time communications between all the vision nodes so all packets must be sent immediately with no packets being queued or delayed. Ethernet is the network that was chosen for the implementation so it will be examined in detail. However, the analysis for other types of network implementations will proceed in a similar fashion.

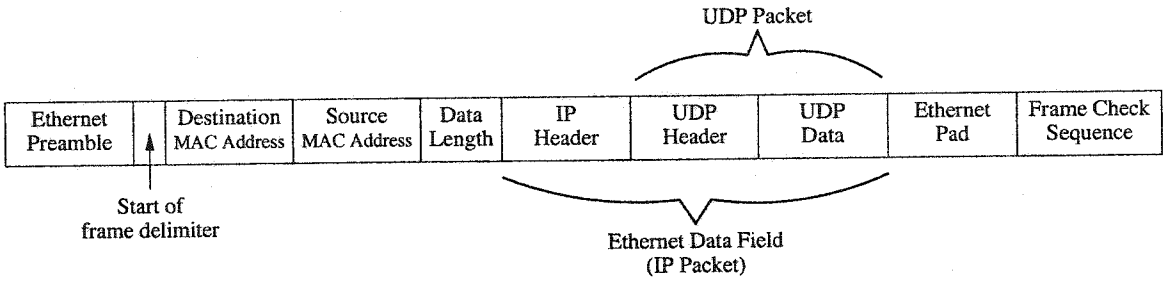


Figure 4.10: Structure of an Ethernet Frame encapsulating a UDP packet.

To determine the network limitations the Ethernet frame lengths and packet rates must be analyzed. The application layer sends all information encapsulated in UDP packets. The UDP packets are encapsulated in IP packets, which in turn are encapsulated in Ethernet frames. The structure of an Ethernet frame containing a UDP packet is shown in Figure 4.10. The IEEE 802.3 standard [30] states that a valid frame must have a maximum length of 1518 bytes and a minimum length of 64 bytes. If the frame length falls below 64 bytes a special Ethernet *Pad* field is used to automatically add bytes to satisfy the minimum frame length. The minimum frame length restrictions are necessary to ensure that collisions can be properly detected.

Label	*Size (bytes)*
Ethernet Preamble	7
Start of Frame	1
Destination MAC Address	6
Source MAC Address	6
Ethernet Data Length	2
IP Header	20
UDP Header	8
SYNCH Byte	1
Ethernet Pad	17
Ethernet Checksum	4
Total Bytes	72

(a) SYNCH Packet

Label	*Size (bytes)*
Ethernet Preamble	7
Start of Frame	1
Destination MAC Address	6
Source MAC Address	6
Ethernet Data Length	2
IP Header	20
UDP Header	8
GET_POSITION Byte	1
Ethernet Pad	17
Ethernet Checksum	4
Total Bytes	72

(b) GET_POSITION Packet

Label	*Size (bytes)*
Ethernet Preamble	7
Start of Frame	1
Destination MAC Address	6
Source MAC Address	6
Ethernet Data Length	2
IP Header	20
UDP Header	8
Position Data	8
Ethernet Pad	10
Ethernet Checksum	4
Total Bytes	72

(c) POSITION_REPLY Packet (with no eigenspace coefficients)

Label	*Size (bytes)*
Ethernet Preamble	7
Start of Frame	1
Destination MAC Address	6
Source MAC Address	6
Ethernet Data Length	2
IP Header	20
UDP Header	8
Position and Coefficients	108
Ethernet Pad	0
Ethernet Checksum	4
Total Bytes	162

(d) POSITION_REPLY Packet (with 25 floating-point eigenspace coefficients)

Table 4.1: Tables summarizing the overhead of the various UDP Packets transmitted in an Ethernet Frame.

Computing Network Limits

The size of the Ethernet frames for each of the packet types used in the system are summarized in Table 4.1 (a) to (d). The size of the Ethernet frames for SYNCH packets and GET_POSITION packets are shown in Table 4.1 (a) and (b) respectively. The SYNCH and GET_POSITION packets both contain one byte which is used to differentiate the packets. Since both these packets contain only one data byte the resulting Ethernet frame requires an additional 17 bytes of padding to meet the minimum frame length requirement. The resulting frame, including the preamble and start of frame delimiter, require a total of 72 bytes. The POSITION_REPLY packet summarized in Table 4.1 (c) contains 4 bytes for the position estimate and 4 bytes for the Euclidean distance measurement for a total data length of 8 bytes. The resulting Ethernet frame length with 8 data bytes still requires an additional 10 byte pad to meet the minimum frame length requirement resulting in a total frame length that is also 72 bytes. The time to send a 72 byte Ethernet frame t_{frame} using 100Mbps Ethernet is:

$$t_{frame} = \frac{\#\text{bits}}{\text{Bandwidth}} = \frac{72 \times 8\text{bits}}{100 \times 10^6\text{bits/s}} = 5.76\mu s \tag{4.5}$$

In addition, the IEEE 802.3 specification [30] requires that each packet must be separated by a minimum inter-frame gap of 96 bit times or $0.96\mu s$. Thus the minimum time required to send one packet is:

$$t_{packet} = t_{frame} + t_{gap} = 5.76\mu s + 0.96\mu s = 6.72\mu s \tag{4.6}$$

During each video field three packets are exchanged between each vision node and the master servo computer. First a synch packet is sent to each vision node which are later followed by position request and reply packets. Thus the total network traffic per vision node comprises three UDP packets per video field time. The network traffic for all N_c vision nodes must occur within one video field time:

$$N_c \times (3 \times t_{packet}) \leq \frac{1}{60}\text{second} \tag{4.7}$$

The network traffic will increase linearly with the number of vision nodes N_c until the network utilization saturates. Hence, the maximum number of vision nodes supported by fast Ethernet can be found by rearranging Equation 4.7 to solve for the number of cameras:

$$N_c \leq \left\lfloor \frac{1}{60 \times 3 \times t_{packet}} \right\rfloor \tag{4.8}$$

Substituting the time per packet t_{packet} determined in Equation 4.6 into Equation 4.8 results in a maximum of 826 cameras that can be supported by fast Ethernet. This limit represents an upper bound at which point the network is saturated. Therefore it is possible to contemplate systems ranging from two standard cameras to systems comprising several hundred inexpensive cameras.

Network Limits with Tele-robotic Extensions

However, if the tele-robotic extensions described in section 4.4 are employed, the size of the packets will increase. The tele-robotic implementation requires that, in addition to position and Euclidean distance, the vision nodes must also transmit several eigenspace coefficients. The eigenspace coefficients are sent to a remote client computer where they are used to reconstruct an image of the robot to enable remote monitoring. Assuming that 25 single-precision floating-point eigenspace coefficients are used for reconstruction, the total size of the POSITION_REPLY Ethernet frames will increase to 162 bytes as indicated in Table 4.1 (d). The time to send a 162 byte Ethernet frame t_{frame2} using 100Mbps Ethernet is:

$$t_{frame2} = \frac{\#\text{bits}}{\text{Bandwidth}} = \frac{162 \times 8\text{bits}}{100 \times 10^6 \text{bits/s}} = 12.96\mu s \tag{4.9}$$

With the minimum inter-frame gap of 96 bit times, the minimum time required to send the reply packet is:

$$t_{packet2} = t_{frame2} + t_{gap} = 12.96\mu s + 0.96\mu s = 13.92\mu s \tag{4.10}$$

The SYNCH and GET_POSITION packets remain the same size so equation 4.8 can be modified to give the maximum number of cameras as:

$$N_c \leq \left\lfloor \frac{1}{60 \times (2 \times t_{packet} + t_{packet2})} \right\rfloor \tag{4.11}$$

Substituting the packet times results in a maximum of 609 cameras that can be supported by fast Ethernet. Even with the increased packet payloads required for tele-robotic operations, a large number of cameras can be supported.

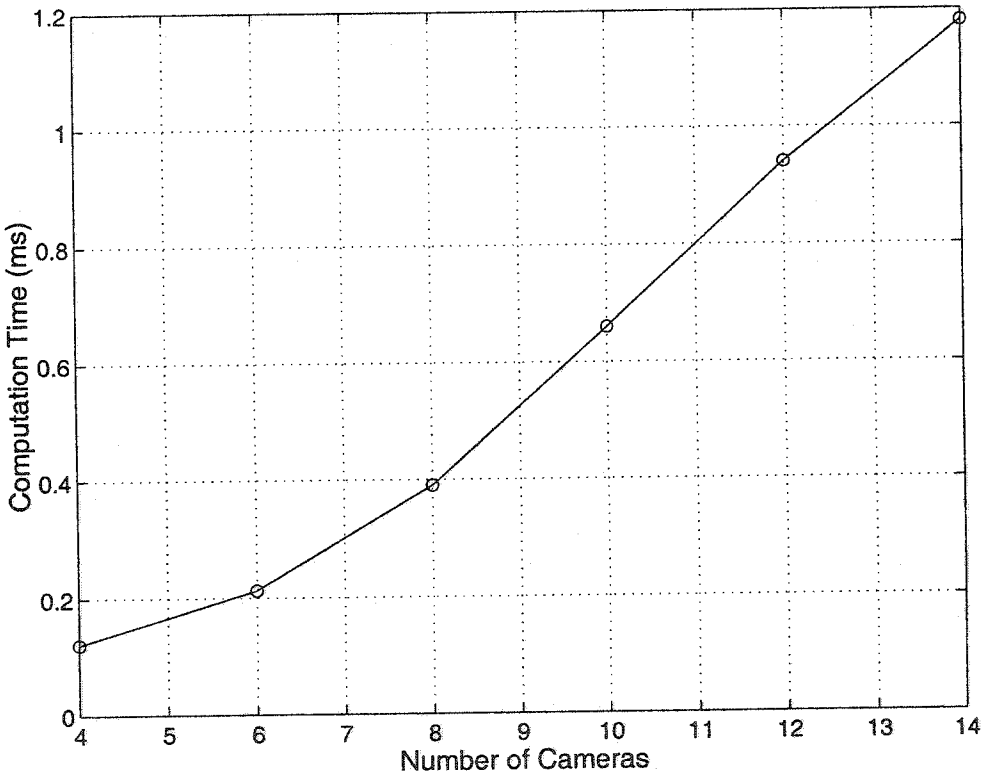


Figure 4.11: Experimental results of master servo computation time vs. the number of cameras.

4.5.2 Processor Time Limitations

Another limitation to the number of vision nodes is available processor time. The computation time required on each individual vision node is constant since the vision nodes run independently and in parallel with all the other vision nodes. However, the processor time required for the control loop on the master servo computer will increase as the number of vision nodes increases. Hence the schedulability of the main control loop becomes a limitation as the number of vision nodes increase.

To analyze the schedulability of the control loop requires information regarding both the computation time and the frequency of the control loop. The frequency of the control loop is simply $f_s = 60N_c$ (from Equation 2.1). Unfortunately, the computation time of the control loop is not constant but also increases as the number of vision nodes increase. This can be shown by noting that $T_s = \frac{1}{f_s}$ and substituting this into Equation 3.42 to form:

$$n = \left\lceil \frac{t_{delay}}{\frac{1}{60N_c}} \right\rceil = \lceil t_{delay} \times 60N_c \rceil \tag{4.12}$$

where n is the number of transport delay states, t_{delay} is the overall transport delay, and N_c is the number of vision nodes. Clearly the number of delay states and hence the total number of state variables increase linearly as the number of vision nodes increase. The dimensions of the matrices in the Kalman filter are determined by the number of state variables. The state estimate matrix, the Kalman gain matrix, and the state input matrix are $(n \times 1)$ column vectors where n is the number of state variables. The storage requirements for these vectors will grow by $O(n)$. The state transition matrix, the system covariance matrix, and the state covariance matrix are all $(n \times n)$ square matrices where n is number of state variables. The storage requirements for these matrices will grow by $O(n^2)$. In addition to increasing storage requirements, the computational requirements for the Kalman filter will also increase as the number of state variables increase. The number of operations to perform multiplication of a $(n \times n)$ matrix with a $(n \times 1)$ column vector will grow by $O(n^2)$. The number of operations to perform matrix multiplications of two $(n \times n)$ matrices will

grow by $O(n^3)$. Therefore as the number of vision nodes increase the matrix dimensions in the Kalman filter also increase resulting in a quadratic increase in both storage requirements and computation time for the control loop.

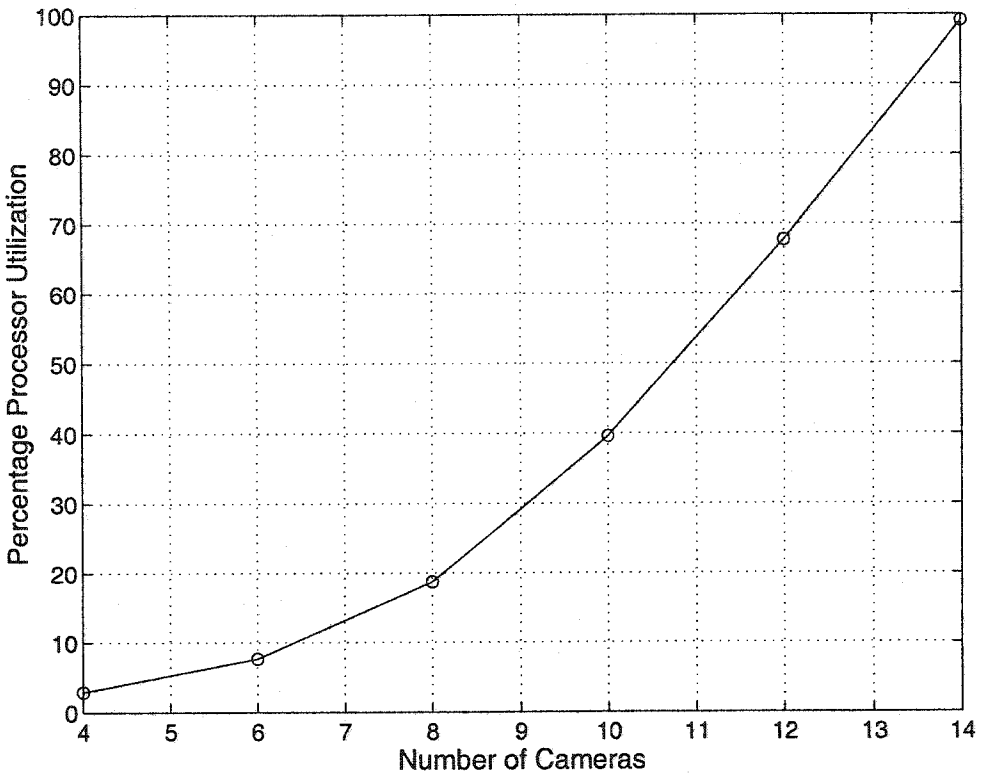


Figure 4.12: Experimental results of master servo processor utilization vs. the number of cameras.

