VISUAL CONTROL OF ROBOTS:

High-Performance Visual Servoing

Peter I. Corke
CSIRO Division of Manufacturing Technology, Australia.

To my family, Phillipa, Lucy and Madeline.

Editorial foreword

It is no longer necessary to explain the word 'mechatronics'. The world has become accustomed to the blending of mechanics, electronics and computer control. That does not mean that mechatronics has lost its 'art'.

The addition of vision sensing to assist in the solution of a variety of problems is still very much a 'cutting edge' topic of research. Peter Corke has written a very clear exposition which embraces both the theory and the practical problems encountered in adding vision sensing to a robot arm.

There is great value in this book, both for advanced undergraduate reading and for the researcher or designer in industry who wishes to add vision-based control.

We will one day come to expect vision sensing and control to be a regular feature of mechatronic devices from machine tools to domestic appliances. It is research such as this which will bring that day about.

John Billingsley
University of Southern Queensland,
Toowoomba, QLD4350
August 1996

Author's Preface

Outline

This book is about the application of high-speed machine vision for closed-loop position control, or visual servoing, of a robot manipulator. The book aims to provide a comprehensive coverage of all aspects of the visual servoing problem: robotics, vision, control, technology and implementation issues. While much of the discussion is quite general the experimental work described is based on the use of a high-speed binary vision system with a monocular 'eye-in-hand' camera.

The particular focus is on accurate high-speed motion, where in this context 'high speed' is taken to mean approaching, or exceeding, the performance limits stated by the robot manufacturer. In order to achieve such high-performance I argue that it is necessary to have accurate dynamical models of the system to be controlled (the robot) and the sensor (the camera and vision system). Despite the long history of research in the constituent topics of robotics and computer vision, the system dynamics of closed-loop visually guided robot systems has not been well addressed in the literature to date.

I am a confirmed experimentalist and therefore this book has a strong theme of experimentation. Experiments are used to build and verify models of the physical system components such as robots, cameras and vision systems. These models are then used for controller synthesis, and the controllers are verified experimentally and compared with results obtained by simulation.

Finally, the book has a World Wide Web home page which serves as a virtual appendix. It contains links to the software and models discussed within the book as well as pointers to other useful sources of information. A video tape, showing many of the experiments, can be ordered via the home page.

Background

My interest in the area of visual servoing dates back to 1984 when I was involved in two research projects; video-rate feature extraction[1], and sensor-based robot control. At that time it became apparent that machine vision could be used for closed-loop control of robot position, since the video-field rate of 50 Hz exceeded the position setpoint rate of the Puma robot which is only 36 Hz. Around the same period Weiss and Sanderson published a number of papers on this topic [224–226, 273] in particular concentrating on control strategies and the direct use of image features — but only in simulation. I was interested in actually building a system based on the feature-extractor and robot controller, but for a number of reasons this was not possible at that time.

[1] This work resulted in a commercial unit — the APA-512 [261], and its successor the APA-512+ [25]. Both devices are manufactured by Atlantek Microsystems Ltd. of Adelaide, Australia.

In the period 1988–89 I was fortunate in being able to spend 11 months at the GRASP Laboratory, University of Pennsylvania on a CSIRO Overseas Fellowship. There I was able to demonstrate a 60 Hz visual feedback system [65]. Whilst the sample rate was high, the actual closed-loop bandwidth was quite low. Clearly there was a need to more closely model the system dynamics so as to be able to achieve better control performance. On return to Australia this became the subject of my PhD research [52].

Nomenclature

The most commonly used symbols used in this book, and their units are listed below. Note that some symbols are overloaded in which case their context must be used to disambiguate them.

$\underline{v}$	a vector	
$\underline{v}_x$	a component of a vector	
$\mathbf{A}$	a matrix	
$\hat{x}$	an estimate of x	
x	error in x	
x_d	demanded value of x	
$\mathbf{A}^T$	transpose of A	
α_x, α_y	pixel pitch	pixels/mm
B	viscous friction coefficient	N.m.s/rad
$\mathbf{C}$	camera calibration matrix (3×4)	
$\mathbf{C}(\underline{q}, \dot{\underline{q}})$	manipulator centripetal and Coriolis term	kg.m^2/s
ceil(x)	returns n, the smallest integer such that $n \geq x$	
E	illuminance (lux)	lx
f	force	N
f	focal length	m
F	f-number	
$F(\dot{q})$	friction torque	N.m
floor(x)	returns n, the largest integer such that $n \leq x$	
G	gear ratio	
ϕ	luminous flux (lumens)	lm
ϕ	magnetic flux (Webers)	Wb
$\mathbf{G}$	gear ratio matrix	
$\mathbf{G}(\underline{q})$	manipulator gravity loading term	N.m
i	current	A
$\mathbf{I}_n$	$n \times n$ identity matrix	
j	$\sqrt{-1}$	
J	scalar inertia	kg.m^2

$\mathbf{J}$	inertia tensor, 3×3 matrix	kg.m^2
${}^A\mathbf{J}_B$	Jacobian transforming velocities in frame A to frame B	
k, K	constant	
K_i	amplifier gain (transconductance)	A/V
K_m	motor torque constant	N.m/A
$\mathtt{K}\{\}$	forward kinematics	
$\mathtt{K}^{-1}\{\}$	inverse kinematics	
L	inductance	H
L	luminance (nit)	nt
m_i	mass of link i	kg
$\mathbf{M}(\underline{q})$	manipulator inertia matrix	kg.m^2
$\mathrm{Ord}()$	order of polynomial	
$\underline{q}$	generalized joint coordinates	
$\underline{Q}$	generalized joint torque/force	
R	resistance	Ω
θ	angle	rad
$\underline{\theta}$	vector of angles, generally robot joint angles	rad
s	Laplace transform operator	
$\underline{s}_i$	COM of link i with respect to the link i coordinate frame	m
$\underline{S}_i$	first moment of link i. $\underline{S}_i = m_i \underline{s}_i$	kg.m
σ	standard deviation	
t	time	s
T	sample interval	s
T	lens transmission constant	
T_e	camera exposure interval	s
$\mathbf{T}$	homogeneous transformation	
${}^A\mathbf{T}_B$	homogeneous transform of point B with respect to the frame A. If A is not given then assumed relative to world coordinate frame 0. Note that ${}^A\mathbf{T}_B = \left({}^B\mathbf{T}_A\right)^{-1}$.	
τ	torque	N.m
τ_C	Coulomb friction torque	N.m
v	voltage	V
ω	frequency	rad/s
$\underline{x}$	3-D pose, $\underline{x} = \left[x\ y\ z\ r_x\ r_y\ r_z\right]^T$ comprising translation along, and rotation about the X, Y and Z axes.	
x, y, z	Cartesian coordinates	
X_0, Y_0	coordinates of the principal point	pixels
${}^i x, {}^i y$	camera image plane coordinates	m
${}^i X, {}^i Y$	camera image plane coordinates	pixels
${}^i\underline{X}$	camera image plane coordinates ${}^i\underline{X} = ({}^i X, {}^i Y)$	pixels
${}^i X$	image plane error	

z	z-transform operator
$\mathrm{Z}\{\}$	Z-transform

The following conventions have also been adopted:

- Time domain variables are in lower case, frequency domain in upper case.
- Transfer functions will frequently be written using the notation

$$K(a)[\zeta, \omega_n] = K\left(\frac{s}{a}+1\right)\left[\frac{1}{\omega_n^2}s^2+\frac{2\zeta}{\omega_n}s+1\right]$$

  A free integrator is an exception, and (0) is used to represent s.

- When specifying motor motion, inertia and friction parameters it is important that a consistent reference is used, usually either the motor or the load, denoted by the subscripts m or l respectively.

  For numeric quantities the units radm and radl are used to indicate the reference frame.

- In order to clearly distinguish results that were experimentally determined from simulated or derived results, the former will always be designated as 'measured' in the caption and index entry.
- A comprehensive glossary of terms and abbreviations is provided in Appendix A.

Acknowledgements

The work described in this book is largely based on my PhD research [52] which was carried out, part time, at the University of Melbourne over the period 1991–94. My supervisors Professor Malcolm Good at the University of Melbourne, and Dr. Paul Dunn at CSIRO provided much valuable discussion and guidance over the course of the research, and critical comments on the draft text.

That work could not have occurred without the generosity and support of my employer, CSIRO. I am indebted to Dr. Bob Brown and Dr. S. Ramakrishnan for supporting me in the Overseas Fellowship and PhD study, and making available the necessary time and laboratory facilities. I would like to thank my CSIRO colleagues for their support of this work, in particular: Dr. Paul Dunn, Dr. Patrick Kearney, Robin Kirkham, Dennis Mills, and Vaughan Roberts for technical advice and much valuable discussion; Murray Jensen and Geoff Lamb for keeping the computer systems running; Jannis Young and Karyn Gee, the librarians, for tracking down all manner of references; Les Ewbank for mechanical design and drafting; Ian Brittle's Research Support Group for mechanical construction; and Terry Harvey and Steve Hogan for electronic construction. The PhD work was partially supported by a University of Melbourne/ARC small grant. Writing this book was partially supported by the Cooperative Research Centre for Mining Technology and Equipment (CMTE), a joint venture between AMIRA, CSIRO, and the University of Queensland.

Many others helped as well. Professor Richard (Lou) Paul, University of Pennsylvania, was there at the beginning and made facilities at the GRASP laboratory available to me. Dr. Kim Ng of Monash University and Dr. Rick Alexander helped in discussions on camera calibration and lens distortion, and also loaned me the SHAPE system calibration target used in Chapter 4. Vision Systems Ltd. of Adelaide, through their then US distributor Tom Seitzler of Vision International, loaned me an APA-512 video-rate feature extractor unit for use while I was at the GRASP Laboratory. David Hoadley proof read the original thesis, and my next door neighbour, Jack Davies, fixed lots of things around my house that I didn't get around to doing.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Visual servoing

Visual servoing is a rapidly maturing approach to the control of robot manipulators that is based on visual perception of robot and workpiece location. More concretely, visual servoing involves the use of one or more cameras and a computer vision system to control the position of the robot's end-effector relative to the workpiece as required by the task.

Modern manufacturing robots can perform assembly and material handling jobs with speed and precision, yet compared to human workers robots are at a distinct disadvantage in that they cannot 'see' what they are doing. In industrial applications, considerable engineering effort is therefore expended in providing a suitable work environment for these blind machines. This entails the design and manufacture of specialized part feeders, jigs to hold the work in progress, and special purpose end-effectors. The resulting high non-recurrent engineering costs are largely responsible for robots failing to meet their initial promise of being versatile reprogrammable workers [84] able to rapidly change from one task to the next.

Once the structured work environment has been created, the spatial coordinates of all relevant points must then be *taught*. Ideally, teaching would be achieved using data from CAD models of the workplace, however due to low robot accuracy manual teaching is often required. This low accuracy is a consequence of the robot's tool-tip pose being inferred from measured joint angles using a model of the robot's kinematics. Discrepancies between the model and the actual robot lead to tool-tip pose errors.

Speed, or cycle time, is the critical factor in the economic justification for a robot. Machines capable of extremely high tool-tip accelerations now exist but the overall cycle time is dependent upon other factors such as settling time and overshoot. High

speed and acceleration are often achieved at considerable cost since effects such as rigid-body dynamics, link and transmission flexibility become significant. To achieve precise end-point control using joint position sensors the robot must be engineered to minimize these effects. The AdeptOne manipulator for instance, widely used in high-speed electronic assembly, has massive links so as to achieve high rigidity but this is at the expense of increased torque necessary to accelerate the links. The problems of conventional robot manipulators may be summarized as:

1. It is necessary to provide, at considerable cost, highly structured work environments for robots.
2. The limited accuracy of a robot frequently necessitates time-consuming manual teaching of robot positions.
3. Mechanical dynamics in the robot's structure and drive train fundamentally constrain the minimum cycle time.

A visually servoed robot does not need to know *a priori* the coordinates of its workpieces or other objects in its workspace. In a manufacturing environment visual servoing could thus eliminate robot teaching and allow tasks that were not strictly repetitive, such as assembly without precise fixturing and with incoming components that were unoriented or perhaps swinging on overhead transfer lines.

Visual servoing also provides the potential to relax the mechanical accuracy and stiffness requirements for robot mechanisms and hence reduce their cost. The deficiencies of the mechanism would be compensated for by a vision sensor and feedback so as to achieve the desired accuracy and endpoint setting time. Jägersand [133] for example shows how positioning accuracy of a robot with significant backlash was improved using visual servoing. Such issues are significant for ultra-fine pitch electronic assembly [126] where planar positioning accuracy of 0.5μm and rotational accuracy of $0.1°$ will be required and settling time will be significant. Moore's Law[1] provides an economic motivation for this approach. Mechanical engineering is a mature technology and costs do not decrease with time. Sensors and control computers on the other hand have, and will continue to, exhibit dramatic improvement in performance to price ratio over time.

Visual servoing is also applicable to the unstructured environments that will be encountered by field and service robots. Such robots must accomplish tasks even though the exact location of the robot and workpiece are not known and are often not practicably measurable. Robotic fruit picking [206], for example, requires the robot whose location is only approximately known to grasp a fruit whose position is also unknown and perhaps varying with time.

[1] Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double approximately every 18 months.

The use of vision with robots has a long history [291] and today vision systems are available from major robot vendors that are highly integrated with the robot's programming system. Capabilities range from simple binary image processing to more complex edge- and feature-based systems capable of handling overlapped parts [35]. The common characteristic of all these systems is that they are static, and typically image processing times are of the order of 0.1 to 1 second. In such systems visual sensing and manipulation are combined in an open-loop fashion, 'looking' then 'moving'.

The accuracy of the 'look-then-move' approach depends directly on the accuracy of the visual sensor and the robot manipulator. An alternative to increasing the accuracy of these sub-systems is to use a visual-feedback control loop which will increase the overall accuracy of the system. Taken to the extreme, machine vision can provide closed-loop position control for a robot end-effector — this is referred to as *visual servoing*. The term *visual servoing* appears to have been first introduced by Hill and Park [116] in 1979 to distinguish their approach from earlier 'blocks world' experiments where the robot system alternated between picture taking and moving. Prior to the introduction of this term, the less specific term *visual feedback* was generally used. For the purposes of this book, the task in visual servoing is to use visual information to control the *pose*[2] of the robot's end-effector relative to a target object or a set of target features (the task can also be defined for mobile robots, where it becomes the control of the vehicle's pose with respect to some landmarks). The great benefit of feedback control is that the accuracy of the closed-loop system can be made relatively insensitive to calibration errors and non-linearities in the open-loop system. However the inevitable downside is that introducing feedback admits the possibility of closed-loop instability and this is a major theme of this book.

The camera(s) may be stationary or held in the robot's 'hand'. The latter case, often referred to as the eye-in-hand configuration, results in a system capable of providing endpoint relative positioning information directly in Cartesian or task space. This presents opportunities for greatly increasing the versatility and accuracy of robotic automation tasks.

Vision has not, to date, been extensively investigated as a high-bandwidth sensor for closed-loop control. Largely this has been because of the technological constraint imposed by the huge rates of data output from a video camera (around 10^7 pixels/s), and the problems of extracting meaning from that data and rapidly altering the robot's path in response. A vision sensor's raw output data rate, for example, is several orders of magnitude greater than that of a force sensor for the same sample rate. Nonetheless there is a rapidly growing body of literature dealing with visual servoing, though dynamic performance or bandwidth reported to date is substantially less than could be expected given the video sample rate. Most research seems to have concentrated on the computer vision part of the problem, with a simple controller sufficiently detuned to ensure stability. Effects such as tracking lag and tendency toward instability have

[2] Pose is the 3D position and orientation.

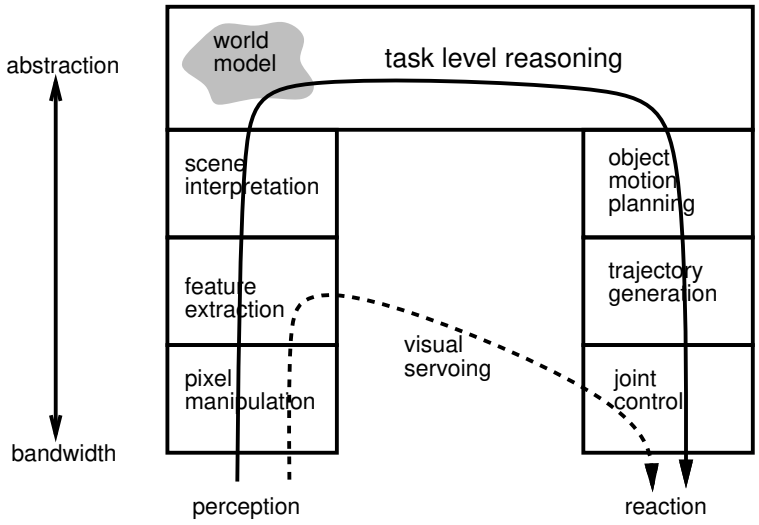


Figure 1.1: General structure of hierarchical model-based robot and vision system. The dashed line shows the 'short-circuited' information flow in a visual servo system.

been noted almost in passing. It is exactly these issues, their fundamental causes and methods of overcoming them, that are the principal focus of this book.

Another way of considering the difference between conventional look-then-move and visual servo systems is depicted in Figure 1.1. The control structure is hierarchical, with higher levels corresponding to more abstract data representation and lower bandwidth. The highest level is capable of reasoning about the task, given a model of the environment, and a look-then-move approach is used. Firstly, the target location and grasp sites are determined from calibrated stereo vision or laser rangefinder images, and then a sequence of moves is planned and executed. Vision sensors have tended to be used in this fashion because of the richness of the data they can produce about the world, in contrast to an encoder or limit switch which is generally dealt with at the lowest level of the hierarchy. Visual servoing can be considered as a 'low level' shortcut through the hierarchy, characterized by high bandwidth but moderate image processing (well short of full scene interpretation). In biological terms this could be considered as *reactive* or *reflexive* behaviour.

However not all 'reactive' vision-based systems are visual servo systems. Andersson's well known ping-pong playing robot [17], although fast, is based on a real-time expert system for robot path planning using ball trajectory estimation and considerable domain knowledge. It is a highly optimized version of the general architecture

shown in Figure 1.1.

1.1.1 Related disciplines

Visual servoing is the fusion of results from many elemental disciplines including high-speed image processing, kinematics, dynamics, control theory, and real-time computing. Visual servoing also has much in common with a number of other active research areas such as *active computer vision*, [9, 26] which proposes that a set of simple visual behaviours can accomplish tasks through action, such as controlling attention or gaze [51]. The fundamental tenet of active vision is not to interpret the scene and then model it, but rather to direct attention to that part of the scene relevant to the task at hand. If the system wishes to learn something of the world, rather than consult a model, it should consult the world by directing the sensor. The benefits of an *active* robot-mounted camera include the ability to avoid occlusion, resolve ambiguity and increase accuracy. Researchers in this area have built robotic 'heads' [157, 230, 270] with which to experiment with perception and gaze control strategies. Such research is generally motivated by neuro-physiological models and makes extensive use of nomenclature from that discipline. The scope of that research includes visual servoing amongst a broad range of topics including 'open-loop' or saccadic eye motion, stereo perception, vergence control and control of attention.

Literature related to *structure from motion* is also relevant to visual servoing. Structure from motion attempts to infer the 3D structure and the relative motion between object and camera, from a sequence of images. In robotics however, we generally have considerable *a priori* knowledge of the target and the spatial relationship between feature points is known. Aggarwal [2] provides a comprehensive review of this active field.

1.2 Structure of the book

Visual servoing occupies a niche somewhere between computer vision and robotics research. It draws strongly on techniques from both areas including image processing, feature extraction, control theory, robot kinematics and dynamics. Since the scope is necessarily broad Chapters 2–4 present those aspects of robotics, image formation and computer vision respectively that are relevant to development of the central topic. These chapters also develop, through analysis and experimentation, detailed models of the robot and vision system used in the experimental work. They are the foundations upon which the later chapters are built.

Chapter 2 presents a detailed model of the Puma 560 robot used in this work that includes the motor, friction, current, velocity and position control loops, as well as the more traditional rigid-body dynamics. Some conclusions are drawn regarding the

significance of various dynamic effects, and the fundamental performance limiting factors of this robot are identified and quantified.

Image formation is covered in Chapter 3 with topics including lighting, image formation, perspective, CCD sensors, image distortion and noise, video formats and image digitization. Chapter 4 discusses relevant aspects of computer vision, building upon the previous chapter, with topics including image segmentation, feature extraction, feature accuracy, and camera calibration. The material for Chapters 3 and 4 has been condensed from a diverse literature spanning photography, sensitometry, video technology, sensor technology, illumination, photometry and photogrammetry.

A comprehensive review of prior work in the field of visual servoing is given in Chapter 5. Visual servo kinematics are discussed systematically using Weiss's taxonomy [273] of image-based and position-based visual servoing.

Chapter 6 introduces the experimental facility and describes experiments with a single-DOF visual feedback controller. This is used to develop and verify dynamic models of the visual servo system. Chapter 7 then formulates the visual servoing task as a feedback control problem and introduces performance metrics. This allows the comparison of compensators designed using a variety of techniques such as PID, pole-placement, Smith's method and LQG. Feedback controllers are shown to have a number of limitations, and feedforward control is introduced as a means of overcoming these. Feedforward control is shown to offer markedly improved tracking performance as well as great robustness to parameter variation.

Chapter 8 extends those control techniques and investigates visual end-point damping and 3-DOF translational manipulator control. Conclusions and suggestions for further work are given in Chapter 9.

The appendices contain a glossary of terms and abbreviations and some additional supporting material. In the interests of space the more detailed supporting material has been relegated to a virtual appendix which is accessible through the World Wide Web. Information available via the web includes many of the software tools and models described within the book, cited technical reports, links to other visual servoing resources on the internet, and errata. Ordering details for the accompanying video tape compilation are also available. Details on accessing this information are given in Appendix B.

Chapter 2

Modelling the robot

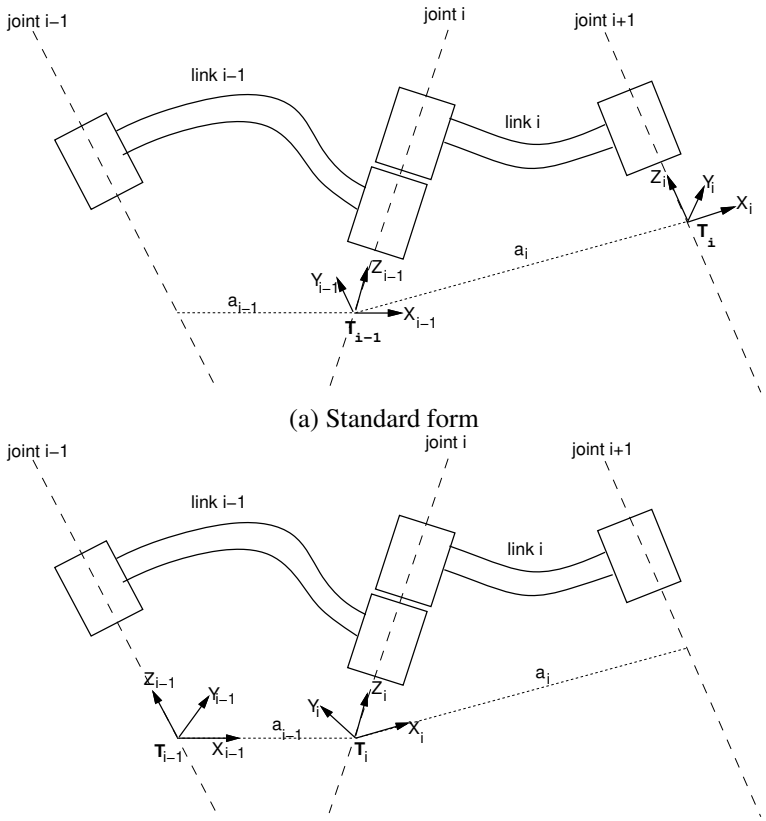
This chapter introduces a number of topics in robotics that will be called upon in later chapters. It also develops models for the particular robot used in this work — a Puma 560 with a Mark 1 controller. Despite the ubiquity of this robot detailed dynamic models and parameters are difficult to come by. Those models that do exist are incomplete, expressed in different coordinate systems, and inconsistent. Much emphasis in the literature is on rigid-body dynamics and model-based control, though the issue of model parameters is not well covered. This work also addresses the significance of various dynamic effects, in particular contrasting the classic rigid-body effects with those of non-linear friction and voltage saturation. Although the Puma robot is now quite old, and by modern standards has poor performance, this could be considered to be an 'implementation issue'. Structurally its mechanical design (revolute structure, geared servo motor drive) and controller (nested control loops, independent axis control) remain typical of many current industrial robots.

2.1 Manipulator kinematics

Kinematics is the study of motion without regard to the forces which cause it. Within kinematics one studies the position, velocity and acceleration, and all higher order derivatives of the position variables. The kinematics of manipulators involves the study of the geometric and time based properties of the motion, and in particular how the various links move with respect to one another and with time.

Typical robots are *serial-link manipulators* comprising a set of bodies, called *links*, in a chain, connected by *joints*[1]. Each joint has one degree of freedom, either translational or rotational. For a manipulator with n joints numbered from 1 to n, there are

[1] Parallel link and serial/parallel hybrid structures are possible, though much less common in industrial manipulators. They will not be discussed in this book.



(a) Standard form

(b) Modified form

Figure 2.1: Different forms of Denavit-Hartenberg notation.

$n+1$ links, numbered from 0 to n. Link 0 is the base of the manipulator, generally fixed, and link n carries the end-effector. Joint i connects links i and $i-1$.

A link may be considered as a rigid body defining the relationship between two neighbouring joint axes. A link can be specified by two numbers, the *link length* and *link twist*, which define the relative location of the two axes in space. The link parameters for the first and last links are meaningless, but are arbitrarily chosen to be 0. Joints may be described by two parameters. The *link offset* is the distance from one link to the next along the axis of the joint. The *joint angle* is the rotation of one link with respect to the next about the joint axis.

To facilitate describing the location of each link we affix a coordinate frame to it — frame i is attached to link i. Denavit and Hartenberg [109] proposed a matrix

method of systematically assigning coordinate systems to each link of an articulated chain. The axis of revolute joint i is aligned with z_{i-1}. The x_{i-1} axis is directed along the common normal from z_{i-1} to z_i and for intersecting axes is parallel to $z_{i-1} \times z_i$. The link and joint parameters may be summarized as:

link length	a_i	the offset distance between the z_{i-1} and z_i axes along the x_i axis;
link twist	α_i	the angle from the z_{i-1} axis to the z_i axis about the x_i axis;
link offset	d_i	the distance from the origin of frame $i-1$ to the x_i axis along the z_{i-1} axis;
joint angle	θ_i	the angle between the x_{i-1} and x_i axes about the z_{i-1} axis.

For a revolute joint θ_i is the joint variable and d_i is constant, while for a prismatic joint d_i is variable, and θ_i is constant. In many of the formulations that follow we use generalized coordinates, q_i, where

$$q_i = \begin{cases} \theta_i & \text{for a revolute joint} \\ d_i & \text{for a prismatic joint} \end{cases}$$

and generalized forces

$$Q_i = \begin{cases} \tau_i & \text{for a revolute joint} \\ f_i & \text{for a prismatic joint} \end{cases}$$

The Denavit-Hartenberg (DH) representation results in a 4x4 homogeneous transformation matrix

$$ {}^{i-1}\mathbf{A}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

representing each link's coordinate frame with respect to the previous link's coordinate system; that is

$$ {}^0\mathbf{T}_i = {}^0\mathbf{T}_{i-1}\, {}^{i-1}\mathbf{A}_i \tag{2.2}$$

where ${}^0\mathbf{T}_i$ is the homogeneous transformation describing the pose of coordinate frame i with respect to the world coordinate system 0.

Two differing methodologies have been established for assigning coordinate frames, each of which allows some freedom in the actual coordinate frame attachment:

1. Frame i has its origin along the axis of joint $i+1$, as described by Paul [199] and Lee [96, 166].

2. Frame i has its origin along the axis of joint i, and is frequently referred to as 'modified Denavit-Hartenberg' (MDH) form [69]. This form is commonly used in literature dealing with manipulator dynamics. The link transform matrix for this form differs from (2.1).

Figure 2.1 shows the notational differences between the two forms. Note that a_i is always the length of link i, but is the displacement between the origins of frame i and frame $i+1$ in one convention, and frame $i-1$ and frame i in the other[2]. This book will consistently use the standard Denavit and Hartenberg methodology[3].

2.1.1 Forward and inverse kinematics

For an n-axis rigid-link manipulator, the *forward kinematic solution* gives the coordinate frame, or pose, of the last link. It is obtained by repeated application of (2.2)

$$
\begin{aligned}
{}^0\mathbf{T}_n &= {}^0\mathbf{A}_1\,{}^1\mathbf{A}_2 \cdots {}^{n-1}\mathbf{A}_n && (2.3)\\
&= \mathrm{K}\left(\underline{q}\right) && (2.4)
\end{aligned}
$$

which is the product of the coordinate frame transform matrices for each link. The pose of the end-effector has 6 degrees of freedom in Cartesian space, 3 in translation and 3 in rotation, so robot manipulators commonly have 6 joints or degrees of freedom to allow arbitrary end-effector pose. The overall manipulator transform ${}^0\mathbf{T}_n$ is frequently written as $\mathbf{T}_n$, or $\mathbf{T}_6$ for a 6-axis robot. The forward kinematic solution may be computed for any manipulator, irrespective of the number of joints or kinematic structure.

Of more use in manipulator path planning is the *inverse kinematic solution*

$$\underline{q} = \mathrm{K}^{-1}(\mathbf{T}) \tag{2.5}$$

which gives the joint coordinates required to reach the specified end-effector position. In general this solution is non-unique, and for some classes of manipulator no closed-form solution exists. If the manipulator has more than 6 joints it is said to be *redundant* and the solution for joint coordinates is under-determined. If no solution can be determined for a particular manipulator pose that configuration is said to be *singular*. The singularity may be due to an alignment of axes reducing the effective degrees of freedom, or the point $\mathbf{T}$ being out of reach.

The manipulator Jacobian matrix, $\mathbf{J}_\theta$, transforms velocities in joint space to velocities of the end-effector in Cartesian space. For an n-axis manipulator the end-effector

[2] It is customary when tabulating the 'modified' kinematic parameters of manipulators to list a_{i-1} and α_{i-1} rather than a_i and α_i.

[3] It may be argued that the MDH convention is more 'logical', but for historical reasons this work uses the standard DH convention.

Cartesian velocity is

$$^{0}\dot{\underline{x}}_{n} = {}^{0}\mathbf{J}_{\theta}\dot{\underline{q}} \tag{2.6}$$

$$^{t_n}\dot{\underline{x}}_{n} = {}^{t_n}\mathbf{J}_{\theta}\dot{\underline{q}} \tag{2.7}$$

in base or end-effector coordinates respectively and where $\underline{x}$ is the Cartesian velocity represented by a 6-vector [199]. For a 6-axis manipulator the Jacobian is square and provided it is not singular can be inverted to solve for joint rates in terms of end-effector Cartesian rates. The Jacobian will not be invertible at a kinematic singularity, and in practice will be poorly conditioned in the vicinity of the singularity, resulting in high joint rates. A control scheme based on Cartesian rate control

$$\dot{\underline{q}} = {}^{0}\mathbf{J}_{\theta}^{-1}\,{}^{0}\dot{\underline{x}}_{n} \tag{2.8}$$

was proposed by Whitney [277] and is known as *resolved rate motion control*. For two frames A and B related by ${}^{A}\mathbf{T}_{B} = [\underline{n}\ \underline{o}\ \underline{a}\ \underline{p}]$ the Cartesian velocity in frame A may be transformed to frame B by

$$^{B}\dot{\underline{x}} = {}^{B}\mathbf{J}_{A}\,{}^{A}\dot{\underline{x}} \tag{2.9}$$

where the Jacobian is given by Paul [200] as

$$^{B}\mathbf{J}_{A} = f({}^{A}\mathbf{T}_{B}) = \begin{bmatrix} [\underline{n}\ \underline{o}\ \underline{a}]^{T} & [\underline{p}\times\underline{n}\ \underline{p}\times\underline{o}\ \underline{p}\times\underline{a}]^{T} \\ 0 & [\underline{n}\ \underline{o}\ \underline{a}]^{T} \end{bmatrix} \tag{2.10}$$

2.1.2 Accuracy and repeatability

In industrial manipulators the position of the tool tip is inferred from the measured joint coordinates and assumed kinematic structure of the robot

$$\hat{T}_6 = \hat{\mathrm{K}}\,(\underline{q}_{meas})$$

Errors will be introduced if the assumed kinematic structure differs from that of the actual manipulator, that is, $\hat{\mathrm{K}} \neq \mathrm{K}$. Such errors may be due to manufacturing tolerances in link length or link deformation due to load. Assumptions are also frequently made about parallel or orthogonal axes, that is link twist angles are exactly 0° or exactly ±90°, since this simplifies the link transform matrix (2.1) by introducing elements that are either 0 or 1. In reality, due to tolerances in manufacture, these assumption are not valid and lead to reduced robot accuracy.

Accuracy refers to the error between the measured and commanded pose of the robot. For a robot to move to a commanded position, the inverse kinematics must be solved for the required joint coordinates

$$q_6 = \hat{\mathrm{K}}^{-1}(\mathbf{T})$$

Joint	α	a_i	d_i	θ_{min}	θ_{max}
1	90	0	0	-180	180
2	0	431.8	0	-170	165
3	-90	20.3	125.4	-160	150
4	90	0	431.8	-180	180
5	-90	0	0	-10	100
6	0	0	56.25	-180	180

Table 2.1: Kinematic parameters and joint limits for the Puma 560. All angles in degrees, lengths in mm.

While the servo system may move very accurately to the computed joint coordinates, discrepancies between the kinematic model assumed by the controller and the actual robot can cause significant positioning errors at the tool tip. Accuracy typically varies over the workspace and may be improved by calibration procedures which seek to identify the kinematic parameters for the particular robot.

Repeatability refers to the error with which a robot returns to a previously taught or commanded point. In general repeatability is better than accuracy, and is related to joint servo performance. However to exploit this capability points must be manually taught by 'jogging' the robot, which is time-consuming and takes the robot out of production.

The AdeptOne manipulator for example has a quoted repeatability of 15μm but an accuracy of 76μm. The comparatively low accuracy and difficulty in exploiting repeatability are two of the justifications for visual servoing discussed earlier in Section 1.1.

2.1.3 Manipulator kinematic parameters

As already discussed the kinematic parameters of a robot are important in computing the forward and inverse kinematics of the manipulator. Unfortunately, as shown in Figure 2.1, there are two conventions for describing manipulator kinematics. This book will consistently use the standard Denavit and Hartenberg methodology, and the particular frame assignments for the Puma 560 are as per Paul and Zhang [202]. A schematic of the robot and the axis conventions used is shown in Figure 2.2. For zero joint coordinates the arm is in a right-handed configuration, with the upper arm horizontal along the X-axis and the lower arm vertical. The upright or READY position[4] is defined by $\underline{q} = [0\ 90\ -90\ 0\ 0\ 0]^\circ$. Others such as Lee [166] consider the zero-angle pose as being left-handed.

[4] The Unimation VAL language defines the so called 'READY position' where the arm is fully extended and upright.

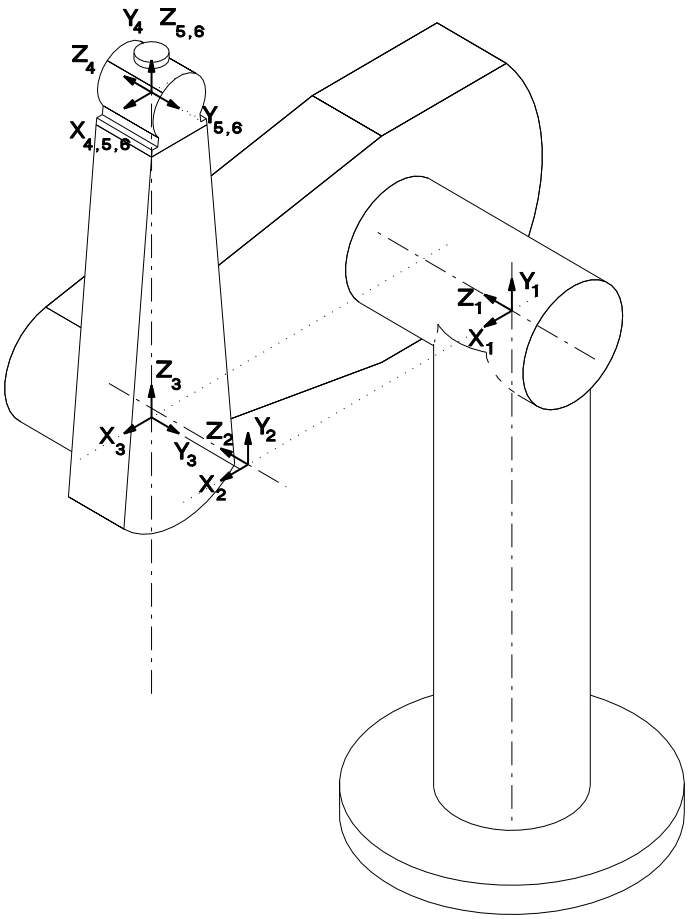


Figure 2.2: Details of coordinate frames used for the Puma 560 shown here in its zero angle pose (drawing by Les Ewbank).

The non-zero link offsets and lengths for the Puma 560, which may be measured directly, are:

- distance between shoulder and elbow axes along the upper arm link, a_2;
- distance from the elbow axis to the center of spherical wrist joint; along the lower arm, d_4;
- offset between the axes of joint 4 and the elbow, a_3;

- offset between the waist and joint 4 axes, d_3.

The kinematic constants for the Puma 560 are given in Table 2.1. These parameters are a consensus [60, 61] derived from several sources [20, 166, 202, 204, 246]. There is some variation in the link lengths and offsets reported by various authors. Comparison of reports is complicated by the variety of different coordinate systems used. Some variations in parameters could conceivably reflect changes to the design or manufacture of the robot with time, while others are taken to be errors. Lee alone gives a value for d_6 which is the distance from wrist center to the surface of the mounting flange.

The kinematic parameters of a robot are important not only for forward and inverse kinematics as already discussed, but are also required in the calculation of manipulator dynamics as discussed in the next section. The kinematic parameters enter the dynamic equations of motion via the link transformation matrices of (2.1).

2.2 Manipulator rigid-body dynamics

Manipulator dynamics is concerned with the equations of motion, the way in which the manipulator moves in response to torques applied by the actuators, or external forces. The history and mathematics of the dynamics of serial-link manipulators are well covered by Paul [199] and Hollerbach [119]. There are two problems related to manipulator dynamics that are important to solve:

- *inverse dynamics* in which the manipulator's equations of motion are solved for given motion to determine the generalized forces, discussed further in Section 2.5, and
- *direct dynamics* in which the equations of motion are integrated to determine the generalized coordinate response to applied generalized forces discussed further in Section 2.2.3.

The equations of motion for an n-axis manipulator are given by

$$\underline{Q} = \mathbf{M}(\underline{q})\ddot{\underline{q}} + \mathbf{C}(\underline{q}, \dot{\underline{q}})\dot{\underline{q}} + \mathbf{F}(\dot{\underline{q}}) + \mathbf{G}(\underline{q}) \tag{2.11}$$

where

$\underline{q}$	is the vector of generalized joint coordinates describing the pose of the manipulator
$\dot{\underline{q}}$	is the vector of joint velocities;
$\ddot{\underline{q}}$	is the vector of joint accelerations
$\mathbf{M}$	is the symmetric joint-space inertia matrix, or manipulator inertia tensor

- **C** describes Coriolis and centripetal effects — Centripetal torques are proportional to $\dot{q}_i^2$, while the Coriolis torques are proportional to $\dot{q}_i\dot{q}_j$
- **F** describes viscous and Coulomb friction and is not generally considered part of the rigid-body dynamics
- **G** is the gravity loading
- $\underline{Q}$ is the vector of generalized forces associated with the generalized coordinates $\underline{q}$.

The equations may be derived via a number of techniques, including Lagrangian (energy based), Newton-Euler, d'Alembert [96, 167] or Kane's [143] method. The earliest reported work was by Uicker [254] and Kahn [140] using the Lagrangian approach. Due to the enormous computational cost, $O(n^4)$, of this approach it was not possible to compute manipulator torque for real-time control. To achieve real-time performance many approaches were suggested, including table lookup [209] and approximation [29,203]. The most common approximation was to ignore the velocity-dependent term **C**, since accurate positioning and high speed motion are exclusive in typical robot applications. Others have used the fact that the coefficients of the dynamic equations do not change rapidly since they are a function of joint angle, and thus may be computed at a fraction of the rate at which the equations are evaluated [149, 201, 228].

Orin et al. [195] proposed an alternative approach based on the Newton-Euler (NE) equations of rigid-body motion applied to each link. Armstrong [23] then showed how recursion might be applied resulting in $O(n)$ complexity. Luh et al. [177] provided a recursive formulation of the Newton-Euler equations with linear and angular velocities referred to link coordinate frames. They suggested a time improvement from 7.9s for the Lagrangian formulation to 4.5 ms, and thus it became practical to implement 'on-line'. Hollerbach [120] showed how recursion could be applied to the Lagrangian form, and reduced the computation to within a factor of 3 of the recursive NE. Silver [234] showed the equivalence of the recursive Lagrangian and Newton-Euler forms, and that the difference in efficiency is due to the representation of angular velocity.

"Kane's equations" [143] provide another methodology for deriving the equations of motion for a specific manipulator. A number of 'Z' variables are introduced which, while not necessarily of physical significance, lead to a dynamics formulation with low computational burden. Wampler [267] discusses the computational costs of Kane's method in some detail.

The NE and Lagrange forms can be written generally in terms of the Denavit-Hartenberg parameters — however the specific formulations, such as Kane's, can have lower computational cost for the specific manipulator. Whilst the recursive forms are computationally more efficient, the non-recursive forms compute the individual dynamic terms (**M**, **C** and **G**) directly.

Method	Multiplications	Additions	For N=6	
			Mul	Add
Lagrangian [120]	$32\frac{1}{2}n^4+86\frac{5}{12}n^3$ $+171\frac{1}{4}n^2+53\frac{1}{3}n$ -128	$25n^4+66\frac{1}{3}n^3$ $+129\frac{1}{2}n^2+42\frac{1}{3}n$ -96	66,271	51,548
Recursive NE [120]	$150n-48$	$131n-48$	852	738
Kane [143]			646	394
Simplified RNE [189]			224	174

Table 2.2: Comparison of computational costs for inverse dynamics from various sources. The last entry is achieved by symbolic simplification using the software package ARM.

A comparison of computation costs is given in Table 2.2. There are considerable discrepancies between sources [96, 120, 143, 166, 265] on the computational burdens of these different approaches. Conceivable sources of discrepancy include whether or not computation of link transform matrices is included, and whether the result is general, or specific to a particular manipulator.

2.2.1 Recursive Newton-Euler formulation

The recursive Newton-Euler (RNE) formulation [177] computes the inverse manipulator dynamics, that is, the joint torques required for a given set of joint coordinates, velocities and accelerations. The forward recursion propagates kinematic information — such as angular velocities, angular accelerations, linear accelerations — from the base reference frame (inertial frame) to the end-effector. The backward recursion propagates the forces and moments exerted on each link from the end-effector of the manipulator to the base reference frame[5]. Figure 2.3 shows the variables involved in the computation for one link.

The notation of Hollerbach [120] and Walker and Orin [265] will be used in which the left superscript indicates the reference coordinate frame for the variable. The notation of Luh et al. [177] and later Lee [96, 166] is considerably less clear.

[5] It should be noted that using MDH notation with its different axis assignment conventions the Newton Euler formulation is expressed differently [69].

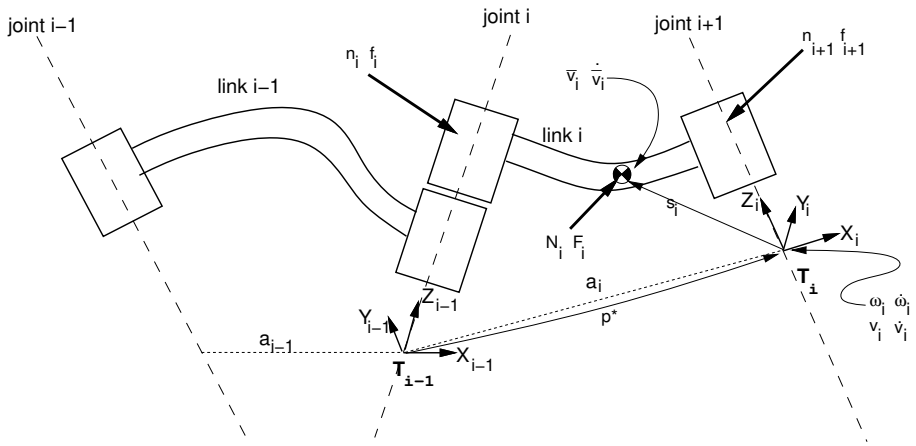


Figure 2.3: Notation used for inverse dynamics, based on standard Denavit-Hartenberg notation.

Outward recursion, $1 \leq i \leq n$**.**

If axis $i+1$ is rotational

$$
\begin{aligned}
{}^{i+1}\underline{\omega}_{i+1} &= {}^{i+1}\mathbf{R}_i \left({}^{i}\underline{\omega}_i + \underline{z}_0 \dot{q}_{i+1} \right) && (2.12)\\
{}^{i+1}\dot{\underline{\omega}}_{i+1} &= {}^{i+1}\mathbf{R}_i \left\{ {}^{i}\dot{\underline{\omega}}_i + \underline{z}_0 \ddot{q}_{i+1} + {}^{i}\underline{\omega}_i \times \left(\underline{z}_0 \dot{q}_{i+1} \right) \right\} && (2.13)\\
{}^{i+1}\underline{v}_{i+1} &= {}^{i+1}\underline{\omega}_{i+1} \times {}^{i+1}\underline{p}^*_{i+1} + {}^{i+1}\mathbf{R}_i \, {}^{i}\underline{v}_i && (2.14)\\
{}^{i+1}\dot{\underline{v}}_{i+1} &= {}^{i+1}\dot{\underline{\omega}}_{i+1} \times {}^{i+1}\underline{p}^*_{i+1} + {}^{i+1}\underline{\omega}_{i+1} \times \left\{ {}^{i+1}\underline{\omega}_{i+1} \times {}^{i+1}\underline{p}^*_{i+1} \right\} \\
&\quad + {}^{i+1}\mathbf{R}_i \, {}^{i}\dot{\underline{v}}_i && (2.15)
\end{aligned}
$$

If axis $i+1$ is translational

$$
\begin{aligned}
{}^{i+1}\underline{\omega}_{i+1} &= {}^{i+1}\mathbf{R}_i\,{}^{i}\underline{\omega}_i && (2.16)\\
{}^{i+1}\dot{\underline{\omega}}_{i+1} &= {}^{i+1}\mathbf{R}_i\,{}^{i}\dot{\underline{\omega}}_i && (2.17)\\
{}^{i+1}\underline{v}_{i+1} &= {}^{i+1}\mathbf{R}_i\left(\underline{z}_0\dot{q}_{i+1}+{}^{i}\underline{v}_i\right)+{}^{i+1}\underline{\omega}_{i+1}\times{}^{i+1}\underline{p}^*_{i+1} && (2.18)\\
{}^{i+1}\dot{\underline{v}}_{i+1} &= {}^{i+1}\mathbf{R}_i\left(\underline{z}_0\ddot{q}_{i+1}+{}^{i}\dot{\underline{v}}_i\right)+{}^{i+1}\dot{\underline{\omega}}_{i+1}\times{}^{i+1}\underline{p}^*_{i+1}\\
&\quad +2\,{}^{i+1}\underline{\omega}_{i+1}\times\left({}^{i+1}\mathbf{R}_i\underline{z}_0\dot{q}_{i+1}\right)\\
&\quad +{}^{i+1}\underline{\omega}_{i+1}\times\left({}^{i+1}\underline{\omega}_{i+1}\times{}^{i+1}\underline{p}^*_{i+1}\right) && (2.19)
\end{aligned}
$$

$$
\begin{aligned}
{}^{i}\dot{\underline{v}}_i &= {}^{i}\dot{\underline{\omega}}_i\times\underline{s}_i+{}^{i}\underline{\omega}_i\times\left\{{}^{i}\underline{\omega}_i\times\underline{s}_i\right\}+{}^{i}\dot{\underline{v}}_i && (2.20)\\
{}^{i}\underline{F}_i &= m_i\,{}^{i}\dot{\underline{v}}_i && (2.21)\\
{}^{i}\underline{N}_i &= \mathbf{J}_i\,{}^{i}\dot{\underline{\omega}}_i+{}^{i}\underline{\omega}_i\times\left(\mathbf{J}_i\,{}^{i}\underline{\omega}_i\right) && (2.22)
\end{aligned}
$$

Inward recursion, $n\geq i\geq 1$**.**

$$
\begin{aligned}
{}^{i}\underline{f}_i &= {}^{i}\mathbf{R}_{i+1}\,{}^{i+1}\underline{f}_{i+1}+{}^{i}\underline{F}_i && (2.23)\\
{}^{i}\underline{n}_i &= {}^{i}\mathbf{R}_{i+1}\left\{{}^{i+1}\underline{n}_{i+1}+\left({}^{i+1}\mathbf{R}_i\,{}^{i}\underline{p}^*_i\right)\times{}^{i+1}\underline{f}_{i+1}\right\}\\
&\quad +\left({}^{i}\underline{p}^*_i+\underline{s}_i\right)\times{}^{i}\underline{F}_i+{}^{i}\underline{N}_i && (2.24)\\
\underline{Q}_i &= \begin{cases}\left({}^{i}\underline{n}_i\right)^T\left({}^{i}\mathbf{R}_{i+1}\underline{z}_0\right) & \text{if link } i+1 \text{ is rotational}\\ \left({}^{i}\underline{f}_i\right)^T\left({}^{i}\mathbf{R}_{i+1}\underline{z}_0\right) & \text{if link } i+1 \text{ is translational}\end{cases} && (2.25)
\end{aligned}
$$

where

i	is the link index, in the range 1 to n
$\mathbf{J}_i$	is the moment of inertia of link i about its COM
$\underline{s}_i$	is the position vector of the COM of link i with respect to frame i
$\underline{\omega}_i$	is the angular velocity of link i
$\dot{\underline{\omega}}_i$	is the angular acceleration of link i
$\underline{v}_i$	is the linear velocity of frame i
$\dot{\underline{v}}_i$	is the linear acceleration of frame i
$\overline{\underline{v}}_i$	is the linear velocity of the COM of link i
$\dot{\overline{\underline{v}}}_i$	is the linear acceleration of the COM of link i
$\underline{n}_i$	is the moment exerted on link i by link $i-1$

$\underline{f}_i$ is the force exerted on link i by link $i-1$

$\underline{N}_i$ is the total moment at the COM of link i

$\underline{F}_i$ is the total force at the COM of link i

$\underline{Q}_i$ is the force or torque exerted by the actuator at joint i

${}^{i-1}\mathbf{R}_i$ is the orthonormal rotation matrix defining frame i orientation with respect to frame $i-1$. It is the upper 3×3 portion of the link transform matrix given in (2.1).

$$ {}^{i-1}\mathbf{R}_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i \end{bmatrix} \tag{2.26} $$

$$ {}^{i}\mathbf{R}_{i-1} = ({}^{i-1}\mathbf{R}_i)^{-1} = ({}^{i-1}\mathbf{R}_i)^T \tag{2.27} $$

${}^{i}\underline{p}_i^*$ is the displacement from the origin of frame $i-1$ to frame i with respect to frame i.

$$ {}^{i}\underline{p}_i^* = \begin{bmatrix} a_i \\ d_i \sin\alpha_i \\ d_i \cos\alpha_i \end{bmatrix} \tag{2.28} $$

It is the negative translational part of $({}^{i-1}\mathbf{A}_i)^{-1}$.

$\underline{z}_0$ is a unit vector in Z direction, $\underline{z}_0 = [0\ 0\ 1]$

Note that the COM linear velocity given by equation (2.14) or (2.18) does not need to be computed since no other expression depends upon it. Boundary conditions are used to introduce the effect of gravity by setting the acceleration of the base link

$$ \dot{v}_0 = -\underline{g} \tag{2.29} $$

where $\underline{g}$ is the gravity vector in the reference coordinate frame, generally acting in the negative Z direction, downward. Base velocity is generally zero

$$ v_0 = 0 \tag{2.30} $$

$$ \omega_0 = 0 \tag{2.31} $$

$$ \dot{\omega}_0 = 0 \tag{2.32} $$

2.2.2 Symbolic manipulation

The RNE algorithm is straightforward to program and efficient to execute in the general case, but considerable savings can be made for the specific manipulator case. The general form inevitably involves many additions with zero and multiplications with 0,

1 or -1, in the various matrix and vector operations. The zeros and ones are due to the trigonometric terms[6] in the orthonormal link transform matrices (2.1) as well as zero-valued kinematic and inertial parameters. Symbolic simplification gathers common factors and eliminates operations with zero, reducing the run-time computational load, at the expense of a once-only off-line symbolic computation. Symbolic manipulation can also be used to gain insight into the effects and magnitudes of various components of the equations of motion.

Early work in symbolic manipulation for manipulators was performed with special tools generally written in Fortran or LISP such as ARM [189], DYMIR [46] and EMDEG [40]. Later development of general purpose computer algebra tools such as Macsyma, REDUCE and MAPLE has made this capability more widely available.

In this work a general purpose symbolic algebra package, MAPLE [47], has been used to compute the torque expressions in symbolic form via a straightforward implementation of the RNE algorithm. Compared to symbolic computation using the Lagrangian approach, computation of the torque expressions is very efficient. These expressions are in sum of product form, and can be extremely long. For example the expression for the torque on the first axis of a Puma 560 is

$$\begin{aligned}\tau_1 = & -C_{23} I_{yz3} \ddot{q}_2 \\ & + {s_{x1}}^2 m_1 \ddot{q}_1 \\ & + S_{23} I_{yz3} \dot{q}_3^2 \\ & + I_{yy2} {C_2}^2 \ddot{q}_1 \\ & + C_2 I_{yz2} \ddot{q}_2 \\ & + \cdots\end{aligned}$$

and continues on for over 16,000 terms. Such expressions are of little value for on-line control, but are appropriate for further symbolic manipulation to analyze and interpret the significance of various terms. For example the symbolic elements of the **M**, **C** and **G** terms can be readily extracted from the sum of product form, overcoming what is frequently cited as an advantage of the Lagrangian formulation — that the individual terms are computed directly.

Evaluating symbolic expressions in this simple-minded way results in a loss of the factorization inherent in the RNE procedure. However with appropriate factorization during symbolic expansion, a computationally efficient form for run-time torque computation can be generated, see Section 2.6.2. MAPLE can then produce the 'C' language code corresponding to the torque expressions, for example, automatically generating code for computed-torque control of a specific manipulator. MAPLE is also capable of generating LaTeX style equations for inclusion in documents.

[6] Common manipulators have link twists of 0°, 90° or -90° leading to zero or unity trigonometric results.

As discussed previously, the dynamic equations are extremely complex, and thus difficult to verify, but a number of checks have been made. The equations of motion of a simple two-link example in Fu et al. [96] was computed and agreed with the results given. For the Puma 560 this is prohibitive, but some simple checks can still be performed. The gravity term is readily extracted and is simple enough to verify manually. The manipulator inertia matrix is positive definite and its symmetry can be verified symbolically. A colleague [213] independently implemented the dynamic equations in MAPLE, using the Lagrangian approach. MAPLE was then used to compute the difference between the two sets of torque expressions, which after simplification was found to be zero.

2.2.3 Forward dynamics

Equation (2.11) may be used to compute the so-called inverse dynamics, that is, actuator torque as a function of manipulator state and is useful for on-line control. For simulation the direct, integral or *forward dynamic* formulation is required giving joint motion in terms of input torques.

Walker and Orin [265] describe several methods for computing the forward dynamics, and all make use of an existing inverse dynamics solution. Using the RNE algorithm for inverse dynamics, the computational complexity of the forward dynamics using 'Method 1' is $O(n^3)$ for an n-axis manipulator. Their other methods are increasingly more sophisticated but reduce the computational cost, though still $O(n^3)$. Featherstone [89] has described the 'articulated-body method' for $O(n)$ computation of forward dynamics, however for $n < 9$ it is more expensive than the approach of Walker and Orin. Another $O(n)$ approach for forward dynamics has been described by Lathrop [160].

2.2.4 Rigid-body inertial parameters

Accurate model-based dynamic control of a manipulator requires knowledge of the rigid-body inertial parameters. Each link has ten independent inertial parameters:

- link mass, m_i;
- three first moments, which may be expressed as the COM location, $\underline{s}_i$, with respect to some datum on the link or as a moment $\underline{S}_i = m_i \underline{s}_i$;
- six second moments, which represent the inertia of the link about a given axis, typically through the COM. The second moments may be expressed in matrix or tensor form as

$$\mathbf{J} = \begin{bmatrix} J_{xx} & J_{xy} & J_{xz} \\ J_{xy} & J_{yy} & J_{yz} \\ J_{xz} & J_{yz} & J_{zz} \end{bmatrix} \tag{2.33}$$

Parameter	Value
m_1	13.0
m_2	17.4
m_3	4.80
m_4	0.82
m_5	0.35
m_6	0.09

Table 2.3: Link mass data (kg).

where the diagonal elements are the *moments of inertia*, and the off-diagonals are *products of inertia*. Only six of these nine elements are unique: three moments and three products of inertia.

For any point in a rigid-body there is one set of axes known as the *principal axes of inertia* for which the off-diagonal terms, or products, are zero. These axes are given by the eigenvectors of the inertia matrix (2.33) and the eigenvalues are the principal moments of inertia. Frequently the products of inertia of the robot links are zero due to symmetry.

A 6-axis manipulator rigid-body dynamic model thus entails 60 inertial parameters. There may be additional parameters per joint due to friction and motor armature inertia. Clearly, establishing numeric values for this number of parameters is a difficult task. Many parameters cannot be measured without dismantling the robot and performing careful experiments, though this approach was used by Armstrong et al. [20]. Most parameters could be derived from CAD models of the robots, but this information is often considered proprietary and not made available to researchers. The robot used in this work, the Puma 560, was designed in the late 1970's and probably predates widespread CAD usage. There is also a considerable literature regarding estimation of inertial parameters from online measurement of manipulator state and joint torques [130].

Tarn and Bejczy [245, 247], Armstrong [22] and Leahy [161, 165, 257] have all reported successful model-based control of the Puma 560, yet there is significant difference in the parameter sets used. This may in fact indicate that the rigid-body effects do not dominate the dynamics of this robot, or that "some feedforward is better than no feedforward". This issue will be revisited in Section 2.4. Comparisons of the published model data are complicated by the different coordinate frames, kinematic conventions and measurement units used in the original reports. The first step is to convert all reported data to a common set of units and coordinate frames, and these data are reported and compared in [60, 61]. Some data sets are to be preferred to others due to the methodologies used to obtain them. The remainder of this section comprises an abbreviated report of that comparison work and tabulates the preferred

Parameter	Value
s_{x_1}	-
s_{y_1}	-
s_{z_1}	-
s_{x_2}	-363.8
s_{y_2}	6
s_{z_2}	77.4
s_{x_3}	0
s_{y_3}	-14
s_{z_3}	70
s_{x_4}	0
s_{y_4}	-19
s_{z_4}	0
s_{x_5}	0
s_{y_5}	0
s_{z_5}	0
s_{x_6}	0
s_{y_6}	0
s_{z_6}	32

Table 2.4: Link COM position with respect to link frame (mm).

inertial parameter values.

Link mass data for joints 2-6 given in Table 2.3 are based on the results of Armstrong et al. [20] who actually disassembled the robot and weighed the links. The often cited data of Tarn et al. [246] is based on estimated mass from models, dimensional data and assumed mass distribution and densities, which is likely to be less accurate. A similar approach appears to have been taken by Paul et al. [204]. Armstrong however does not give a value for m_1, so Tarn's value is presented in the table. It can be shown however that the parameter m_1 does not appear in the equations of motion — it is not a *base parameter* [148, 153].

Link center of gravity data given in Table 2.4 is again based on Armstrong et al. who measured the COM of the disassembled links on a knife-edge. Tarn et al.'s data is again an estimate based on models of the arm structure.

It is difficult to meaningfully compare these data sets, and contrast them with those for the robot used in this work. The approach proposed here is to compare the gravity loading terms for joints 2 and 3 — those links for which gravity load is significant. A small number of gravity load coefficients encapsulate a larger number of mass and COM parameters. The gravity loadings are readily generated from the symbolic torque

Parameter	Armstrong	Tarn	RCCL
g_1	-0.896	-0.977	-0.928 (CP30/g)
g_2	0.116	0.235	0.0254 (CP21/g)
g_3	0	-0.00980	
g_4	0	0	
g_5	-2.88e-3	0.34e-3	-2.88e-3 (CP50/g)
g_6	0	0	
g_7	-0.104	-0.112	0.104 (CP22/g)
g_8	3.80	5.32	-3.79 (CP20/g)

Table 2.5: Comparison of gravity coefficients (N.m) from several sources.

equations established earlier, and are given by

$$\begin{aligned}
\frac{\tau_{g3}}{g} &= -\left((m6+m5+m4)\,D4+sz3\,m3+m4\,sy4\right)S23 \\
&\quad +\left((m3+m4+m5+m6)\,A3+m3\,sx3\right)C23 \\
&\quad +\left(sz4\,m4-sy5\,m5-sy6\,m6\,C6\right)C23\,S4 \\
&\quad +S23\,S5\,sy6\,m6\,S6 \\
&\quad -\left(sz6\,m6+sz5\,m5\right)\left(C23\,S5\,C4+S23\,C5\right) \\
&\quad -S6\,sy6\,m6\,C23\,C4\,C5 \\
\frac{\tau_{g2}}{g} &= \left(m2\,sx2+\left(m2+m3+m4+m5+m6\right)A2\right)C2 \\
&\quad -sy2\,m2\,S2+\frac{\tau_{g3}}{g}
\end{aligned} \tag{2.34}$$

where A_i and D_i are kinematic constants, $C_i \equiv \cos\theta_i$ and $S_i \equiv \sin\theta_i$. These may be written more succinctly in terms of a number of coefficients which are functions of link mass, center of gravity and some kinematic length parameters;

$$\begin{aligned}
\frac{\tau_{g3}}{g} &= g_1 S23+g_2 C23+g_3 C23\,S4+g_4 S23\,S5\,S6 \\
&\quad +g_5\left(S5\,C4\,C23+S23\,C5\right)+g_6 C5\,C4\,C23\,S6
\end{aligned} \tag{2.35}$$

$$\frac{\tau_{g2}}{g} = g_7 S2+g_8 C2+\frac{\tau_{g3}}{g} \tag{2.36}$$

These coefficients are evaluated and compared in Table 2.5 along with those used by the RCCL robot control package [115, 175]. There is close agreement between the magnitudes of the coefficients from Armstrong and those used in RCCL. Different kinematic conventions used by RCCL, or the sign of the gear ratios, may explain the difference in sign for the joint 2 coefficients g_7 and g_8. The RCCL values were

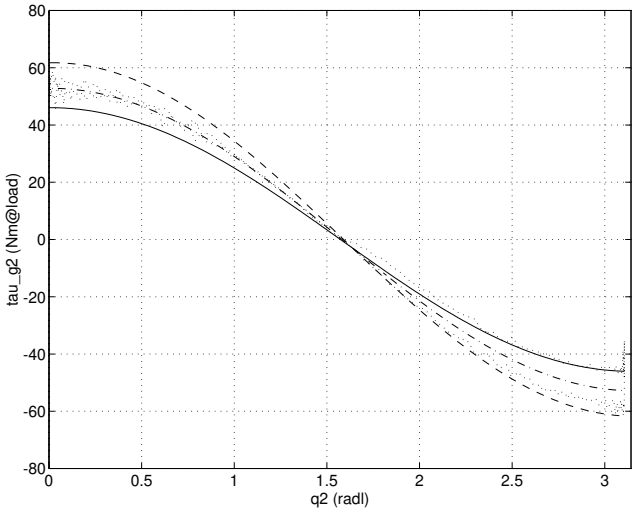


Figure 2.4: Measured and estimated gravity load on joint 2, $\tau_{g_2}(\theta_2)$, for $\theta_3 = -\pi/2$. Torque measurements (shown dotted) are derived from measured motor current and corrected to eliminate the effect of Coulomb friction. Also shown is the estimated gravity load based on maximum likelihood fit (dot dash), parameter values of Armstrong (solid) and Tarn (dashed).

determined using an experimental procedure similar to that described by Lloyd [175] for a Puma 260[7].

Figure 2.4 shows the joint 2 torque due to gravity, versus changing shoulder joint angle. The shoulder joint was moved forward and backward over the angular range at very low speed to eliminate any torque component due to viscous friction. Joint torque is derived from measured motor current using motor torque constants from Table 2.14. The Coulomb friction effect is very pronounced, and introduces significant hysteresis in the torque versus angle plot. The torque in Figure 2.4 has been corrected for Coulomb friction using the identified friction parameters from Table 2.12, but some hysteresis remains at $q_2 \approx \pi$. It is speculated that this is due to position-dependent Coulomb friction effects outside the range of joint angles over which the friction estimation experiments were conducted.

A maximum likelihood fit to the experimental data is also shown in the figure. It can be seen that the estimated torque using Armstrong's data is slightly lower than

[7] John Lloyd, private communication.

Parameter	β (N.m)	ϕ (rad)
Armstrong	46.1	2.6e-3
Tarn	61.8	19.5e-3
Max.Likelihood	52.9	-10.8e-3

Table 2.6: Comparison of shoulder gravity load models in cosine form.

Parameter	Value
J_{xx_1}	-
J'_{yy_1}	0.35†
J_{zz_1}	-
J_{xx_2}	0.130
J_{yy_2}	0.524
J_{zz_2}	0.539
J_{xx_3}	0.066
J_{yy_3}	0.0125
J_{zz_3}	0.086
J_{xx_4}	1.8e-3
J_{yy_4}	1.8e-3
J_{zz_4}	1.3e-3
J_{xx_5}	0.30e-3
J_{yy_5}	0.30e-3
J_{zz_5}	0.40e-3
J_{xx_6}	0.15e-3
J_{yy_6}	0.15e-3
J_{zz_6}	0.04e-3

Table 2.7: Link inertia about the COM (kg.m^2). †-This value, due to Armstrong, is in fact the inertia about the link frame $J_{yy_1} + m_1(s_{x_1}^2 + s_{z_1}^2)$ not about the COM.

that measured, while that based on Tarn's data is somewhat higher. The gravity load may be written in the form

$$\tau_{g_2} = \beta cos(\theta_2 + \phi) \tag{2.37}$$

where β is the magnitude and ϕ the phase. The coefficients for the various forms are compared in Table 2.6, and in terms of magnitude the maximum likelihood fit is bracketed by the models of Tarn and Armstrong, as is also evident from Figure 2.4. Despite the previous objections to the methodology of Tarn et al. their data gives a fit for the gravity load of joint 2 that is as good as that of Armstrong.

Link inertia about the center of gravity is given in Table 2.7, based largely on Armstrong. Armstrong's data for links 1 to 3 was determined experimentally, while that for

the wrist links was estimated. However Armstrong's value of J_{yy1}[8] is in fact the inertia measured about the link frame, $J'_{yy_1} = J_{yy_1} + m_1(s_{x_1}^2 + s_{z_1}^2)$, not the COM as indicated in [20], since the two inertial components cannot be separated by measurements at the link[9].

2.2.5 Transmission and gearing

For a G:1 reduction drive the torque at the link is G times the torque at the motor. The inertia of the motor at the link is amplified by G^2, as is the viscous friction coefficient. For rotary joints the quantities measured at the link, subscript l, are related to the motor referenced quantities, subscript m, as shown in Table 2.8. The manipulator gear ratios[10] given in Table 2.9 are derived from several sources [20, 190, 262].

The design of the robot's wrist is such that there is considerable coupling between the axes, that is, rotation of one motor results in the rotation of several wrist links. The relationship between link and motor angles is clearly expressed in matrix form

$$\underline{\theta}_m = \mathbf{G}\underline{\theta}_l, \quad \underline{\theta}_l = \mathbf{G}^{-1}\underline{\theta}_m \tag{2.38}$$

where

$$\mathbf{G} = \begin{bmatrix} G_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & G_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & G_4 & 0 & 0 \\ 0 & 0 & 0 & -G_{45}G_5 & G_5 & 0 \\ 0 & 0 & 0 & -G_{46}G_6 & -G_{56}G_6 & G_6 \end{bmatrix} \tag{2.39}$$

$$\mathbf{G}^{-1} = \begin{bmatrix} \frac{1}{G_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{G_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{G_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_4} & 0 & 0 \\ 0 & 0 & 0 & \frac{G_{45}}{G_4} & \frac{1}{G_5} & 0 \\ 0 & 0 & 0 & \frac{G_{46}+G_{56}G_{45}}{G_4} & \frac{G_{56}}{G_5} & \frac{1}{G_6} \end{bmatrix} \tag{2.40}$$

The cross-coupling ratios have been determined from examination of the engineering drawings and correspond with the numerical values given by [262]. These have also been verified experimentally.

[8] J_{zz1} in the source paper, due to different coordinate conventions used.

[9] B. Armstrong, private communication.

[10] The sign of the ratio is due to the convention for direction of rotation of the motor (defined by the digital position-loop encoder counter, see Section 2.3.6), and the convention for direction of link rotation which is defined by the kinematic model. Negative motor current results in positive motor torque.

J_l	$=$	$G^2 J_m$
B_l	$=$	$G^2 B_m$
τ_{C_l}	$=$	$G\tau_{C_m}$
τ_l	$=$	$G\tau_m$
$\dot{\theta}_l$	$=$	$\dot{\theta}_m / G$
$\ddot{\theta}_l$	$=$	$\ddot{\theta}_m / G$

Table 2.8: Relationship between load and motor referenced quantities for gear ratio G.

Joint	Gear ratio
G_1	-62.6111
G_2	107.815
G_3	-53.7063
G_4	76.03636
G_5	71.923
G_6	76.686
G_{45}	$-1/G_5$
G_{46}	$-1/G_6$
G_{56}	$-13/72$

Table 2.9: Puma 560 gear ratios.