The actual processor time required on the master servo computer for the main control loop including the Kalman filter computations was determined experimentally. A digital storage oscilloscope was employed to measure the computation time for the control loop running on a 400MHz Pentium II processor. Several experimental measurements were repeated for the computation time of the control loop as the number of state variables were varied to simulate the presence of more vision nodes. The results are shown in Figure 4.11 with a plot of the computation time vs. the number of simulated vision nodes. This plot

verifies that the computation time is increasing quadratically with the number of cameras. From this data it is possible to determine the processor utilization for the control loop in the master servo computer. The processor utilization of the main control loop can be expressed as:

$$\frac{T_c}{T_s} \times 100\% \tag{4.13}$$

where T_c is the computation time of the control loop and T_s is the period of control loop. T_c can be obtained from Figure 4.11 and T_s is $\frac{1}{60N_c}$ where N_c represents the number of cameras or vision nodes. Substituting these values into Equation 4.13 and plotting yields the results shown in Figure 4.12. The plot shows that the processor utilization increases quadratically with the number of vision nodes. The upper limit is evident as the processor utilization approaches 100% as the system approaches 14 cameras. Hence the processor time required by the main control loop on the master servo computer represents the main limiting factor in determining the maximum number of vision nodes. A system employing 14 cameras corresponds to an effective visual sampling rate of 840Hz.

For the system components used in this experiment, the limits dictated by available processor time will be encountered long before any network saturation occurs. This upper limit on the number of vision nodes is not fixed but can be increased by optimizing the code in the control loop, by using a faster processor, or by using multiple processors. Alternatively the computation time of the Kalman filter could be reduced by employing a stationary Kalman filter. However, this is not an attractive option since the stationary Kalman filter performs poorly under varying noise and becomes unstable in the presence of occlusions.

4.5.3 Imaging Sensor Limitations

The imaging sensors also have limitations due to the required integration time when capturing an image. This can become significant as the number of cameras and hence the

effective sample rate of the system increase. If the pixel exposure time exceeds the effective sample time, there will be cameras with overlapping samples. Pixel exposure time can be reduced when using a camera equipped with an electronic shutter but only up to a certain point. Therefore, increasing the number of cameras beyond a certain point will only have a limited effect.

4.6 Summary

This chapter has described both the hardware and software implementation details for a direct visual servoing planar robot. The hardware uses off-the-shelf components with a network of distributed vision nodes connected via Ethernet to a master servo computer. Various techniques to ensure a quiet Ethernet network capable of providing deterministic service were described. These techniques included steps to avoid packet collisions by suppressing superfluous packets and managing multiple access through the use of a master/slave round robin polling protocol using UDP packets. The vision nodes employ a soft real-time operating system based on the POSIX.1b standard. The master servo computer uses a hard real-time operating system with priority scheduling to drive the overall system timing and control loops. Some practical extensions for tele-robotic applications were also described including support for remote visual monitoring and local task supervision. Finally, the practical limits to the maximum number of vision nodes was analyzed. It was found that the fast Ethernet network could theoretically support up to 826 vision nodes. However, the main limitation was found to be the processor time on the master servo computer. Using a 400MHz Pentium II processor, the limit was found to be 14 cameras, which corresponds to an effective visual sampling rate of 840Hz. However, this limit is not fixed and can be increased by using a faster computer.

Chapter 5

Experimental Results

5.1 Introduction

A direct visual servo system was implemented as described in Chapter 4 and a variety of experiments were performed. The actual performance of the direct visual servoing system was tested under different conditions. The experimental results add confidence to the simulation results and verify that the concept can be practically implemented in the "real world". Each of the experiments described in the following sections are accompanied by actual plots of various state variables that was logged in real-time by the master servo computer while the robot was running. The experiments that were performed include the transient step response, servo-hold performance, and external disturbance rejection. The first experiments were run without any occlusions, then later, repeated under full and partial occlusions. The occlusions used for the experiment were arbitrary "real-world" occlusions that varied over time. An experiment was also performed to gauge the performance of the direct visual servo under varying illumination conditions. Finally, a simple tele-robotic experiment was also run to verify that the operation of the tele-robotic extensions were feasible.

5.2 Experimental Setup

The master servo computer was configured to close the position loop using only vision feedback to form a direct visual servoing system. The system was configured with four cameras using the scheme illustrated in Figure 2.1. Using four cameras the effective sample rate is 4 times 60Hz or 240Hz (from Equation 2.1). The equipment and its configuration are described in the following subsections.

Component	*Description*
Vision Node Computer	AMD Athalon (550MHz) with 128M DRAM
Frame-Grabber	Matrox Meteor 1 PCI Card
Master Servo Computer	Intel Pentium II (400MHz) with 128M DRAM
Network Adapters	100baseT Ethernet PCI Card
Ethernet Hub	3COM 8 port Hub
Camera	Panasonic WV-BP334
Servo I/O Card	Servo-To-Go Model 2 Servo I/O ISA Card
Power Amplifier	AMC B15A8F
Servo Motor	Reliance Electric model #1842419031
Oscilloscope	Tektronix TDS 210

Table 5.1: List of equipment used for the experiment.

5.2.1 Equipment

The hardware that was used for the experiments was described in section 4.2. The key components are all based on off-the-shelf hardware and are summarized in Table 5.1. The vision nodes used a Linux kernel which was compiled to eliminate all superfluous features and to provide built-in support for the network card. A kernel patch was applied to reserve a large contiguous memory space for a buffer for the frame-grabber card. The master servo computer used a real-time Linux kernel and was equipped with a servo I/O card to control the power amplifier. A rotary encoder with a resolution of 4000 counts/revolution was

attached to the servo motor as a position reference for training and for logging the actual position during experiments.

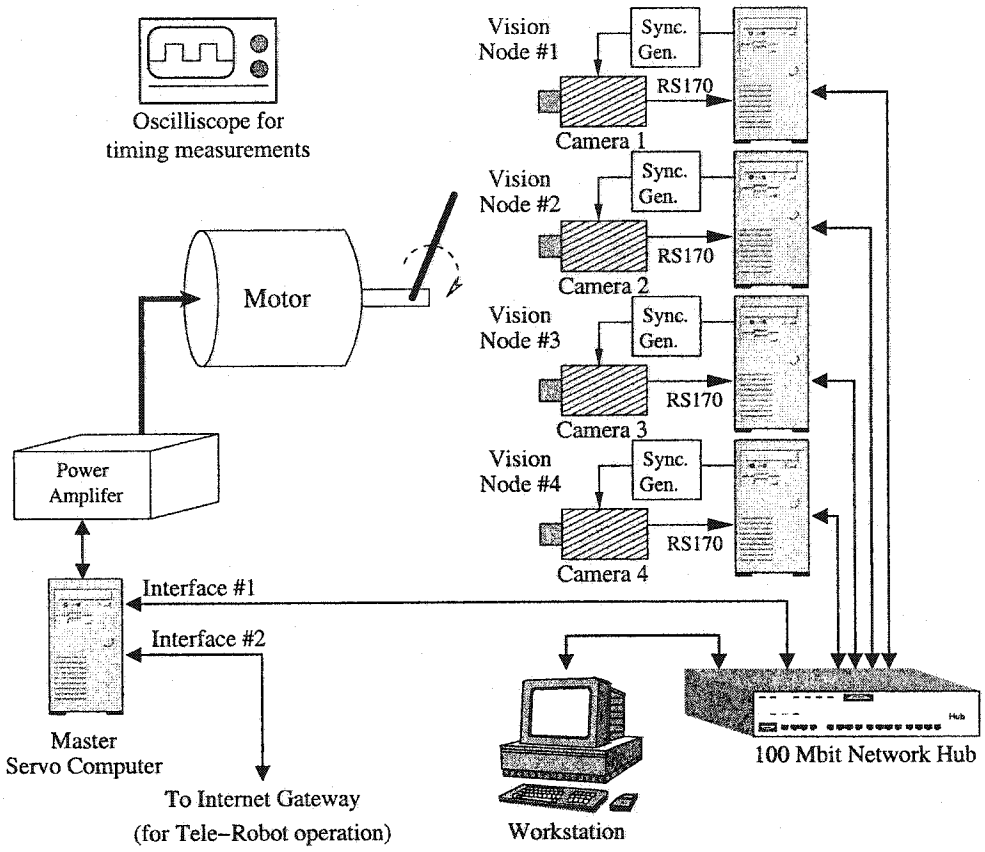


Figure 5.1: Experimental setup.

5.2.2 Equipment Configuration

The components listed in Table 5.1 were connected as illustrated in Figure 5.1. A total of four vision nodes were assembled and inter-connected using 100baseT Ethernet. The master servo computer was also connected to the network and was responsible for controlling the planar robot. A second network interface connected to the Internet was added to the master servo computer to support the tele-robotic experiments. An additional workstation

was introduced to enable remote logins to each of the network nodes from a central console for the configuration of software and settings and for the retrieval of data logging information. The workstation also served to assist in protocol verification by "sniffing" the network traffic between the master servo computer and the vision nodes allowing all the network packets to be observed and time-stamped. An oscilloscope was also employed to monitor software timing and verify sample times.

5.2.3 Training

The experimental setup was first configured for learning by using the 4000 count/rev rotary encoder as a reference to measure the actual position of the planar robot joint. A set of 101 images like the ones depicted in Figure 2.14 were captured by each of the four cameras as the joint was moved in 20 encoder count increments throughout a rotation of 180^o. Once the parameterized manifolds for each vision node were computed and interpolated, the system was configured for direct visual servoing by closing the position loop using feedback from four vision nodes arranged according to the scheme depicted in Figure 5.1.

5.3 Experimental Results Without Occlusions

This section presents the first set of experiments which were performed without any occlusions present. These experiments include measurements of the step response and the response to external disturbances using both stationary and non-stationary Kalman filters.

5.3.1 Step Response Without Occlusions

The experiments described in this section explore direct visual servoing using four cameras with no occlusions present and with varying step magnitudes. The transient step response using traditional encoder feedback is used as a basis for comparison.

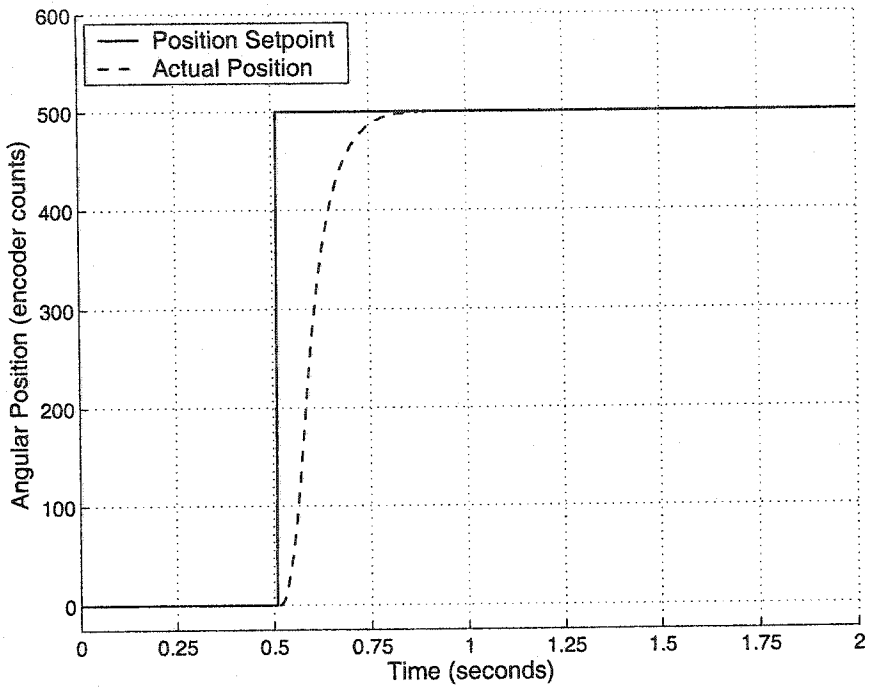
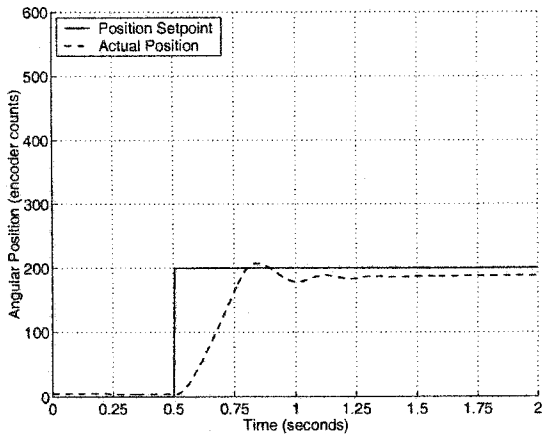


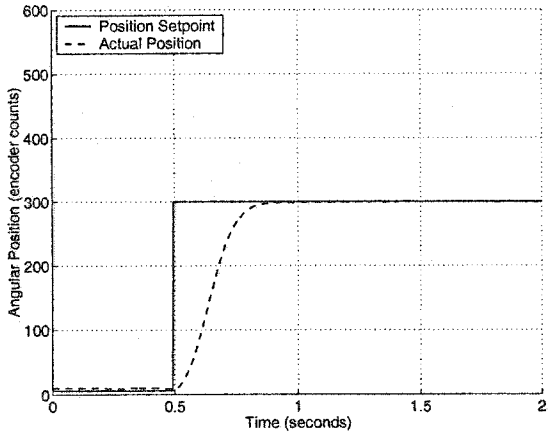
Figure 5.2: Step response using traditional encoder feedback.