2.2.6 Quantifying rigid body effects

The numeric values of the inertial parameters obtained above may be substituted into the equations of motion. With some manipulation this allows the various dynamic effects to be quantified and the bounds due to change in configuration established. For instance Figures 2.5, 2.6 and 2.7 show respectively the inertia at joint 1 and 2 and the gravity load at joint 2, all plotted as functions of manipulator configuration.

There are two components of inertia 'seen' by the motor. One is due to the rotating armature, and the other due to the rigid-body dynamics of the link reflected through the gear system. The total inertia sets the upper bound on acceleration, and also affects the dynamics of the axis control loop. It is insightful to plot total inertia normalized with respect to the armature inertia, J_m, since this clearly shows the inertia compared with the unloaded ($\eta = 1$) case. The normalized inertia is defined to be

$$\eta_{ii}(\underline{q}) = 1 + \frac{\mathbf{M}_{ii}(\underline{q})}{G_i^2 J_{m_i}} \tag{2.41}$$

where $\mathbf{M}$ is the manipulator inertia matrix from (2.11), and J_{m_i} and G_i are the motor armature inertia and the reduction gear ratio respectively for joint i. The normalized

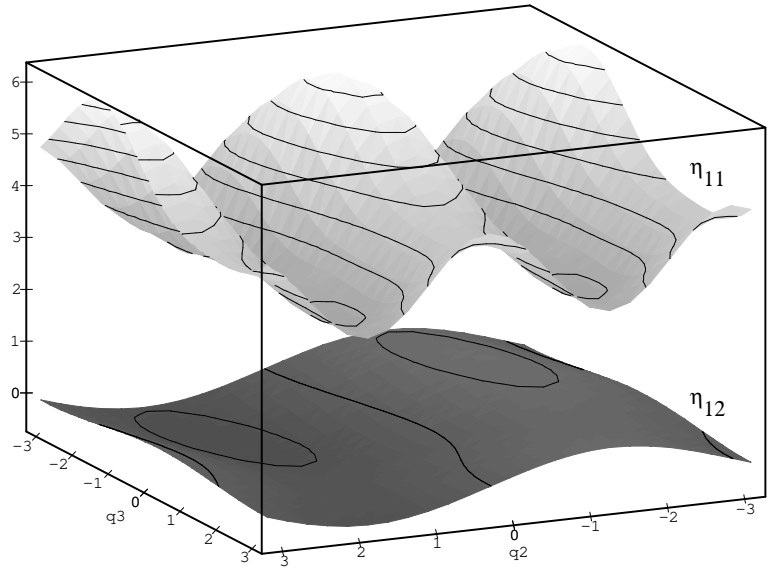


Figure 2.5: Plot of normalized inertia for joint 1 as a function of shoulder and elbow configuration. Shown is the diagonal term η_{11} and the inertial coupling term η_{12}. Angles in rads.

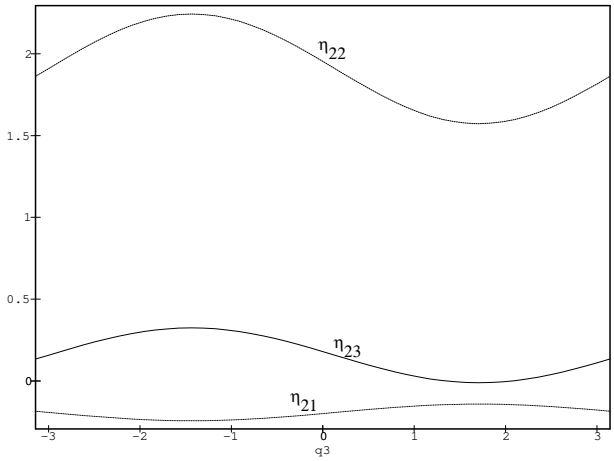


Figure 2.6: Plot of normalized inertia for joint 2 as a function of elbow configuration. Shown is the diagonal term η_{22} and the inertial coupling terms η_{21} and η_{23}. Angles in rads.

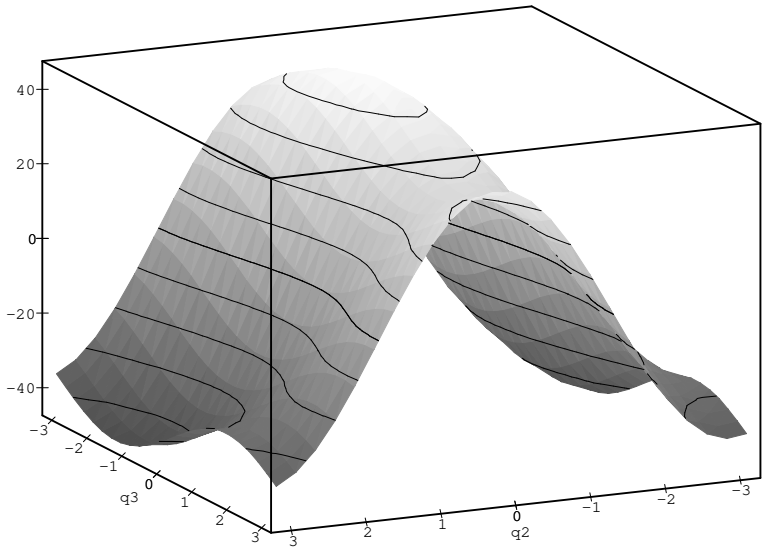


Figure 2.7: Gravity load (N.m) on joint 2, $\tau_{g_2}(q_2, q_3)$. Angles in rads. Gravity load varies between ± 46 N.m or ± 48 % of the fuse limited torque.

inertial coupling is defined to be

$$\eta_{ij}(\underline{q}) = \frac{\mathbf{M}_{ij}(\underline{q})}{G_i^2 J_{m_i}} \tag{2.42}$$

The normalized diagonal and off-diagonal inertia elements for Puma joints 1 and 2 are shown in Figures 2.5 and 2.6 as a function of configuration. The off-diagonal terms are relatively small in magnitude compared to the diagonal values. The rigid-body dynamic equations from Section 2.2 can be used to compute the minima and maxima of the normalized inertias, and these are summarized in Table 2.10. The variation is most pronounced for the waist and shoulder joints. Gravity load, plotted in Figure 2.7 for joint 2, shows that gravity torque is significant compared to the torque limit of the actuator. The relative significance of various dynamic terms is examined further in Section 2.4.

2.2.7 Robot payload

The inertia of the load, in this work a camera, has been computed from mass and dimensional data assuming uniform mass distribution within each of the camera body and lens. The results are tabulated in Table 2.11. The camera inertia when referred to

Quantity	Minimum	Maximum	Max/Min
η_{11}	2.91	6.35	2.2
η_{12}	-0.75	0.75	-
η_{22}	2.57	3.24	1.3
η_{33}	1.63		-
η_{44}	1.01		-
η_{55}	1.01		-
η_{66}	1.00		-

Table 2.10: Minimum and maximum values of normalized inertia, based on preferred armature inertia data from Table 2.13.

Component	Value
m_{lens}	$0.270\,\mathrm{kg}$
m_{cam}	$0.155\,\mathrm{kg}$
I_{zz}	$1.0 \times 10^{-3}\,\mathrm{kg.m^2}$
I_{xx}	$6.2 \times 10^{-3}\,\mathrm{kg.m^2}$
I_1	$0.417\,\mathrm{kg.m^2}$
η_5	0.034
η_6	0.005
η_1	0.532

Table 2.11: Mass and inertia of end-mounted camera. Inertia I_{xx} and I_{zz} are computed with respect to the center of the robot wrist, see Figure 4.11. I_1 is camera inertia with respect to the joint 1 axis at maximum arm extension. η_i is the normalized camera inertia with respect to joint i, that is, $I_{cam}/G_i^2 J_{m_i}$.

the wrist motors and normalized is insignificant. However the inertia contribution to joint 1 when the arm is fully extended, I_1, is significant.

2.3 Electro-mechanical dynamics

This section provides details about the dynamic effects due to the robot's control electronics, actuators and mechanical transmission. These effects are at least as significant as the rigid-body effects just reviewed though they are less well covered in the literature, perhaps due to the robot specific nature of these effects.

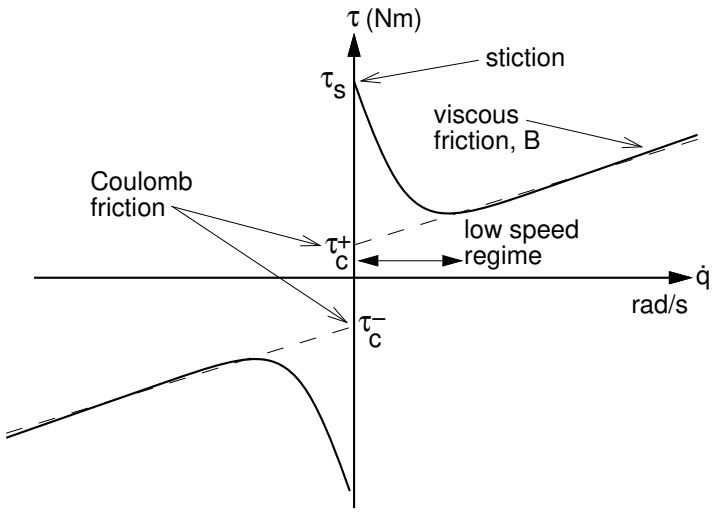


Figure 2.8: Typical friction versus speed characteristic. The dashed lines depict a simple piecewise-linear friction model characterized by slope (viscous friction) and intercept (Coulomb friction).

2.3.1 Friction

Dynamic effects due to the transmission or drive system are significant, particularly as the drive train becomes more complex. The addition of gears to amplify motor torque leads to increased viscous friction and non-linear effects such as backlash and Coulomb friction.

For a geared manipulator, such as the Puma, friction is a dominant dynamic characteristic. A typical friction torque versus speed characteristic is shown in Figure 2.8. The dashed line represents the simple friction model

$$\tau_f = B\dot{\theta} + \tau_c \tag{2.43}$$

where slope represents viscous friction, and offset represents Coulomb friction. The latter is frequently modelled by the non-linear function

$$\tau_c = \begin{cases} 0 & \text{if } \dot{q} = 0 \\ \tau_c^+ & \text{if } \dot{q} > 0 \\ \tau_c^- & \text{if } \dot{q} < 0 \end{cases} \tag{2.44}$$

and in general $|\tau_c^+| \neq |\tau_c^-|$. Static friction, or stiction, is the torque that is necessary to bring a stationary joint into motion, and can be considerably greater than the Coulomb

friction value. The more complex model, represented by the solid line, is significant only for very low velocities [22]. The negative slope can be attributed to the Stribeck effect due to mixed lubrication — the load is partially, but increasingly, lifted by lubricant, resulting in decreasing friction. This negative slope characteristic can result in instability with simple PID joint control schemes at low velocity. When the contact is fully lubricated viscous friction is evident.

The friction parameters discussed represent total lumped friction due to motor brushes, bearings and transmission. They are dependent upon many factors including temperature, state of lubrication, and to a small extent shaft angle. Armstrong [21, 22] provides detailed investigations of the low speed and angular dependence of joint friction for a Puma 560 manipulator.

Classic techniques for determining friction are based on the simple piecewise-linear friction model of (2.43). A joint is moved at constant velocity and the average torque (typically determined from motor current) is measured. This is repeated for a range of velocities, both positive and negative, from which the slope (viscous friction coefficient) and intercepts (Coulomb friction torques) can be determined. Measurement of joint friction characteristics using this approach have been previously reported for a Puma 260 [175] and a Puma 560 [173].

The friction values for the robot used in this work have been determined experimentally and are summarized in Table 2.12. Coulomb friction and viscous friction were determined by measuring average joint current for various joint speeds over a short angular range about the vertical 'READY' position. This was done to eliminate the torque component due to gravity which would otherwise influence the experiment. A typical plot of current versus velocity is shown in Figure 2.9. Given knowledge of the motor torque constant from Table 2.14, viscous and Coulomb friction values may be determined from the slope and intercept respectively. A robot 'work out' program was run prior to the measurements being taken, so as to bring joints and lubricant up to 'typical' working temperature. There is no evidence of the negative slope on the friction curve at the velocities used here. The lowest velocity in each test was 5°/s at the link, which is approximately 5% and 2% of the peak velocities for the base and wrist joints respectively.

From Table 2.12 it is clear that some friction parameters show considerable dependence on the direction of rotation. Statistical analysis of the mean and variance of the sample points [266] for positive and negative velocity for each joint indicate that at the 95% confidence level the friction values are not equal, apart from viscous friction values for joints 1 and 6. Armstrong [22] showed statistically that Coulomb and viscous friction had separate values for positive and negative velocities. For linear system design and simulation the mean viscous friction value will be used.

Stiction, τ_s, was measured by increasing the joint current until joint motion occurred[11]. For those joints subject to gravity load the robot was positioned so as to

[11] Taken as increasing encoder value for 5 consecutive sample intervals.

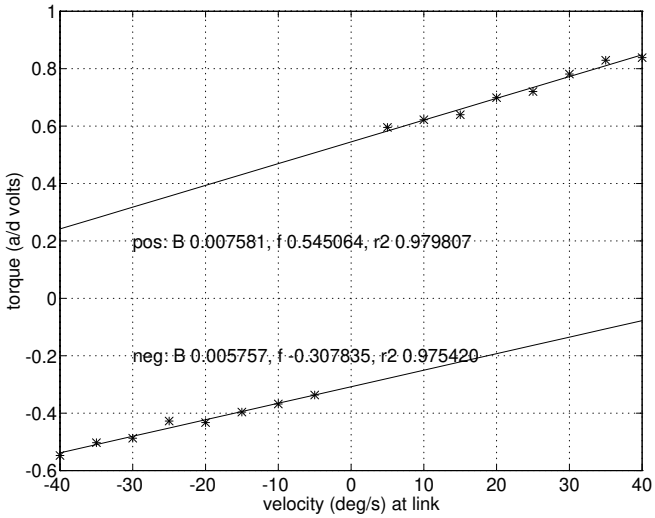


Figure 2.9: Measured motor current (actually motor shunt voltage) versus joint velocity for joint 2. Experimental points and lines of best fit are shown.

Joint	τ_s^+	τ_C^+	B^+	τ_s^-	τ_C^-	B^-	$\overline{B}$
1	0.569	0.435	1.46e-3	-0.588	-0.395	-1.49e-3	1.48e-3
2	0.141	0.126	0.928e-3	-95.1e-3	-70.9e-3	-0.705e-3	0.817e-3
3	0.164	0.105	1.78e-3	-0.158	-0.132	-0.972e-3	1.38e-3
4	14.7e-3	11.2e-3	64.4e-6	-21.8e-3	-16.9e-3	-77.9e-6	71.2e-6
5	5.72e-3	9.26e-3	93.4e-6	-13.1e-3	-14.5e-3	-71.8e-6	82.6e-6
6	5.44e-3	3.96e-3	40.3e-6	-9.21e-3	-10.5e-3	-33.1e-6	36.7e-6

Table 2.12: Measured friction parameters — motor referenced (N.m and N.m.s/rad). Positive and negative joint velocity are indicated by the superscripts. The column $\overline{B}$ is the mean of B^+ and B^-.

eliminate gravity torque. The average stiction over 20 trials was taken. The standard deviation was very high for joint 1, around 16% of the mean, compared to 5% of the mean for the wrist joints.

2.3.2 Motor

The Puma robot uses two different sizes of motor — one for the base axes (joints 1-3) and another for the wrist axes (joints 4-6). Data on these motors is difficult to find, and the motors themselves are unlabelled. There is speculation about the manufacturer and model in [12], and it is strongly rumoured that the motors are 'specials' manufactured by Electrocraft for Unimation. It is conceivable that different types of motor have been used by Unimation over the years. Tarn and Bejczy et al. [245, 247] have published several papers on manipulator control based on the full dynamic model, and cite sources of motor parameter data as Tarn et al. [246] and Goor [102]. The former has no attribution for motor parameter data quoted, while the latter quotes "manufacturer's specifications" for the base motors only. The source of Tarn's data for the wrist motors [247] is not given. Kawasaki manufacture the Puma 560 under licence, and data on the motors used in that machine was provided by the local distributor. Those motors are Tamagawa TN3053N for the base, and TN3052N for the wrist. However some of these parameters appear different to those quoted by Tarn and Goor.

A complete block diagram of the motor system dynamics is shown in Figure 2.10 and assumes a rigid transmission. The motor torque constant, K_m, is a gain that relates motor current to armature torque

$$\tau_m = K_m i_m \tag{2.45}$$

and is followed by a first-order stage representing the armature dynamics

$$\Omega_m = \frac{\tau}{J_{eff}s + B} \tag{2.46}$$

where Ω_m is motor velocity, J_{eff} the effective inertia due to the armature and link, and B the viscous friction due to motor and transmission. The so-called *mechanical pole* is given by

$$p_m = -\frac{B}{J_{eff}} \tag{2.47}$$

Coulomb friction, τ_c, described by (2.44), is a non-linear function of velocity that opposes the armature torque. The friction and inertia parameters are lumped values representing the motor itself and the transmission mechanism. Finally, there is a reduction gear to drive the manipulator link.

An equivalent circuit for the servo motor is given in Figure 2.11. This shows motor impedance comprising resistance, R_m, due to the armature winding and brushes, and inductance, L_m, due to the armature winding. R_s is the shunt resistor which provides the current feedback signal to the current loop. The electrical dynamics are embodied in the relationship for motor terminal voltage

$$V_m = sK_m\Theta + sL_mI_m + (R_s + R_m)I_m + E_c \tag{2.48}$$

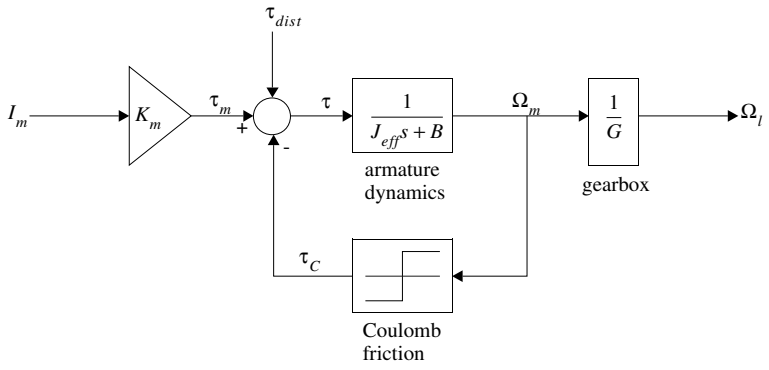


Figure 2.10: Block diagram of motor mechanical dynamics. τ_{dist} represents disturbance torque due to load forces or unmodeled dynamics.

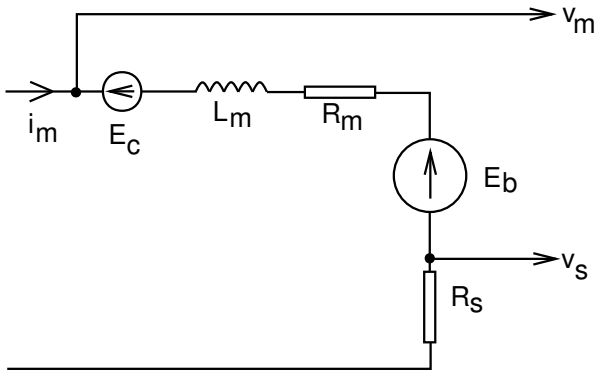


Figure 2.11: Schematic of motor electrical model.

which has components due to back EMF, inductance, resistance and *contact potential difference* respectively. The latter is a small constant voltage drop, typically around 1 to 1.5V [146], which will be ignored here. The so-called *electrical pole* is given by

$$p_e = -\frac{R_m}{L_m} \tag{2.49}$$

2.3.2.1 Inertia

As mentioned earlier there are two components of inertia 'seen' by the motor. One is due to the the rigid-body link dynamics 'reflected' through the gear system, and the

Parameter	Armstrong	Tarn	Kawasaki	Preferred
J_{m1}	291e-6	198-6	200e-6	200e-6
J_{m2}	409e-6	203e-6	200e-6	200e-6
J_{m3}	299e-6	202e-6	200e-6	200e-6
J_{m4}	35e-6	18.3e-6	20e-6	33e-6
J_{m5}	35e-6	18.3e-6	20e-6	33e-6
J_{m6}	33e-6	18.3e-6	20e-6	33e-6

Table 2.13: Comparison of motor inertia values from several sources — motor referenced ($\mathrm{kg.m^2}$).

other due to the rotating armature. The total inertia sets the upper bound on acceleration, and also affects the location of the mechanical pole by (2.47).

Several sources of armature inertia data are compared in Table 2.13. Armstrong's [20] values (load referenced and including transmission inertia) were divided by G_i^2, from Table 2.9, to give the values tabulated. These estimates are based on total inertia measured at the joint with estimated link inertia subtracted, and are subject to greatest error where link inertia is high. From knowledge of motor similarity the value for motor 2 seems anomalous. Values given by Tarn [246] are based on an unknown source of data for armature inertia, but also include an estimate for the inertia of the shafts and gears of the transmission system for the base axes. These inertia contributions are generally less than 2% of the total and could practically be ignored. The very different estimates of armature inertia given in the literature may reflect different models of motor used in the robots concerned. The preferred values for the base axes are based on a consensus of manufacturer values rather than Armstrong, due to the clearly anomalous value of one of his base motor inertia estimates. Inertia of the drive shaft, flexible coupling and gear will be more significant for the wrist axes. Frequency response measurements in Section 2.3.4 are consistent with the higher values of Armstrong and therefore these are taken as the preferred values.

Change in link inertia with configuration, as shown in Figure 2.5, has a significant effect on the dynamics of the axis control loop. The mechanical pole of the motor and link is

$$p_m = -\frac{B_m}{\eta(\underline{q})J_m} \tag{2.50}$$

The variation of the mechanical pole, due to configuration change, represents a significant challenge for control design if it is to achieve stability and performance over the entire workspace. It is clear from (2.41) that without gearing this effect would be far more significant, making independent joint control generally infeasible for direct-drive robots.

Parameter	Armstrong	Paul [204]	CSIRO		Preferred
			Load test	Back EMF	
K_{m_1}	0.189	0.255	0.223	0.227	0.227
K_{m_2}	0.219	0.220	0.226	0.228	0.228
K_{m_3}	0.202	0.239	0.240	0.238	0.238
K_{m_4}	0.075	0.078	0.069	0.0675	0.0675
K_{m_5}	0.066	0.070	0.072	0.0718	0.0718
K_{m_6}	0.066	0.079	0.066	0.0534	0.0534

Table 2.14: Measured motor torque constants - motor referenced (N.m/A).

2.3.2.2 Torque constant

For a permanent magnet DC motor the torque and back EMF are given by [146]

$$\tau = \frac{Z}{2\pi}\phi i_m = K_m i_m \tag{2.51}$$

$$E_b = \frac{Z}{2\pi}\phi\dot{\theta} = K_m\dot{\theta} \tag{2.52}$$

where ϕ is the magnetic flux due to the field, Z the number of armature windings, and θ the motor shaft angle. If a consistent set of units is used, such as SI, then the torque constant in N.m/A and back-EMF constant in V.s/rad will have the same numerical value.

Armature reaction is the weakening of the flux density of the permanent magnet field, by the MMF (*magneto-motive force* measured in Ampere-turns) due to armature current. This could potentially cause the torque constant to decrease as armature current is increased. However according to Kenjo [146] the flux density increases at one end of the pole and decreases at the other, maintaining the average flux density. Should the flux density become too low at one end of the pole, permanent de-magnetization can occur. A frequent cause of de-magnetization is over-current at starting or during deceleration. A reduction of flux density leads to reduction of torque constant by (2.51).

Table 2.14 compares measurements of torque constants of the robot used in this work, with those obtained by other researchers for other Puma 560 robots. The values in the column headed 'CSIRO load test' were obtained by a colleague using the common technique of applying known loads to robot joints in position control mode and measuring the current required to resist that load. Values in the column headed 'Armstrong' were computed from the maximum torque and current data in [20]. Tarn [245] gives the torque constant for the base axis motors as 0.259 N.m/A (apparently from the manufacturer's specification). The Kawasaki data indicate torque constants of 0.253 and 0.095 N.m/A for base and wrist motors respectively.

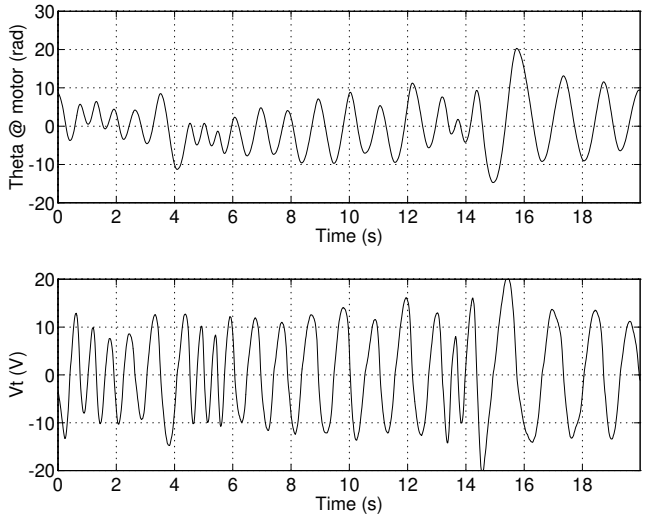


Figure 2.12: Measured joint angle and voltage data from open-circuit test on joint 2.

Considerable variation is seen in Table 2.14, but all values for the base motor are less than Tarn's value. This may be due to loss of motor magnetization [60] in the robots investigated, compared to the "as new" condition of the servo motors[12]. Another source of error is the measurement procedure itself — since the joint is not moving, the load torque is resisted by both the motor and the stiction mechanism. Lloyd [175] used an elaborate procedure based on the work involved in raising and lowering a mass so as to cancel out the effects of friction.

In this work a different approach is used, based on the equivalence of the motor torque constant and the back EMF constant from (2.51) and (2.52). This technique is well known for bench testing of motors and involves driving the motor, as a generator, at constant speed with another motor — however for a motor fitted to a robot this is not practical. A novel approach using a system identification technique allows this test to be applied in situ where the motors are back-driven as the links are manipulated manually. At open-circuit, that is $i_m = 0$, the motor terminal voltage from (2.48) is equal to the back EMF

$$v_m = K_m \dot{\theta} \tag{2.53}$$

The experimental procedure is simpler and less time consuming than the conven-

[12] Field magnetization decreases with time, and current overload or abrupt power shut-down events.

tional load-test approach. The motor is disconnected from the power amplifier[13] and a time history of terminal voltage and motor shaft angle or velocity is recorded, see Figure 2.12, as the joint is moved around by hand. The problem is transformed into one of establishing the relationship between back EMF and motor shaft speed, which is immune to the effects of static friction.

If the motor contains a tachometer then solution of (2.53) is straightforward, but requires accurate knowledge of the tachometer gain. In the more common situation where the robot motor does not incorporate a tachometer, velocity must be estimated from the derivative of measured joint angle. Using a 2-point derivative approximation[14] equation (2.53) can be written in ARX form as

$$V_m = \frac{K_m}{T}(1 - z^{-1})\Theta \tag{2.54}$$

to facilitate parameter identification. In this work MATLAB and the System Identification Toolbox [174] function `arx()` were used to fit a model of the form

$$V_m = (b_1 + z^{-1}b_2)\Theta \tag{2.55}$$

to the measured data using a batch least-squares procedure. The magnitude of the estimated coefficients $\hat{b}_1$ and $\hat{b}_2$, ideally the same from (2.54), agreed to within 0.3%. From these identified coefficients the torque constant is taken as being

$$K_m = \frac{T\left(\hat{b}_1 - \hat{b}_2\right)}{2} \tag{2.56}$$

The results of this experimental procedure are summarized in the rightmost column of Table 2.14, and the results from this method agree with the load-test method on the same robot to within $\pm 2\%$. The armature reaction effect, if present, would be expected to give open-circuit values of K_m that were consistently greater than the load test values, due to field weakening in the latter case. There is no evidence of this being a significant effect.

The considerable discrepancy with the load-test method for joint 6 is due to cross-coupling in the wrist mechanism. In the load-test procedure, a torque applied to link 6 is transferred to the all the wrist motors. The torque relationship following from (2.38) is

$$\underline{\tau}_m = \mathbf{G}^{-1}\underline{\tau}_l \tag{2.57}$$

where $\underline{\tau}_m$ is the vector of motor torques, $\underline{\tau}_l$ the vector of torques applied to the links. Using knowledge of the gear ratios from Table 2.9 a unit torque applied to link 6 results in motor torques of

$$\underline{\tau}_m = \begin{bmatrix} 0 & 0 & 0 & -0.000138 & -0.002510 & 0.013040 \end{bmatrix} \tag{2.58}$$

[13] Removing the fuse isolates the motor from the power amplifier.

[14] The use of other derivative approximations such as 3-point and 5-point derivatives has not been investigated.

Axis	Low speed. expt.	ARX expt.	Kawasaki	Tarn
Base	2.1	-	1.6	1.6
Wrist	6.7	5.1	3.83	-

Table 2.15: Comparison of measured and manufacturer's values of armature resistance (Ω). Experimental results obtained using (2.59) and (2.61).

Only 83% of the applied torque is transferred to joint 6, 16% to joint 5 and around 1% to joint 4. Thus the torque constant will be overestimated, the true value being 83% of the experimental value or 0.0548. This is close to the value determined directly by the open-circuit method. The values determined by means of the back EMF test will be chosen as the preferred values since the methodology is free from the errors present in the load test approach.

2.3.2.3 Armature impedance

Figure 2.11 shows the motor armature impedance $R_m + sL_m$, where R_m is the resistance due to the armature winding and brushes, and L_m is inductance due to the armature winding. For the Puma 560 armature inductance is low (around 1 mH [247]) and of little significance since the motor is driven by a current source.

Armature resistance, R_m, is significant in determining the maximum achievable joint velocity by (2.74) but is difficult to measure directly. Resistance measurement of a static motor exhibits a strong motor position dependence due to brush and commutation effects. A conventional *locked-rotor test* also suffers this effect and introduces the mechanical problem of locking the motor shaft without removing the motor from the robot. Measurements on a moving motor must allow for the effect of back EMF. Combining (2.48) and (2.45) and ignoring inductance we can write

$$\frac{v_m}{v_s} = \frac{R_m + R_s}{R_s} + \frac{K_m^2 \dot{\theta}}{R_s \tau_m} \tag{2.59}$$

where the second term represents the effect of back EMF, which may be minimized by increasing the torque load on the motor, and reducing the rotational speed. v_m and v_s are directly measurable and were fed to an FFT analyzer which computed the transfer function. The system exhibited good coherence, and the low frequency gain was used to estimate R_m since R_s is known. The resistance values for the base and wrist motors determined in this fashion are summarized in Table 2.15 along with manufacturer's data for the 'similar' Kawasaki Puma motors. The experiments give the complete motor circuit resistance including contributions due to the long umbilical cable, internal robot wiring, and connectors.

It is possible to simultaneously estimate R_m and L_m using the back EMF constant already determined, and time records of motor terminal voltage, current and shaft angle. First it is necessary to compute the motor voltage component due only to armature impedance

$$\hat{v}_{Z_m} = v_m - K_m\hat{\dot{\theta}} \tag{2.60}$$

by subtracting the estimated back EMF. Equation (2.48) may be rewritten in discrete-time ARX form as

$$\begin{aligned} V_{Z_m} &= (R_m + R_s)I_m + \frac{L_m}{T}(1 - z^{-1})I_m \\ &= \left\{ ((R_m + R_s) + \frac{L_m}{T}) - \frac{L_m}{T}z^{-1} \right\} I_m \end{aligned} \tag{2.61}$$

and a parameter identification technique used as for the torque constant case. For joint 6 the identification results in estimates of $R_m = 5.1\,\Omega$ and $L_m = 0.83\,\text{mH}$. The inductance is of a similar order to that reported by Tarn et al. [247], but the resistance estimate is somewhat lower than that obtained using the low-speed test described above. Given the approximations involved in the low-speed test, (2.59), the result from (2.61) is to be preferred, although the experimental procedure is less convenient.

2.3.2.4 MATLAB simulation model

The model of armature motion including friction and stiction that has been developed is a simplified version of that proposed by Hill [117] for simulation of radio-telescope antennae. Like a geared robot these antennae have high levels of friction relative to maximum torque. The armature motion is modelled by a non-linear system with two states corresponding to the motor being stationary or moving. If the motor is stationary the applied torque must exceed the stiction torque for motion to commence. If the speed falls below a threshold ε then the motor enters the stationary state.

$$\Omega = \begin{cases} 0 & \text{if stationary} \\ \frac{1}{Js+B}(\tau - \tau_C) & \text{if moving} \end{cases} \tag{2.62}$$

Hill's approach is somewhat more sophisticated and requires a fixed step integration algorithm. The algorithm is implemented as a MATLAB `S-function' and is available for use in SIMULINK [182] models. It works satisfactorily with the built in variable-length-step integration routines.

2.3.3 Current loop

The Unimate current loop is implemented in analog electronics on the so called *analog servo* board, one per axis, within the controller backplane. A block diagram of the

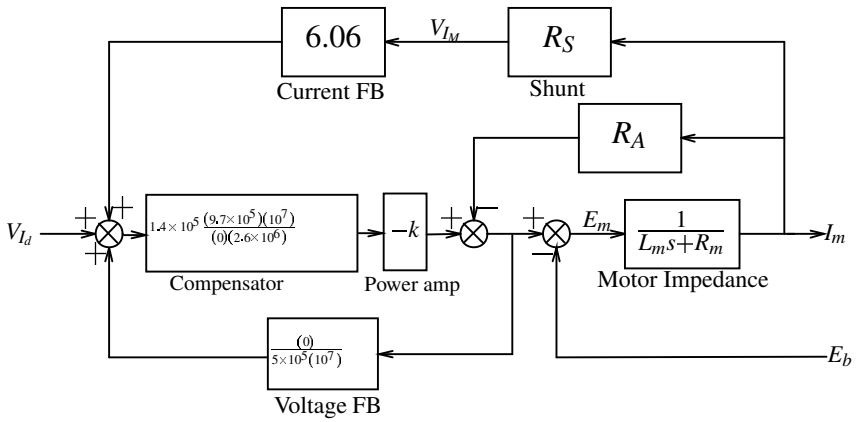


Figure 2.13: Block diagram of motor current loop. E_b is the motor back EMF, V_m motor terminal voltage, and R_A the amplifier's output impedance.

Unimate current loop is shown in Figure 2.13. The transfer function of the block marked *compensator* was determined from analysis of schematics [255] and assuming ideal op-amp behaviour. The compensator includes an integrator, adding an open-loop pole at the origin in order to achieve Type 1 system characteristics in current following. The remaining dynamics are at frequencies well beyond the range of interest, and will be ignored in subsequent modelling. The voltage gain of the block marked *power amplifier*, $-k$, has been measured as approximately -50.

Current feedback is from a shunt resistor in series with the motor, see Figure 2.11. The shunts are $0.2\,\Omega$ and $0.39\,\Omega$ for the base and wrist joints respectively. The high forward path gain, dominated by the compensator stage, results in the closed-loop gain, K_i, being governed by the feedback path

$$K_i = \frac{I_m}{V_{I_d}} = \frac{-1}{6.06 \times R_s} \tag{2.63}$$

The measured frequency response of the joint 6 current loop is shown in Figure 2.14, and the magnitude and phase confirm the analytic model above. The response is flat to 400 Hz which is consistent with Armstrong's [22] observation that current steps to the motor settle in less than $500\,\mu$s. The use of a current source to drive the motor effectively eliminates the motor's electrical pole from the closed-loop transfer function.

The measured current-loop transconductance of each axis is summarized in Table 2.16. The gains are consistently higher than predicted by (2.63), but no more than

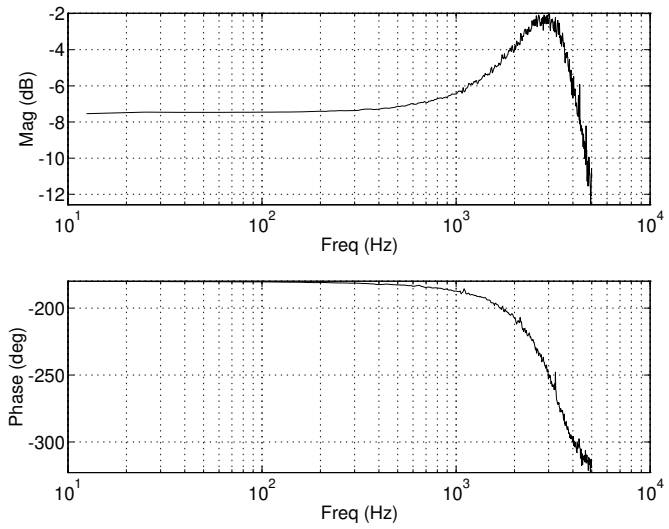


Figure 2.14: Measured joint 6 current-loop frequency response I_m/V_{I_d}.

Joint	K_i (A/V)	$i_{m_{max}}$ (A)
1	-0.883	8.83
2	-0.877	8.77
3	-0.874	8.74
4	-0.442	4.42
5	-0.442	4.42
6	-0.449	4.49

Table 2.16: Measured current-loop transconductances and estimated maximum current. The measurement is a lumped gain from the DAC terminal voltage to current-loop output.

expected given the tolerance of components used in the various gain stages[15]. The maximum current in Table 2.16 is estimated from the measured transconductance and the maximum current-loop drive of $v_{i_d} = 10\,\text{V}$.

[15]The circuitry uses only standard precision, 10%, resistors.

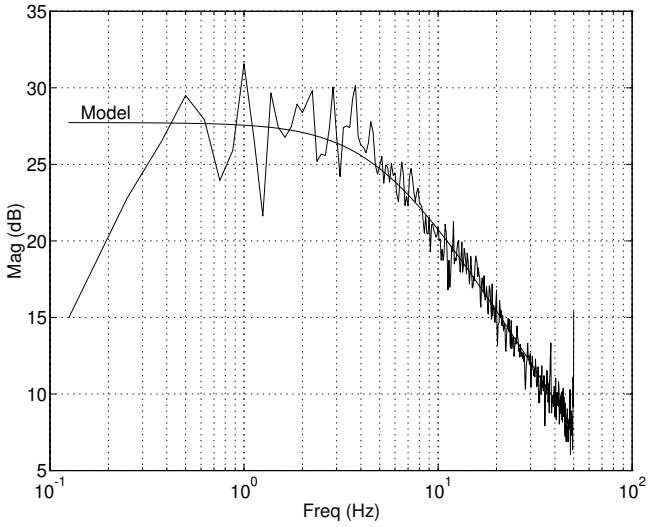


Figure 2.15: Measured joint 6 motor and current-loop transfer function, Ω_m/V_{I_d}, with fitted model response.

2.3.4 Combined motor and current-loop dynamics

The measured transfer function between motor current and motor velocity for joint 6 is given in Figure 2.15, along with a fitted transfer function. The transfer function corresponds to a linearization of the non-linear model of Figure 2.10 for a particular level of excitation. The fitted model is

$$\frac{\Omega_m}{V_{I_d}} = \frac{24.4}{(31.4)} \text{rad/s/V} \tag{2.64}$$

which has a pole at 5Hz. From Figure 2.10 and (2.63) the model response is given by

$$\frac{\Omega_m}{V_{I_d}} = \frac{K_m K_i}{J_{eff}s + B_{eff}} \tag{2.65}$$

where J_{eff} and B_{eff} are the effective inertia and viscous friction due to motor and transmission. From the measured break frequency and Armstrong's inertia value from Table 2.13 the estimated effective viscous friction is

$$B_{eff} = 33 \times 10^{-6} \times 31.4 = 1.04 \times 10^{-3} \text{N.m.s/radm} \tag{2.66}$$

Substituting these friction and inertia values into equation (2.65) gives

$$\frac{\Omega_m}{V_{I_d}} = \frac{K_m K_i}{(33 \times 10^{-6})s + (1.04 \times 10^{-3})} \tag{2.67}$$

Equating the numerator with (2.64), and using known K_i from Table 2.16 leads to an estimate of torque constant of 0.060 which is similar to the value determined earlier in Table 2.14.

The estimated effective viscous friction is significantly greater than the measured value of 37×10^{-6} N.m.s/rad from Table 2.12. This is a consequence of the linear identification technique used which yields the 'small-signal' response. The effective viscous friction includes a substantial contribution due to Coulomb friction particularly at low speed. Coulomb friction may be linearized using sinusoidal or random input describing functions giving an effective viscous damping of

$$B_{eff} = B + \frac{k\tau_C}{\sqrt{2}\sigma_{\dot{\theta}}} \tag{2.68}$$

where $\sigma_{\dot{\theta}}$ is the RMS velocity, and $k = 1.27$ (sinusoidal) or $k = 1.13$ (random).

The large-signal response may be measured by step response tests, and these indicate a much higher gain — approximately 200 rad/s/V. From (2.65) the DC gain is

$$\frac{K_m K_i}{B_{eff}} \tag{2.69}$$

which leads to an estimate of effective viscous friction of 126×10^{-6} N.m.s/rad. As expected at the higher speed, this estimate is closer to the measured viscous friction coefficient of Table 2.12.

2.3.4.1 Current and torque limits

Maximum motor current provides an upper bound on motor torque. When the motor is stationary, the current is limited only by the armature resistance

$$i_{max} = \frac{v_a}{R_m} \tag{2.70}$$

where v_a is the maximum amplifier voltage which is 40V for the Puma. From the armature resistance data given in Table 2.15 it can be shown that the maximum current given by (2.70) is substantially greater than the limit imposed by the current loop itself. Maximum current is thus a function of the current loop, not armature resistance. The sustained current is limited further by the fuses, or circuit breakers, which are rated at 4A and 2A for the base and wrist motors respectively. The fuse limits are around half the maximum achievable by the current loop.

Joint	τ_{Iloop}	τ_{fuse}
1	120	56
2	200	97
3	110	52
4	22	10
5	22	10
6	21	10

Table 2.17: Maximum torque (load referenced) at current loop and fuse current limits. All torques in N.m.

Using maximum current data from Table 2.16, and known motor torque constants, the maximum joint torques can be computed. Table 2.17 shows these maxima for the case of current limits due to the current loop or fuse.

2.3.4.2 Back EMF and amplifier voltage saturation

A robot manipulator with geared transmission necessarily has high motor speeds, and thus high back EMF. This results in significant dynamic effects due to reduced motor torque and amplifier voltage saturation. Such effects are particularly significant with the Puma robot which has a relatively low maximum amplifier voltage. Figure 2.16 shows these effects very clearly — maximum current demand is applied but the actual motor current falls off rapidly as motor speed rises. Combining (2.52) and (2.48), and ignoring motor inductance since steady state velocity and current are assumed, the voltage constraint can be written as

$$\left|\dot{\theta}K_m + \frac{\tau}{K_m}R_T\right| \leq v_a \tag{2.71}$$

where R_T is the total circuit resistance comprising armature resistance, R_m, and amplifier output impedance R_a. Rising back EMF also diminishes the torque available from the actuator during acceleration

$$\tau_{avail} = (v_a - \dot{\theta}K_m)\frac{K_m}{R_T} \tag{2.72}$$

However during deceleration, back EMF works to increase motor current which is then limited by the current loop or fuse. When the frictional torque equals the available torque

$$\tau_C + \dot{\theta}B = (v_a - \dot{\theta}K_m)\frac{K_m}{R_T} \tag{2.73}$$

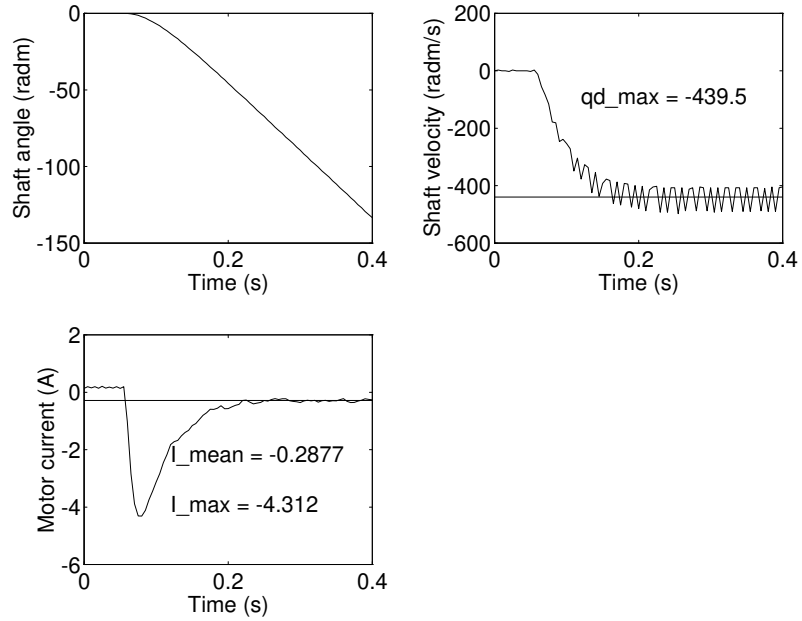


Figure 2.16: Measured motor and current loop response for a step current demand of $V_{I_d} = 10$. Motor angle, velocity and current versus time are shown. $\dot{q}_{max}$ is the steady state velocity, and I_{mean} is the steady state current. The finite rise-time on the current step response is due to the anti-aliasing filter used. The sampling interval is 5ms.

the joint velocity limit due to amplifier voltage saturation

$$\dot{\theta}_{vsat} = \frac{v_a K_m - R_T \tau_C}{R_T B + K_m^2} \tag{2.74}$$

is attained. Armature resistance, R_T, and friction, serve to lower this value below that due to back EMF alone. The corresponding motor current is

$$i_{vsat} = \frac{B v_a + K_m \tau_C}{R_T B + K_m^2} \tag{2.75}$$

Experiments were conducted in which the maximum current demand was applied to each current loop, and the motor position and current history recorded, as shown in Figure 2.16. The initial current peak is approximately the maximum current expected

Joint	Measured		Estimated	
	$\dot{\theta}_{vsat}$	i_{vsat}	$\dot{\theta}_{vsat}$	i_{vsat}
1	120	2.1	149	3.2
2	163	0.26	165	1.3
3	129	0.42	152	1.7
4	406	0.56	534	0.84
5	366	0.39	516	0.75
6	440	0.29	577	0.51

Table 2.18: Comparison of experimental and estimated velocity limits due to back EMF and amplifier voltage saturation. Velocity and current limits are estimated by (2.74) and (2.75) respectively. Velocity in radm/s and current in Amps.

from Table 2.16, but falls off due to voltage saturation as motor speed rises. Without this effect the motor speed would ultimately be limited by friction. Table 2.18 compares computed maximum joint velocities and currents with the results of experiments similar to that leading to Figure 2.16. The estimates for $\dot{\theta}_{vsat}$ are generally higher than the measured values, perhaps due to neglecting the amplifier output resistance (which has not been measured). Measurements for joints 2 and 3 are complicated by the effect of gravity which effectively adds a configuration-dependent torque term to (2.72). To counter this, gravity torque was computed using (2.36) and (2.35) and the corresponding current subtracted from that measured. At voltage saturation, the current is generally around 30% of the maximum current achievable by the power amplifier.

2.3.4.3 MATLAB simulation

A complete SIMULINK model of the motor and current loop is shown in Figure 2.17. This model is a stiff non-linear system and is thus very slow to simulate. The high-order poles due to motor inductance and the current-loop compensator are around 300 times faster than the dominant mechanical pole of the motor. Non-linearities are introduced by voltage saturation and Coulomb friction. The reduced order model, Figure 2.18, has similar non-linear characteristics but does not have the high-order poles, is much faster to integrate, and is to be preferred for simulation purposes. Increasing the value of R_m above that in Table 2.15 allows the model to more accurately predict the saturation speed and current. This could be considered as allowing for the currently unmodeled amplifier output impedance.

2.3.5 Velocity loop

Like the current loop, the Unimate velocity loop is implemented in analog electronics on the *analog servo* board. A block diagram of the velocity loop is shown in Figure

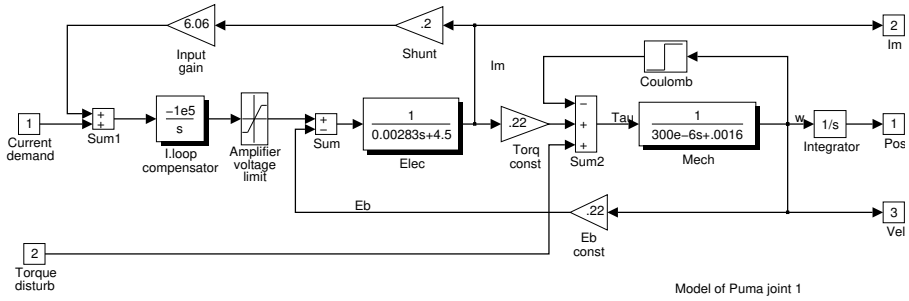


Figure 2.17: SIMULINK model MOTOR: joint 1 motor and current-loop dynamics.

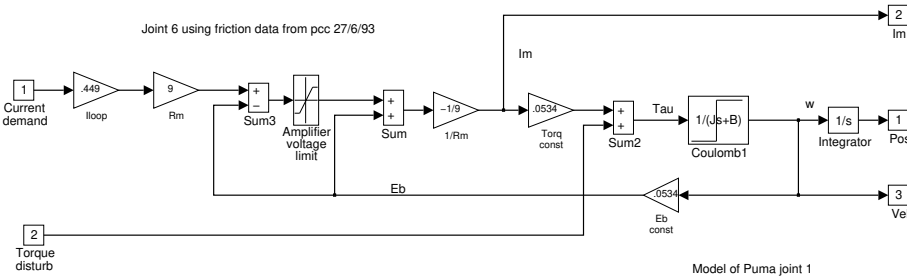


Figure 2.18: SIMULINK model LMOTOR: reduced order model of joint 6 motor and current-loop dynamics.

2.19, and includes the motor and current loop blocks already discussed. The gains K_{gain} and K_{vel} are set by trimpots R31 (GAIN) and R29 (VELOCITY) respectively on each analog servo board. The Unimation Puma does not use a tachometer to measure motor velocity. Instead, a velocity signal is synthesized from the triangular output signal from the motor's incremental encoder by an analog differentiator and a switching network. The gain of the synthetic tachometer has been experimentally determined to be

$$K_{tach} = \frac{V_{\Omega_m}}{\Omega_m} = 34 \times 10^{-3}\,\mathrm{V.s/radm} \tag{2.76}$$

with some increase in gain at frequencies below 2Hz. From Figure 2.19 the closed-loop transfer function for the motor, current and velocity loop is

$$\frac{\Omega_m}{V_{\Omega_d}} = \frac{K_{gain}K_iK_m}{Js + B + \underbrace{K_iK_mK_{gain}K_{vel}K_{tach}}_{B'}} \tag{2.77}$$

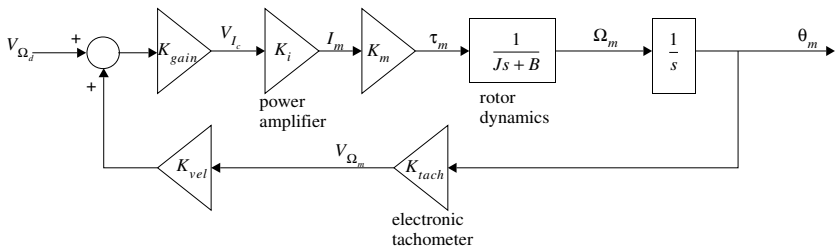


Figure 2.19: Velocity loop block diagram.

It is desirable that $B' \gg B$ in order that the closed-loop dynamics are insensitive to plant parameter variation, but in practice this is not the case. The adjustable gains K_{gain} and K_{vel} provide interacting control of the DC gain and closed-loop pole location. The measured transfer function between motor velocity and velocity demand for joint 6 is given in Figure 2.20 and the fitted model is

$$\frac{\Omega_m}{V_{\Omega_d}} = \frac{-11.9}{(148.6)} \text{radm/s/V} \tag{2.78}$$

Observe that the mechanical pole of the motor at 5 Hz has been 'pushed out' to nearly 25 Hz by the action of the velocity loop. Substituting known numeric values[16] into (2.77) results in the transfer function

$$\frac{\Omega_m}{V_{\Omega_d}} = \frac{-12.8}{(137)} \text{ radm/s/V} \tag{2.79}$$

and compares well with the measured result (2.78). Again, this transfer function represents the small-signal response of the system.

The large signal gain for each axis has also been determined experimentally by providing a step demand to the velocity loop and measuring the resultant slope of the joint position versus time curve. These results, summarized in Table 2.20, are highly dependent on the way the particular analog velocity loop has been 'tuned', as given by (2.77). It can be seen that joint 2 has a significantly higher velocity gain than the others, probably to compensate for its much higher gear reduction ratio.

A SIMULINK model of the velocity loop is shown in Figure 2.21. This model makes use of the motor and current-loop model LMOTOR developed earlier.

[16]The gains $K_{gain} = 2.26$, and $K_{vel} = 1.75$ were determined by measuring the transfer function between adjacent points in the circuitry with an FFT analyzer. $K_{vel} = 6.06K'_{vel}$ where $K'_{vel} = 0.288$ is the gain of the trimpot R29.

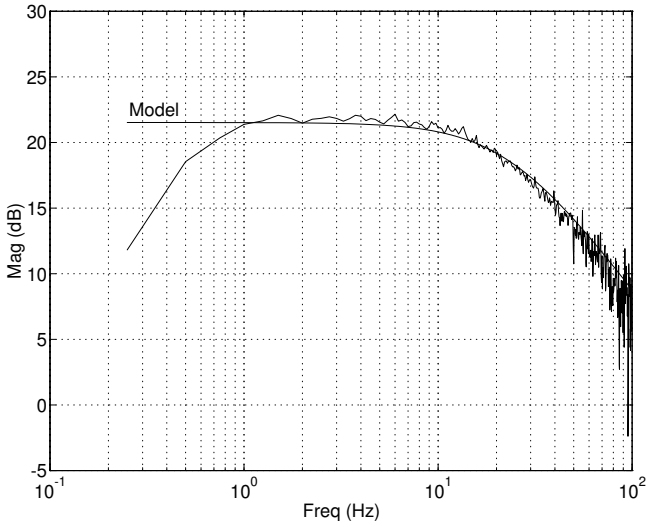


Figure 2.20: Measured joint 6 velocity-loop transfer function, Ω_m/V_{Ω_m}, with fitted model response. Phase data is not shown but is indicative of a sign reversal.

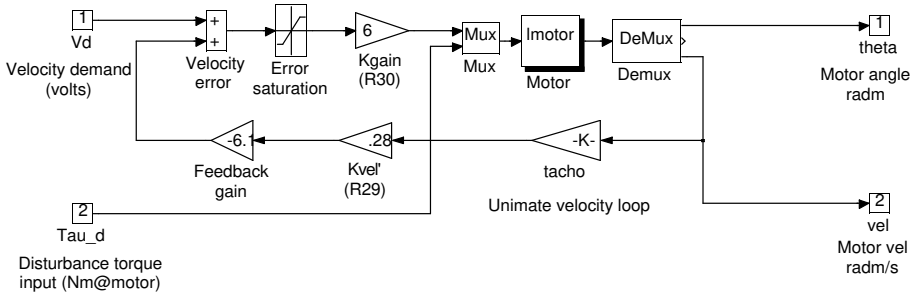


Figure 2.21: SIMULINK model VLOOP: motor velocity loop.

2.3.6 Position loop

The position loop is implemented on a *digital servo* board — one per axis. This board utilizes a 6503, 8-bit microprocessor clocked at 1 MHz to close an axis position loop at approximately 1 kHz, and also to execute commands from the host computer [56]. Position feedback is from incremental encoders, whose characteristics are summarized in Table 2.19, fitted to the motor shafts. The analog control signal, v_{DAC}, is generated

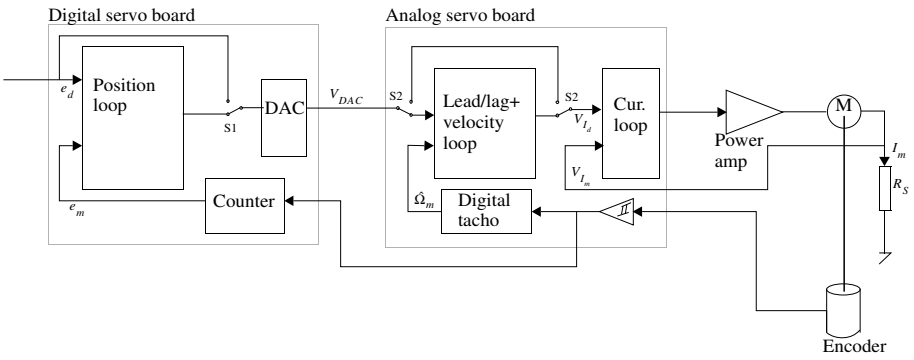


Figure 2.22: Unimation servo position control mode. The switch S2 routes the output of the DAC to the lead compensator and velocity loop.

Joint	Encoder resolution	Counts/motor rev.	Counts/radl
1	250	1000	9965
2	200	800	13727
3	250	1000	8548
4	250	1000	12102
5	250	1000	11447
6	250	500†	6102

Table 2.19: Puma 560 joint encoder resolution. †Note that joint 6 is generally run in 'divide by two mode', so that the encoder count maintained by the digital servo board is half the number of actual encoder counts. This is necessary since joint 6 can rotate through a range of more than 2^{16} counts.

by a 12-bit DAC with a bipolar output voltage in the range -10 V to 9.995 V and drives, via an analog compensation network, the analog velocity demand. The overall control system structure is shown in Figure 2.22.

A block diagram of the position loop is given in Figure 2.23. The digital control loop operates at a sample interval of $T_{servo} = 924\mu$s. The encoder demand, e_d, is provided by the host computer at intervals of $2^N T_{servo}$ where N is in the range 2 to 7 resulting in host setpoint intervals of approximately 4.5, 7, 14, 28, 56 or 112 ms. The microprocessor linearly interpolates between successive position setpoints, at the servo rate. At that same rate, the microprocessor implements the fixed-gain control

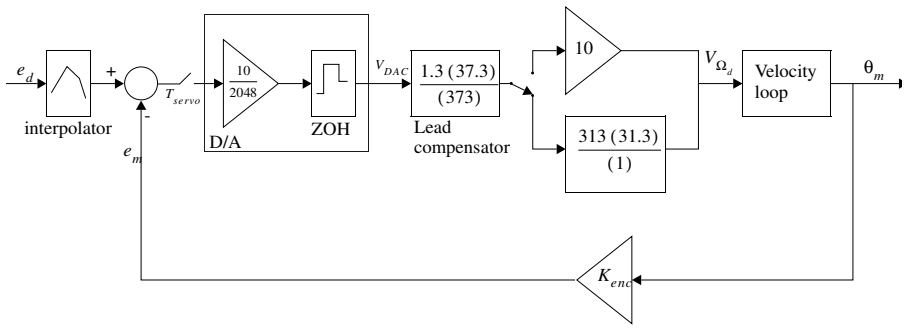


Figure 2.23: Block diagram of Unimation position control loop.

law[17]

$$v_{DAC} = \frac{10}{2048}(e_d' - e_m) \tag{2.80}$$

where e_d' is the desired encoder value from the interpolator, and e_m is the measured encoder count. This control law has been verified by disassembly of the microprocessor's EPROM and by measuring the transfer function V_{DAC}/e_m. The output voltage is a velocity command which is conditioned by a phase lead network

$$1.3\frac{(37.3)}{(373)} \tag{2.81}$$

so as to increase the closed-loop bandwidth. The gain of 1.3 was determined by analysis of the circuit, but the measured gain for joint 6 was found to be 1.13. A switchable integral action stage

$$I(s) = \begin{cases} 10 & \text{if the robot is moving, or} \\ 313\frac{(31.3)}{(1)} & \text{if the robot is 'on station'} \end{cases} \tag{2.82}$$

is enabled by the microprocessor when the axis is within a specified range of its setpoint. Integral action boosts the gain at low frequency, increasing the disturbance rejection.

The transfer function of the lead network (2.81) and integral stage (2.82) introduce a gain of around 11.3 before the velocity loop. The velocity command voltage, V_{Ω_d}, is limited by the circuit to the range ± 10 V, so the DAC voltage must in turn be limited such that

$$|v_{DAC}| \leq \frac{10}{11.3} \tag{2.83}$$

[17] Since the digital controller has a fixed gain, loop gain is adjusted by the velocity-loop gain control.

Joint	ω/v_{DAC} radm/s/V	$\dot{\theta}_{max}$ radm/s	$\dot{\theta}_{max}/\dot{\theta}_{vsat}$ radm/s
1	-101	89	74%
2	-318	281	172%
3	-90.2	80	62%
4	-366	324	80%
5	-235	208	57%
6	-209	185	42%

Table 2.20: Measured step-response gains, ω/v_{DAC}, of velocity loop. Step magnitude was selected so as to avoid voltage saturation effects. These gain values include the effect of the position-loop lead network and switchable integral/gain stage. $\dot{\theta}_{max}$ is the estimated maximum velocity due to the velocity loop when $v_{\Omega_d} = 10\,\text{V}$. Rightmost column is ratio of velocity limits due to velocity loop and voltage saturation effects.

This limits the usable voltage range from the DAC to only $\pm 0.88\,\text{V}$ — that is only 9% of the available voltage range, and gives an effective resolution of less than 8 bits on velocity demand. This saturation of the velocity-loop demand voltage is readily apparent in experimentation. From measured velocity-loop gains, ω/v_{DAC}, shown in Table 2.20 and knowledge of maximum DAC voltage from (2.83), the maximum joint velocities can be determined. These maxima are summarized in Table 2.20 along with the ratio of velocity limits due to velocity loop and voltage saturation effects. For most axes the maximum demanded joint velocity is significantly less than the limit imposed by voltage saturation. The only exception is joint 2 which, as observed earlier, has an abnormally high velocity-loop gain.

Root-locus diagrams for the joint 6 position loop are shown in Figures 2.24 and 2.25. For the case with no integral action the dominant pole is on the real axis due to the open-loop pole at the origin moving toward the compensator zero, resulting in a closed-loop bandwidth of 32 Hz. The complex pole pair has a natural frequency of 57 Hz and a damping factor of 0.62. With integral action enabled the dominant mode is a lightly damped complex pole pair with a natural frequency of around 1.2 Hz and a damping factor of 0.25.

A SIMULINK model of the position controller is shown in Figure 2.26. The structure of the switchable integral action stage is somewhat different to Figure 2.23 but more closely represents the actual controller, in particular with respect to 'bumpless' switching of the integrator. This model is used extensively in later chapters when investigating the behaviour of visual-loop closed systems.

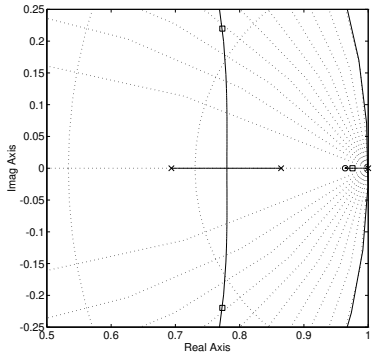


Figure 2.24: Root-locus diagram of position loop with no integral action.

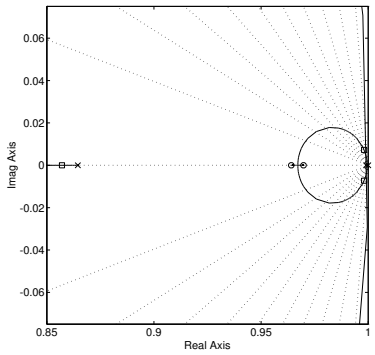


Figure 2.25: Root-locus diagram of position loop with integral action enabled.

2.3.6.1 Host current control mode

The Unimate digital servo board also allows the motor current to be controlled directly by the host. In this mode the DAC output voltage is connected directly to the current loop as shown in Figure 2.27. The DAC is updated within T_{servo} of the setpoint being given. This mode is useful when the host computer implements its own axis control strategy.

2.3.7 Fundamental performance limits

A summary of the robot's performance limits is given in Table 2.21. The velocity limits are due to back EMF and voltage saturation, given previously in Table 2.18. The

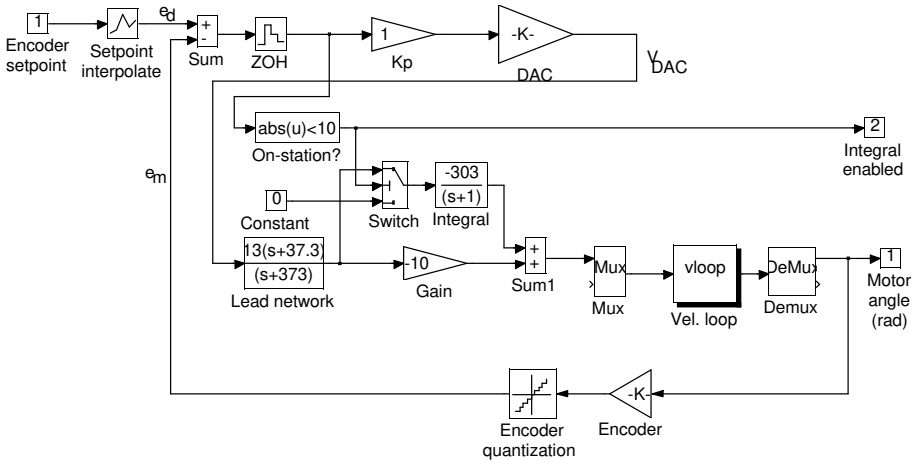


Figure 2.26: SIMULINK model POSLOOP: Unimation digital position control loop.

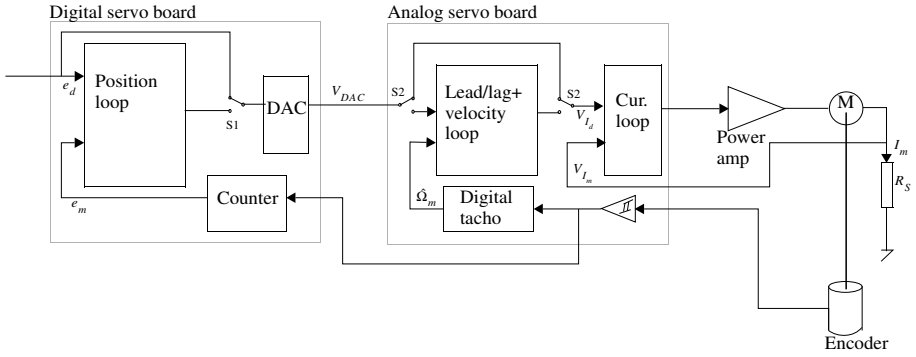


Figure 2.27: Block diagram of Unimation servo current control mode. The switch S2 is used to route the host setpoint directly to the current-loop demand via the DAC, bypassing the velocity loop and lead compensator.

torque limits are derived from the fuse-limited current and torque constant data from Table 2.14. Acceleration limits are estimated from the torque limits, maximum normalized link inertia from Table 2.10, and armature inertia from Table 2.13. Note that the velocity and acceleration maxima are mutually exclusive — maximum acceleration can be achieved only at zero velocity or during deceleration. Maximum velocity occurs when achievable acceleration is zero.

Joint	Motor referenced			Load referenced		
	$\dot{\theta}$	τ	$\ddot{\theta}$	$\dot{\theta}$	τ	$\ddot{\theta}$
1	120	0.91	840	1.92	56	13
2	163	0.91	3200	1.51	97	30
3	129	0.97	3000	2.40	52	55
4	406	0.14	3800	5.34	10	49
5	366	0.14	4000	5.09	10	55
6	440	0.11	3700	5.74	10	48

Table 2.21: Summary of fundamental robot performance limits.

2.4 Significance of dynamic effects

When numeric parameters are substituted into the symbolic torque expressions from Section 2.2.2 it becomes clear that the various torque contributions vary widely in significance. For example, the acceleration of joint 6 exerts a very small torque on joint 2 compared to the gravity load on that axis. While considerable literature addresses dynamic control of manipulators, less attention is paid to the significance, or relative magnitude, of the dynamic effects. In much early work the velocity terms were ignored, partly to reduce the computational burden, but also to acknowledge the reality that accurate positioning and high speed motion are exclusive in typical robot applications. Several researchers [119, 153, 164] have investigated the dynamic torque components for particular robots and trajectories. Unfortunately the significance of the torque components is highly dependent upon both robot and trajectory. Khosla [153] for example found that, for the CMU DD-II arm, inertia torques dominate.

Recently Leahy [162] has proposed standard trajectories for the comparison of model-based controllers for the Puma 560 robot[18]. Figure 2.28 shows the torque components for the proposed test trajectory. Leahy's minimum jerk trajectory algorithm was not available so a seventh order polynomial was used instead, and this seems to have resulted in marginally higher peak velocities and accelerations. It is clear that friction, gravity and inertia torques are significant. Another significant factor is torque limitation due to voltage saturation. Figure 2.28 shows the available torque computed as a function of joint velocity, and this limit is clearly exceeded in the middle of the trajectory. The proposed trajectory is perhaps too close to the robot's performance limit to be of use as a benchmark.

Symbolic manipulation of the dynamic equations provides another way to gain

[18] Unfortunately the proposed standard is expressed in terms of start and finish joint angles, but does not specify the joint angle convention used. However study of earlier work by Leahy [164] indicates that he uses the convention of Lee [166]. Another unfortunate omission was the "minimum jerk" trajectory generator algorithm used.

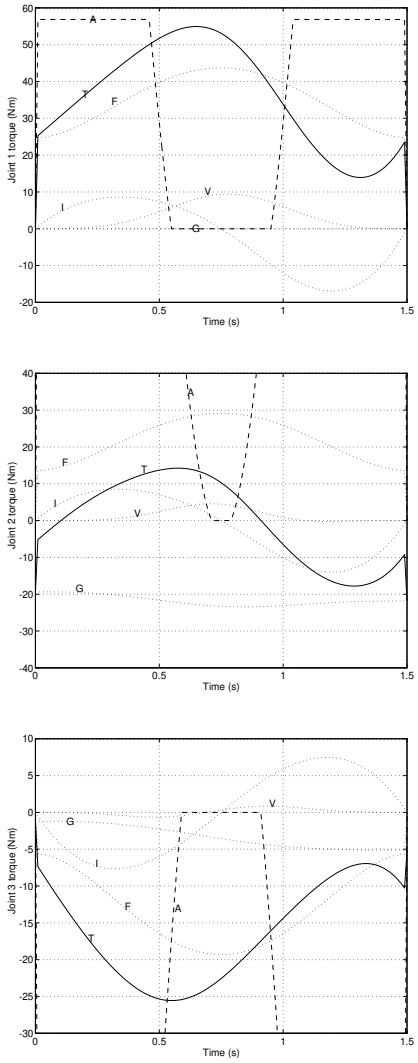


Figure 2.28: Breakdown of torque components for Leahy's proposed test trajectory. Joints 1, 2 and 3 shown down the page. The curves are marked; T total torque, F friction, G gravity, I inertia, and V velocity terms. The curve marked A is the available torque due to amplifier voltage saturation, (2.72), and fuse current limit.