Step Response Using Encoder Feedback

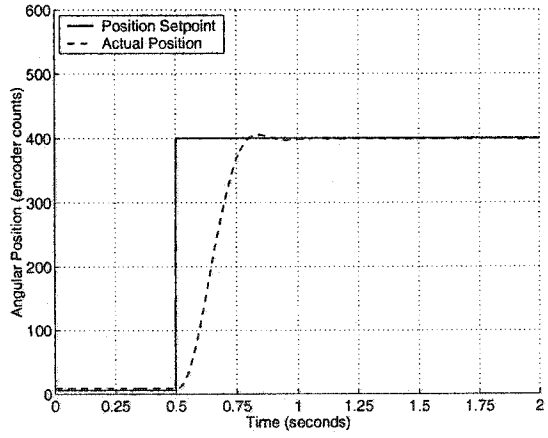
The step response using encoder feedback was obtained to provide a benchmark for comparison with the step response of the direct visual servo system. In contrast to a direct visual servo system, the latency of encoder feedback is negligible. The step response of the planar robot using only a traditional rotary encoder for position feedback is shown in Figure 5.2. A PID controller was employed having a sample rate of 240Hz which matches the sample rate of a direct visual servo using four cameras. The gains were adjusted for a critically damped response resulting in a rise-time of approximately $150ms$ with no overshoot and no steady state errors.



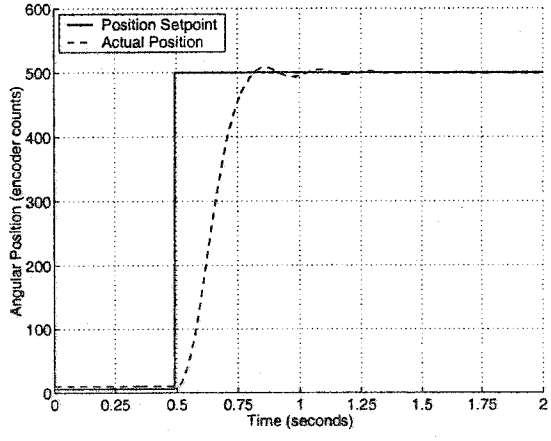
(a) Step magnitude of 200 counts.



(b) Step magnitude of 300 counts.



(c) Step magnitude of 400 counts.



(d) Step magnitude of 500 counts.

Figure 5.3: Step responses for various input step magnitudes for a system with 4 cameras.

Step Response Using Multiple Cameras

In contrast to encoder feedback, vision feedback involves a substantial latency so the direct visual servo relies on the Kalman filter predictor for stability.

The first experiment demonstrated the transient step response using a direct visual servo employing a non-stationary Kalman filter and using a PI position loop with an inner velocity loop. Four vision nodes were used yielding an effective sample rate of 4 times 60Hz or 240Hz. After fine tuning the system gains, a measurement of the actual transient step responses for various input step magnitudes were obtained. Step responses for step magnitudes of 200, 300, 400, and 500 counts are shown in Figure 5.3(a)-(d). The system gains were fine-tuned for each step input to achieve a response that was close to critical damping. Small amounts of noise in the state estimates when using vision feedback limit the magnitude to which the gains can be increased before vibrations begin to occur in the planar robot. The actual position in each plot was measured using a 4000 count/revolution reference encoder. The encoder was attached to the motor shaft and was used for training and data logging purposes only.

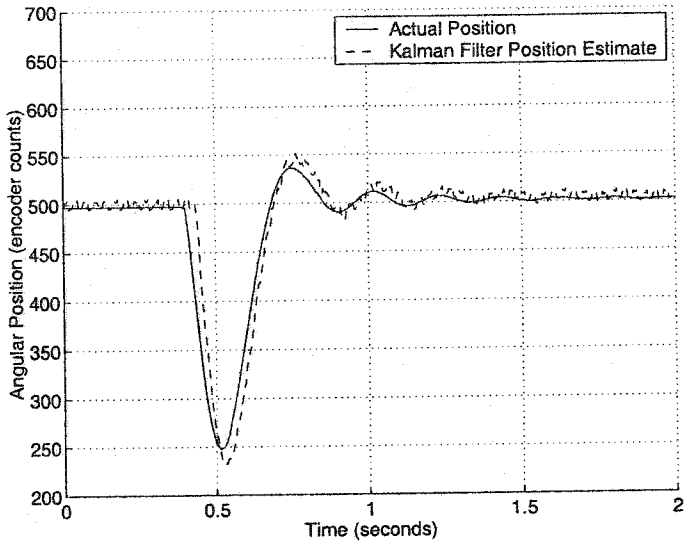
The results show a rise time of approximately $190ms$ with a slight overshoot for a 500 count input step. This closely matches the predicted risetime for four cameras shown in Figure 3.23. The transient oscillations are due to higher order effects which are not modelled in the Kalman filter predictor. Some positions exhibit small steady state errors. This occurs when the gravity feed-forward does not exactly cancel the actual holding torque causing the predictor to assume a slight acceleration. The actual holding torque is due to a combination of gravity and torque ripple due to the saliency in the motor poles. The feedback from the cameras corrects the predictor estimates but the effect produces a small increase in error for the overall state estimate.

These results are also comparable to the rise-time using only encoder feedback shown in Figure 5.2 which was also run at a sample rate of 240Hz. The performance using encoder

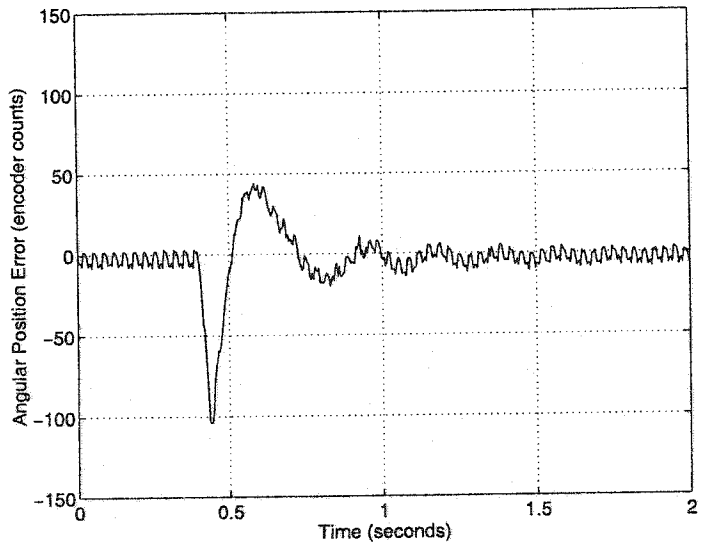
feedback shows a somewhat faster risetime and does not exhibit the small transient oscillations which are present in the direct visual servo. The encoder feedback is better because it does not require a predictor and it does not need to contend with long feedback delays. Hence the gains using encoder feedback can be made larger before the onset of instability. The proportional gain in the PID loop used with encoder feedback was an order of magnitude higher than the $K_p K_v$ gain product used with vision feedback. This high gain resulted in some output current saturation when using encoder feedback with large step inputs.

5.3.2 Response to External Disturbances

Another experiment was performed to observe the response to an external torque disturbance. An external torque disturbance was applied by hitting the planar robot joint during a servo-hold and observing the response. The actual transient position response along with the estimated position from the Kalman filter is shown in Figure 5.4. The performance of the step responses shown in the preceding sections indicates that the Kalman filter predictor is generally able to accurately predict the output position of the system under normal conditions. However, the Kalman filter predictor cannot anticipate any unmodelled target motion such as external disturbance inputs. Therefore, a direct visual servoing system cannot sense an external disturbance until is propagates through the entire vision transport delay. This explains the initial lag between the actual measured position and the estimated position in Figure 5.4(a). The estimated position momentarily remains unchanged for approximately $30ms$ while the actual position changes dramatically in response to the external hit. The delay in the response of the position estimate is due to the transport delay in the vision feedback. The position estimation error vs. time shown in Figure 5.4(b) indicates a large initial position error from the disturbance. The state estimates are corrected when the vision feedback eventually propagates through forcing the Kalman filter to converge and the position error returns to zero. These results are similar to the simulated results in Figure 3.24.



(a) Actual and estimated position vs. time.



(b) Position estimate error vs. time.

Figure 5.4: Transient response to an external disturbance applied to the robot.

5.4 Experimental Results In the Presence of Occlusions

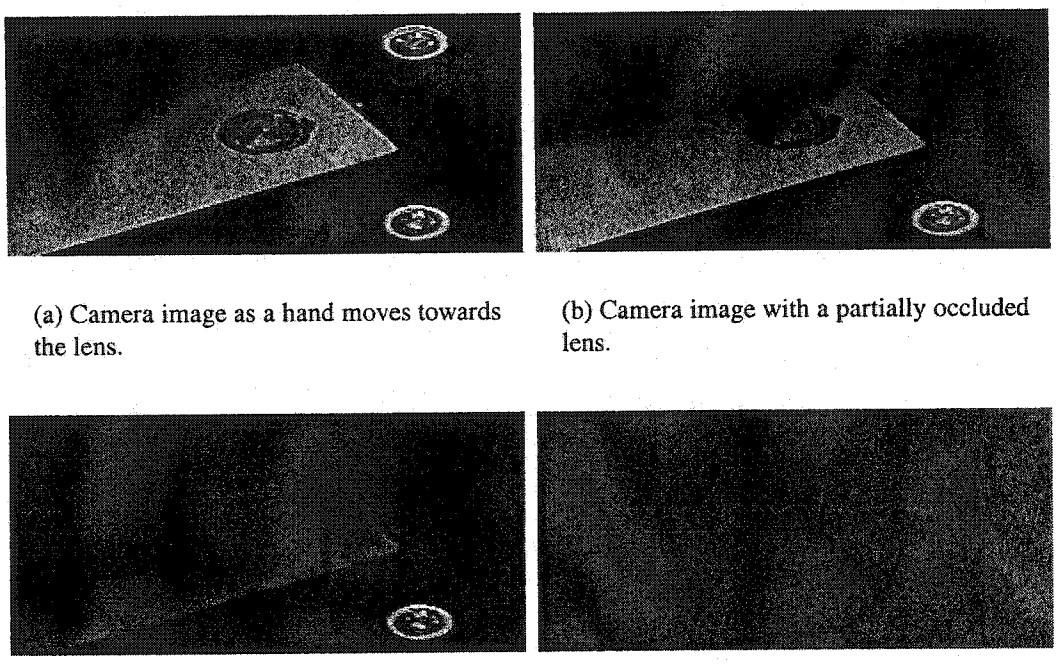
This section presents the second set of experiments which were performed in the presence of various occlusions. The types of occlusions that were explored include both full occlusions in one camera and partial occlusions in all cameras. The full occlusion in one camera was accomplished by placing a hand over the lens of a camera as illustrated in Figure 5.5. Partial occlusions in all cameras were accomplished by placing a wrench in front of the robot joint as illustrated in Figure 5.8. The experiments proceeded by observing the performance of the robot as these occlusions were dynamically introduced and removed from the scene. Both the actual position of the robot and the Euclidean distance over time were recorded. The experiments include the analysis of the servo-hold performance, the step response, and the response to external disturbances.

5.4.1 Servo-Hold with Full Occlusions in One Camera

The servo-hold performance was investigated when a full occlusion is placed in front of one of the cameras. A hand was placed in front of a camera to fully occlude its view as illustrated in Figure 5.5. Experiments were performed using both a stationary and a non-stationary Kalman filter.

Servo-Hold with Full Occlusions in One Camera Using Stationary Kalman Filter

The servo-hold performance was first investigated using a stationary Kalman filter using fixed Kalman gains that were precomputed off-line. One camera was exposed to a full occlusion as a hand was moved in front of the lens of camera #3. The effect was as shown in Figure 5.5. Actual plots of the Euclidean distance vs. time is shown in Figure 5.6(a) showing the change in Euclidean distance in camera #3 as a hand occluded the lens and then was removed. The corresponding position disturbance is illustrated in Figure 5.6(b). This indicates a high sensitivity to the presence of occlusions as the planar robot begins



(a) Camera image as a hand moves towards the lens.

(b) Camera image with a partially occluded lens.

(c) Camera image of almost fully occluded lens.

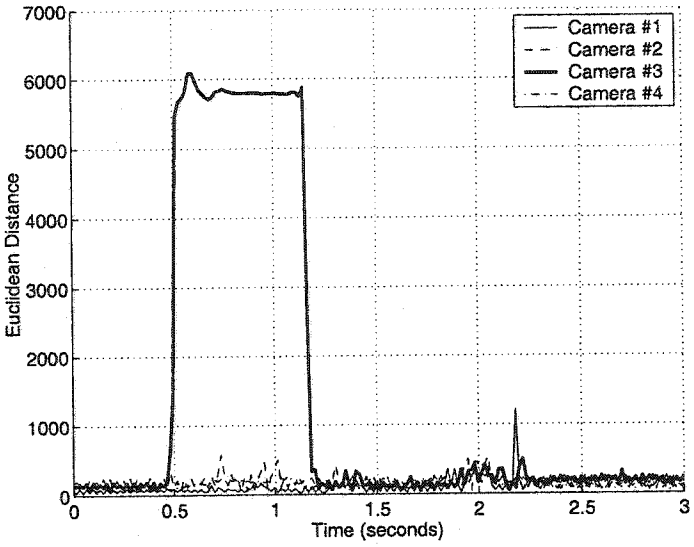
(d) Camera image of fully occluded lens.

Figure 5.5: Camera images as a full occlusion is gradually placed over the lens.