Rank	Joint 1	Joint 2	Joint 3
1	$-0.6916\,S_2\,\ddot{q}_2$	$37.23\,C_2$	$-8.744\,S_{23}$
2	$1.505\,{C_2}^2\,\ddot{q}_1$	$4.182\,\ddot{q}_2$	$0.6864\,\ddot{q}_3$
3	$-1.943\,{S_3}^2\,C_2\,S_2\,\dot{q}_1\,\dot{q}_2$	$-8.744\,S_{23}$	$0.2498\,{C_4}^2\,\ddot{q}_3$
4	$1.301\,{C_2}^2\,{C_3}^2\,\ddot{q}_1$	$-0.7698\,S_3\,\ddot{q}_2$	$-0.3849\,S_3\,\ddot{q}_2$
5	$-1.637\,{C_3}^2\,C_2\,S_2\,\dot{q}_1\,\dot{q}_2$	$-0.3849\,S_3\,\ddot{q}_3$	$0.1688\,{S_4}^2\,\ddot{q}_3$

Rank	Joint 4	Joint 5	Joint 6
1	$0.1921\,\ddot{q}_4$	$0.1713\,\ddot{q}_5$	$0.1941\,\ddot{q}_6$
2	$0.02825\,S_{23}\,S_4\,S_5$	$-0.02825\,C_{23}\,S_5$	0
3	$-0.001244\,S_4\,S_5\,\ddot{q}_3$	$-0.02825\,C_4\,C_5\,S_{23}$	0
4	$0.001244\,S_3\,S_4\,S_5\,\ddot{q}_2$	$0.001244\,C_4\,C_5\,\ddot{q}_3$	0
5	$-0.001244\,S_4\,S_5\,\ddot{q}_2$	$-0.001244\,C_4\,C_5\,S_3\,\ddot{q}_2$	0

Table 2.22: Ranking of terms for joint torque expressions. Computed at 20% peak acceleration and 80% peak velocity. Motor armature inertia is included. Coefficients are shown to 4 figures.

insight into the significance of various dynamic effects. After numerical substitution into the symbolic equations the terms in the torque expressions comprise a numeric coefficient multiplied by a function of manipulator state variables. The magnitude of the coefficient is related to the significance of that term to the total torque. Velocity terms contain either a velocity squared or a product of velocities and may be significant despite a small coefficient. The procedure introduced here involves setting joint velocities and accelerations to nominal values prior to ranking the magnitude of the terms. The nominal values chosen are 20% of the maximum acceleration, and 80% of the maximum velocity as given in Table 2.21. Table 2.22 shows the 5 most significant dynamic terms for each torque expression at the nominal velocity and acceleration. As expected, gravity ranks highly for joints 2 and 3, followed by inertia. Joint 3 has an off-diagonal inertial component indicating some coupling with joint 2. Joint 1 has significant Coriolis coupling with joint 2 and to a lesser extent joint 3. The wrist joints are dominated by inertia, with gravity and inertial coupling effects one and two orders of magnitude down respectively.

2.5 Manipulator control

2.5.1 Rigid-body dynamics compensation

In conventional manipulator control, for instance the standard Unimate controller, each axis is independently controlled, and torques due to inertial coupling, Coriolis,

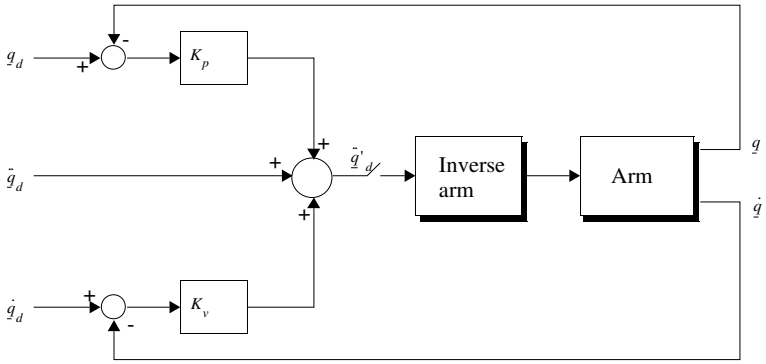


Figure 2.29: Computed torque control structure.

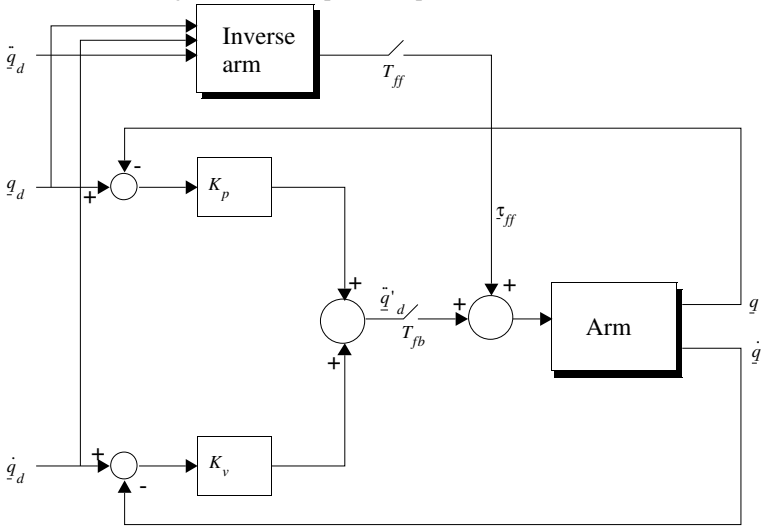


Figure 2.30: Feedforward control structure.

centripetal and gravity effects are treated as disturbances. Gearing helps to reduce the configuration dependence of some dynamic effects, and also reduces the magnitude of disturbance torques at the motor. The quality of trajectory tracking is directly related to the disturbance rejection capability of the axis servo. However given knowledge of the manipulator's equations of motion, inertial parameters, and manipulator state it is possible to compute the joint torque required to follow a trajectory, and explicitly counter all the disturbance torques.

The two major forms of control incorporating manipulator dynamics that have

been proposed [69] are:

1. Computed torque control, shown in Figure 2.29. The torque demand for the actuators is

$$\underline{Q} = \mathbf{M}(\underline{q}) \left\{ K_v(\underline{\dot{q}}_d - \underline{\dot{q}}) + K_p(\underline{q}_d - \underline{q}) + \underline{\ddot{q}}_d \right\} + \mathbf{C}(\underline{q}, \underline{\dot{q}})\underline{\dot{q}} + \mathbf{F}(\underline{\dot{q}}) + \mathbf{G}(\underline{q}) \quad (2.84)$$

   where K_p is the position gain, and K_v the velocity gain, or damping term. The inverse dynamics are 'in the loop' and must be evaluated each servo interval, although the coefficients of **M**, **C**, and **G** could be evaluated at a lower rate. Assuming ideal modelling and parameterization, the error dynamics of the linearized system are

$$\ddot{e} + K_v\dot{e} + K_p e = 0 \quad (2.85)$$

   where $e = q_d - q$. The error dynamics of each axis are independent and a function only of the chosen gain matrices. In the case of model error there will be coupling between axes, and the right-hand side of (2.85) will be a non-zero forcing function.

2. Feedforward control, shown in Figure 2.30, linearizes the dynamics about the operating points q_d, $\dot{q}_d$ and $\ddot{q}_d$, by providing the gross torques

$$\underline{Q} = \mathbf{M}(\underline{q}_d)\underline{\ddot{q}}_d + \mathbf{C}(\underline{q}_d, \underline{\dot{q}}_d)\underline{\dot{q}}_d + \mathbf{F}(\underline{\dot{q}}_d) + \mathbf{G}(\underline{q}_d) + \left\{ K_v(\underline{\dot{q}}_d - \underline{\dot{q}}) + K_p(\underline{q}_d - \underline{q}) \right\} \quad (2.86)$$

   where K_p is the position gain, and K_v is the velocity gain, or damping term. The inverse dynamics are not 'in the loop' and can be evaluated at a lower rate, T_{ff}, than the error feedback loops, T_{fb}. Again assuming ideal linearization of the plant, the error dynamics are given by

$$\mathbf{M}(\underline{q}_d)\ddot{e} + K_v\dot{e} + K_p e = 0 \quad (2.87)$$

   which are seen to be dependent upon manipulator configuration.

In each case the position and velocity loops are only required to handle modelling errors and disturbances. Both techniques require accurate determination of manipulator state and in particular joint velocity. Commonly, robots have only joint position sensors so velocity must be estimated. Khosla [151] describes a "least squares digital filter" for this purpose, but other reports on model-based control experiments do not discuss this issue.

There is relatively little experimental evaluation of the dynamic control of robot manipulators. One reason is the lack of manipulators for which these rigid-body dynamic effects are significant, though this situation has changed over the last 5 years. Most commercial robots are characterized by high gear ratios, substantial joint friction, and relatively low operating speed — thus independent joint control suffices.

Gravity loading is commonly countered by introducing integral action to the position loop when near the destination.

A number of studies have investigated the efficacy of the control strategies (2.84) and (2.86) for the Puma 560 and various direct drive robots. In general the performance metric used is high-speed position tracking error. Valavanis, Leahy and Saridis [257] reported on computed torque control for a Puma 600 robot, and found the performance inferior to individual joint control. The discussion is not specific, but it would seem problems in modelling, and the low sample rate may have contributed to the result. They conclude that for such a highly geared manipulator, gravity and friction overwhelm the joint interaction forces. Later work by Leahy et al. [161, 162, 165] concludes that tracking performance improves with the completeness of the dynamic model which ultimately includes friction, velocity terms and payload mass. Leahy also finds that computed torque control is superior to feedforward control.

As already alluded to, it is possible to compute the feedforward torque at a lower rate than the feedback torque. Leahy [162] investigates this for feedforward torque control and finds acceptable performance when T_{ff} is up to eight times longer than T_{fb}. Accuracy is increased if the feedforward torque is computed at the feedback rate, but with the feedforward torque coefficients, $\mathbf{M}(\underline{q}_d)$, $\mathbf{C}(\underline{q}_d, \underline{\dot{q}}_d)$, and $\mathbf{G}(\underline{q}_d)$, evaluated at the lower rate. These approaches can reduce the online computational burden. Sharkey et al. [228] provide a stronger reason for such a control strategy. They argue that the inertial compensation need not be computed at a rate beyond the desired closed-loop bandwidth. Inertial parameter uncertainty means that high-bandwidth control should be restricted to local joint feedback so as to control unmodeled dynamics. High-bandwidth inertial compensation, in conjunction with poor inertial models, was found to couple noise between axes, leading to poor control.

In the last decade, with the availability of high-torque actuators, a number of experimental direct-drive robots have been built to exploit the potentially greater performance achievable by eliminating the complex transmission. Several studies have compared the two control strategies described, and Khosla [152] concludes, like Leahy, that the computed torque scheme gives slightly better performance than feedforward control when modelling errors are present. If an exact model is available then both schemes would give the same results. Khosla also found that the off-diagonal terms in the manipulator inertia matrix were significant for the CMU DD-II arm. An et al. [11] evaluated a spectrum of control strategies for the MIT SLDD Arm, including independent joint control, gravity feedforward, and full dynamics feedforward. They conclude that feedforward control significantly improves following accuracy at high speed, and that model accuracy was important. Further work [10] finds no significant difference between feedforward and computed torque control.

A more radical approach is to design the manipulator so as to simplify the dynamics. The MIT 3DOF direct drive arm [287] used clever mechanical design to ensure that the manipulator's inertia matrix is diagonal and configuration invariant.

This is achieved by placing all actuators remotely and transmitting the torque via a transmission, eliminating the reaction torques transmitted between adjacent links. This approach simplifies dynamic control, and ensures uniformity of response over the work volume. The Minnesota direct drive arm [145] is statically balanced to eliminate gravity forces. The principal benefits accrue not so much from simplification of the dynamics expressions, but from the use of smaller (and lighter) motors due to the elimination of steady state holding torques.

2.5.2 Electro-mechanical dynamics compensation

As seen from Section 2.4 friction and voltage saturation effects are significant. The latter cannot be compensated for — it is a fundamental constraint in motion planning [17]. However a number of strategies have been proposed to counter friction and include [43]:

- high-gain control loops to reduce the effect of non-linearities, but with limit-cycles as a possible consequence
- adding a high frequency 'dither' signal to the torque command, at the expense of possibly exciting high order structural dynamics, and possible fatigue of actuators and bearings;
- compensation by non-linear feedforward of friction torque, which requires an accurate model of the frictional characteristics of the joint.

Canudas De Wit [43] describes the effect of under and over compensation of estimated friction on system stability for PD joint controllers. He then describes an adaptive compensation scheme which estimates the unknown frictional characteristics. A major source of difficulty is noise-free determination of low velocity. He proposes a combined feedback and feedforward compensator where the feedforward torque is computed from measured velocity above a velocity threshold, and from desired velocity below the threshold. Experimental results confirmed the operation of the proposed algorithm. The controllers described in Sections 8.1 and 8.2 use this technique for friction compensation.

2.6 Computational issues

Model-based dynamic control, either by computed torque or feedforward, requires the rapid computation of the manipulator's inverse dynamics. Through the 1980s much literature was devoted to ways of computing the inverse dynamics sufficiently fast on microprocessor hardware to allow online control. The approaches have included DSP devices [138], parallel processing, table lookup [209], and special architectures [18, 168, 172, 208].

Many of the reports seem to have focussed on a target of 6-axis torque computation time in around 1 ms but this level of performance may be unwarranted. Luh et al. [177] indicate a servo rate of at least 60 Hz is required and Paul [199] suggests a servo rate of 15 times the highest structural resonance. However the coefficients of the dynamic equations do not change rapidly since they are a function of joint angle and may be computed at a fraction of the rate at which the servo equations are evaluated [201]. Sharkey et al. [228] go further and contend that it may be disadvantageous to compute rigid-body dynamics at a high rate since the dynamic model is unlikely to be accurate at those high frequencies. Computation of the rigid-body dynamics for all 6 joints is also unnecessary since the dynamic effects are most significant for the base axes as shown in Table 2.22. Another trend in the 1980s was the increasing power of general purpose symbolic algebra packages, enabling researchers to readily manipulate the very complex dynamic equations of motion. Symbolic simplification, in conjunction with modern microprocessors, offers an attractive alternative to specialized hardware architectures. The next sections review work in the areas of parallel computation and symbolic simplification and then contrast the two approaches.

2.6.1 Parallel computation

Lathrop [159] provides a good summary of the computational issues involved and some approaches to parallelism. A significant problem in using multiple processors is scheduling the component computational tasks. This difficult problem is tackled in many different ways, but is essentially an 'off-line' process that need be done only once. Important issues in parallelization include:

- Level of decomposition. Is the minimum scheduled computation a dynamic variable, matrix operation or scalar multiplication? As the parallelism becomes more fine grained, interprocessor communications bandwidth will become a limiting factor.
- Utilization profile or load balancing. It is desirable to keep all processors as fully utilized as possible.
- The 'speed up' achieved, that is, the execution time ratio of the multiprocessor implementation to the single processor implementation. The speed-up can never exceed the number of processors, and the ratio of speed-up to number of processors is some indication of the benefit gained, or efficiency of multiprocessing.

Luh [176] demonstrated how six processors could be used to achieve a speed-up of 2.64 for computation of the manipulator inverse dynamics. A “variable branch-and-bound” technique was developed to schedule the computing tasks. Nigham and Lee [194] propose an architecture based on six 68020 processors, with the parallel

computations scheduled manually. Integer arithmetic is proposed, and with 16.7 MHz devices a time of 1.5 ms is predicted, but issues such as shared resource contention or synchronization are not discussed; nor is the improvement over single CPU performance given. Khosla [150] also describes a manual approach to scheduling the computations onto 8 processors for the NE formulation, to achieve an 81% time reduction (speed-up of 5.3). Kasahara and Narita [144] describe an automatic approach to the np-hard problem of multiprocessor scheduling. They show the decomposition of the dynamics of a six-axis revolute robot into 104 computational tasks, and a maximum speed-up of 3.53 for four CPUs.

An alternate approach is to partition the problem into one axis per processor. Unfortunately the recursive nature of the RNE computation is such that the inward recursion for link i cannot be initiated until the outward recursion for links $i+1$ to n, and inward recursion for links n to $i+1$, have been completed. However the interaction force and moment from the inward recursion can be predicted from previous values, and this allows each joint processor to operate independently. Vuskovic [263] investigates the errors introduced by prediction; zero-order prediction gave acceptable results, and the error is shown to reduce with sample interval. First-order prediction was less successful. A speed-up factor of 6 over a single processor was achieved, without the need for complex off-line task scheduling, and in fact the processors can operate completely asynchronously. Yii et al. [286] also describe a zero-order predictive system, but provide no experimental results.

Lathrop [159] uses a directed graph to represent the RNE formulation, with groups of parallel processors at each node. Conventional and systolic pipelined structures are proposed with a speed-up of two orders of magnitude requiring 180 'simple' matrix-vector processors for the 6-axis case. This approach is then extended to a new parallel implementation which has $O(\log_2 n)$ cost requiring 637 matrix-vector processors for 6 axes.

Hashimoto et al. [112] describe a parallel form of the RNE equations derived from analysis of data dependency graphs. For n joints the computation can be divided into n approximately equal parallel tasks. Later work [113] discusses the implementation of this algorithm for 3-axis dynamic control of a Puma 560 with three transputers, achieving a dynamics computation time of 0.66 ms.

2.6.2 Symbolic simplification of run-time equations

Symbolic expansion of the equations of motion has been previously discussed in Section 2.2.2, but the sum-of-product form is too expensive for run-time evaluation since the inherent factorization of the RNE form has been lost, and symbolic re-factorization is prohibitively expensive. A far better approach is to retain factors during symbolic evaluation. In this work whenever an intermediate expression grows to become a sum of more than one product term then that expression is replaced by a single new vari-

Method	3-axis			6-axis		
	Mul	Add	Factors	Mul	Add	Factors
General solution	402	345		852	738	
Symbolically simplified by MAPLE	238	108	45	538	254	106
Symbolic simplified by MAPLE after parameter value substitution	203	59	39	358	125	93
ARM closed form				461	352	
ARM recursive form				224	174	
Izaguirre and Paul [131]				331	166	
Leahy [164]				304	212	

Table 2.23: Operation count for Puma 560 specific dynamics after parameter value substitution and symbolic simplification.

able, which is equated to the previous expression. Table 2.23 shows the operation count for this factored symbolic representation for a Puma 560, as well as the number of factors so generated. The operation count is considerably less than the general form given in Table 2.2. Substituting numeric values for inertial parameters, many of which are zero, reduces the operation count still further.

ARM has been used [192] to generate customized computations for both direct and inverse dynamics. The inverse dynamics can be in closed or recursive form, the latter being most advantageous for higher numbers of joints. The results of ARM are compared with the results obtained using the writer's MAPLE program in Table 2.23. The MAPLE results compare very favorably with those of ARM for the comparable closed form solution. ARM's factored recursive form has not been implemented with MAPLE. Another symbolic approach generates computationally efficient forms of the inertia and velocity terms via a Lagrangian formulation [131]. Written in LISP, the programs read a manipulator parameter file, and write 'C' functions to compute the **M**, **C** and **G** matrices as functions of manipulator state.

2.6.3 Significance-based simplification

As shown in Table 2.22 the terms in the torque expressions vary greatly in significance. This can be seen clearly in Figure 2.31 which shows a histogram of the distribution of these coefficient magnitudes for the Puma 560's joint 1 torque expression. The median coefficient magnitude is nearly four orders of magnitude below that of the greatest coefficient. In this section we investigate the possibility of 'culling' the terms, keeping only those that contribute 'significantly' to the total joint torque. Using the

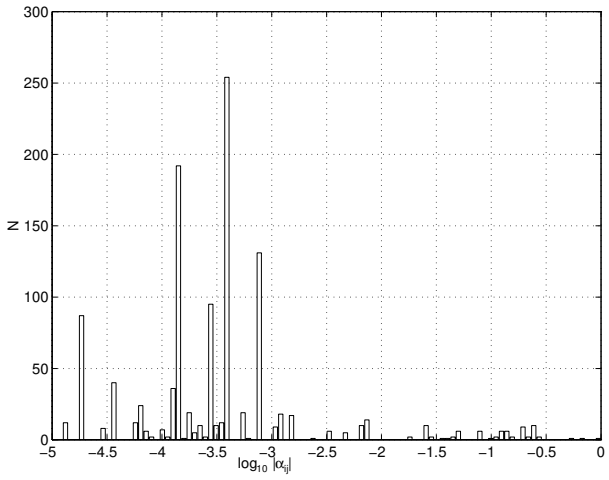


Figure 2.31: Histogram of $\log_{10}|\alpha_{1j}|$ coefficient magnitude for τ_1, normalized with respect to the greatest coefficient, for the Puma 560. The median value is -3.67.

nominal velocity and acceleration values as in Section 2.4 the torque expressions were truncated at coefficient magnitude less than 5% and 1% of the greatest coefficient. The number of remaining terms for each axis are summarized in Table 2.24. A comparison of more elaborate culling methods is given in [53].

To investigate the effect of culling on accuracy a simple Monte-Carlo style simulation was conducted. Error statistics were collected on the difference between the full and truncated torque expressions for N random points in manipulator state space. The joint angles were uniformly distributed in the joint angle range, while velocity and acceleration were normally distributed with the 2σ values equated to the limits from Table 2.21. These results are also summarized in Table 2.24. Truncation to 5% introduces negligible errors, except for joint 2. A good compromise would appear to be culling to 1% significance for joint 2 and 5% for all others. At the 5% level only 4% of the terms remain in the torque expression. Online torque computation based on these truncated torque expressions is used in the controller described in Section 8.2.

2.6.4 Comparison

To place some of these efforts in context, the execution times of the RNE formulation for a 6-axis Puma 560 are given in Table 2.25. The matrix numeric examples by the author were written in 'C' in a straightforward manner, while the symbolically simplified examples were generated automatically by MAPLE with some minor code

Joint	N_{orig}	5% significance			
		N_{terms}	$\overline{\tau}_d$	σ_{τ_d}	$\max\Delta\tau$
1	1145	162	0.0027	0.0978	0.4599
2	488	8	0.0814	6.6640	22.7200
3	348	19	0.0218	0.6881	2.5660
4	81	3	-0.0002	0.0188	0.0847
5	90	4	-0.0003	0.0217	0.0891
6	18	2	0.0000	0.0010	0.0032
	2170	199			

Joint	N_{orig}	1% significance			
		N_{terms}	$\overline{\tau}_d$	σ_{τ_d}	$\max\Delta\tau$
1	1145	333	-0.0013	0.0251	0.1024
2	488	36	0.0158	0.7514	2.8470
3	348	39	-0.0041	0.1420	0.5907
4	81	30	-0.0001	0.0027	0.0134
5	90	36	-0.0002	0.0031	0.0165
6	18	2	0.0000	0.0010	0.0032
	2170	476			

Table 2.24: Significance-based truncation of the torque expressions. This shows the original number of terms in the expression, and the number after truncating to 5% and 1% of the most significant coefficient. Also shown are the mean, standard deviation and maximum (all in N.m) of the error due to truncation, computed over 1000 random points in manipulator state space. All torques are link referenced.

massaging performed by `sed` scripts. Table 2.25 shows that, for the same processor, the factored symbolic form executes approximately 3 times faster than the numeric matrix form of (2.12) to (2.25). This ratio of 3 would be expected from examining the operation counts summarized in Table 2.23. The fairly 'ordinary' single board computer for robot control in this work, a 33 MHz 68030, is able to compute the torque for the first 3 axes in only 800 μs.

In order to compare scalar and parallel implementations a computational efficiency metric is proposed

$$\eta = \frac{n \times 10^6}{N f_{clock} T} \tag{2.88}$$

where n is the number of axes, N the number of processors, f_{clock} the CPU clock rate, and T the execution time. Clock rate is included to enable comparison between scalar and parallel implementations on the same CPU type, but is not meaningful in comparison across processor types.

Table 2.26 compares two reported parallel implementations with scalar results generated by the author from Table 2.25. The parallel implementations are shown

Processor	Approach	Axes	Year	Time (μs)
PDP-11/45 [177]	assembler, RNE	6	1982	4500
Sun3/60†	'C' general RNE	6	1988	13400
μPD77230 DSP [138]	assembler, general RNE	6	1989	550
Sun4/20 (SLC)†	'C' general RNE	6	1990	1630
T800 x 3 [112]	Occam parallel processing	3	1989	660
T805 25 MHz†	'C' + symbolic simplification	3	1989	570
T805 25 MHz†	'C' + symbolic simplification	6	1989	1140
68030+68882 33 MHz†	'C' + symbolic simplification	6	1989	1580
68030+68882 33 MHz†	'C' + symbolic simplification	3	1989	795
Sun4/20 (SLC)†	”	6	1990	548
Sun4/20 (SLC)†	”	3	1990	362
SparcStation 10†	”	6	1993	65

Table 2.25: Comparison of computation times for Puma 560 equations of motion based on RNE formulation. The given year is an indication only of when the particular computing technology was first introduced. Those marked with a † were benchmarked by the writer, using the GNU C cross compiler v2.2.3 for 68030, gcc v2.4.5 for the Sparc and Helios C v2.05 for the transputer, T805. Compiler optimization, `-O`, was enabled in all cases.

to make poor use of the computing hardware. This is fundamentally because they are using additional hardware to perform arithmetic operations which can be eliminated off-line. It can be concluded that off-line symbolic manipulation, high-level languages and state-of-the-art computers provide a simple and powerful means of computing manipulator dynamics at a sufficient rate. The generation of symbolically simplified run-time code in 'C' from a Denavit-Hartenberg parameter file executes in less than 15 s on a Sun SparcStation 2. Specialized hardware or parallel software with their inherent long development times must now be seen as an unattractive approach to the online computation of manipulator dynamics. The off-line computational effort devoted to scheduling parallel computation [144, 176, 194], described in Section 2.6.1, would perhaps have been better devoted to symbolic simplification.

Approach	CPU	f_{clock} (MHz)	N	T (ms)	n	η
Nigham&Lee	68020	16.7	6	1.5	6	39.9
Corke†	68030	33	1	2.0	6	119
Hashimoto	T800	20‡	3	0.66	3	76
Corke†	T805	25	1	3.8	0.57	211

Table 2.26: Comparison of efficiency for dynamics computation. †Written by the author, see Table 2.25. ‡Clock speed of 20 MHz is assumed only.

Chapter 3

Fundamentals of image capture

This chapter introduces some fundamental aspects of image formation and capture that are particularly relevant to visual servoing. Cameras are often treated as black boxes with a couple of adjustment rings, but before an image becomes available in a framestore for computer analysis a number of complex, but frequently overlooked, transformations occur. These processing steps are depicted in Figure 3.1 and include illumination, lens, sensor, camera electronics and digitizer, and each will introduce artifacts into the final digital image. Particularly important for visual servoing are the temporal characteristics of the camera, since the camera's shutter acts as the sampler in a visual control loop.

This chapter will systematically examine the process of image formation and acquisition, generally overlooked, prior to it being digitally processed. The effects of each stage are examined and a detailed model built up of the camera and image digitizing system. A particular camera, Pulnix TM-6, and digitizer, Datacube DIGIMAX [73], are used as concrete examples for model development, but are typical of CCD cameras and digitizers in general.

3.1 Light

3.1.1 Illumination

Light is radiant energy with a capacity to produce visual sensation and *photometry* is that part of the science of *radiometry* concerned with measurement of light. Radiant energy striking a surface is called *radiant flux* and is measured in watts. Radiant flux evaluated according to its visual sensation is *luminous flux* and is measured in *lumens*. The ratio of luminous flux to radiant flux is *luminosity* measured in lumens/watt. The

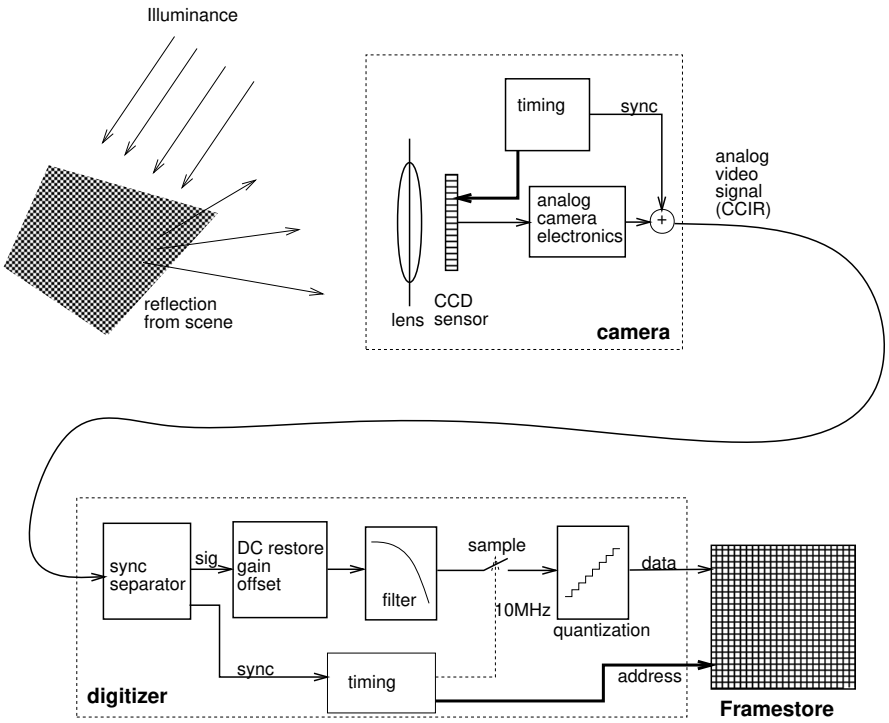


Figure 3.1: Steps involved in image processing.

photopic[1] luminosity curve for a 'standard' observer is shown in Figure 3.2.

The *luminous intensity* of a source is the luminous flux per unit solid angle[2] measured in lumens/sr or candelas. Some common photometric units are given in Table 3.1. For a point source of luminous intensity I the *illuminance* E falling normally onto a surface is

$$E = \frac{I}{l^2} \text{lx} \tag{3.1}$$

where l is the distance between source and the surface. Outdoor illuminance on a bright sunny day is approximately 10,000 lx, whereas office lighting levels are typically around 1,000 lx. The *luminance* or *brightness* of a surface is

$$L_s = E_i \cos\theta \, \text{nt} \tag{3.2}$$

[1] The eye's light-adapted response using the cone photoreceptor cells. The dark adapted, or scotopic, response using the eye's monochromatic rod photoreceptor cells is shifted toward the longer wavelengths.

[2] Solid angle is measured in steradians, a sphere is 4π sr.

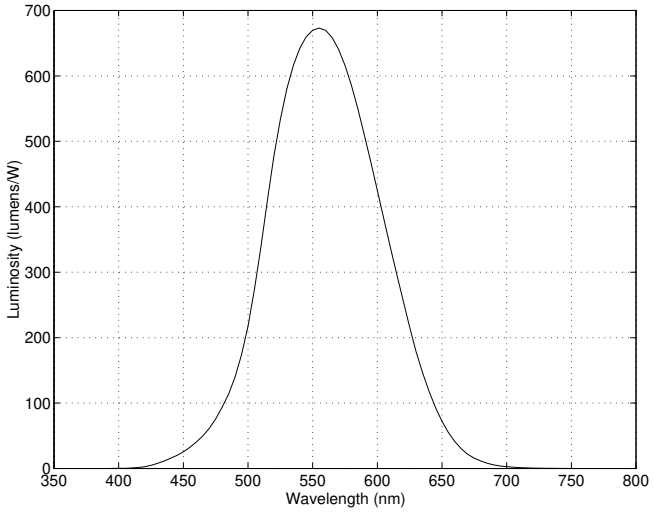


Figure 3.2: CIE luminosity curve for the 'standard observer'. Peak luminosity is 673 lumens/W at a wavelength of 555 nm (green).

Quantity	Unit	Symbol	Equivalent units
Luminous flux	lumen	lm	
Solid angle	steradian	sr	
Luminous intensity	candela	cd	lm/sr
Illuminance	lux	lx	lm/m^2
Luminance	nit	nt	lm/m^2/sr

Table 3.1: Common SI-based photometric units.

where E_i is the incident illuminance at an angle θ to the surface normal.

3.1.2 Surface reflectance

Surfaces reflect light in different ways according to surface texture and wavelength — the two extremes are specular ('glossy') and diffuse ('matte') reflection as shown in Figure 3.3. A specular, or mirror like, surface reflects a ray of light at an angle, θ_r, equal to the angle of incidence, θ_i. A diffuse surface scatters incident light in all directions. A Lambertian surface is a perfectly diffusing surface with a matte appearance

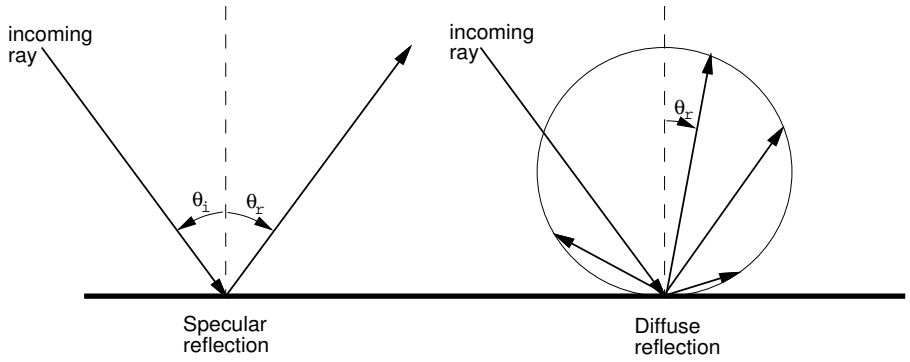


Figure 3.3: Specular and diffuse surface reflectance. Length of the ray is proportional to luminous intensity.

and has consistent luminance irrespective of viewing angle[3]. The luminous intensity of a surface point is

$$I = rE cos\theta_r \,\mathrm{cd} \tag{3.3}$$

where E is the illuminance, and r the surface reflectivity ($0 \leq r \leq 1$). Typical reflectivities are; white paper 0.68, photographic grey card 0.18, and dark velvet 0.004. The *luminance* or 'brightness' of the Lambertian surface is

$$L = \frac{r}{\pi} E \,\mathrm{nt} \tag{3.4}$$

which is independent of θ_i and θ_r. In practice, real surfaces are a combination of these extremes and light is reflected in a lobe centered about the angle $\theta_r = \theta_i$.

3.1.3 Spectral characteristics and color temperature

Many illumination sources, such as incandescent lamps and the sun, have radiation spectra that closely approximate a blackbody radiator [4] at a temperature known as the *color temperature* of the source. The color temperature of solar radiation is 6500 K, and a standard tungsten lamp is 2585 K.

Figure 3.4 compares solar and tungsten spectra and also shows the spectral response of the human eye and a typical silicon CCD sensor. The peak for a tungsten lamp is in the infra-red and this radiation serves to heat rather than illuminate the

[3] Luminous intensity decreases with angle from the normal, but so too does the apparant area of the light-emitting surface patch.

[4] The solar spectrum at ground level is substantially modified by atmospheric absorption.

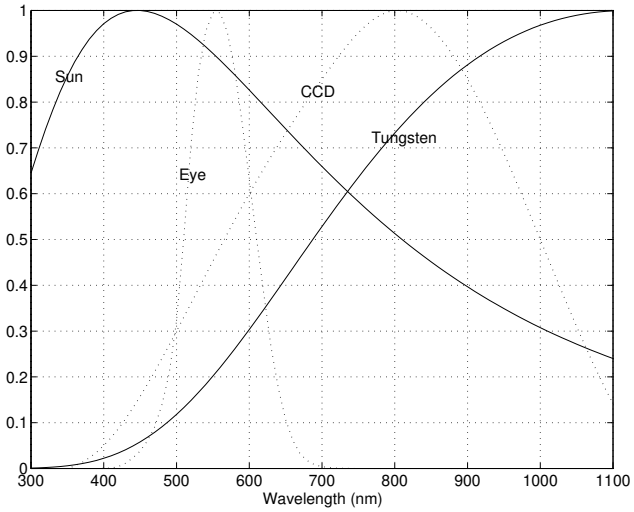


Figure 3.4: Blackbody emissions for solar and tungsten illumination. The photopic response of the human eye and that of a typical silicon CCD sensor are shown for comparison purposes.

Symbol	Name	Value
C_1		$3.741844 \times 10^{-16}\mathrm{W.m^2}$
C_2		$1.438835 \times 10^{-2}\mathrm{m.K}$
σ	Stefan-Boltzmann constant	$5.669572 \times 10^{-8}\mathrm{W/m^2.K^4}$
h	Planck's constant	$6.626197 \times 10^{-34}\mathrm{J.s}$
k	Boltzmann's constant	$1.38062 \times 10^{-23}\mathrm{J/K}$
c	Speed of light in vacuum	$2.99792459 \times 10^{8}\mathrm{m/s}$

Table 3.2: Relevant physical constants.

subject — only a small fraction of the radiation is emitted in the visible spectrum. Radiation from fluorescent tubes cannot be approximated by a blackbody curve, since the light is due to fluorescence of various phosphors, each with a narrow spectral emission.

The radiation spectrum of a blackbody at temperature T as a function of wave-

length, λ, is given by Planck's radiation formula

$$\mathtt{M}\ (\lambda) = \frac{2\pi hc^2}{\lambda^5(e^{hc/k\lambda T} - 1)}$$

where h is Planck's constant, k is Boltzmann's constant, and c is the speed of light. The equation is frequently written in the form

$$\mathtt{M}\ (\lambda) = \frac{C_1}{\lambda^5(e^{C_2/\lambda T} - 1)}$$

where C_1 and C_2 are constants given in Table 3.2. The units of $\mathtt{M}$ are $\mathrm{W/m^3}$ which is interpreted as watts emitted per unit area per unit wavelength. The integral of this function is the total power radiated per unit area

$$\mathtt{M}_{tot} = \int_0^\infty \mathtt{M}\ (\lambda) d\lambda = \sigma T^4\ \mathrm{W/m^2}$$

where σ is the Stefan-Boltzmann constant. The wavelength corresponding to peak emission is given by Wien's displacement law

$$\lambda_{max} = \frac{0.0028978}{T}\ \mathrm{m}$$

The SI values of the relevant physical constants are summarized in Table 3.2.

For a given radiation spectrum, $\mathtt{M}\ (\lambda)$, the corresponding luminous flux can be obtained by

$$\phi_{tot} = \int_0^\infty \mathtt{M}\ (\lambda) K(\lambda) d\lambda$$

where $K(\lambda)$ is luminosity in lumens per watt. The energy of a photon is given by Planck's equation

$$E_{phot}(\lambda) = \frac{hc}{\lambda}\ \mathrm{J}$$

and thus the photon-flux density is

$$n_{phot} = \int_0^\infty \frac{\mathtt{M}\ (\lambda)}{E_{phot}(\lambda)} d\lambda \tag{3.5}$$

$$= \int_0^\infty \frac{\lambda \mathtt{M}\ (\lambda)}{hc} d\lambda\ \mathrm{photons/m^2} \tag{3.6}$$

Thus the number of photons per lumen for a given radiation spectrum is

$$\frac{\displaystyle\int_0^\infty \frac{\lambda \mathtt{M}\ (\lambda)}{hc} d\lambda}{\displaystyle\int_0^\infty \mathtt{M}\ (\lambda) K(\lambda) d\lambda}\ \mathrm{photons/lm}$$

Source	Photons/lumen
Tungsten lamp at 2885 K	2.0×10^{16}
Tungsten lamp at 3200 K	1.8×10^{16}
Tungsten lamp at 5600 K	1.4×10^{16}
Red LED at 670 nm	1.6×10^{17}

Table 3.3: Photons per lumen for some typical illuminants.

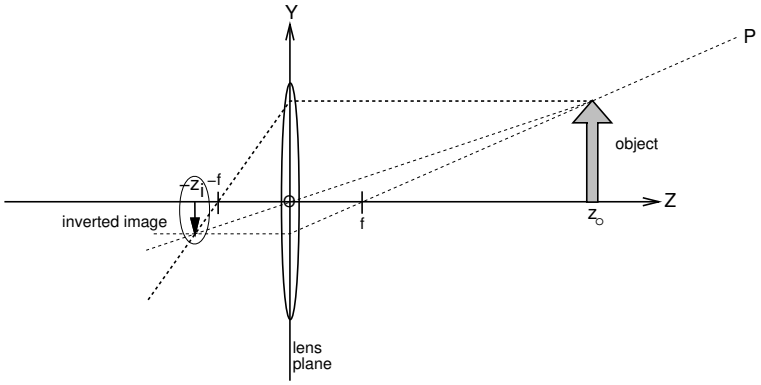


Figure 3.5: Elementary image formation, showing an inverted real image of the object.

Results for a number of standard illuminants are summarized in Table 3.3. These values can be useful in analyzing the sensitivity of CCD sensors which are typically quoted in terms of electrons per lux for a nominated standard illuminant. Knowing the charge well capacity of the photosite, in electrons, allows us to estimate the luminous flux over the photosite required for saturation.

CCD sensors are sensitive to infra-red radiation but sensitivity expressed in electrons/lux cannot quantify this since electrons are generated by infra-red photons which do not contribute to luminous flux. Infra-red filters are frequently fitted to CCD cameras to prevent saturation from infra-red radiation, particularly when working with tungsten lamps. Infra-red radiation reduces image clarity since the longer wavelengths are focussed differently by the lens and such photons cause higher pixel cross-talk in the sensor, see Section 3.3.

3.2 Image formation

The elementary aspects of image formation with a simple lens are shown in Figure 3.5. The positive Z-axis is the camera's *optical axis*. The variables are related by the

lens law

$$\frac{1}{z_o}+\frac{1}{z_i}=\frac{1}{f} \tag{3.7}$$

where z_o is the distance to the object, z_i the distance to the image, and f the focal length. For $z_o > f$ a real inverted image is formed at $z_i > f$ ($\lim_{z_o\to\infty} z_i = f$). For an object at infinity the film or solid state sensor is placed on the focal plane at $z = -f$. For near objects the image is formed behind the focal plane, so the lens must be moved out of the camera in order to focus the image on the film or sensor.

The image height, y_i, is related to the object height, y_o, by the magnification

$$M=\frac{y_i}{y_o}=\frac{f}{f-z_o}\approx -\frac{f}{z_o} \tag{3.8}$$

which is negative, since $z_o > f$, representing the image inversion. Note that magnification decreases as the object distance, z_o, increases.

The f-number of a lens is the dimensionless quantity

$$F = f/d \tag{3.9}$$

where d is the diameter of the lens. f-number is inversely related to the light gathering ability of the lens. To reduce light falling on the image plane, the effective diameter may be reduced by a mechanical aperture or iris, which increases the f-number. Illuminance at the image plane is reduced by F^2 since it depends on light gathering area — to increase illuminance by a factor of 2, the f-number must be reduced by a factor of $\sqrt{2}$ or 'one stop'. The minimum f-number is marked on a lens, and is related to its light gathering capability. The f-number graduations on the aperture control increase by a factor of $\sqrt{2}$ at each stop. An f-number is conventionally written in the form f/1.4 for $F = 1.4$.

The horizontal and vertical subtended angles of the cone of view, or angles of view, are given by

$$\theta_H = 2\tan^{-1}\frac{W}{2f} \tag{3.10}$$

$$\theta_V = 2\tan^{-1}\frac{H}{2f} \tag{3.11}$$

where W and H are respectively the horizontal and vertical dimensions of the camera's active sensing area. Standard 35 mm film is 24 mm × 36 mm, whereas a CCD sensor is around 10 mm × 10 mm. Table 3.4 compares the angles of view for different focal length lenses and sensor sizes. Clearly for the relatively small CCD sensor, the field of view is much narrower compared to film. Frequently the *semi-angles of view* are given which are half the angles of view given above. The angle of view is a maximum when the lens is focussed at infinity.

f (mm)	Pulnix CCD		35 mm film	
	θ_H	θ_V	θ_H	θ_V
8	44°	33°	132°	113°
25	15°	11°	72°	51°

Table 3.4: Angles of view for Pulnix CCD sensor and 35 mm film. Computed for various lenses and with CCD dimensions taken from Table 3.8

In practice, compound lenses are used, comprising a number of simple lens elements of different shapes fabricated from different glasses. This reduces the size of the lens and minimizes aberrations, but at the expense of increased light loss due to reflection at each optical interface.

3.2.1 Light gathering and metering

The illuminance on the image plane of a camera, E_i, is given by

$$E_i \approx \frac{\pi L T \cos^4\theta}{4F^2(M+1)^2} \text{lx} \tag{3.12}$$

where L is the scene luminance, T the transmission efficiency of the lens, M the magnification, and θ is the angle from the optical axis to the image plane point[5]. The $\cos^4\theta$ term models the fall off in illuminance away from the optical axis of the lens. For large object distance $(M+1)^2 \approx 1$ and the illuminance at the sensor is independent of object distance[6].

A lightmeter is a device with fixed optics that measures the radiant flux falling on a sensor of known area A. By suitable choice of spectral characteristic the sensor output can be interpreted as luminous flux, ϕ. Photographers employ two methods of light measurement:

1. *Incident light measurements* measure the illuminance of the scene, and are taken by placing the meter in front of the subject aimed at the camera. The illuminance is given by
$$E_s = \frac{\phi}{A} \tag{3.13}$$
The luminance of the scene is computed using (3.4) and an assumed reflectivity of 18%[7]
$$L_s = \frac{0.18}{\pi}\frac{\phi}{A} \tag{3.14}$$

[5] This effect is pronounced in short focal lenses since $max(\theta) = W/2f$ where W is the sensor width.

[6] This may seem counter intuitive, but as object distance increases the image size is reduced as is the spatial integral of luminous flux.

[7] $18\% = 2^{-2.5}$ and corresponds to the geometric mean of the 5 stop reflectance range of 'typical' scenes.

2. *Reflected light measurements* measure the luminance of the scene directly, and are taken by placing the meter near the camera aimed at the scene. The illumination measured by the sensor (3.13) is related to the scene luminance by (3.12) so that
$$L_s = \frac{4F^2}{\pi T}\frac{\phi}{A} \tag{3.15}$$
where F and T are known characteristics of the lightmeter's optics.

The two types of measurement are simply different scalings of the lightmeter sensor output. A lightmeter generally returns the logarithm of luminance called exposure value, or EV, which is related to luminance by

$$L = k2^{EV} \tag{3.16}$$

The calculation dials on a lightmeter are a circular sliderule implementing equation (3.12) which relates exposure time to aperture for a given scene luminance and film sensitivity or 'speed'. *Exposure* of the film or sensor is

$$e = E_i T_e \tag{3.17}$$

in units of lux.s, where E_i is the illuminance and T_e is the exposure time. The ISO film speed, S, is determined from the exposure necessary to "discernibly fog" the film

$$S = \frac{0.8}{e} \tag{3.18}$$

The factor k in (3.16) is typically around 0.14 and results in the film exposure being 'centered' within its dynamic range[8].

An alternative approach, used by photographers, is to use a *spot-reading lightmeter* to measure the brightest and darkest regions of the scene separately, and then manually compute the required exposure.

3.2.2 Focus and depth of field

Maintaining focus over a wide range of camera-object distances is a non-trivial problem when the camera is mounted on the end of a robot. The options are to establish a large depth of field or use a lens with servo controlled focus. The latter approach has a number of disadvantages since such lenses are generally heavy and bulky, have limited adjustment rates, relative rather than absolute focus setting, and target distance must be somehow determined.

[8]The exposure computed on the basis of film speed is only sufficient to "discernibly fog" the film. A good photograph requires a higher exposure, which is achieved by calculations based on a fraction of the real luminance.

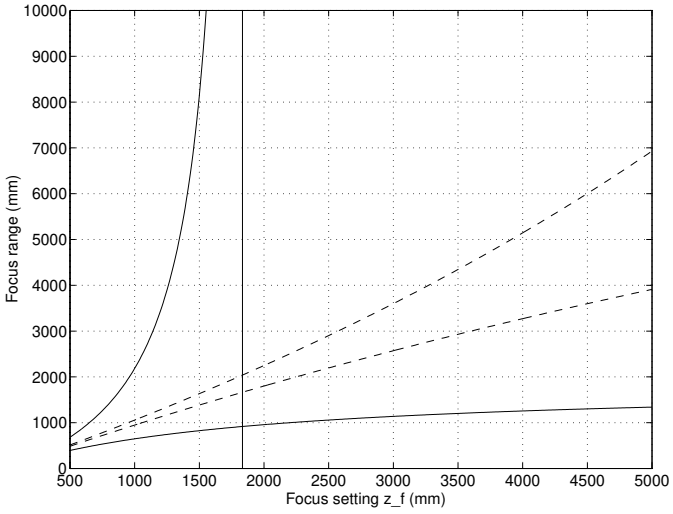


Figure 3.6: Depth of field bounds for 8 mm (solid) and 25 mm lenses (dashed) at f/1.4. Circle of confusion is 25 μm ($\approx$ 3 pixels) in diameter. The vertical line at $z_f = 1820$ mm is the singularity in z_F at the hyperfocal distance.

Depth of field is the range in object distance over which the lens can form an image without significant deterioration. This is defined such that the image of a point source object is a circle of diameter a or smaller — the so called *circle of confusion*. The bounds of the acceptable focus range are given by

$$z_N = z_f - \frac{z_f}{\frac{Mf}{aF} + 1} \tag{3.19}$$

$$z_F = z_f + \frac{z_f}{\frac{Mf}{aF} - 1} \tag{3.20}$$

where z_N and z_F are the near and far bounds of the acceptable focus range, and z_f is the focus setting of the lens. The locus of near and far bounds for a range of focus settings is shown in Figure 3.6. When the lens is focussed at the *hyperfocal* distance

$$z_f = f\left(1 - \frac{f}{aF}\right) \tag{3.21}$$

the depth of field extends from half the hyperfocal distance to infinity. Rearranging (3.19) and (3.20) gives the focus and aperture setting for a specified focus range

$$z_f = \frac{2z_N z_F}{z_N - z_F} \tag{3.22}$$

$$F = \frac{f^2(z_F - z_N)}{2az_N z_F} \tag{3.23}$$

The f^2 term in (3.23) means that longer focal length lenses will require a very large f-number in order to maintain a given depth of field, and will thus have very little illumination on the image plane.

For 35 mm film the circle of confusion diameter is typically taken as 25μm [250], and for a CCD array is the pixel-to-pixel spacing [82] (see Table 3.8). Large depth of field is achieved by using a small aperture, but at the expense of light falling on the sensor. The requirement for large depth of field and short exposure time to eliminate motion blur both call for increased ambient illumination, or a highly sensitive image sensor.

3.2.3 Image quality

In practice the image formed by a lens is not ideal. Lenses used for imaging are usually compound lenses containing several simple lenses in order to achieve a compact optical system. Imperfections in the shape or alignment of the simple lenses leads to degraded image quality. Non-idealities include aberrations which lead to image blur, and geometric distortions which cause the image to fall in the wrong place. It is important to match the lens with the sensor size used[9] , since lenses are designed to minimize effects such as geometric distortion and vignetting only over the sensor or film area.

3.2.3.1 Aberrations and MTF

Aberrations reduce image clarity by introducing blur. This leads to reduced contrast on fine image detail. Common aberrations include [28]:

- *Spherical aberration*, *coma*, *astigmatism* and *field curvature* which introduce variations in focus across the scene.
- Chromatic aberrations due to different wavelengths of light being focussed at different distances from the lens.
- Vignetting, where image brightness falls off at the edges of the field of view due to the effect of the lens barrel.

[9] Lenses are typically manufactured for 1/2, 2/3 and 1 inch sensors.

- Diffraction due to the aperture, which will reduce the definition of the image. The image of a point source will be an Airy pattern[10] and the minimum separation at which two point sources can be discerned is

$$d = 2{,}4\lambda F \tag{3.24}$$

  where λ is the wavelength of the illumination (λ is often taken as 555 nm, the eye's peak sensitivity). Diffraction effects are thus most pronounced at small aperture or high f-number.

The effect of aberrations can be reduced by closing the aperture since this restricts the light rays to a small central region of each optical element. However this increases the magnitude of diffraction effects. It is generally desirable to operate between these two extremes, where the magnitudes of the two effects are approximately equal.

In order to quantify image sharpness, the relationship between contrast and resolution is important. This can be considered as the magnitude of a spatial transfer function which is generally referred to as the *modulation transfer function* or MTF. Aberrations result in MTF roll-off at high spatial frequency. MTF is normally expressed as a percentage of the uniform illumination response (or DC gain). The MTF is the magnitude of the normalized Fourier transform of the *line spread function*, the distribution of intensity along a line orthogonal to the image of a line of negligible width. A related measurement derived from the spatial square wave response is the *contrast transfer function* or CTF. Useful spatial resolution is typically given in lines, determined as the highest distinguishable line pitch in a resolution test chart. The spatial resolution of the complete imaging system is obtained by multiplying the MTF due to the lens, sensor and and analog electronics, and is discussed further in Sections 3.3.2 and 3.5.5.

The MTF at the spatial Nyquist frequency is related to the rise distance[11] of an image of a sharp edge by

$$\text{MTF}_{f_s/2} = \frac{1}{N} \tag{3.25}$$

where N is the numbers of pixels for the image intensity to rise from 10% to 90% [121].

3.2.3.2 Geometric distortion

Unlike aberration, geometric distortion leaves the image sharp but causes it to fall in the wrong place [28]. Geometric distortions are classified as *radial* or *tangential*.

[10] A finite sized disk with faint concentric circular rings, a 2D Sinc pattern.

[11] The spatial analog of rise time.

Radial distortion causes image points to be translated along radial lines from the principal point[12] (outward distortion is considered positive). Tangential distortion occurs at right angles to the radii, but is generally of less significance than radial distortion.

Radial distortion may be approximated by a polynomial [282]

$$\Delta r = k_1 r^3 + k_2 r^5 + k_3 r^7 + \ldots \tag{3.26}$$

where r is the distance between the image point and the principal point. The corrected, or true, image point radius is

$$r' = r - \Delta r \tag{3.27}$$

For Cartesian image plane coordinates, $({}^i x, {}^i y)$, with their origin at the principal point the distortion may be written

$$\Delta {}^i x = \Delta r \frac{{}^i x}{r} = {}^i x (k_1 r^2 + k_2 r^4 \ldots) \tag{3.28}$$

$$\Delta {}^i y = \Delta r \frac{{}^i y}{r} = {}^i y (k_1 r^2 + k_2 r^4 \ldots) \tag{3.29}$$

The polynomial coefficients, k_i, would be determined by a calibration procedure such as described by Tsai [252] or Wong [283]. Andersson [17] mapped the distortion vector at a number of points in the image in order to determine the polynomial coefficients. Geometric distortion is generally worse for short focal length lenses.

3.2.4 Perspective transform

The simple lens from Figure 3.5 performs a mapping from 3D space to the 2D image plane. Using similar triangles it can be shown that the coordinates of a point on the image plane $({}^i x, {}^i y)$ are related to the world coordinates (x, y, z) by:

$${}^i x = \frac{fx}{f - z} \tag{3.30}$$

$${}^i y = \frac{fy}{f - z} \tag{3.31}$$

$$\tag{3.32}$$

which is the *projective-perspective* transform from the world to the image plane. Such a transform has the following characteristics:

- World lines are mapped to lines on the image plane.
- Parallel world lines, not in the plane orthogonal to the optical axis, are projected to lines that intersect at a vanishing point.

[12] The point where the camera's optical axis intersects the image plane.

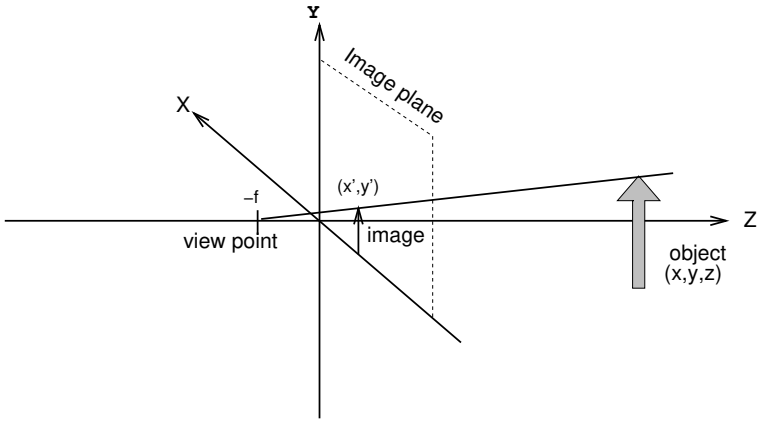


Figure 3.7: Central perspective geometry.

- Conics in world space are projected to conics on the image plane, for example, a circle is projected as a circle or an ellipse.
- The mapping is not one-to-one, and a unique inverse does not exist. In general the location of an object point cannot be determined uniquely by its image. All that can be said is that the point lies somewhere along the projecting ray OP shown in Figure 3.5. Other information, such as a different view, or knowledge of some physical constraint on the object point (for example we may know that the object lies on the floor) is required in order to fully determine the object's location in 3D space.

In graphics texts the perspective transform is often shown as in Figure 3.7, where a non-inverted image is formed on the image plane at $z = 0$, from a viewpoint at $z = -f$. This is also referred to as *central projection* [80]. Such a transform lacks two important characteristics of the lens; image inversion, and a singularity at $z = f$.

3.3 Camera and sensor technologies

Early computer vision work was based on vidicon, or thermionic tube, image sensors. These devices were large and heavy, lacked robustness, and suffered from poor image stability and memory effect [68]. Since the mid 1980s most researchers have used some form of solid-state camera based on an NMOS, CCD or CID sensor.

Most visual servo work has been based on monochrome sensors, but color has been used for example in a fruit picking robot [108] to differentiate fruit from leaves. Given real-time constraints the advantages of color vision for object recognition may

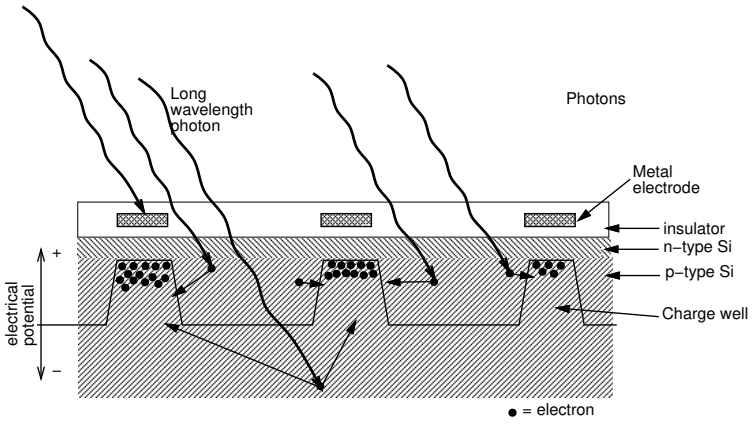


Figure 3.8: Notional depiction of CCD photosite charge wells and incident photons. Silicon is more transparent at long wavelengths and such photons will generate electrons deeper within the substrate. These may diffuse to charge wells some distance away resulting in poor resolution at long wavelength.

be offset by the increased cost and high processing requirements of up to three times the monochrome data rate.

3.3.1 Sensors

Solid state sensors comprise a number of discrete *photosites* or *pixels* — each site accumulates a charge proportional to the illumination of the photosite integrated over the exposure period. Line scan sensors are a 1D array of photosites, and are used in applications where linear motion of a part can be used to build up a 2D image over time. Area sensors are a 2D array of photosites, and are the basis of cameras for television and most machine vision work. The following discussion will be limited to area sensors although line-scan sensors have been used for visual servoing [271].

The most common types of solid-state area-imaging sensor are:

1. CCD (Charge Coupled Device). A CCD sensor consists of a rectangular array of photosites each of which accumulates a charge related to the time integrated incident illumination over the photosite. Incident photons generate electron-hole pairs in the semiconductor material, and one of these charges, generally the electron, is captured by an electric field in a *charge well* as depicted in Figure 3.8. Charge packets are moved about the chip using multi-phase clock voltages applied to various gate electrodes. Such a transport mechanism shifts the charge

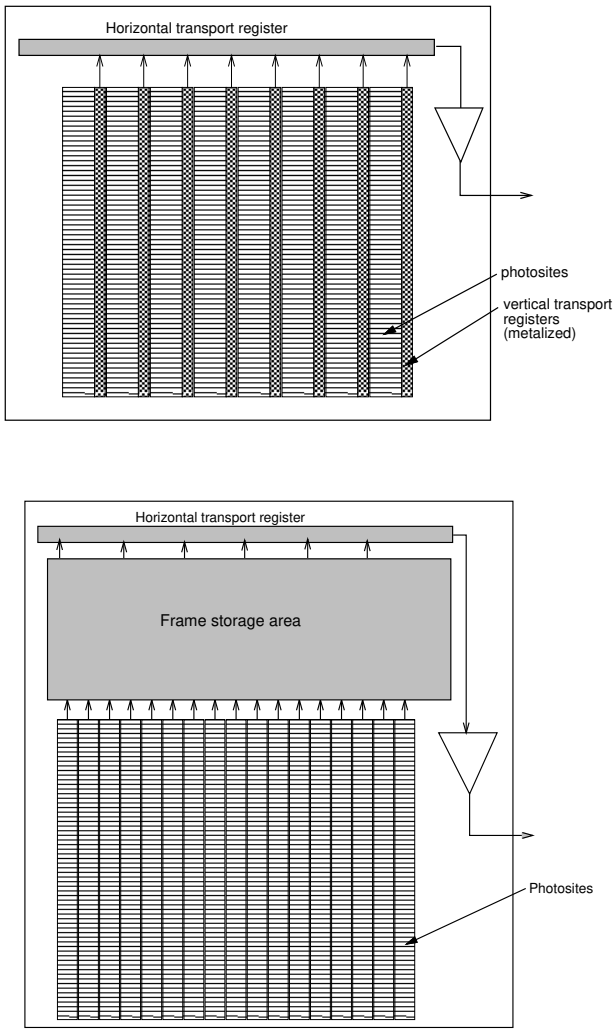


Figure 3.9: CCD sensor architectures. Top is interline transfer, bottom is frame transfer.

from the photosite to the output amplifier. The two common architectures for CCD sensors, shown in Figure 3.9, are:

(a) Interline transfer. At the end of each field time the odd or even photosites transfer their charge to vertical transport registers, and are themselves discharged. During the next field time, the vertical transport registers shift

one line at a time into the horizontal transport register from where it is shifted out at pixel rate to the amplifier.

(b) Frame transfer. At the end of each field time the odd or even photosites are shifted into the vertical transport registers and are themselves discharged. The charge is then rapidly shifted, typically at 100 times the line rate, into the storage area. During the next field, the lines are sequentially loaded into the horizontal transport register where they are shifted out at pixel rate to the amplifier.

The common features of CCD sensors are that the photosites are sampled simultaneously, and are read destructively. Under high illumination the charge wells can overflow into adjacent photosites leading to *blooming*. The sensor manufacturer will quote the capacity of the charge well in terms of electrons, commonly 10^5 to 10^6 electrons. For interline transfer devices the vertical transport registers are covered by metalization to minimize the effect of light on the stored charge, but this reduces the effective sensitive area of the device. As shown in Figure 3.8 infra-red photons penetrate a considerable distance [121] into the substrate and the electrons generated may be collected by charge wells some distance away. This introduces cross-talk between pixels, where the pixel values are not truly independent spatial samples of incident illumination.

2. NMOS, or photodiode array. Each site contains a photodiode whose junction capacitance is precharged and which photoelectrons cause to discharge. Each site is read to determine the remaining charge after which the capacitor is recharged by 'charge injection'. While conceptually capable of random access to pixels, the sensor devices typically have counters to selectively output pixels in raster scan order.

3. CID (Charge Injection Device). A CID sensor is very similar to the NMOS sensor except that the charge at the photosite can be read non-destructively. Charge injection clears the accumulated charge, and may be inhibited, allowing for some control over exposure time. While conceptually capable of random access to pixels, the sensor devices typically have counters to selectively output pixels in raster scan order.

The most significant difference between the CCD and NMOS sensors is that the CCD sensor samples all photosites simultaneously, when the photosite charge is transferred to the transport registers. With the other sensor types pixels are exposed over the field-time prior to their being read out. This means that a pixel at the top of the frame is exposed over a substantially different time interval to a pixel in the middle of the frame, see Figure 3.10. This can present a problem in scenes with rapidly moving targets as discussed by Andersson [17]. Andersen et al. [13] discuss the analogous

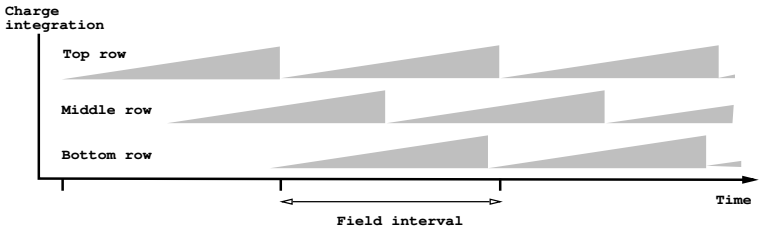


Figure 3.10: Pixel exposure intervals for NMOS or CID sensor as a function of pixel row.

sampling problem for a vidicon image sensor, and propose a modified Z-transform approach.

3.3.2 Spatial sampling

The original image function $I(x,y)$ is sampled by the discrete CCD photosites to form the signal $I^*(i,j)$, where i and j are the pixel coordinates. The Nyquist period is 2 pixels, so fine image detail, that with a period of less than 2 pixels, will result in aliasing which is manifested visually as a Moiré fringing effect.

An ideal spatial sampler would require an infinitesimally small photosite that would gather no light. In reality photosites have significant width and exhibit cross-talk due to capture of photoelectrons from adjacent areas of the substrate. The spatial frequency response of the photosite array is a function of the photosite capture profile. This issue is touched on in a qualitative way by Purll [28], and some sensor manufacturers provide experimental results on spatial frequency response, or sensor MTF.

Consider a one dimensional sinusoidal illumination pattern as shown in Figure 3.11

$$I(x) = a\sin(\omega x + \phi) + b \tag{3.33}$$

with a spatial frequency of ω and arbitrary phase ϕ and offset b so that illumination $I(x) \geq 0, \forall x$. For pixels with active sensing width h and a spacing p, the response of the k^{th} pixel

$$I^*(k) = \frac{1}{h}\int_{kp}^{kp+h} I(x)\,dx, \tag{3.34}$$

$$= \frac{a}{\omega h}\{\cos(\omega kp + \phi) - \cos(\omega(kp+h) + \phi)\} + b, \tag{3.35}$$

is normalized so that response magnitude is independent of h. The camera output as a

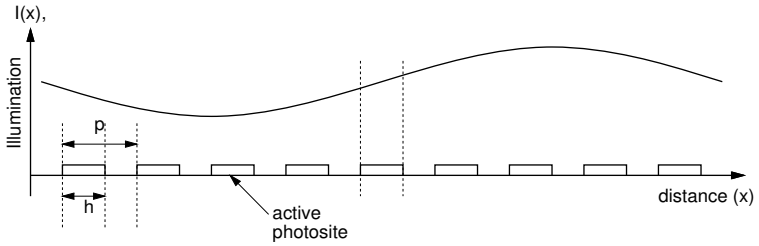


Figure 3.11: Camera spatial sampling in one dimension. h is the width of the photosite and p is the pixel pitch. Each pixel responds to the spatial integral of illumination.

function of distance, x, is piecewise constant

$$I^*(x) = \sum_{k=-\infty}^{\infty} I^*(k)\left\{H(x-kp) - H\left(x-(k+1)p\right)\right\} \tag{3.36}$$

where $H(x)$ is the Heaviside unit-step function. This function is periodic and can be represented by a Fourier series

$$I^*(x) = \sum_{n=-\infty}^{\infty} \left\{c(n\omega)e^{jn\omega x}\right\} \tag{3.37}$$

where $c(n\omega)$ is the n^{th} Fourier coefficient given by

$$c(n\omega) = \frac{1}{T}\int_T I^*(x)e^{-jn\omega x}dx \tag{3.38}$$

and $\int_T$ is the integral over one period $T = 2\pi/\omega$.

MTF is the ratio of the magnitude of the fundamental component of the camera output to the magnitude of the camera input

$$MTF = \frac{|c(\omega)|}{a} \tag{3.39}$$

which can be shown to be

$$MTF = \frac{2}{\omega h}\sin(\frac{wh}{2})\ \frac{2}{w}\sin\frac{\omega p}{2} \tag{3.40}$$

Substituting $f_p = 1/p$, $f = \omega/2\pi$, $h' = h/p$, and the normalized sampling frequency $f' = f/f_p$ leads to

$$MTF = p\,\text{sinc}(\pi f' h')\,\text{sinc}(\pi f') \tag{3.41}$$

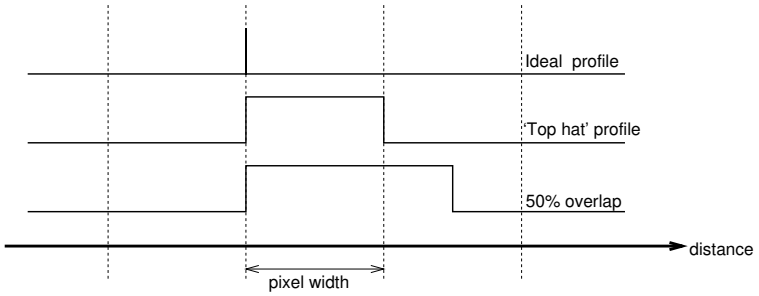


Figure 3.12: Some photosite capture profiles.