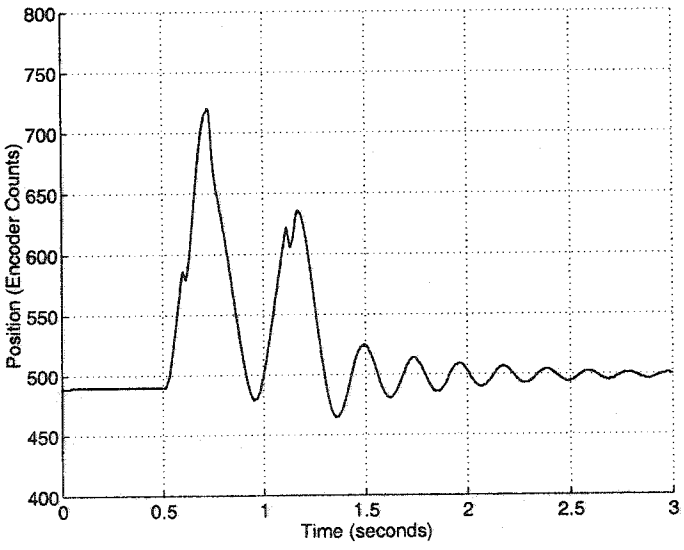
to oscillate and becomes unstable. The instability is dramatic despite the fact that the occlusion is isolated to only one of the four cameras in the system. Stability is regained once the occlusion is removed from the camera. Clearly, the stationary Kalman filter is unable to cope with full occlusions even in only one of the cameras.

Servo-Hold with Full Occlusions in One Camera Using Non-Stationary Kalman Filter

Next, the experiment was repeated with the non-stationary Kalman filter using an estimate of measurement error variance $\mathbf{R}$ obtained using Equation 3.56. Once again, a full occlusion was introduced into one camera while the robot was performing a servo hold. The occlusion was introduced by arbitrarily sweeping a hand over each camera lens in sequence

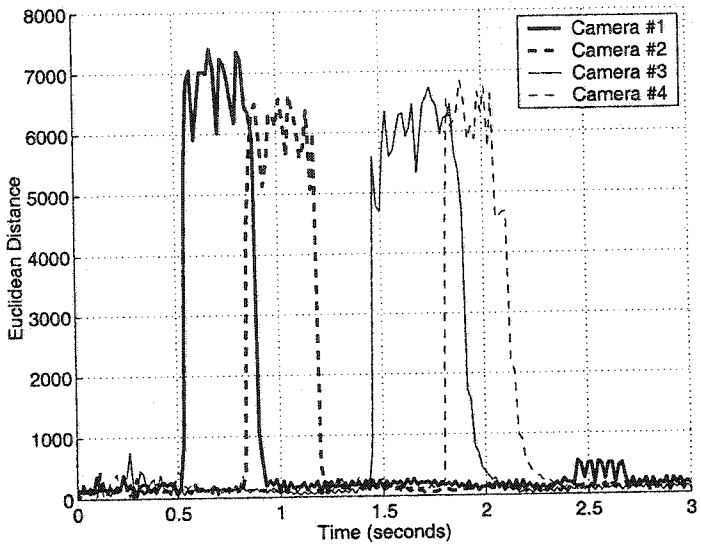


(a) Euclidean distance vs. Time

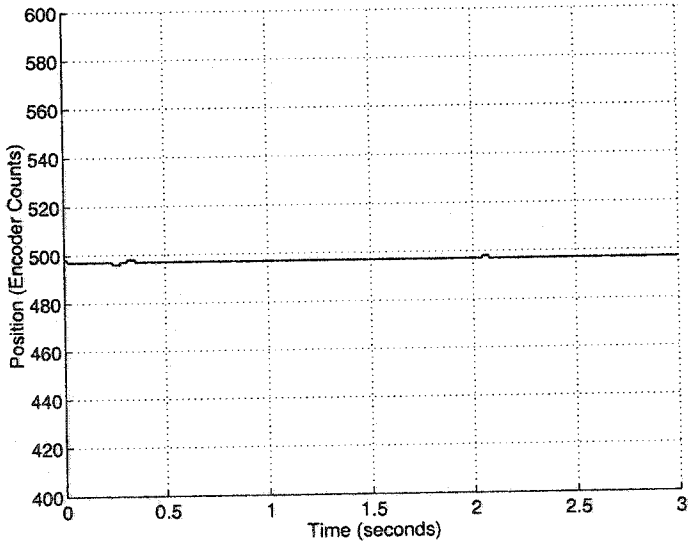


(b) Actual position vs. Time

Figure 5.6: Euclidean distance and position vs. time during a servo-hold as a full occlusion is introduced into camera #3 using a stationary Kalman filter.



(a) Euclidean distance vs. Time



(b) Actual position vs. Time

Figure 5.7: Euclidean distance and position vs. time during a servo-hold as a full occlusion is swept across all four cameras using a non-stationary Kalman filter.

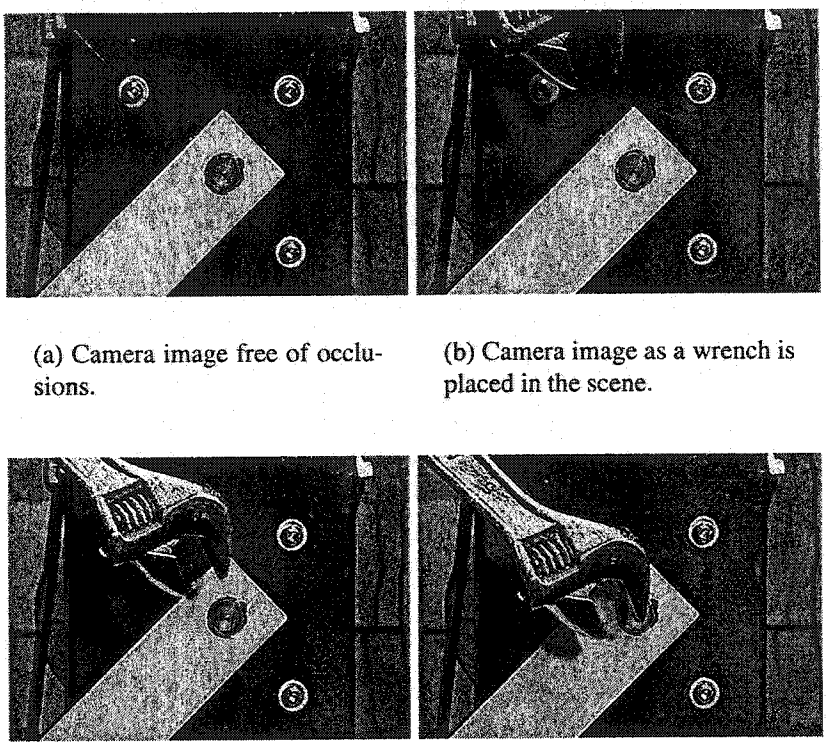
producing full occlusions in each camera similar to those shown in Figure 5.5. Figure 5.7(a) shows actual plots of the Euclidean distances vs. time as the occlusion passes in front of each camera lens in turn. Because pairs of cameras are located close together as shown in Figure 4.5, there are moments when two cameras are simultaneously occluded. This is due to overlap as the occlusion remains briefly in the view of one camera as it enters the view of the adjacent camera. The corresponding actual position of the robot during this time is illustrated in Figure 5.7(b) and demonstrates good stability even in the presence of full occlusions in one or two of the cameras. Further experiments indicated that even if three of the four cameras are completely occluded the robot remained stable although the Kalman filter had to rely on only one camera and the predictor which results in poorer dynamic performance. Experiments with partial occlusions in all cameras were further explored in the next section.

5.4.2 Servo Hold with Partial Occlusions in All Cameras

Next, the servo-hold performance was investigated when partial occlusions were present in all cameras. A wrench was placed in front of the planar robot thereby occluding the view for all cameras as illustrated by the sequence of images shown in Figure 5.8. The position of the wrench was strategically selected to occlude the view in all cameras. However, the degree of the occlusion varied in each camera depending on the angle of the camera with respect to the robot. Consequently, the cameras to the left of the robot experienced slightly greater occlusions than the cameras on the right.

Servo Hold with Partial Occlusions Using a Stationary Kalman Filter

First, a system running a stationary Kalman filter was configured using the same pre-computed Kalman gains as were used in section 5.4.1. All cameras were exposed to a partial occlusion as a wrench was placed in front of the planar robot as shown in Figure 5.8.



(a) Camera image free of occlusions.

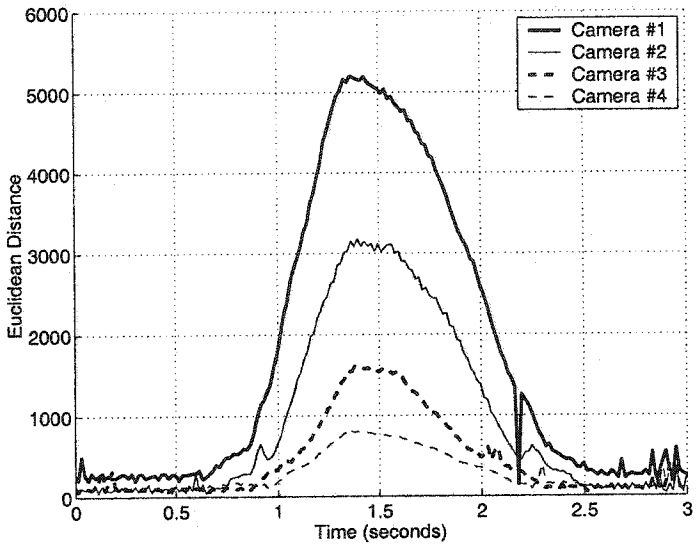
(b) Camera image as a wrench is placed in the scene.

(c) Camera image as the wrench begins to occlude the motor shaft.

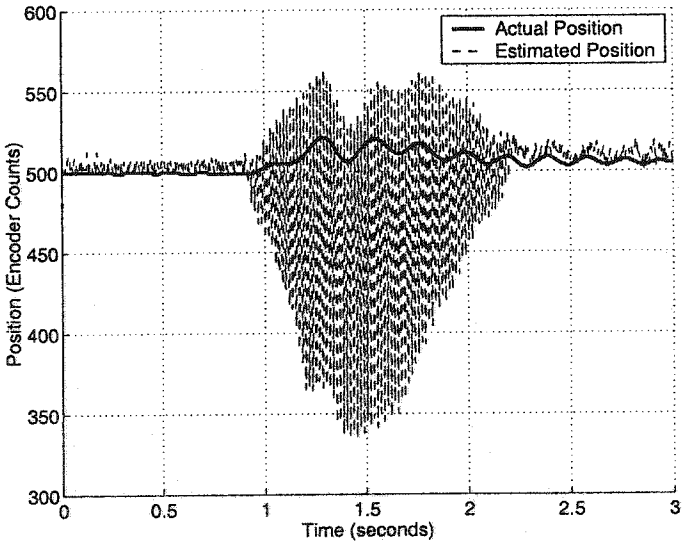
(d) Camera image of the wrench partially occluding the robot.

Figure 5.8: Placing a wrench in directly front of the planar robot causes a partial occlusion to occur in all cameras.

An actual plot of the Euclidean distance vs. time is shown in Figure 5.9(a) which indicate the change in Euclidean distance as a wrench occluded the robot and then was removed. Each camera reported varying degrees of occlusions with peak Euclidean distances varying from 1500 to over 5000. Camera #1 experienced the greatest occlusion due to the fact that the wrench entered the scene from the same side that camera #1 was located. The peak Euclidean distances diminished for camera #2 through camera #4 which were sequentially positioned to the right of the robot.



(a) Euclidean distance vs. Time



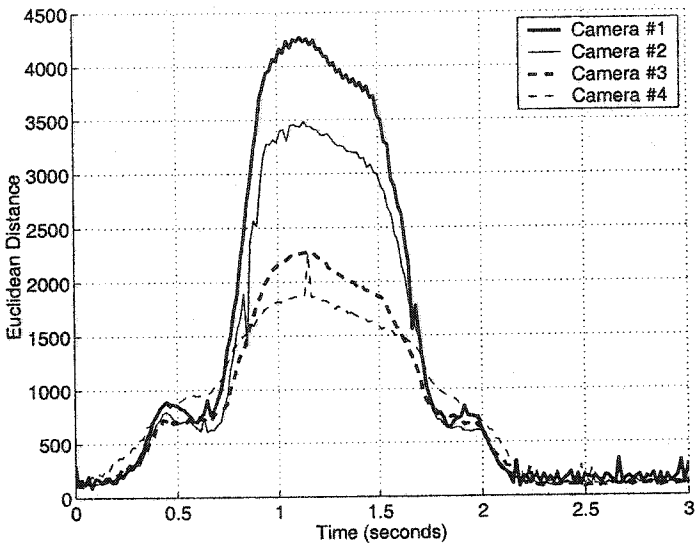
(b) Actual position vs. Time

Figure 5.9: Euclidean distance and position vs. time during a servo-hold as a partial occlusion is introduced in all cameras using a stationary Kalman filter.

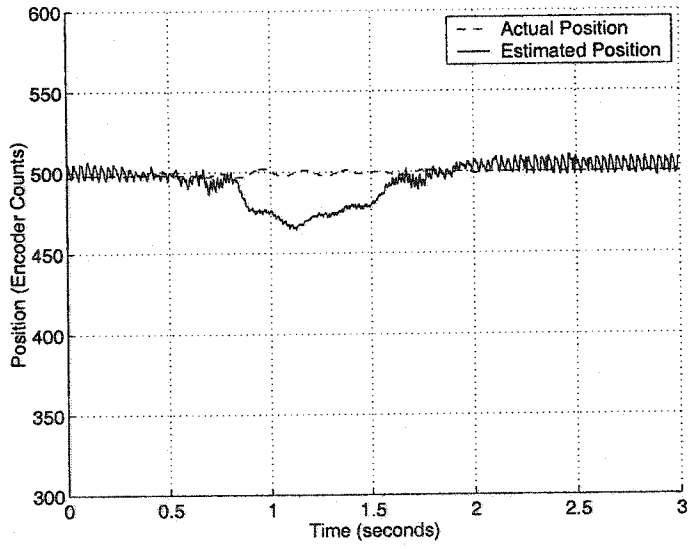
The corresponding actual and estimated position are shown in Figure 5.9(b). The position estimated by the stationary Kalman filter becomes quite noisy in the presence of occlusions. The large noise present in the position estimates form the feedback to the position compensator which produces correspondingly large variations in the output current. These output current variations cause large torque variations in the servo motor. However, the frequency of the torque variations is large compared to the mechanical bandwidth of the robot so only a small amount of variation in the actual robot position is visible. Nevertheless, the variations in motor current do cause a large increase in the RMS motor current which increases the power dissipation and temperature in both the servo motor and the power amplifier.