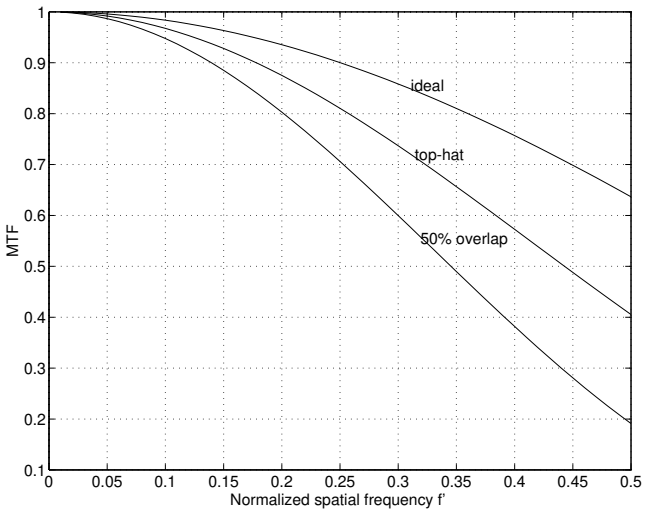


Figure 3.13: MTF for the case of ideal sampler, $h' = 0$, full width sampler, $h' = 1$, and 50% pixel overlap, $h' = 1.5$.

For an ideal sampler, that is, $h' = 0$, the MTF is given by

$$MTF|_{h'=0} = p\,\mathrm{sinc}(\pi f') \tag{3.42}$$

which is the expected frequency response for a sampler followed by a zero-order hold. A sampler that integrates over the entire pixel width will have an MTF given by

$$MTF|_{h'=1} = p\,\mathrm{sinc}^2(\pi f') \tag{3.43}$$

For the pixel capture profiles shown in Figure 3.12, the corresponding spatial frequency responses are given in Figure 3.13. These indicates that cross-talk and finite pixel width cause the spatial frequency response to roll off more sharply at high spatial frequency. The vertical lines of metalization in an interline transfer sensor reduce the active width of the photosite, and should improve the MTF in the horizontal direction.

3.3.3 CCD exposure control and motion blur

In the discussion so far it is has been assumed that the photosites are being charged, or integrating, for one whole field time. High-speed relative motion between the camera and scene results in a blurred image, since the photosites respond to the integral of illumination over the exposure period. A blurred object will appear elongated in the direction of motion[13].

In the case where an object moves more than its width during the exposure interval the illumination will be spread over a greater number of pixels and each will 'see' less light. That is, as the image blurs, it elongates and becomes dimmer. A machine vision system which uses thresholding to differentiate a bright object from its background can thus 'lose sight' of the object. In a visual servoing system this can lead to 'rough' motion [65].

A conventional film camera uses a mechanical shutter to expose the film for a very short period of time relative to the scene dynamics. Electronic shuttering is achieved on CCD sensors by discharging the photosites until shortly before the end of field by means of the anti-blooming or integration control gate. Only the charge integrated over the short remaining period, T_e, is transferred to the transport registers. The accumulated charge is reduced proportionally with the exposure time which reduces the signal to noise ratio of the image. This effect can be countered by opening the camera's aperture or providing additional scene illumination.

The timing of the Pulnix camera's integration period has been determined experimentally by viewing an LED emitting a very short light pulse at a time that is adjustable relative to the vertical synchronization pulse. As shown in Figure 3.14 the integration period always ends 1 ms after the onset of vertical blanking.

Consider the one dimensional case of a moving object whose centroid location on the image plane is ${}^iX(t)$. The CCD sensor responds to the integral of the incident illumination, so the perceived centroid will be the mean over the integration period

$$ {}^i\overline{X} = \frac{1}{T_e} \int_{-T_e}^{0} {}^iX(t)dt \tag{3.44} $$

[13] It may be possible to determine the velocity of a known symmetric object from the shape of its blurred image.

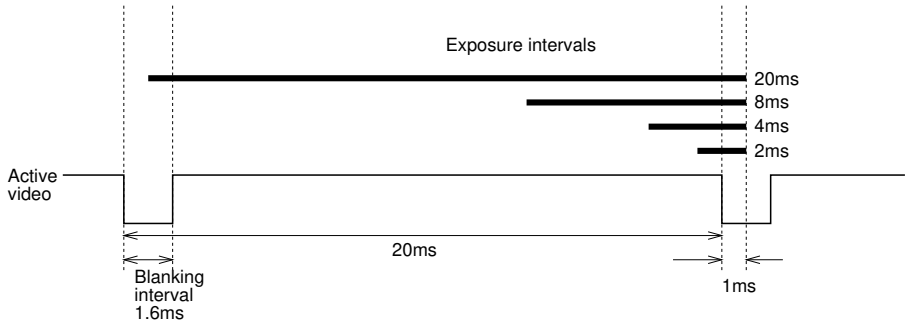


Figure 3.14: Experimentally determined exposure interval of the Pulnix camera with respect to vertical active timing.

If the target is moving at a constant velocity ${}^i\dot{X}$ the perceived centroid will be

$$
\begin{aligned}
{}^i\overline{X} &= \frac{1}{T_e}\int_{-T_e}^{0} \left({}^iX(0) + {}^i\dot{X}t\right) dt && (3.45)\\
&= {}^iX(0) - {}^i\dot{X}\frac{T_e}{2} && (3.46)\\
&= {}^iX(-T_e/2) && (3.47)
\end{aligned}
$$

which lags the actual centroid by half the exposure interval. Thus for a finite exposure interval, the actual exposure time should be taken as halfway through the exposure period.

3.3.4 Linearity

The luminance response of a CRT monitor

$$L_{CRT} = v^{\gamma_{CRT}} \tag{3.48}$$

is highly non-linear, where v is the input signal and γ_{CRT} is a value typically in the range 2.2 to 2.8. The camera is normally adjusted for the inverse response

$$v = (L_{cam})^{\gamma_{cam}} \tag{3.49}$$

where L_{cam} is the luminance of the observed scene and γ_{cam} is chosen as 0.45 to correct for the non-linearity of the CRT and render an image with correct contrast. Early vacuum tube sensors such as iconoscopes in fact had the appropriate non-linear

characteristic. Linear sensors such as the orthicon tube and CCDs require a non-linear amplifier[14].

The transfer function, or linearity of the camera's illumination response, is often referred to as *gamma*, and many solid state cameras have a switch to select $\gamma_{cam} = 1$ or 0.45. For machine vision work, where the image is not being displayed on a CRT, a non-linear camera response serves no useful purpose.

3.3.5 Sensitivity

Sensitivity is the DC gain of the sensor to incident illumination. Factors contributing to sensitivity include:

- Quantum efficiency, η_q, the fraction of incident photons converted to electrons. This quantum efficiency is typically around 80% but is a function of the semiconductor material and wavelength of the photon.
- Area efficiency or pixel fill factor is the ratio of photosite active area to total area
$$\eta_f = \frac{p'_x p'_y}{p_x p_y} \tag{3.50}$$
where p'_x and p'_y are the dimensions of the active photosite sensing area. Areas of metalization in each photosite reduce the sensitive area. Frame transfer sensors have a higher area efficiency than interline transfer devices since there is no metalized vertical transport register.
- Charge transfer efficiency. Each time a packet of charge is moved about the substrate, some charge carriers will be lost. Although this fraction is very small, charge packets from distant photosites undergo a greater number of transfers. As well as reducing the apparent intensity of the pixel, the 'lost' charge from bright regions will be picked up by subsequent charge packets, thus reducing contrast in areas of fine detail. Bright regions leaving a trail of charge behind them result in *streaking* of the image.
- The gain of the on-chip charge amplifier.
- The gain of output amplifier within the camera electronics.
- The gain of the AGC (Automatic Gain Control) stage.

[14]Transmission of the non-linear intensity signal has an advantageous side effect in noise reduction. Signals corresponding to low intensity are selectively boosted by the non-linearity prior to transmission and then reduced on display, also reducing the amplitude of additive noise in transmission.

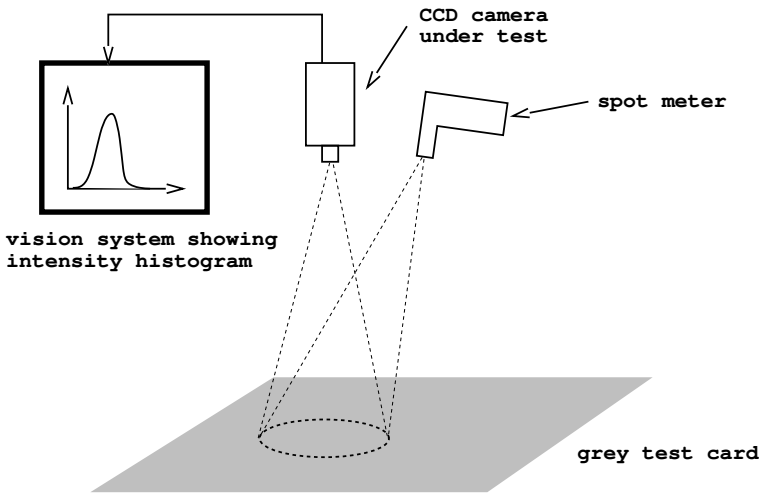


Figure 3.15: Experimental setup to determine camera sensitivity.

f-number	grey-level	sensor luminance (lx)
f/5.6	saturated	2.0
f/8	177	0.98
f/11	120	0.52
f/16	44.2	0.25

Table 3.5: Grey level response of Pulnix TM-6, with 25 mm lens, 20 ms exposure time, no AGC, $\gamma = 1$, luminance 80 cd/m^2.

Sensor sensitivity is typically quoted in units of electrons/lux, and may be found in a data sheet. However the sensitivity or overall gain of a complete camera is rarely given and must be determined experimentally. A suitable experimental setup is shown in Figure 3.15 where the the camera is aimed at a uniformly illuminated grey test card and defocused so as to eliminate the effect of fine texture.

A spot-reading meter is used to determine the luminance of the card, L, and the mean grey-level of a small region in the center of the image, I, is taken as the camera's response. For a range of f-number settings shown in Table 3.5 the response of the Pulnix camera was determined to be

$$I = \frac{11000}{F^2} + 12 \tag{3.51}$$

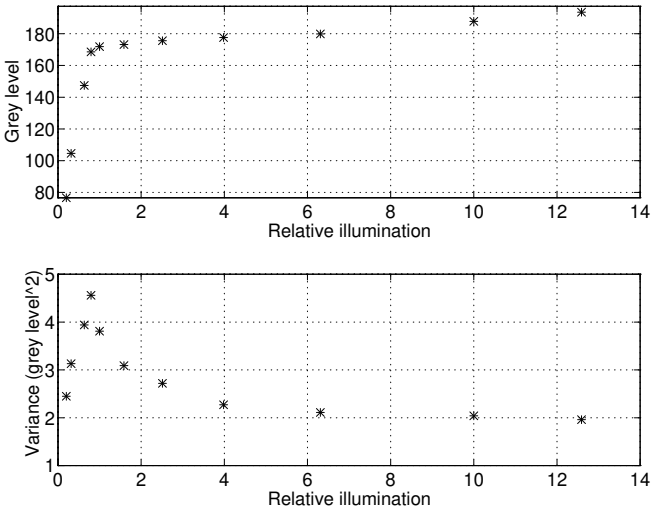


Figure 3.16: Measured response of AGC circuit to changing illumination. Top curve shows mean grey-level response. Bottom curve shows the grey-level spatial variance.

which has the form expected by (3.12) plus a small offset. Luminance L is measured by the spot-reading meter as $80\,\mathrm{cd/m^2}$ and if the intensity gain of the camera is $K_{cam}\,\mathrm{greylevel/lx}$ then from (3.12) we may write

$$11000 = K_{cam}\frac{\pi 80}{4} \tag{3.52}$$

giving a camera gain of

$$K_{cam} = 180\,\mathrm{greylevel/lx} \tag{3.53}$$

This is a lumped gain value which includes lens transmission, sensor response, and camera and digitizer analog gain[15]. Extrapolating from the experimental data, saturation of the digitizer will occur when the illuminance at the sensor exceeds 1.4 lx.

Let us assume that the minimum discernable grey value (due to noise) is 5, which corresponds to an illuminance of $1.4 \times 5/256 = 27 \times 10^{-3}$ lux. Using the film speed relationship (3.18) and an exposure interval of 40 ms we can equate this sensor to an ISO film speed of around 730.

[15] DIGIMAX is set to nominal 0 dB gain, see Section 3.5.2.

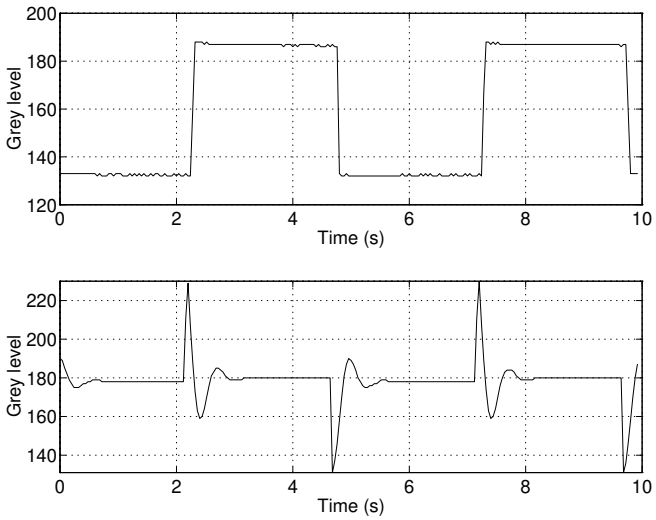


Figure 3.17: Measured response of AGC circuit to step illumination change. Top curve is with AGC disabled, and the bottom with AGC enabled.

Cameras commonly have an automatic gain control (AGC) function which attempts to maintain the 'average' grey-level in the scene. The Pulnix camera has a quoted AGC gain range of 16 dB or 2.7 stops. Several experiments were conducted to investigate the behaviour of the AGC circuit:

1. With constant illumination, images were captured of a grey card. As increasingly large white squares were added to the card the intensity of the grey background area was reduced. The AGC strategy thus appears to be maintenance of the mean grey-level in the scene even if this forces bright scene areas into saturation.

2. With constant illumination and no lens fitted, images were captured with a variety of neutral density filters over the camera. The results plotted in Figure 3.16 show a plateau in the camera response for the range of illuminance levels over which the output is held roughly constant by the AGC.

   At very low illuminance the gain of the AGC is insufficient to maintain the camera output level. In this regime it can be seen that the grey-level spatial variance[16] has increased, as the amplifier boosts the weak signal and accentuates

[16]The variance of pixel values within a 64×64 window centered in the image.

the noise component.

3. The temporal response of the AGC was investigated by measuring the response to an average illumination level which is an offset square wave. The ratio of intensities is approximately 1.5, requiring only a 3 dB gain range which is well within the 16 dB range of the camera's AGC circuit. The response with and without AGC is shown in Figure 3.17. The former exhibits substantial overshoot taking around 1 s to settle. In a normal situation a bright object entering the scene would cause only a moderate change in average illumination and this oscillatory effect would be less marked. An AGC is a non-linear feedback system, and the response will be related to the amplitude of the input.

3.3.6 Dark current

In addition to the photo-generated electrons, a photosite accumulates charge due to *dark current*. Dark current results from random electron-hole pairs generated thermally or by tunnelling and is indistinguishable from photon generated current. A temperature rise of 7 K will double the dark current [83]. To minimize accumulated charge due to dark current, sensors designed for long exposure times are cooled. Short exposure times reduce charge due to both dark current and illumination.

The mean dark current is determined by dark-reference photosites which are generally columns on each side of the array. These are manufactured in the normal manner but are covered by metalization so that only dark current is accumulated. The camera electronics use the signal from these pixels as a black reference, subtracting it from the output of the uncovered pixels.

3.3.7 Noise

A number of factors contribute to noise in a CCD sensor:

- Photon arrival statistics result in *photon shot noise*. Photon arrival is a random process [221] and the probability of detecting a photon in the time interval $[t, t+\delta t]$ is proportional to the radiometric intensity $E_r(t)$ ($\mathrm{W/m^2}$) at time t.

  The photon-number statistics are generally modelled by a Poisson distribution with a mean and variance

$$\overline{n} = \frac{E_r}{h\nu} \tag{3.54}$$

$$\sigma_n^2 = \overline{n} \tag{3.55}$$

  where ν is the photon frequency and h is Planck's constant.

- Photon conversion statistics, or photoelectron noise. A photon incident on a sensor of quantum efficiency η_q will produce an electron with a probability of η_q and fail to do so with a probability $1 - \eta_q$.
- Dark current shot noise, due to the random thermal generation of electrons captured by the photosite.
- Receiver shot and thermal noise introduced during amplification of the accumulated charge. To minimize this, a high-gain low-noise amplifier is generally fabricated on-chip with the CCD sensor.
- Readout noise due to coupling of CCD clocking signals into the sensor output signal.
- Pattern noise or response non-uniformity. This is not a true noise source but reflects a variation in gain between photosites. This effect arises from variations in material characteristics and process variations in chip manufacture. The non-uniformity is highly dependent on color which affects photon penetration of the material and thus the defects encountered.

If x is the camera output signal, the signal to noise ratio

$$\text{SNR} = \frac{\overline{x^2}}{\Sigma\sigma^2} \tag{3.56}$$

is a function of the various noise sources just discussed.

The mean and variance of grey-level within a small window in the center of the image were computed for a range of illumination levels. To achieve uniform illumination of the sensor the lens was removed and the illuminance controlled by means of neutral density filters placed in front of the camera. The variance is plotted as a function of the mean intensity in Figure 3.18. The variance increases linearly with mean camera response as would be expected for photon shot noise by (3.55). The use of short exposure times does not appear to have any significant effect on the output noise level. The line corresponds to a SNR of 36 dB[17].

If the value of each pixel within the sample window is averaged over time the sample mean will approach the expected value given by (3.54), and the spatial variance of the time-averaged pixels would then be due to pixel response non-uniformity. For the brightest sample in Figure 3.18 the spatial variance falls from 3.5 to 1.0 after averaging 100 frames.

[17] The digitizer quantization noise from (3.70) is insignificant compared to the noise measured here.

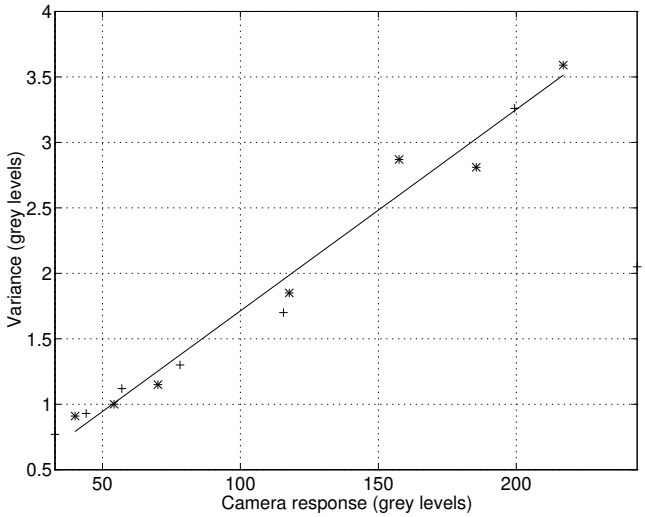


Figure 3.18: Measured spatial variance of illuminance as a function of illuminance. Points marked '*' are measured with a 20 ms exposure time, and '+' with a 2 ms exposure time.

3.3.8 Dynamic range

The *dynamic range* of a CCD sensor is the ratio of the largest output signal to the smallest discernible output. The largest signal, at saturation, is directly related to the capacity of the charge well. At very low illumination levels the response of the sensor is totally overwhelmed by the dark current and noise effects described above. The smallest discernible output is thus the output noise level. For a CCD with, for example, noise of 100 electrons and a charge well of 100,000 electrons the dynamic range is 1000 or nearly 10 bits.

3.4 Video standards

Broadcast and closed-circuit television industries dominate the manufacture and consumption of video equipment, thus most cameras used for machine vision work conform to television standards. The two most widely used standards are:

- RS170 used in the USA and Japan with 525 line frames at 30 frames per second;

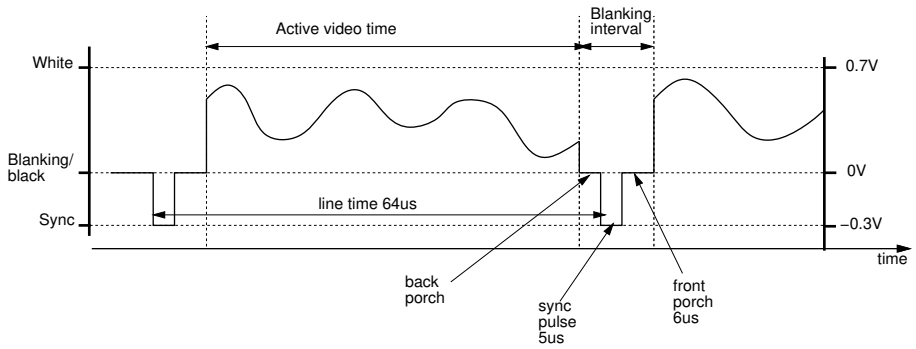


Figure 3.19: CCIR standard video waveform, 1 V peak to peak.

- CCIR used in Europe and Australia with 625 line frames at 25 frames per second.

The requirements of broadcast television are quite different to those for machine vision. Broadcast television requires low transmission bandwidth, ease of decoding in the viewer's set and minimal human perception of flicker. Interlacing, an artifact introduced to address the constraints of broadcast television is particularly problematic for machine vision and is discussed further in Section 3.4.1. The camera frame rate, effectively the sample rate in a visual servo system, is now a significant limiting factor given the current and foreseeable capabilities of image processing hardware. High frame rate and non-interlaced cameras are becoming increasingly available but are expensive due to low demand and require specialized digitization hardware. Introduction is also hampered by a lack of suitable standards. An important recent development is the AIA standard for cameras with digital rather than analog output, which supports interlaced and non-interlaced images of arbitrary resolution.

A television signal is an analog waveform whose amplitude represents the spatial image intensity $I(x,y)$ sequentially as $v(t)$ where time is related to the spatial coordinates by the rasterizing functions

$$x = R_x(t) \tag{3.57}$$
$$y = R_y(t) \tag{3.58}$$

which are such that the image is transmitted in a *raster* scan pattern. This proceeds horizontally across each line, left to right, and each line from top to bottom. Between each line there is a *horizontal blanking* interval which is not displayed but provides a short time interval in which the display CRT beam can return to the left side of screen prior to displaying the next line, and contains a synchronization pulse, to keep the display point in step with the transmitted waveform. The video waveform for

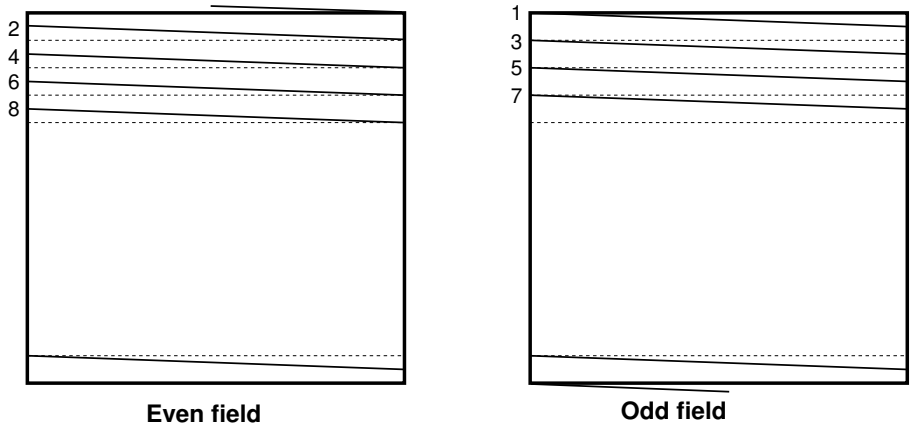


Figure 3.20: CCIR format interlaced video fields. Note the half line at top and bottom which distinguishes the two field types.

one line time is shown in Figure 3.19, and details of timing are given in Table 3.6. Typically video signals are 1 V peak-to-peak. The average value of the luminance signal is referred to as the *pedestal* voltage. The *back-porch* period is used to provide a reference blanking level for the intensity waveform. For RS170 video the voltage corresponding to black is raised slightly, 54 mV, above the blanking level by the *black-setup* voltage.

An *interlaced* video signal comprises pairs of sequential half-vertical-resolution *fields*, displaced vertically by one line, as shown in Figure 3.20. All even lines are transmitted sequentially in one field, followed by all odd lines in the next field. This artifice allows a screen update rate of 50 Hz which is above the flicker perception threshold of human viewers. A field is not an integral number of line times — for CCIR standard video there are 287.5 lines per field. Even fields begins with a half-line, and odd fields end with a half-line as shown in Figure 3.20. A CCIR frame is defined as comprising an even field followed by an odd field. Between each field there is a *vertical blanking interval*, which also serves for beam retrace and synchronization. Details of vertical waveform timing are given in Table 3.7.

A composite color signal comprises the luminance signal already described with a superimposed suppressed-carrier chrominance signal. The chrominance signal is phase modulated to encode the two color component signals. To demodulate this signal the carrier signal[18] must be reinserted locally. A few cycles of the carrier frequency, the *color burst*, are transmitted during the back porch time to synchronize the

[18] For CCIR the color subcarrier frequency is approximately 4.34 MHz.

Period	Fraction	Time (μs)
Total line (H)	1H	64.0
H blanking	0.16H	12.0±0.25
H sync pulse	0.08H	4.7 ± 0.2

Table 3.6: Details of CCIR horizontal timing.

Period	Fraction	Time
Total field (V)	1V	20 ms
	312.5H	
Total frame	2V	40 ms
	625H	
Active frame time	575H	
Active field time	287.5H	
V blanking	25H	1.6 ms
	0.08V	

Table 3.7: Details of CCIR vertical timing.

local color subcarrier oscillator. This method of encoding color is chosen so as to minimize interference with the luminance signal and to provide backward compatibility, that is, allowing monochrome monitors to satisfactorily display a color video signal. Bright fully-saturated regions cause the CCIR color video signal to peak at 1.23 V.

3.4.1 Interlacing and machine vision

As already described a frame in an interlaced image comprises two video fields. Before the frame can be processed in a machine vision system both fields must be read into a framestore to recreate the frame. This process is referred to as *deinterlacing*.

Ordinarily deinterlacing does not cause any problem, but when imaging a rapidly moving object the time at which the field images are captured, or the type of shuttering used, is critical. A *frame shuttered* camera exposes both fields simultaneously, whereas a *field shuttered* camera exposes each field prior to readout. Frame shuttering is feasible with a frame transfer CCD sensor. The Pulnix camera used in this work employs an interline transfer CCD sensor and is capable only of field shuttering. Figure 3.21 shows the effect of deinterlacing an image of a rapidly moving object from a field shuttered camera. For moderate velocity the object becomes ragged around the edges, but for high velocity it appears to disintegrate into a cloud of short line segments. One approach to circumventing this problem is to treat the fields as frames in their own right, albeit with half the vertical resolution. This has the advantages of providing twice the visual sample rate, while eliminating deinterlacing which requires

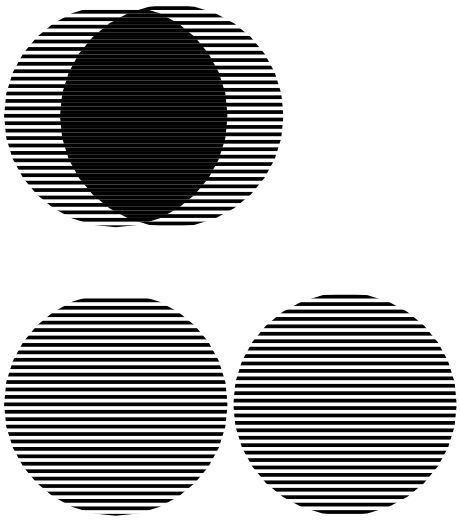


Figure 3.21: The effects of field-shuttering on an object with moderate (top) and high (bottom) horizontal velocity.

additional hardware and increases latency.

When using video fields in this manner it should be remembered that the photosites for the two fields are offset vertically by one line in the sensor array, and thus odd and even field pixels correspond to vertically disparate points in the scene[19]. This results in the vertical coordinates being superimposed with a 25 Hz square wave with 1 pixel peak-to-peak amplitude. Although the image plane displacement is small, the apparent velocity is very high, around 12 pixel/s, and may cause problems with target velocity estimators. To be rigorous when using field rate data this effect should be corrected for (different camera calibrations made for each field) but in practice this is rarely done.

3.5 Image digitization

The digital images used in machine vision are a spatially sampled representation of the continuous image function $I(x,y)$. The first step in machine vision processing is to digitize the analog video waveform which represents the image function. The waveform is sampled and quantized and the values stored in a two-dimensional memory array known as a *framestore*. These samples are referred to as picture elements or

[19] Some cameras are able to provide an interlaced output that uses only a single field from the sensor. That is, they output even field, even field, even field ...

pixels, and their magnitude is often referred to as *grey-level* or *grey value*. Each row of data in the framestore corresponds to one line time of the analog video signal.

This section will examine the various processing stages involved in image digitization, both in general terms and also with particular reference to the Datacube DIGIMAX [73] hardware used in this work.

3.5.1 Offset and DC restoration

The analog waveform from the camera may accumulate DC offset during transmission and amplification. DC restoration is the process of eliminating these offsets which would otherwise be manifest as brightness offsets, introducing errors in perceived image contrast. The waveform is sampled during the back porch period, and this value (maintained by a sample and hold network) is subtracted from the video signal for the next line so as to restore the DC level. Ideally the signal's black reference voltage is related to the output of the dark reference photosites within the camera.

The DIGIMAX digitizer allows an offset to be specified, so that the signal level corresponding to black can be set to correspond with a grey value of 0. The Pulnix TM-6 camera has been observed to output a black level 25 mV above blanking level, despite its claimed CCIR output format.

3.5.2 Signal conditioning

Prior to digitization the analog signal from the camera must be filtered to reduce aliasing and possibly to screen out the chrominance signal from a composite video source. The DIGIMAX has a 6th order filter with settable break frequency. For a 512 pixel sensor a break frequency of 4.5 MHz is used with a 40 dB/octave rolloff. However such a filter will have a substantial effect on frequency components within the pass band which will be manifested in reduced contrast and blurring of fine image detail, compounding the effect due to the camera's MTF. More importantly the signal will be delayed due to the phase characteristic, see Figure 3.22, of the anti-aliasing filter. This must be considered in the camera calibration process since it shifts the entire image horizontally as shown by the step response Figure 3.23.

The signal conditioning stage may also introduce a gain, K_{dig}, prior to digitization to improve the signal to noise ratio for dark scenes. The DIGIMAX digitizer allows gain settings in the range -4 dB to +10 dB in 2 dB steps, which corresponds to +1.3 to -3.3 stops at the camera.

3.5.3 Sampling and aspect ratio

At the sampler the piecewise constant output signal corresponding to the camera photosites has been degraded and 'rounded off' due to the limited bandwidth electronics

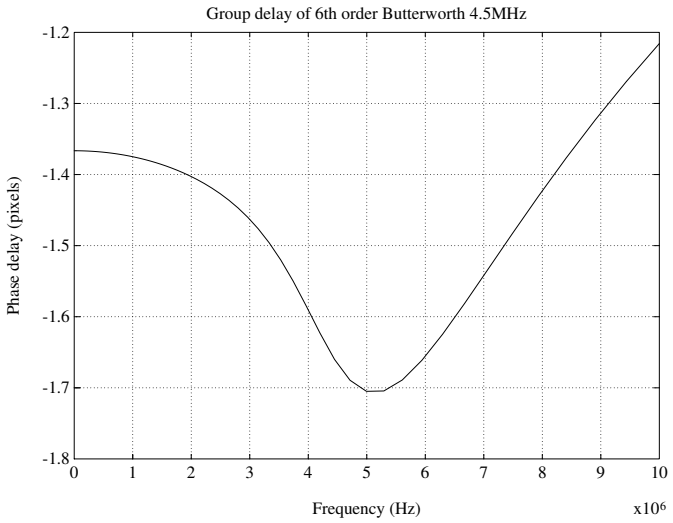


Figure 3.22: Phase delay of 6th order, 4.5 MHz Butterworth low-pass filter showing delay in pixels as a function of frequency of the video signal. For video digitization the Nyquist frequency is 5 MHz.

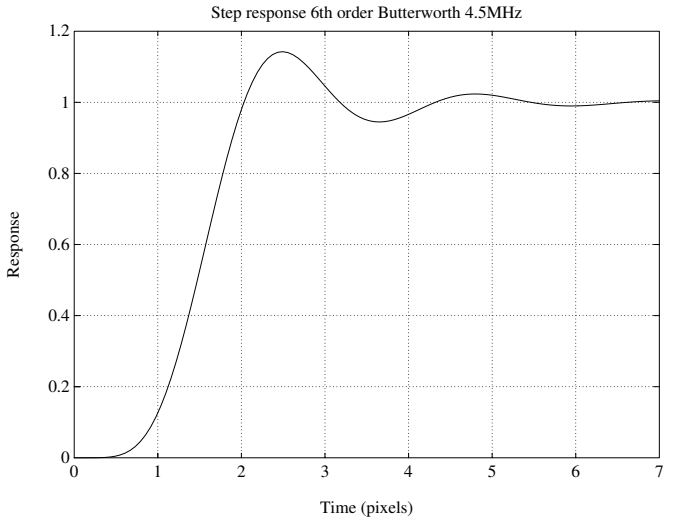


Figure 3.23: Simulated step response of 6th order, 4.5MHz Butterworth low-pass filter.

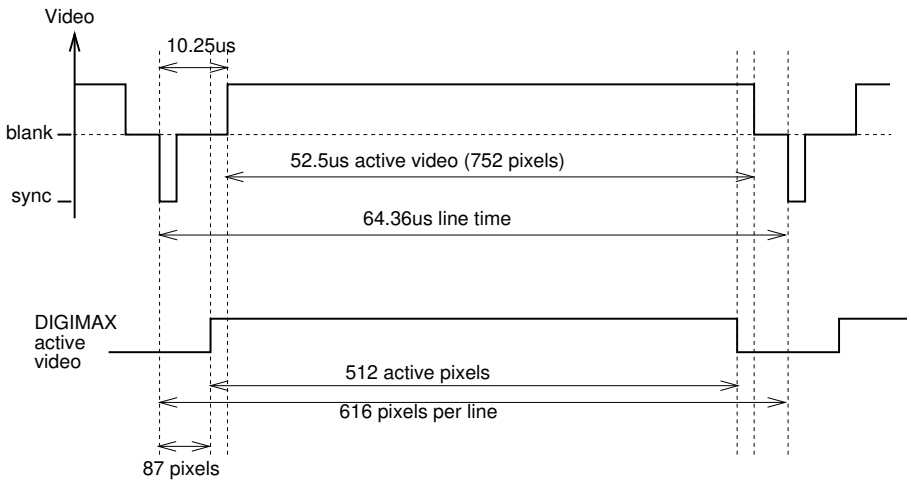


Figure 3.24: Measured camera and digitizer horizontal timing.

and transmission. This signal must be sampled to generate the framestore pixel values $I_f^*(i,j)$ where i and j are framestore row and column addresses. Importantly, the framestore pixels, I_f^*, are not necessarily mapped exactly onto camera pixels. In order to correctly sample the camera signal and address the framestore the camera's rasterizing functions (3.57) and (3.58) must be replicated within the digitizer. The synchronization information within the video signal is used for this purpose.

A digitizer samples the video signal $v(t)$ at a frequency f_d, which is generally an integral multiple of the horizontal synchronization pulse frequency, f_h. For the DIGIMAX digitizer [75]

$$f_d = 616 f_h \tag{3.59}$$

Such sampling does not necessarily align the samples with the output of spatially adjacent photosites. Typical digitizers sample the incoming video into an array of 512×512 samples[20], so each line has 512 samples irrespective of the number of photosites per line in the sensor. Each digitized pixel is therefore not an independent point sample of the incident illumination, an effect which has serious ramifications for many machine vision algorithms, particularly edge detectors, which make assumptions about independent pixels [15]. This pixel re-sampling further reduces the MTF, and also alters the aspect ratio of the stored image.