Servo Hold with Partial Occlusions Using a Non-Stationary Kalman Filter

The robustness of the system employing a non-stationary Kalman filter when all cameras are simultaneously subjected to a partial occlusion was also investigated. Once again, a partial occlusion was introduced in all cameras by placing a wrench in front of the robot as shown in Figure 5.8. Actual plots of the Euclidean distance vs. time are shown in Figure 5.10(a) showing the change in Euclidean distance in each camera as the partial occlusion is introduced in front of the robot. The peak Euclidean distances varied from approximately 4200 for camera #1 which was located to the left of the robot down to 1800 for camera #4 which was located to the far right of the robot. Since the occlusion was introduced from the left side of the robot as shown in Figure 5.8, the cameras placed farther to the right of the robot experienced less of an occlusion. A Euclidean distance of 2500 corresponds to the threshold where the vision feedback is considered "too noisy" and the position measurement is effectively ignored. The corresponding actual position of the robot during this time is illustrated in Figure 5.10(b). The results demonstrate good stability during a servo-hold despite the partial occlusion and the wild oscillations that were evident with the stationary Kalman filter are completely gone. The estimated position does stray



(a) Euclidean distance vs. Time



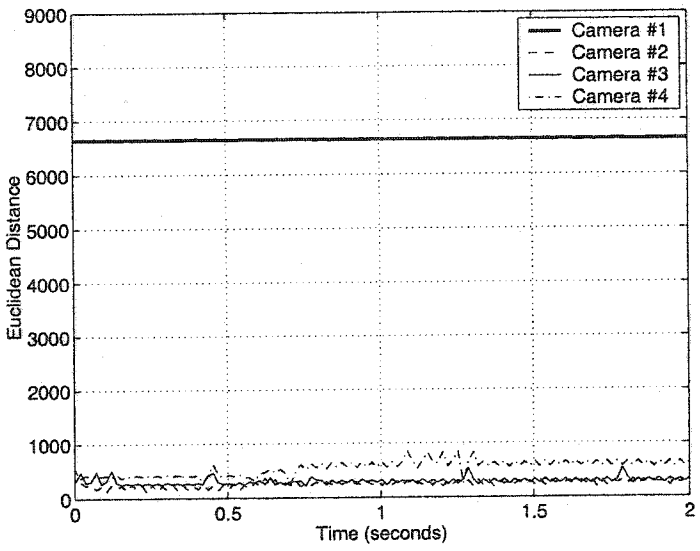
(b) Actual position vs. Time

Figure 5.10: Euclidean distance and position vs. time during a servo-hold as a partial occlusion is introduced in all cameras using a non-stationary Kalman filter.

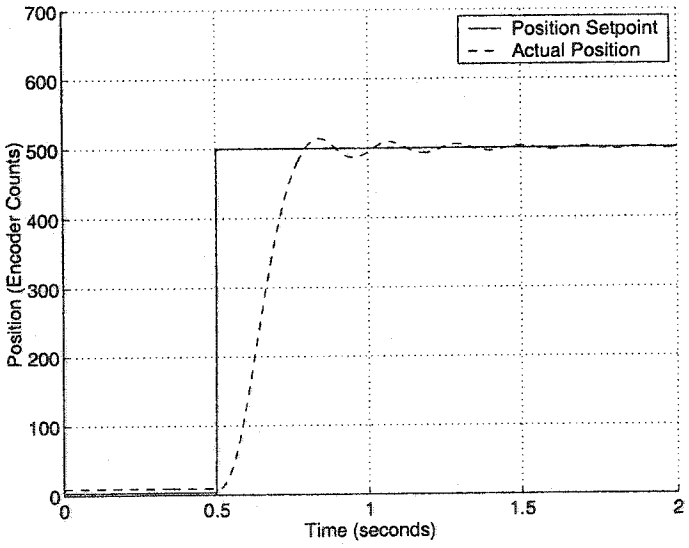
from the actual position by up to 25 encoder counts when the Euclidean distance reaches its peak. This is due in part to the fact that cameras #3 and #4 have exceeded a Euclidean distance of 2500 and are therefore ignored. The result is that the Kalman filter relies on the predictor without any corrective feedback for half of the sample times which causes the position estimate to stray somewhat from the actual position. This suggests that strategic camera placement can help ensure that some cameras always have a partially unobstructed view can make the system robust to partial occlusions when using a non-stationary Kalman filter.

5.4.3 Step Response with Full Occlusions in One Camera

The step response performance of the four camera system with one camera fully occluded was also experimentally investigated. The step responses was only obtained using a non-stationary Kalman filter since the stationary Kalman filter was not even capable of maintaining a stable servo-hold as indicated in section 5.4.1. Camera #1 was fully occluded and the remaining three cameras were free of occlusions. This is apparent in Figure 5.11(a) where a constant large Euclidean distance is apparent in camera #1 while the remaining three cameras report small Euclidean distances. The Euclidean distance for camera #1 is fixed at around 6800 which is well above the limit of 2500 causing all measurements from this camera to be discarded by the Kalman filter. Therefore every fourth sample time, only the predictor will be used to yield the position estimate which will reduce the rate at which the actual vision feedback can correct the state estimates. Despite the loss of one camera, the system maintained a stable servo-hold. Next, a 500 count step input was applied and the response is shown in Figure 5.11(b). The plot shows a risetime of roughly $190ms$ which is identical to the risetime for the unoccluded step response using the same gains plotted in Figure 5.3. However, the magnitude and the duration of the oscillations in the step response is greater for the occluded step response. This is due to the degraded state estimates which



(a) Euclidean distance vs. Time



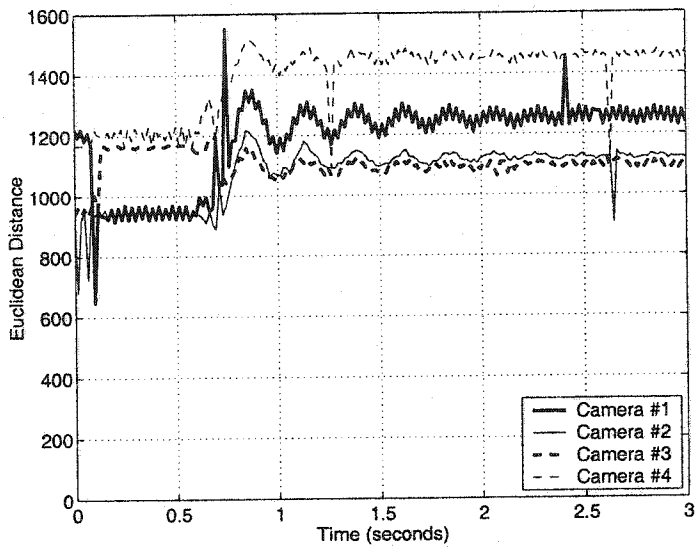
(b) Actual position vs. Time

Figure 5.11: Euclidean distance and position vs. time during a step input with a full occlusion in one camera using a non-stationary Kalman filter.

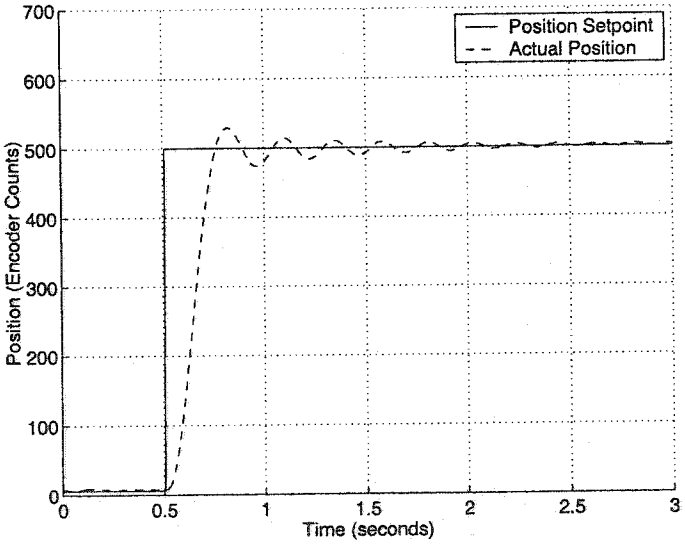
are produced without the benefit of visual feedback from one of the four cameras.

5.4.4 Step Response with Partial Occlusions in All Cameras

The step response performance of the four camera system with all the cameras partially occluded was also experimentally investigated. Once again, the step response was only obtained using a non-stationary Kalman filter. All cameras were exposed to a partial occlusion with a wrench which was statically placed in front of the planar robot as shown in Figure 5.8. An actual plot of the Euclidean distance vs. time is shown in Figure 5.12(a) and the corresponding actual position vs. time is shown in Figure 5.12(b). The Euclidean distance ranges from just under 1000 to over 1500 which are all well below the limit of 2500. Despite the fact that the wrench did not move during the robot move, the Euclidean distance is seen to change and even oscillate in a fashion similar to that shown in the position plot. This is due to the fact that as the robot moves, the static occlusion covers different parts of the robot corresponding to different parts of the feature vectors. The result is that the position oscillations are visible in the Euclidean distance plot since the Euclidean distance is now also a function of robot position. The response to a 500 count step input is shown in Figure 5.12(b). Once again, the plot shows a risetime of roughly the same as the risetime for the unoccluded step response using the same gains plotted in Figure 5.3 but with substantially larger oscillations. The oscillations with partial occlusions in all cameras are also larger than those observed in the previous section with a full occlusion in one camera. This is due to the fact that the remaining three cameras in the previous section were still providing good feedback, whereas in this case the feedback from all cameras is degraded and thus weighted lower in the overall state estimation process in the Kalman filter. This results in state estimates which rely more heavily on the predictor and less on actual feedback reducing the dynamic performance. Nevertheless, these results indicate that the system is robust and can still function in the presence of partial occlusions.



(a) Euclidean distance vs. Time



(b) Actual position vs. Time

Figure 5.12: Euclidean distance and position vs. time during a step input with partial occlusions in all cameras using a non-stationary Kalman filter.

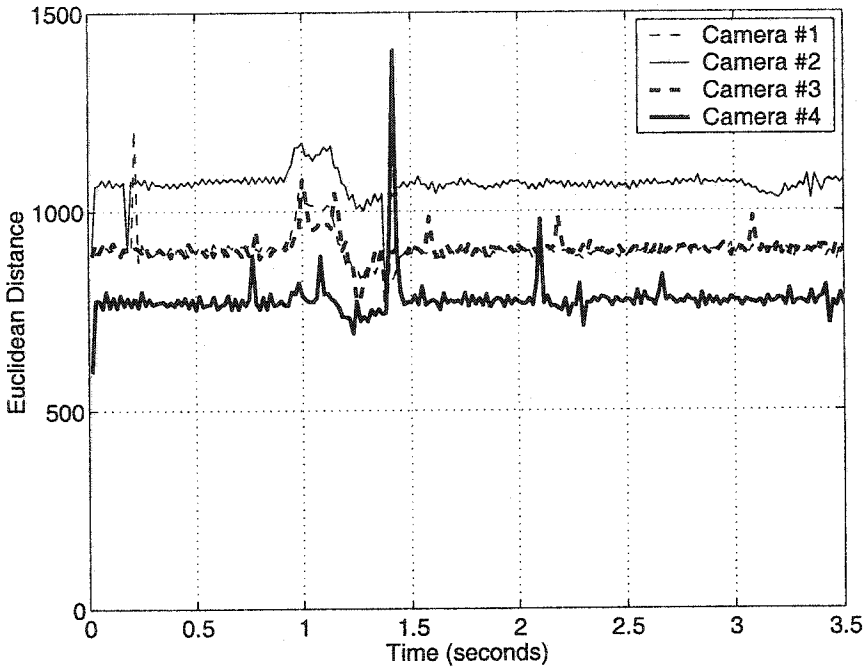


Figure 5.13: Euclidean distance vs. time in all cameras while an external disturbance is applied to the robot in the presence of occlusions.

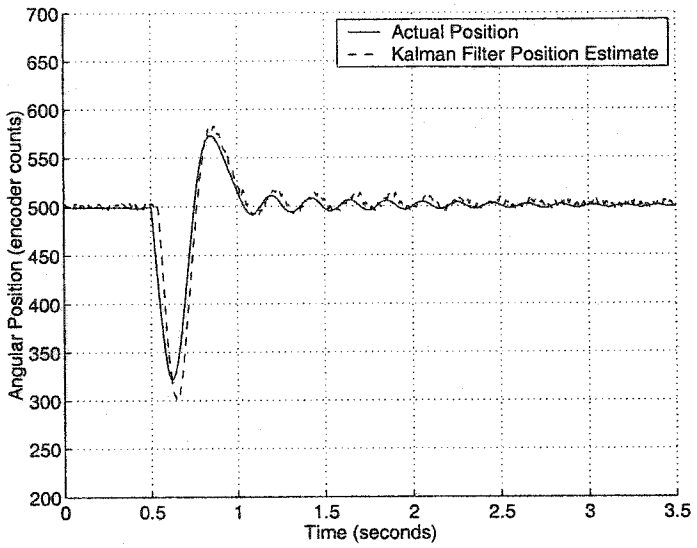
5.4.5 Response to External Disturbances in the Presence of Occlusions

Another experiment was performed to observe the response to an external torque disturbance in the presence of partial occlusions. An occlusion similar to the one shown in Figure 5.8 was introduced in front of the robot. The resulting Euclidean distance is shown in Figure 5.13. An external torque disturbance was applied by hitting the planar robot joint during a servo-hold and observing the response. The actual position and the estimated position are plotted in Figure 5.14(a) and the position estimate error vs. time is shown below in Figure 5.14(b). Once again, the Kalman filter predictor is unable to anticipate any unmodelled target motion such as external disturbance inputs. Therefore an initial delay occurs in the response of the position estimate due to the transport delay in the vision feedback. This explains the large initial position error in Figure 5.14 which was also predicted by

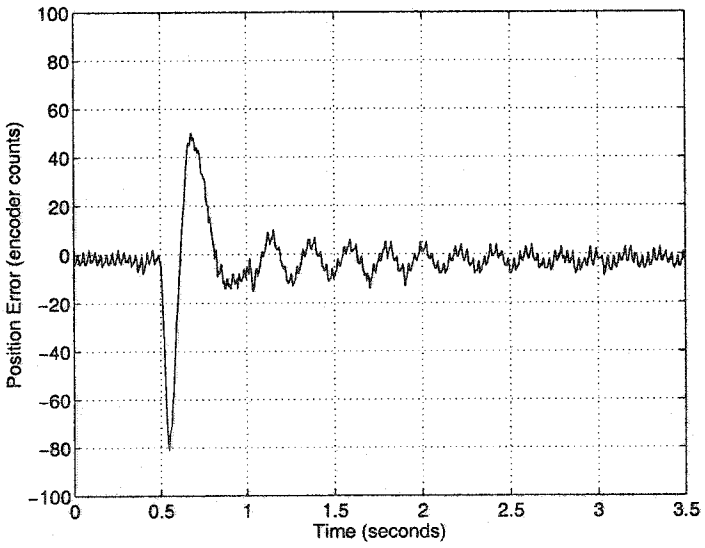
the simulation in Figure 3.24. However, the performance has degraded in comparison to the disturbance response with no occlusions shown in Figure 5.4. With no occlusions the position estimation error decays to almost zero in less than one second, whereas with occlusions the estimation error oscillates for almost 3 seconds. Once again, the partial occlusions present in all cameras degrade the feedback and make the Kalman filter more reliant on the predictor which is totally unable to predict external torque disturbances. Hence the visual feedback is weighted lower and it takes longer for the Kalman filter to converge to the actual position. Despite the decrease in dynamic performance, the direct visual servo is still stable in the presence of external disturbances with partial occlusions.

5.5 Experimental Results Under Varying Illumination

Although occlusions were the focus of much of the experiments, the effects of varying illumination were also briefly explored. In section 3.4.6, it was hypothesized that the error variance as determined by Euclidean distance could potentially improve robustness to variations in illumination. A simple experiment was performed with the planar robot performing a servo-hold under changing illumination. The illumination around the planar robot was varied by gradually covering and uncovering the main source of light (an incandescent lamp equipped with a 100 Watt bulb). This had the effect of varying the intensity of illumination while the direction of the illumination remained fixed. The background room lighting remained constant but by covering the lamp an dramatic change in lighting appearance was obtained as shown in Figure 5.15. The experiment was repeated using both a stationary and a non-stationary Kalman filter.

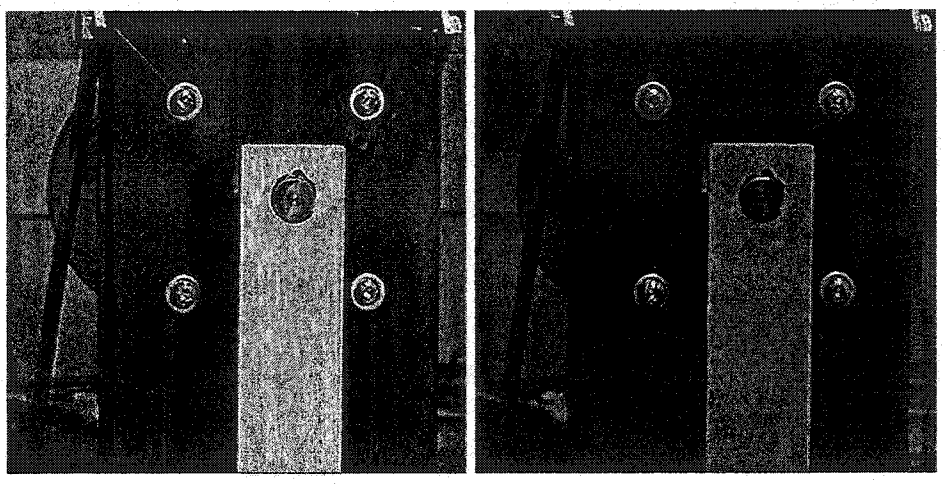


(a) Actual and estimated position vs. time.



(b) Position estimate error vs. time.

Figure 5.14: Transient response to an external disturbance applied to the robot in the presence of partial occlusions.



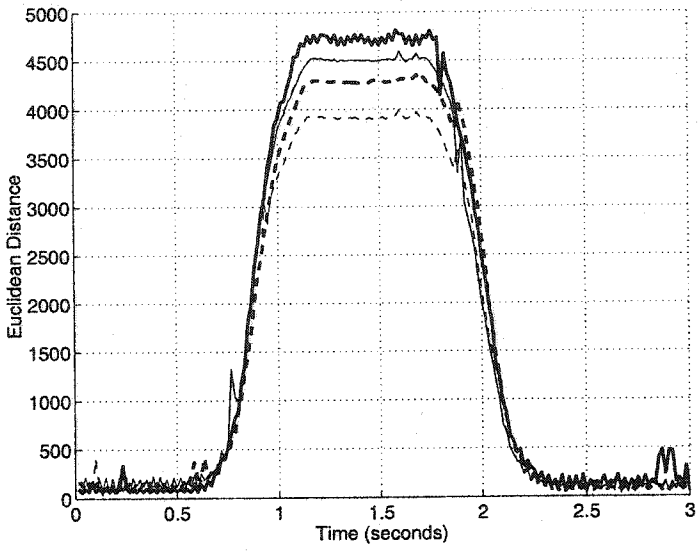
(a) Planar robot under normal illumination.

(b) Planar robot with lamp covered.

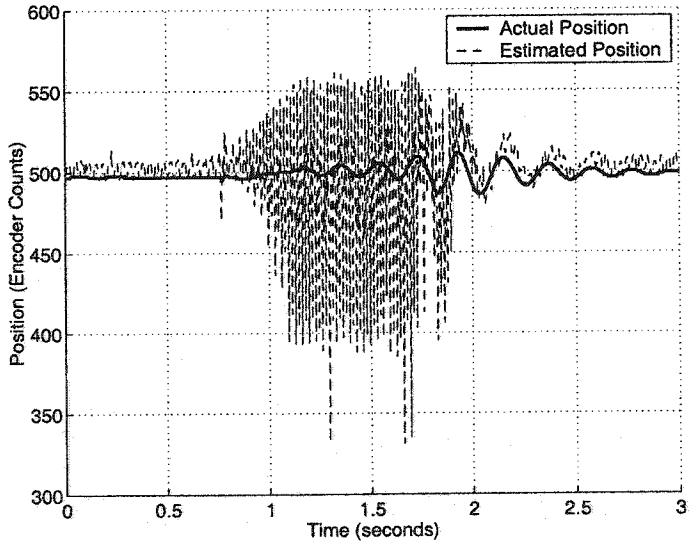
Figure 5.15: Camera images as the illumination is varied.

5.5.1 Servo-Hold Under Varying Illumination Using a Stationary Kalman Filter

First, a system running a stationary Kalman filter was configured using the pre-computed Kalman gains. The workcell was then exposed to large variation in illumination as shown in Figure 5.5. Actual plots of the Euclidean distance vs. time are shown in Figure 5.16(a) which show the change in Euclidean distance as the primary light source is gradually covered and then uncovered. The Euclidean distances all change simultaneously and by a similar amount since the illumination variation is global for all cameras except for the small variations in background lighting. The corresponding position disturbance illustrated in Figure 5.16(b) indicates a sensitivity to illumination variation as the planar robot begins to oscillate and becomes unstable. Stability is restored once the illumination is returned to normal (i.e. the same as it was during the training phase).

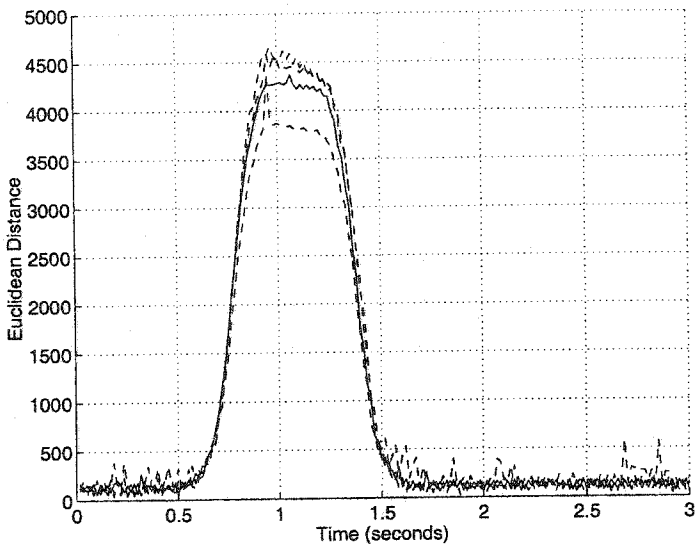


(a) Euclidean distance for all four cameras vs. Time

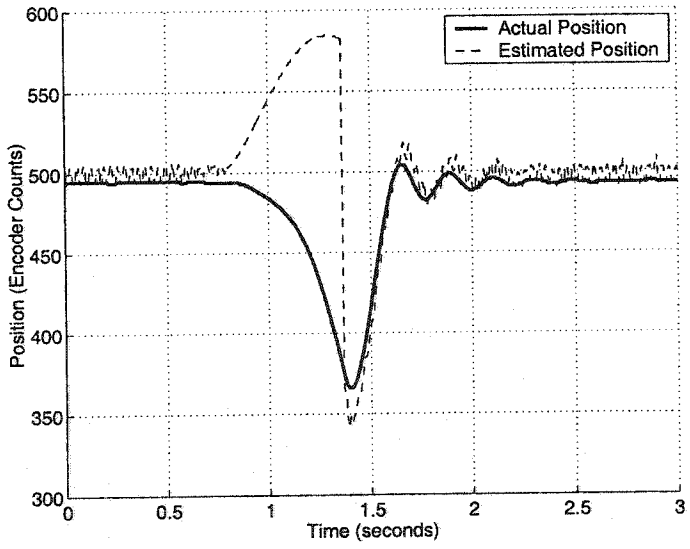


(b) Actual and estimated position vs. Time

Figure 5.16: Euclidean distance and position vs. time under varying illumination using a stationary Kalman filter.



(a) Euclidean distance for all four cameras vs. Time



(b) Actual and estimated position vs. Time

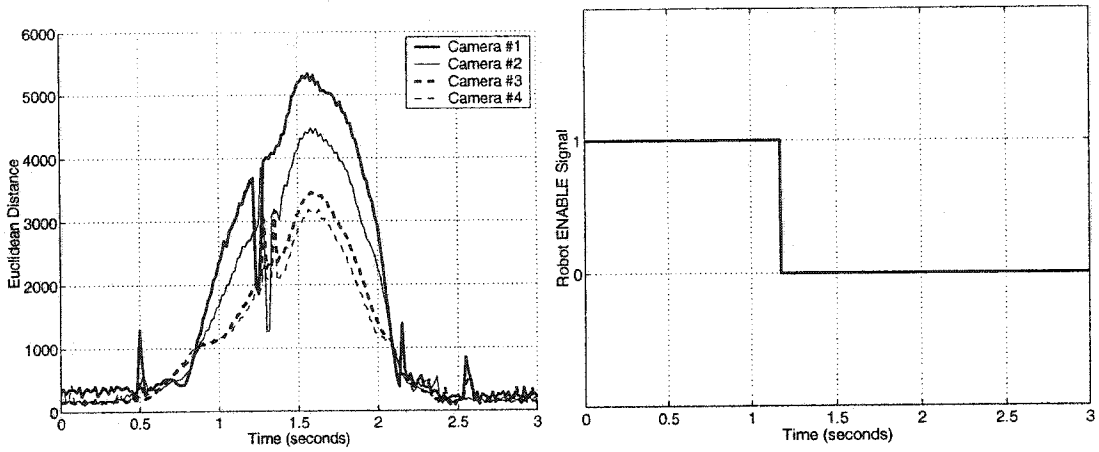
Figure 5.17: Euclidean distance and position vs. time under varying illumination using a non-stationary Kalman filter.

5.5.2 Servo-Hold Under Varying Illumination Using a Non-Stationary Kalman Filter

Next, the robustness of the system employing a non-stationary Kalman filter under varying illumination was also investigated. Once again, the workcell was then exposed to large variation in illumination as shown in Figure 5.15. Actual plots of the Euclidean distance vs. time are shown in Figure 5.17(a) showing the change in Euclidean distance as the primary light source is covered and then uncovered. The actual position of the robot during this time is illustrated in Figure 5.17(b). The robot demonstrates good stability until the change in illumination becomes large. A problem appears when the change in illumination is severe enough to result in Euclidean distances that exceed 2500, at which point the robot position begins to stray. A Euclidean distance of 2500 corresponds to the threshold where the vision feedback is considered "too noisy" and the position measurement is effectively ignored. The position begins to stray as the predictor errors accumulate causing the estimated position to diverge from the actual position. However, once the light is restored to the point where the Euclidean distance falls below 2500, the estimated position quickly converges back to the actual position. These results indicate that the non-stationary Kalman filter shows promise for operation under small variations in illumination that result in Euclidean distances that are below the maximum threshold.

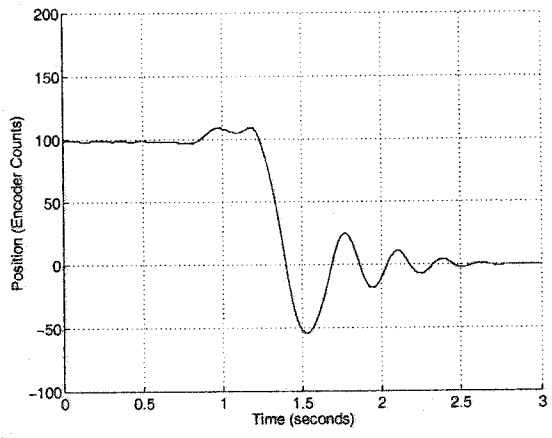
5.6 Experiments in Robot Tele-Operation

A final experiment was performed to demonstrate local task supervision during tele-operation of the planar robot. The planar robot was remotely operated using a workstation connected over a Local Area Network to the master servo computer as shown in Figure 5.1. The workstation performed a TCP connection to the master servo computer and the image dimensions and image eigenvectors were transmitted. Once the basis vectors were received,



(a) Euclidean distance vs. Time

(b) Robot ENABLE Signal vs. Time



(c) Actual Position vs. Time

Figure 5.18: Euclidean distance, robot ENABLE, and position vs. time during tele-robot operation as a partial occlusion is introduced in the workcell.

the coefficients were streamed via a UDP connection to reconstruct an image of the robot on the display of the workstation. Next, an occlusion was introduced into the view of the cameras after commanding the robot to perform a servo-hold. As the occlusion grew, the Euclidean distance increased, as shown in Figure 5.18(a). As the Euclidean distance increased, the position measurement error variance increased. As the measurement error variance increased, the entries in the state estimation error matrix $\mathbf{P}$ in the Kalman filter also grew. This continued until the position state estimation error exceeded a predetermined threshold. At this point, the master servo computer disabled the robot without human intervention as shown ion Figure 5.18(b). The robot enable signal is an internal boolean variable that is set to "1" to enable the robot and set to "0" to disable the robot. With the robot disabled, the applied motor torque becomes zero and gravity takes over causing the robot joint to sag to its lowest position as shown in Figure 5.18(c). This sequence demonstrates the local task supervision that can be realized using data from the Kalman filter to automatically disable the robot when the visual feedback is sufficiently degraded even when no human is present to intervene.

5.7 Summary

The direct visual servoing system described in Chapter 4 was built using four network-synchronized cameras. This system was used to performed various experiments involving the direct visual servoing of a planar robot. The experimental results demonstrate excellent servo-hold performance and a good step response closely matching the step response simulated in Chapter 3. Further experiments demonstrated the response to external disturbance inputs and once again closely matched the predictions simulated in Chapter 3. The robustness of the system was investigated by repeating the experiments with full occlusions in one camera and with partial occlusions in all cameras. The performance of the stationary and non-stationary Kalman filter were also compared and contrasted. As predicted, the

non-stationary Kalman filter proved to be robust in the presence of various degrees of occlusions whereas the stationary Kalman filter was unstable. A final experiment tested the robustness of the direct visual servo under varying illumination intensity. Experimental results show that, besides robustness to occlusions, the approach developed in this work also shows promise for dealing with small variations in illumination. All the results presented in this chapter serve to confirm the efficacy of the approach proposed by this research.

Finally, an experiment was also performed to demonstrate how the system could be extended for tele-robotic applications. The tele-robotic experiment successfully demonstrated remote servoing and local task supervision.

Chapter 6

Conclusions

6.1 Summary Review

This chapter provides a summary of the results along with the conclusions and recommendations for future work. The concept of a robust direct visual servo has been developed in the preceding chapters starting with an introduction to the area of visual servoing in Chapter 1. The following chapters have all focused on different aspects of the direct visual servo culminating with the construction and testing of a working prototype in Chapter 5. Chapter 2 focused on the details of the vision system which was designed to overcome the limitation of 60Hz video by using multiple RS-170 cameras synchronized over a network. Each camera is equipped with a computer to processes video at field rates and Principal Component Analysis is used to determine the position of a planar robot. The effect of occlusions on position measurement errors was simulated and a statistical relationship was discovered between Euclidean distance in eigenspace and the position measurement error. This insight became the key to sensor fusion in Chapter 3 where a non-stationary Kalman filter was developed to weight the feedback from multiple cameras. The Kalman filter also accounts for the transport delay in the vision system and provides timely state estimates necessary for stable closed loop servoing. Chapter 3 also introduced detailed models of

the vision system, the servo motor, the amplifier, the position compensator, and the planar robot. These models were used in simulation to predict the performance of a planar robot using direct visual servoing. Chapter 4 describes the implementation of a working direct visual servo to validate the simulation results of Chapter 3. Chapter 4 highlights numerous practical hardware and software issues including the system bottlenecks that dictate an upper limit to the number of cameras. Chapter 4 leads to the construction of a system consisting of four cameras which is used to perform the various experiments described in Chapter 5. The experiments focused on the transient step response, servo-hold performance, and external disturbance rejection. The experiments were first run without any occlusions then repeated under full and partial occlusions. Additional experiments demonstrated the performance under varying illumination and illustrated local task supervision in a tele-robotic application.

6.2 Results

Both simulated and experimental results are provided in Chapter 3 and Chapter 5 respectively. All results are based on the direct visual servoing of a planar robot. The results are summarized in the following subsections.

6.2.1 Simulation Results

The simulations predicted that the multiple synchronized cameras combined with a suitable Kalman filter could provide stable closed loop position control for a direct visual servo. The simulations also demonstrated how increasing the number of synchronized cameras increased the effective vision sample rate and improved the transient response time. Another simulation that was performed predicted that the response to a disturbance input would require a time equal to the latency in the vision system before the controller could respond.

This is due to the fact that a disturbance input is an unmodelled event that cannot be anticipated by the predictor. Finally, a modest occlusion was simulated and the estimated positions were plotted for both the stationary and the non-stationary Kalman filter. The stationary Kalman filter shows a large sensitivity to occlusions and the estimated position exhibits large variations. In contrast, the position estimates produced by the non-stationary Kalman filter remains much closer to the actual position in the presence of occlusions.

6.2.2 Experimental Results

Actual experimental results were also obtained for the direct visual servoing of a planar robot using the implementation described in Chapter 4. The risetime of the step response of the direct visual servo was experimentally shown to closely match that of a traditional encoder based position loop and the simulated predictions. The response to an input disturbance also closely matched the simulated results.

Experiments were performed introducing various occlusions ranging from full occlusions in a subset of the cameras to partial occlusions in all cameras. The stationary Kalman filter exhibited instability in the presence of full occlusions in one camera. In contrast, a system employing a non-stationary Kalman filter maintained a stable servo-hold in the presence of various occlusions. The step response of systems employing a non-stationary Kalman filter in the presence of occlusions was stable but exhibited larger oscillations due to the degraded feedback from the cameras. Likewise, the response to an external disturbance remained stable in the presence of occlusions but with a poorer dynamic performance. Yet another experiment demonstrated that the non-stationary Kalman filter also shows promise for improved performance in the presence of small variations in illumination. In general, the experimental results demonstrate that a direct visual servoing system employing a non-stationary Kalman filter is robust to occlusions.

Finally, an experiment was performed to demonstrate the tele-operation of the planar

robot. The results demonstrate that the proposed approach also holds promise for telerobotic applications and can provide a degree of local task supervision.

6.3 Conclusions

Several conclusions can be drawn from this work. The results confirm that the effective vision sample rate of the system can be increased by using multiple network-synchronized cameras. When combined with an appropriate Kalman filter, this approach provides a system capable of direct visual servoing. The use of a non-stationary Kalman filter is recommended over that of the simpler stationary Kalman filter. All the results indicate that the non-stationary Kalman filter is far superior in coping with noise and occlusions. The statistical relationship uncovered between Euclidean distance and measurement error variance predicted in simulation works well for improving the robustness of the system. However, each new application will need to determine the constant factor m in equation 3.56 and the maximum Euclidean distance over which this equation is valid. Before determining the constant factor m, each new application will require the selection of a suitable sub-image size and an appropriate number of eigenvectors based on position accuracy specifications and available computational power.

The experimental results also manifest several advantages and shortcomings of this approach to direct visual servoing. These aspects are summarized in the following subsections.

6.3.1 Strengths of the Approach

The advantages of this approach are numerous. Firstly, the system does not rely on any proprietary hardware but uses off-the-shelf components. The approach uses flexible and computationally efficient eigenspace vision methods which work for general objects and backgrounds and require no camera calibrations. Also, the distributed computing power

provides scalable vision sample rates by selecting the number of vision nodes. The number of vision nodes is only limited by the network and the processor utilization of the master servo computer. Besides improving the visual sample rate, the multiple vision nodes provide a measure of redundancy allowing the system to continue functioning even when one camera completely fails. This is evident in the experimental results when the robot continues to servo even when one camera is completely occluded. Besides being resilient to partial failure, the use of multiple cameras improves the likelihood that when some cameras are occluded that others may still have a clear view of the target object. Finally, the results show that this approach is robust in the presence of noise and occlusions.

6.3.2 Shortcomings of the Approach

This approach also has some shortcomings. The Kalman filter predicts the state outputs to reduce the effects of latency in the vision system and provide stable closed loop control. The performance of the predictor is highly sensitive to modelling accuracy and a poor model of the system will lead to inaccurate state estimates and reduced position accuracy in the robot. The model used for feedforward is also critical. If the torque feedforward does not match the amount required to hold the joint in a given position then the torque error will be taken up by the position loop. This torque error will appear to the predictor like an accelerating torque which will result in increased errors in the position predictions. Also, the Kalman filter predictor cannot anticipate unmodelled target motion such as external disturbances. This results in a delayed response to external disturbances due to the latency in the vision feedback.

Another shortcoming of this approach is the difficulty in extending this work to robots with multiple degrees of freedom. Multiple degrees of freedom imply a more comprehensive learning phase when using eigenspace vision methods. This can become unwieldy

since the number of points on the manifold grows exponentially with the number of degrees of freedom of the robot. It is not practical to obtain training images for a typical six degree-of-freedom robot for all robot positions. Therefore, for higher degree-of-freedom robots, the training images may need to be limited to points surrounding predetermined regions of interest.

Despite these shortcomings, with an appropriate number of off-the-shelf cameras and computers, this approach shows promise for many high-speed vision applications such as the direct visual servoing of many practical robotic systems.

6.3.3 Limitations on the Number of Cameras

Finally, this approach has some practical limitations as to the number of cameras that can be supported as discussed in section 4.5. A 100Mbps Ethernet network has sufficient bandwidth to support a maximum of 826 cameras. This number is reduced to 609 cameras if the packet payloads are increased to support tele-robotic operations. However, the main limitation turns out to be available processor time on the master servo computer. The processor utilization on the master servo computer was exhausted long before the before the limitation due to network bandwidth was encountered. The number of states (n) grow linearly as the number of cameras (N_c) increase. The Kalman filter and controller computations grow by $O(n^3)$ as the number of states increase. Subsequent analysis showed a limit of 14 cameras when using a 400MHz Pentium II processor for the master servo computer. This corresponds to an effective visual sampling rate of 840Hz. This limit is not fixed but can be increased by optimizing code and using faster processors.

6.4 Research Contributions

There are several distinctive aspects to this work. First, to be able to perform *direct* visual servoing without using any traditional position or velocity feedback to stabilize the

position loop is unique. The visual feedback is being used to directly control the motor current and hence the torque without the use of any other sensors. Almost all the literature in visual servoing describes work based on the dynamic "look-and-move" approach which relies on an inner position loop using traditional position sensors. Direct visual servoing was selected because of the unique challenges it presents. Direct visual servo control has the potential to achieve faster responses than "look-and-move" systems. Direct visual servoing presents a variety of difficult practical and theoretical design problems including overcoming slow vision sample rates and long transport delays. The direct visual servo described is position-based and is characterized by the use of multiple fixed cameras and planar positioning.

The use of multiple cameras synchronized over a network to increase the vision sample rate is also novel. The technique has the further advantage that the visual sample rate is completely *scalable*. Furthermore, the network-synchronized cameras do not rely on eigenspace techniques and can therefore be employed with alternative vision algorithms in traditional visual servoing applications. The techniques that were used to avoid packet collisions and perform round-robin polling demonstrate how a common Ethernet network can be adapted to provide deterministic performance for use in industrial control systems.

Although the use of eigenspace vision methods is not new, this work has addressed the problem of occlusions and noise which are a particular challenge when employing eigenspace methods. The determination of measurement error variance directly from Euclidean distance provides an elegant means for employing a non-stationary Kalman filter. Often the measurement error variance used in conjunction with a Kalman filter is difficult to obtain and the practical issues of identifying them are seldom discussed in the literature. The technique developed in this work provides a method for quickly computing the measurement error variance which results in a system which is robust to noise and occlusions.

Finally, the tele-robotic extensions demonstrate how eigenspace methods can be used

to simultaneously provide position measurements and compress images for remote monitoring. The issue of local task supervision is also addressed by using information available in the Euclidean distance and the Kalman filter.

Although this work explores a specific category of visual servoing, many of the results can be applied to other types of visual servoing or other machine vision problems. For instance, the techniques used to increase vision sample rates can be applied to "look-and-move" visual servoing to improve dynamic performance. The network synchronized cameras can be applied to general high speed vision problems beyond visual servoing. Likewise, the solutions in this work relating to issues associated with noise and long transport delays are perennial problems in machine vision. In particular, the novel technique for dealing with occlusions can not only be applied to any visual servoing structure but is relevant to a wide variety of machine vision problems.

6.5 Future Work

The area of direct visual servoing as a whole is one branch in the taxonomy of visual servoing which has received very little attention in the literature and is ripe for further research. The lack of work in this area is due in part to numerous practical barriers including limited video frame rates and the latency associated with image processing. This work has strived to address some of these challenges and demonstrate a working concept for direct visual servoing. However, there are numerous opportunities for future work in this area.

6.5.1 Future Theoretical Work

There are some recommended areas that could form the focus for future work. One obvious research task would be to extend the eigenspace methods from the planar robot to robots with more degrees of freedom. This is a challenging problem since the number of points on the manifold grow exponentially with the number of degrees of freedom of

the robot. Therefore, new techniques will need to be found to support higher degree-of-freedom robots.

Another area for further study is examining further the robustness to variations in illumination. The problem of illumination is a perennial problem for machine vision in general and any advances in this area would be welcome in many applications. This may involve evaluating subspaces based on something other than Principal Component Analysis.

The accuracy of the state estimator could be improved by using additional available feedback. There is observable state information that can be gleaned from within the servo motor itself such as the back *emf*. The back *emf* can provide a low latency measurement of motor velocity which could greatly improve position predictions and the response time to external disturbances.

Other future work could rely more heavily on system simulation, especially if the simulation models were improved. The servo motor model could be enhanced to include higher order effects such as magnetic hysteresis, eddy current losses, armature reaction, and commutation effects. The servo amplifier model could also be improved to include higher order effects relating to the PWM control. These effects include power switch conduction and switching losses and current loop dynamics. The physical model of the robot could also be enhanced by more accurately modelling motor inertia along with the friction and windage losses. Some of the improvements to the simulation models could then be incorporated into the state predictor to improve the state estimates produced by the Kalman filter. The torque feedforward could also be improved by modelling not only gravitational effects but other forces such as the cogging torque due to the saliency in the motor poles. This would reduce the state estimation errors produced by the predictor when the torque feedforward does not exactly cancel the static forces on the robot.

Finally, there are opportunities for future work relating to visual servoing in the area of tele-robotics. Some of the challenges include tele-operation over channels with long and sometimes variable transport delays. The difficulties are compounded when a robot must

work in unstructured environments. The applications that could benefit from this research include tele-medicine and space exploration.

6.5.2 Future Implementation Work

There are also several possibilities for future work which could focus more on various implementation issues surrounding the network and the vision nodes.

There are numerous alternatives to using an Ethernet network. All video images are processed within every vision node so the network is not required to transmit any large video frames. The only information passing over the network consists of position and Euclidean distance information. Since the packets consist of a modest 8 byte payload, the network demands are small. This enables this technique to be implemented using networks that have a small MTU (Maximum Transfer Unit) and modest bandwidths such as wireless networks. The most important network attribute is deterministic delays since the Kalman filter system model assumes a fixed transport delay. Consequently, this approach would be suitable for use with with a variety of wireless protocols. Therefore, is is possible to create a wireless network of vision nodes which would result in a reduction in wiring complexity and enable remote control.

Finally, work could be done to consolidate all the components of a vision node into one compact package. This package would constitute a "smart camera" that could be used to form ad-hoc networks of intelligent visual sensors. Such intelligent cameras would be both flexible and convenient. Each camera could be capable of training itself to perform pattern recognition and report only the results back to a central host. As computers continue to increase in performance and decrease in size and cost, the notion of a "smart camera" will become increasingly attractive for a wide variety of machine vision applications.

Bibliography

[1] T.E. Anderson, D.E. Culler, and D.A. Patterson. A case for NOW (Networks Of Workstations). *IEEE Micro Magazine*, pages 54–64, February 1995.

[2] R.L. Andersson. Dynamic sensing in a ping-pong playing robot. *IEEE Transactions on Robotics and Automation*, 5(6):728–739, December 1989.

[3] W.L. Bialkowski. Application of Kalman filters to the regulation of dead time processes. *IEEE Transactions on Automatic Control*, AC-28(3):400–406, March 1983.

[4] G.M. Bone and D.W. Capson. Vision-guided fixtureless assembly of automotive components. Accepted for the journal *Robotics and Computer-Integrated Manufacturing*.

[5] D.P. Bovet and M. Cesati. *Understanding the Linux Kernel*. O'Reilly & Associates Inc., 2001.

[6] R.G. Brown and P.Y.C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering*. John Wiley & Sons, third edition, 1997.

[7] G.C. Buttazzo, B. Allotta, and F.P. Fanizza. Mousebuster: A robot system for catching fast moving objects by vision. In *Proceedings of the International Conference on Robotics and Automation*, Atlanta, GA, May 1993. IEEE.

[8] S. Cass. Robosoccer: A new breed of robots takes to the playing field. *IEEE Spectrum*, May 2001.

[9] F. Chaumette. Potential problems of stability and convergence in image-based and position-based visual servoing. In D. Kriegman, G. Hager, and S. Morse, editors, *The Confluence of Vision and Control*, volume 237, pages 66–78. Springer-Verlag, 1998.

[10] P. Corke, J. Roberts, and G. Winstanley. Vision-based control for mining automation. *IEEE Robotics and Automation Magazine*, pages 44–49, December 1998.

[11] P.I. Corke and M.C. Good. Dynamic effects in visual closed-loop systems. *IEEE Transactions on Robotics and Automation*, 12(5):671–683, October 1996.

[12] P.I. Corke and S.A. Hutchinson. Real-time vision, tracking and control. In *Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA, April 2000. IEEE.

[13] P.I. Corke and S.A. Hutchinson. A new partitioned approach to image-based visual servo control. *IEEE Transactions on Robotics and Automation*, 17(4):507–515, August 2001.

[14] P.A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach.* Prentice Hall Inc., 1982.

[15] K. Dutton, S. Thompson, and B. Barraclough. *The Art of Control Engineering.* Addison-Wesley, 1998.

[16] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(2):313–326, June 1992.

[17] H. Nakai et al. A volleyball playing robot. In *Proceedings of the International Conference on Robotics and Automation*, Leuven, Belgium, May 1998. IEEE.

[18] J.T. Feddema, C.S.G. Lee, and O.R. Mitchell. Weighted selection of image features for resolved rate visual feedback control. *IEEE Transactions on Robotics and Automation*, 7(1):31–47, February 1991.

[19] J. Fortuna, D.C. Schuurman, and D.W. Capson. A comparison of PCA and ICA for object recognition under varying illumination. In *Proceedings of the International Conference on Pattern Recognition*, volume 3, pages 11–15, Quebec City, PQ, August 2002. IAPR.

[20] B.O. Gallmeister. *POSIX.4: Programming for the Real World.* O'Reilly & Associates Inc., 1995.

[21] J.A. Gangloff and M.F. de Mathelin. High speed visual servoing of a 6 DOF manipulator using mimo predictive control. In *Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA, April 2000. IEEE.

[22] G.D. Hager, G. Grunwald, and G. Hirzinger. Feature-based visual servoing and its application to telerobotics. In *Proceedings of the International Conference on Intelligent Robots and Systems*, Munich, Germany, September 1994.

[23] D. Halliday and R. Resnick. *Fundamentals of Physics.* John Wiley & Sons, second edition, 1986.

[24] R.C. Harrell, D.C. Slaughter, and P.D. Adsit. A fruit-tracking system for robotic harvesting. *Machine Vision and Applications*, 2(2):69–80, 1989.

[25] K. Hashimoto and H. Kimura. Visual servoing with nonlinear observer. In *Proceedings of the International Conference on Robotics and Automation*, pages 484–489. IEEE, 1995.

[26] K. Hashimoto and T. Noritsugu. Visual servoing with linearized observer. In *Proceedings of the International Conference on Robotics and Automation*, Detroit, MI., May 1999. IEEE.

[27] C.C.W. Hulls and W.J. Wilson. Design of a real-time computer systems for relative position robot control. In *Proceedings of the Fourth Euromicro Workshop on Real-Time Systems*, pages 38–43, June 1992.

[28] S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, October 1996.

[29] IEEE, New York, NY. *Portable Operating System Interface (POSIX) - Part 1: System Application Program Interface (API) - Amendment 1: Realtime Extension [C Language]*, 1994. IEEE Standard 1003.1b-1993.

[30] IEEE, New York, NY. *IEEE Standard 802.3 Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifictions*, 2002. IEEE Standard 802.3-2002.

[31] M. Ishikawa and T. Komuro. Digital vision chips and high-speed vision systems. In *Symposium on VLSI Circuits Digest of Technical Papers*. IEEE, 2001.

[32] M. Ishikawa, A. Morita, and N. Takayanagi. High speed vision system using massively parallel processing. In *Proceedings of the International Conference on Intelligent Robots and Systems*, Raleigh, NC, July 1992. IEEE/RSJ.

[33] J.E. Jackson. *A User's Guide to Principal Components*. John Wiley & Sons, 1991.

[34] M. Jägersand, O. Fuentes, and R. Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *Proceedings of the International Conference on Robotics and Automation*, Albuquerque, NM, April 1997. IEEE.

[35] Ramesh Jain, Rangachar Kasturi, and Brian G. Schunck. *Introduction to Machine Vision*. McGraw-Hill Ryerson Limited, 1995.

[36] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, pages 35–45, 1960.

[37] *IEEE Transactions on Automatic Control*, Special Issue on Kalman Filter Applications, AC-28(3), March 1983.

[38] G. Kaplan. Ethernet's winning ways. *IEEE Spectrum*, pages 113–115, January 2001.

[39] J. Krumm. Eigenfeatures for planar pose measurement of partially occluded objects. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 55–60. IEEE, 1996.

[40] A. Leonardis and H. Bischof. Dealing with occlusions in the eigenspace approach. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 453–458. IEEE, 1996.

[41] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20:46–61, January 1973.

[42] E. Malis, F. Chaumette, and S. Boudet. 2-1/2-D visual servoing. *IEEE Transactions on Robotics and Automation*, 15(2):238–250, April 1999.

[43] E. Malis, F. Chaumette, and S. Boudet. Multi-cameras visual servoing. In *Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA., April 2000. IEEE.

[44] S. Mann. *Linux TCP/IP Network Administration*. Prentice Hall, 2002.

[45] H. Murase and S.K. Nayar. Visual learning and recognition of 3D objects from appearance. *International Journal of Computer Vision*, 14(1), 1995.

[46] Y. Nakabo, M. Ishikawa, H. Toyoda, and S. Mizuno. 1ms column parallel vision system and it's application of high speed target tracking. In *Proceedings of the International Conference on Robotics and Automation*, San Francisco, CA, April 2000. IEEE.

[47] S.K. Nayar, H. Murase, and S.A. Nene. Learning, positioning, and tracking visual appearance. In *Proceedings of the International Conference on Robotics and Automation*, San Diego, May 1994. IEEE.

[48] S.K. Nayar, S.A. Nene, and H. Murase. Real-time 100 object recognition system. In *Proceedings of the International Conference on Robotics and Automation*, Minneapolis, MN., April 1996. IEEE.

[49] S.K. Nayar, S.A. Nene, and H. Murase. Subspace methods for robot vision. *IEEE Transactions on Robotics and Automation*, 12(5):750–758, October 1996.

[50] P.R. Oliveira and R.F. Romero. A comparison between PCA and neural networks and the JPEG standard for performing image compression. In *Proceedings of the IEEE Workshop on Cybernetic Vision*. IEEE, 1996.

[51] L.L. Peterson and B.S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers Inc., 1996.

[52] R. Ramshaw and R.G. Van Heeswijk. *Energy Conversion, Electric Motors & Generators*. Saunders College Publishing, 1990.

[53] A.A. Rizzi and D.E. Koditschek. Progress in spatial robot juggling. In *Proceedings of the International Conference on Robotics and Automation*, Nice, France, May 1992. IEEE.

[54] A. Rubini. *Linux Device Drivers*. O'Reilly & Associates Inc., 1998.

[55] A. C. Sanderson and L. E. Weiss. Image-based visual servo control using relational graph error signals. In *Proceedings of the International Conference on Cybernetics and Society*, pages 1074–1077. IEEE, 1980.

[56] D.C. Schuurman and D.W. Capson. Direct visual servoing using network-synchronized cameras and Kalman filter. In *Proceedings of the International Conference on Robotics and Automation*, pages 4191–4197, Washington, DC, May 2002. IEEE.

[57] D.C. Schuurman and D.W. Capson. Video-rate eigenspace methods for position tracking and remote monitoring. In *Proceedings of the Southwest Symposium on Image Analysis and Interpretation*, pages 45–49, Santa Fe, NM, April 2002. IEEE.

[58] Y. Shirai and H. Inoue. Guiding a robot by visual fedback in assembling tasks. *Pattern Recognition*, 5:99–108, 1973.

[59] W. Stallings. *Operating Systems*. Prentice Hall, fourth edition, 2001.

[60] W.R. Stevens. *UNIX Network Programming*, volume 1. Prentice Hall, second edition, 1998.

[61] M.E. Stieber, M. McKay, G. Vukovich, and E. Petriu. Vision-based sensng and control for space robotics applications. *IEEE Transactions on Instrumentation and Measurement*, 48(4):807–812, August 1999.

[62] M.E. Stieber, C.P. Trudel, and D.G. Hunter. Robotics systems for the international space station. In *Proceedings of the International Conference on Robotics and Automation*, Albuquerque, NM, April 1997. IEEE.

[63] A.S. Tanenbaum. *Computer Networks*. Prentice Hall, third edition, 1996.

[64] F. Tendick, J. Voichick, G. Tharp, and L. Stark. A supervisory telerobotic control system using model-based vision feedback. In *Proceedings of the International Conference on Robotics and Automation*, Sacramento, CA, April 1991.

[65] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 586–591. IEEE, June 1991.

[66] M. Uenohara and T. Kanade. Use of Fourier and Karhunen-Loève decomposition for fast pattern matching with a large set of templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(8), August 1997.

[67] Markus Vincze and Gregory D. Hager. *Robust Vision for Vision-Based Control of Motion*. SPIE Optical Engineering Press and IEEE Press, 2000.

[68] J. Wang and W.J. Wilson. 3D relative position and orientation estimation using Kalman filtering for robot control. In *Proceedings of the International Conference on Robotics and Automation*, pages 2638–2645, Nice, France, May 1992. IEEE.

[69] C.R. Weisbin and G. Rodriguez. NASA robotics research for planetary surface exploration. *IEEE Robotics and Automation Magazine*, pages 25–34, December 2000.

[70] L.E. Weiss, A.C. Sanderson, and C.P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Transactions on Robotics and Automation*, RA-3(5):404–417, October 1987.

[71] D.B. Westmore and W.J. Wilson. Direct dynamic control of a robot using an end-point mounted camera and Kalman filter position estimation. In *Proceedings of the International Conference on Robotics and Automation*, pages 2376–2384, Sacramento, CA, April 1991. IEEE.

[72] W.J. Wilson, C.C. Williams, and G.S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, pages 684–696, October 1996.

[73] W. Wolf, B. Ozer, and T. Lv. Smart cameras as embedded systems. *IEEE Computer*, pages 48–53, September 2002.

[74] J.L. Wyatt, D.L. Standley, and W. Yang. The MIT vision chip project: Analog VLSI systems for fast image acquisition and early vision processing. In *Proceedings of the International Conference on Robotics and Automation*, Sacramento, CA, April 1991. IEEE.

[75] D.B. Zhang, L. Van Gool, and A. Oosterlinck. Stochastic predictive control of robot tracking systems with dynamic visual feedback. In *Proceedings of the International Conference on Robotics and Automation*, Cincinnati, OH, May 1990. IEEE.