Figure 3.24 shows the timing waveforms measured for the Pulnix camera and DIGIMAX. The active video time of the camera is the response of the unmasked

[20] Since CCIR video has 575 active lines per frame, see Table 3.7, the lowest 63 lines of the frame (11%) are lost. More modern digitizers and framestores allow capture of an entire CCIR frame.

Parameter	Value
Sensor width	6.5 mm
Sensor height	4.8 mm
Horizontal active pixels	752 pixels
Vertical active lines	582 pixels
Pixel width (p_x)	8.6 μm
Pixel height (p_y)	8.3 μm

Table 3.8: Manufacturer's specifications for the Pulnix TM-6 camera.

pixels and is readily measured with an oscilloscope from the video waveform with the sensor illuminated and no lens fitted[21]. The active video region of the DIGIMAX is 512 pixels long beginning 87 pixels after the horizontal synchronization pulse [75]. Clearly the active video times of the camera and digitizer do not overlap correctly. As a consequence the digitized image contains a black stripe, approximately 10 pixels wide, on the left hand edge of the image and the same number of pixels are lost from the right hand edge. The DIGIMAX's timing is appropriate for RS170 where the horizontal blanking periods are shorter than for CCIR.

The camera's 752 camera pixels per line, from Table 3.8, are re-sampled at 512 points, and this ratio can also be expressed in terms of the digitizer and camera pixel frequencies

$$\beta = \frac{f_c}{f_d} \approx \frac{752}{512} \approx 1.47 \tag{3.60}$$

Each framestore pixel thus contains information from nearly 1.5 camera photosites.

The parameter β is important for many of the camera calibration techniques discussed in Section 4.2.2, and a number of techniques have been proposed for its determination since it is highly dependent upon the camera and digitizer used. Tsai [169] suggests observing interference patterns in the digitized image due to beating of the camera's pixel clock with the digitizer, but no such pattern is discernible with this Pulnix camera. Penna [205] describes an approach based on imaging an accurate sphere, and fitting a polynomial curve to the captured image.

An experiment to accurately determine β was conducted using the frequency ratio measurement capability of an HP5115A Universal Counter. The camera pixel clock and the horizontal synchronization pulse signals are both available and their ratio was measured as

$$\frac{f_c}{f_h} = 904.3 \tag{3.61}$$

[21] The measured line time at 64.36 μs is slightly longer than the CCIR standard 64 μs.

Parameter	Value
α_x	79.2 pixel/mm
α_y	120.5 pixel/mm (frame mode)
α_y	60.2 pixel/mm (field mode)

Table 3.9: Derived pixel scale factors for the Pulnix TM-6 camera and DIGIMAX digitizer.

and knowledge of DIGIMAX operation, (3.59), leads to

$$\beta = \frac{f_c}{f_d} = \frac{f_c}{f_h}\frac{f_h}{f_d} = 1.468 \tag{3.62}$$

Framestore pixel aspect ratio is a function of the camera's pixel dimensions and the sampling ratio β. The dimensions[22] of the photosites, $p_x \times p_y$, for the camera used in this work are given in Table 3.8.[23] The scale factors α_x and α_y are readily derived from the photosite dimensions

$$\alpha_x = \frac{1}{\beta \times p_x}\,\text{pixel/m} \tag{3.63}$$

$$\alpha_y = \frac{1}{p_y}\,\text{pixel/m} \tag{3.64}$$

An image on the sensor with dimensions of $W \times H$ will appear in the framestore with dimensions, in pixels, of $\alpha_x W \times \alpha_y H$. In general the aspect ratio, height/width, will be changed by α_y/α_x, and this has been verified experimentally as 1.52. When processing video fields rather than frames, adjacent rows in the image are separated by two rows on the sensor. The vertical scale factor becomes

$$\alpha_y = \frac{1}{2p_y}\,\text{pixel/m} \tag{3.65}$$

and the ratio α_y/α_x is now 0.761. These scale factors are summarized in Table 3.9.

The timing relationships just described, which are a function of the particular camera and digitizer used, directly affect horizontal displacement of the image and digitized image aspect ratio. Both of these factors will affect the intrinsic parameters of the camera, and are significant in camera calibration which will be discussed in Section 4.2.

Figure 3.25 shows the coordinate systems that will be used throughout this work. The direction of pixel scanning within the CCD sensor eliminates the image inversion present in the earlier lens equations (3.30) and (3.31). The world X and Y directions

[22] Manufacturer's specifications do not discuss tolerance on pixel dimensions.

[23] Generally the pixels cells are not square though cameras with square pixels are available.

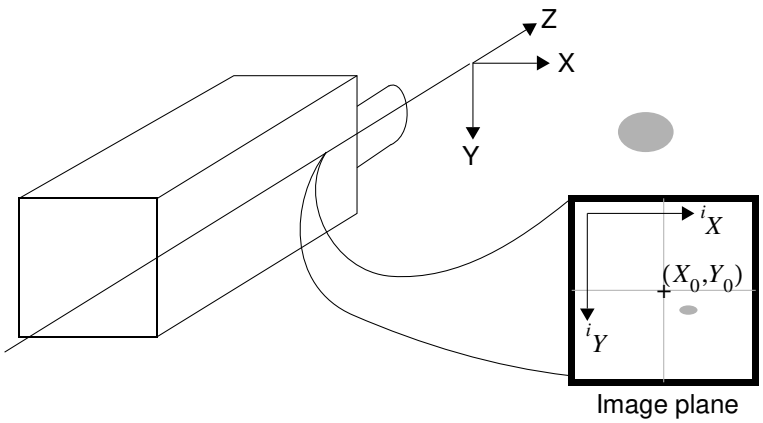


Figure 3.25: Camera and image plane coordinate systems.

correspond to the image plane ^iX and ^iY directions respectively. Those equations can now be written in terms of pixel coordinates as

$$^iX = \frac{\alpha_x f x}{z-f} + X_0 \tag{3.66}$$

$$^iY = \frac{\alpha_y f y}{z-f} + Y_0 \tag{3.67}$$

$$\tag{3.68}$$

where (X_0, Y_0) is the pixel coordinate of the principal axis defined earlier.

3.5.4 Quantization

The final step in the digitization process is to quantize the conditioned and sampled analog signal by means of a high-speed analog to digital converter. The sample is quantized into an n-bit integer with values in the range 0 to $2^n - 1$. Typically the black reference would correspond to 0 and the peak white level to $2^n - 1$. The quantized signal, $x_q(t)$, can be written in terms of the original signal , $x(t)$, as

$$x_q(t) = x(t) + e_q(t) \tag{3.69}$$

where $e_q(t)$ is the quantization noise which is assumed to have a uniform distribution over the range $[-\frac{1}{2}\ \frac{1}{2}]$ and a mean square given by

$$\overline{e^2} = \frac{1}{12} \tag{3.70}$$

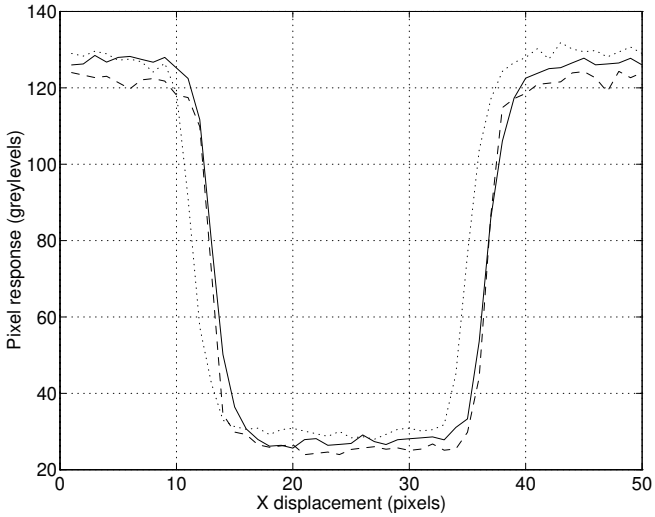


Figure 3.26: Measured camera response to horizontal step illumination change. Curves are f/1.4 with 4.5 MHz analog filter (solid), f/1.4 with no analog filter (dotted), and f/16 with 4.5 MHz analog filter (dashed).

The signal to noise ratio is then

$$\text{SNR} = 12\overline{x^2} \tag{3.71}$$

The DIGIMAX employs an 8-bit converter and for a typical pixel RMS grey-value of 100 the SNR due to quantization would be 51 dB. To increase the SNR for low amplitude signals a gain, K_{dig}, can be introduced prior to quantization, see Section 3.5.2. From Figure 3.18 it is clear that the quantization noise, (3.70), is low compared to the total measured variance.

3.5.5 Overall MTF

The spatial step response of the entire imaging system was measured by capturing an image of a standard test chart under various conditions. The spatial frequency response will be reduced by all the mechanisms described above including lens aberration, aperture diffraction, pixel capture profile, analog signal processing, and digitizer pixel re-sampling. The horizontal step responses are shown in Figure 3.26. Switching out the DIGIMAX analog filter shifts the edge by approximately 1.7 pixels[24] but

[24] Since the phase delay of the filter, shown in Figure 3.22, is eliminated.

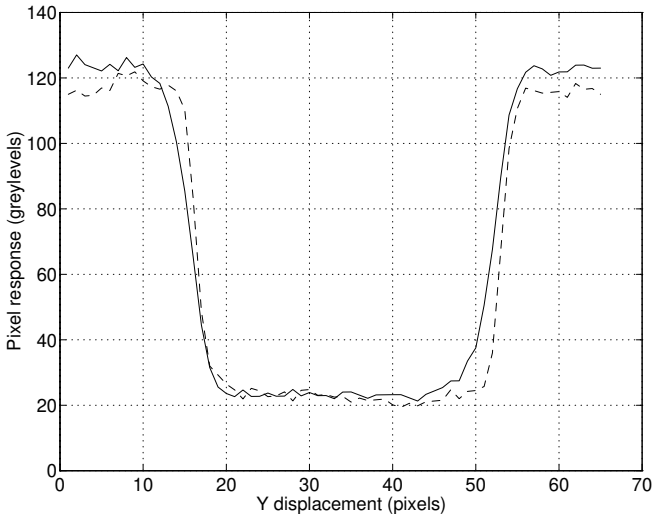


Figure 3.27: Measured camera response to vertical step illumination change. Curves are f/1.4 with 4.5 MHz analog filter (solid), and f/16 with 4.5 MHz analog filter (dashed).

does not increase the edge sharpness. For both horizontal and vertical responses the edge gradient is 50% higher at f/16 compared to f/4. As discussed in Section 3.2.3.1 the effect of aberrations is reduced as the f-number increases until the diffraction limit, (3.24), is encountered. From this it may be concluded that the analog filter and diffraction have minimal effect on the overall MTF which is ultimately limited by lens aberration.

The vertical and horizontal profiles both have approximately the same edge gradient when converted from pixel units to distance units, that is, around 25 greylevel$/\mu$m. The edge width for the f/1.4 case is approximately 3.8 pixels, which from (3.25) gives an estimated MTF at the Nyquist frequency of

$$\mathrm{MTF}_{f_s/2} = \frac{1}{3.8} \approx 0.26$$

This is low compared to the MTF plots given in Figure 3.13 and again suggests that spatial sampling is not a limiting factor in edge resolution. There is strong evidence to suggest that the C-mount CCTV lens[25] used is the dominant source of edge blur.

[25] Cosmicar C814.

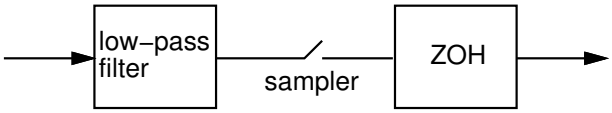


Figure 3.28: Typical arrangement of anti-aliasing (low-pass) filter, sampler and zero-order hold.

These lenses are mass produced commodity items for applications such as surveillance and are likely to be designed for low unit cost rather than image quality. It has been suggested [52] that the format of C-mount lenses, particularly the long back-focal distance severely constrains the optical design. C-mount lenses are however commonly used within the machine vision and robotics research community.

3.5.6 Visual temporal sampling

In a visual servo system the camera performs the function of the sampler. An ideal visual sampler would capture the instantaneous state of the scene and in practice this can be approximated by choice of a suitably short exposure interval. Digital control systems, as shown in Figure 3.28, typically include an analog prefilter between the sensor and the digitizer as an anti-aliasing device. Such filters are low-pass and designed to attenuate signals above the Nyquist frequency so that, when aliased into lower frequencies by the sampler, they will not be detrimental to the control-system performance [95].

The effect of aliasing due to camera sampling results in the well known effect where, in movies, the wheels on wagons can appear to rotate backward. In a visual servo system it is difficult to conceptualize an analog prefilter — this would be some optical device that transmitted low-frequency scene intensity change but attenuated high-frequency change. Intuitively it can be seen that such an effect is achieved by motion blur. A point oscillating at high frequency in the scene will appear, with a long exposure interval, to be a blur and have little if any apparent motion. While rapidly oscillating targets are unlikely to be encountered, camera oscillation due to manipulator structural resonance is a significant issue and is discussed further in Section 8.1.4, where it is shown that resonances exist at frequencies considerably greater than the visual Nyquist frequency.

Simulations of camera response to sinusoidal target motion show the magnitude of the motion detected by the camera is a complex function of target motion magnitude and frequency as well as camera exposure interval and threshold. The simulation models the individual photosite charge integration as the target intensity profile moves with respect to the photosite array. The integration is performed using a large number of time steps within each charge integration interval. The dominant characteristic for

sinusoidal target motion up to the Nyquist frequency is a linear phase characteristic due to the latency associated with the exposure interval given previously by (3.47).

Above the Nyquist frequency it is not possible to examine the magnitude and phase of sinusoidal components since, due to frequency folding, the input and output frequency are not equal. The approach used in simulation is to assume target motion with a uniform spectrum above the visual Nyquist frequency and examine the RMS magnitude of the simulated camera output which represents noise injected into the controller by super-Nyquist target motion. The target motion

$$x(t) = \mathtt{H}\ (r(t)) + \beta t \tag{3.72}$$

is generated by high-pass filtering a vector of random numbers, $r(t)$. In this case the filter selected, $\mathtt{H}$, is a 6^{th} order Type I Chebyshev filter with a break frequency of 30 Hz and 3 dB passband ripple. For small amplitude target motion the camera output is highly dependent on where the target image lies with respect to the pixel boundaries. To counter this the high frequency target motion is added to a ramp signal, βt, which slowly moves the target centroid by 1 pixel over the simulation interval of 64 field times. This ramp is removed before computing the RMS value of the simulated camera output. Figure 3.29 shows the magnitude response of the camera simulated in this manner for two different exposure intervals. It can be seen that the camera attenuates this super-Nyquist signal and that the attenuation increases with exposure interval. Intuitively this is reasonable since image blur will increase with exposure interval. Figure 3.30 shows, for a constant exposure interval, the effect of varying the threshold. The camera response is greatest for a normalized threshold of 0.5 which is the midpoint between background and foreground intensity. Greatest attenuation of super-Nyquist components can be achieved by using a low threshold and a long exposure interval.

3.6 Camera and lighting constraints

This section summarizes the issues involved in the selection of camera settings and scene illumination. The requirements for visual servoing are:

- a large depth of field so as to avoid image focus problems as the camera moves depth wise with respect to the scene;
- short exposure interval to reduce motion blur and have the camera closely approximate an ideal visual sampler;
- the object be of sufficient size and brightness in the image that it can be recognized.

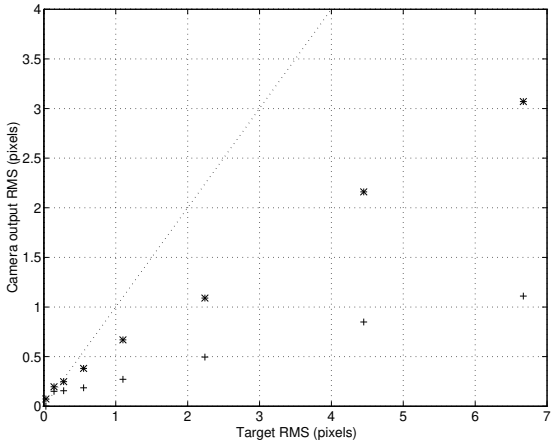


Figure 3.29: Magnitude response of simulated camera output versus target motion magnitude for various exposure intervals: 2 ms (*), 20 ms (+). Dotted line corresponds to unit gain. Threshold is 0.5.

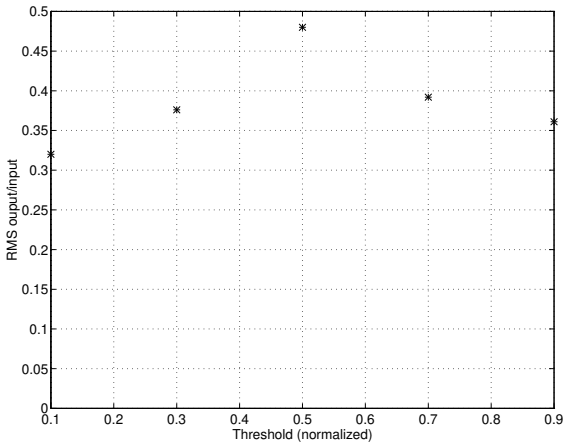


Figure 3.30: Magnitude response of simulated camera output to changing threshold for constant target motion magnitude. Exposure interval is 2 ms.

Quantity	Lower bound	Upper bound
focal length f	magnification (3.8) geometric distortion	field of view (3.10), (3.11)
f-number F	depth of field (3.19), (3.20)	diffraction (3.24) image brightness (3.12) SNR (3.56)
exposure interval T_e	image brightness (3.12) SNR (3.56)	image blur
illuminance E_i	image brightness (3.12) SNR (3.56)	subject heating

Table 3.10: Constraints in image formation. The relevant equations are cited.

These requirements are conflicting — for instance depth-of-field requires a small aperture which combined with the need for short exposure time greatly reduces the image intensity. Table 3.10 lists the parameters and indicates the factors that control the upper and lower bounds of acceptable settings.

A spreadsheet, see Figure 3.31, is a convenient way to appreciate the interactions of these parameters. The top section contains parameters of the lens and digitizer, and these correspond to the 8 mm lens used in this work.[26] The next section computes the diffraction blur by (3.24), and the hyperfocal distance for the specified diameter circle of confusion by (3.21). The near and far focus bounds computed by (3.19) and (3.20) are also shown. For this case a reasonable depth of field can be obtained at f/5.6 with the focus point set to 700 mm. The diffraction effect is negligible at only 0.3 pixels.

The lowest section of the spreadsheet computes field of view, object image plane size at two settable distances from the camera; near and far. Incident lightmeter readings and estimated camera grey-level response are calculated based on the distance between object and light source. In this example the lighting system comprises two 800 W studio spot-lights situated 3 m behind the camera. Given the short exposure time and small aperture setting a significant amount of illumination is required to obtain a reasonable camera response at the far object position.

3.6.1 Illumination

An 800 W photographic floodlamp with a quartz halogen bulb has a luminous efficiency of 20 lm/W. Assuming the emitted light is spread over π sr[27] the luminous intensity is

$$\frac{800 \times 20}{\pi} = 5100 \text{ cd}$$

[26] Lens transmission of 80% is an estimate only.

[27] One quarter sphere solid angle.

Camera and lens settings

Aperture (F)	5.6			
Focal length (f)	8	mm		
Lens tranmission	80	%		
Exposure time (T_e)	2	ms		
Pixel scale (α_x)	80	pixel/mm		
Pixel scale (α_y)	60	pixel/mm		
Focus setting	700	mm		
Hyperfocal distance	922	mm		
Diffraction	3.79	µm =	0.30	pixels
Circle confusion diam	1	pixel		
Depth of field	425	mm, to	INF	mm

Object apparent size

			Near	Far	
Object range:			600	3000	mm
Object diameter (d)	50	mm =	54.1	10.7	pixel
Magnification			0.014	0.003	
Framestore pixel scale factor (X)			1.08	0.21	pixel/mm
Framestore pixel scale factor (Y)			0.81	0.16	pixel/mm
FIeld of view:	width		474	2394	mm
	height		631	3191	mm

Illumination and object brighness

Light position	-3000	mm	10186	10186	cd
Light power	1600	W	786	283	lx
Luminous efficiency	20	lm/W	393	141	nt
Sphere fraction	0.25		11.47	9.99	EV
Surface reflectance	0.5		1.26	0.45	lx
			220	79	grey-levels

Figure 3.31: Spreadsheet program for camera and lighting setup. Input variables are in grey boxes. Image plane object size and field of view are calculated for two camera distances; near and far. Object brightness, in lightmeter units and estimated camera grey-level, is computed based on the distance between object and light source.

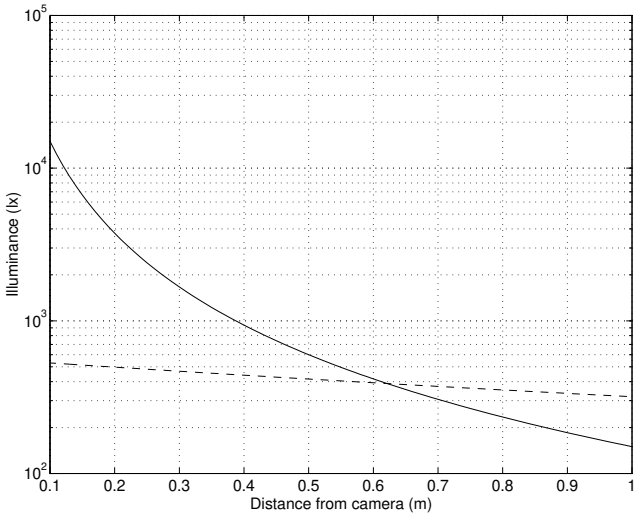


Figure 3.32: Comparison of illuminance due to a conventional 800 W floodlamp (dashed) and 10 camera mounted LEDs (solid) as a function of distance from the camera. It is assumed that the LEDs are 15 cd each and mounted at the camera, and the floodlamp is 3 m behind the camera and radiating light over π sr.

If the lamp is positioned 3 m away from the object the illuminance will be

$$\frac{5100}{3^2} = 570 \text{lx}$$

Recently high-intensity light emitting diodes (LEDs) have become available. A typical device has an intensity of 3 cd at 20 mA with a 7° beam width. At the peak current of 100 mA the intensity would be 15 cd, and if the LED were mounted on the camera and the object distance was 0.5 m the illuminance would be 60 lx. Thus ten camera-mounted high-power LEDs exceed the illuminance of a very hot 800 W studio floodlamp situated some distance away. A comparison of the effective illuminance of floodlamp and LEDs for varying object distance is given in Figure 3.32 showing that the LEDs are competitive for target illumination up to 600 mm. These calculations have been based on photometric units, though as shown in Figure 3.4, a tungsten lamp and a red LED at 670 nm emit most of their radiation in a spectral region where the sensitivity of a CCD sensor greatly exceeds that of the eye.

In the visual servoing application the camera's electronic shutter is open for only

10% of the time[28] so 90% of the scene illumination is essentially wasted. The LED scheme can be further refined by pulsing the LED so that it is lit only while the camera's electronic shutter is 'open'. At a 10% duty-cycle the LED peak current can substantially exceed the continuous rating and experiments have shown that peak currents of 400 mA are possible. However the light output does not increase proportionally to current and this is discussed further in Appendix E.

A pulsed LED light source has been built with a ring of 10 high-intensity LEDs placed around the camera lens, see Figure E.1. A control box, described in Appendix E, allows the current pulse height, width and starting time to be adjusted. The advantages of this light source are that it is lightweight, rugged, has low heat output, and like a miner's cap-lamp directs light on the subject of interest. Its principal disadvantage is that the light output is uneven due to the way in which the pools of light from the individual LEDs overlap.

3.7 The human eye

The human eye has many important differences when compared to a CCD sensor. The eye is approximately spherical with a diameter of 15 mm and light is sensed by photoreceptors located in the retina at the back of the eye. In normal daylight conditions cone photoreceptors are active and these are color sensitive: 65% sense red, 33% sense green and only 2% sense blue. The cones are approximately 3μm in diameter and 34,000 of them are packed into the foveal area of the retina which is only 0.6 mm in diameter. The photoreceptor density in the rest of the retina is considerably lower. The eye has high resolution only over the foveal field of view of a few degrees but subconscious eye motion directs the fovea over the entire field of view. The distance between the lens and retina is approximately constant at 15 mm so focussing is achieved by muscles which change the shape of the lens.

Cone photoreceptors have a dynamic range of 600 and the pupil, equivalent to the iris of a lens, varies in diameter from 2 to 8 mm which provides for a factor of 16 (10 in older people) in dynamic range. At very low light levels the rod photoreceptors become active and provide another factor of 20 in dynamic range. The rod sensors are monochromatic and their density in the fovea is only 7% of that of the cones, but increases in the peripheral region. Rod sensitivity is chemically adapted with a time constant of tens of minutes. The overall dynamic range of the eye is thus approximately 100,000.

The eye has three degrees of rotational motion. The muscles that actuate the human eye are the fastest acting in the body allowing the eye to rotate at up to 600 deg/s and 35,000 deg/s^2 for saccadic motion [270]. Smooth pursuit eye motions, involved in tracking a moving object, operate at up to 100 deg/s [214]. Rotation about the view-

[28] A 2 ms exposure for every 20 ms video field.

ing axis, cyclotorsion, is limited and the maximum, ranging from 5 to 20 deg, varies between individuals.

Chapter 4

Machine vision

Computer vision is the application of a computer system for receiving and processing visual information. It comprises two broad areas: image processing and image interpretation. The latter, often referred to as *machine vision*, is typically applied to part inspection and quality control, and involves the extraction of a small number of generally numeric *features* from the image. The former is the enhancement of images such that the resulting image more clearly depicts some desired characteristics for a human observer, and is commonly applied to medical and remote sensing imagery.

Section 4.1 introduces the conventional computer vision topics of segmentation and feature extraction. Particular emphasis is given to binary image processing and moment features since these provide a cost-effective and tractable solution to video-rate image feature extraction. It is important to note that visual servo techniques based on binary image features are equally applicable to more elaborately obtained features, provided that the processing rate is high and the latency is low.

Section 4.2 discusses the important topics of close-range photogrammetry, camera calibration and eye-hand calibration. These are important in understanding prior work introduced in the next chapter, and also to complete the characterization of the camera and digitizer used in this work.

4.1 Image feature extraction

Image interpretation, or scene understanding, is the problem of describing physical objects in a scene given an image, or images, of that scene. This description is in terms of, generally numeric, image features.

The principal role of image feature extraction is to reduce the camera output data rate to something manageable by a conventional computer, that is, extracting the 'essence' of the scene. An *image feature* is defined generally as any measurable re-

lationship in an image. Jang [135] provides a formal definition of features as image functionals

$$f = \int\int_{Image} \mathtt{F}\ (x, y, I(x,y))\, dxdy \tag{4.1}$$

where $I(x,y)$ is the pixel intensity at location (x,y). The function F () is a linear or non-linear mapping depending on the feature, and may also include delta functions. Some specific examples of image features include:

- the $(p+q)^{th}$ order moments

$$m_{pq} = \int\int_{Image} x^p y^q I(x,y) dxdy; \tag{4.2}$$

  For a binary image m_{00} is the object area, and $(m_{10},\ m_{01})$ is the centroid. Moment features, discussed further in Section 4.1.2, are widely used in visual servo systems [17, 65, 88, 92, 116, 212] due to the simplicity of computation;

- template matching by cross-correlation or sum of squared differences [198] to determine the coordinate of some distinctive pixel pattern in the image. The pattern may be an edge, corner or some surface marking;

- lengths, or orientation, of the edges of objects;

- lengths, or orientation, of line segments connecting distinct objects in the scene such as holes or corners [104, 198, 236]. Feddema [91, 92] used lines between the centroids of holes in a gasket to determine the gasket's pose.

Two broad approaches to image feature extraction have been used for visual servoing applications; whole scene segmentation, and feature tracking, and are described in sections 4.1.1 and 4.1.4 respectively.

4.1.1 Whole scene segmentation

Segmentation is the process of dividing an image into meaningful segments, generally homogeneous with respect to some characteristic. The problem of robustly segmenting a scene is of key importance in computer vision, and much has been written about the topic and many methods have been described in the literature. Haralick [106] provides a survey of techniques applicable to static images but unfortunately many of the algorithms are iterative and time-consuming and thus not suitable for real-time applications. In a simple or contrived scene the segments may correspond directly to objects in the scene, but for a complex scene this is rarely the case.

The principal processing steps in segmentation, shown in Figure 4.1, are:

1. *Classification*, where pixels are classified into spatial sets according to 'low-level' pixel characteristics.

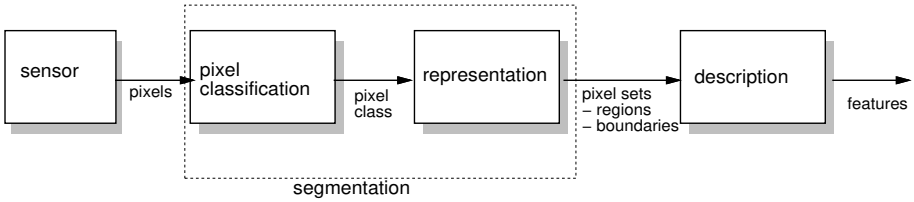


Figure 4.1: Steps involved in scene interpretation

2. *Representation*, where the spatial sets are represented in a form suitable for further computation, generally as either connected regions or boundaries.

3. *Description*, where the sets are described in terms of scalar or vector valued features.

Pixel values may be scalar or vector, and can represent intensity, color, range, velocity or any other measurable scene property. The classification may take into account the neighbouring pixels, global pixel statistics, and even temporal change in pixel value. General scenes have too much 'clutter' and are difficult to interpret at video rates unless pixels of 'interest' can be distinguished. Harrell [108] describes the use of color classification to segment citrus fruit from the surrounding leaves in a fruit picking visual servo application. Haynes [39] proposes sequential frame differencing or background subtraction to eliminate static background detail. Allen [7, 8] uses optical flow calculation to classify pixels as moving or not moving with respect to the background, which is assumed stationary.

The simplest classification is into two sets, leading to *binary segmentation*. Commonly this is achieved by applying a threshold test to the pixel values, and for an intensity image may be written

$$P_{ij} \in \begin{cases} S_b & \text{if } I_{ij} < T \\ S_f & \text{if } I_{ij} \geq T \end{cases}$$

where P_{ij} is the pixel (i, j), and S_b and S_f are respectively the sets of background and foreground pixels. This technique is widely used in laboratory situations where the lighting and environment can be contrived (for instance using dark backgrounds and white objects) to yield high contrast, naturally distinguishing foreground objects from the background. Many reported real-time vision systems for juggling [212], ping-pong [17, 88] or visual servoing [65, 92, 116] use this simple, but non-robust, approach.

Selection of an appropriate threshold is a significant issue, and many automated approaches to threshold selection have been described [219, 275]. Adaptive thresholding has also been investigated, the principal problem being how to automatically rate

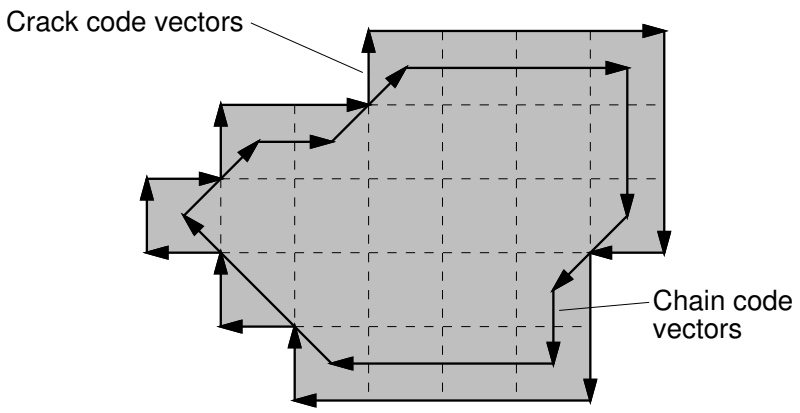


Figure 4.2: Boundary representation as either crack codes or chain code.

the 'success' or quality of a segmentation so that an automatic system may modify or adapt its parameters [15, 156]. Corke and Anderson [54] use the capability of a hardware region-grower to perform many trial segmentations per second, and judge appropriate threshold from the number of regions found.

Once the pixels of interest have been identified they must be represented in some form that allows features such as position and shape to be determined. Two basic representations of image segments are natural: two-dimensional *regions*, and *boundaries*. The first involves grouping contiguous pixels with similar characteristics. Edges represent discontinuities in pixel characteristics that often correspond to object boundaries. These two representations are 'duals' and one may be converted to the other, although the descriptions used are quite different.

Edges may be represented by fitted curves, *chain code* or *crack code*, as shown in Figure 4.2. Crack code represents the edge as a series of horizontal and vertical line segments following the 'cracks' between pixels around the boundary of the pixel set. Chain code represents the edge by direction vectors linking the centers of the edge pixels. Feddema [91] describes the use of chain code for a visual servoing application. Crack codes are represented by 2-bit numbers giving the crack direction as $90i°$, while chain code is represented by 3-bit numbers giving the next boundary point as $45i°$. Rosenfeld and Kak [217] describe a single-pass algorithm for extracting crack-codes from run-length encoded image data.

The dual procedure to boundary tracing is *connected component* analysis (also *connectivity analysis* or *region growing*), which determines contiguous regions of pixels. Pixels may be 4 way, 6 way or 8 way connected with their neighbours [217, 268]. This analysis involves one pass over the image data to assign region labels to all pixels. During this process it may be found that two regions have merged, so a table records

their equivalence [27, 70] or a second pass over the data may be performed [217].

While edge representations generally have fewer points than contained within the component, computationally it is advantageous to work with regions. Boundary tracking cannot commence until the frame is loaded, requires random-access to the image memory, and on average 4 memory accesses to determine the location of the next edge pixel. Additional overhead is involved in scanning the image for boundaries, and ensuring that the same boundary is traced only once.

Next each segmented region must be *described*, the process of *feature extraction*. The regions, in either edge or connected component representation, can be analyzed to determine area, perimeter, extent and 'shape'.

4.1.2 Moment features

A particularly useful class of image features are moments. Moments are easy to compute at high speed using simple hardware, and can be used to find the location of an object (centroid) and ratios of moments may be used to form invariants for recognition of objects irrespective of position and orientation as demonstrated by Hu [125] for planar objects. For a binary image the image function $I(x,y)$is either 0 or 1 and the moments describe the set of points S_i, not the grey-level of those points.

From (4.2) the $(p+q)^{th}$ order moment for a digitized image is

$$m_{pq} = \sum_R \sum x^p y^q I(x,y) \tag{4.3}$$

Moments can be given a physical interpretation by regarding the image function as mass distribution. Thus m_{00} is the total mass of the region and the centroid of the region is given by

$$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}} \tag{4.4}$$

For a circular object it should be noted that if it is not viewed along the surface normal, the centroid of the image (which will be an ellipse) does not correspond with the centroid of the object. Figure 4.3 shows this in exaggerated form via geometric construction, where clearly $b < a$. The effect becomes more pronounced as the viewing axis departs from the surface normal.

The central moments μ_{pq} are computed about the centroid

$$\mu_{pq} = \sum_R \sum (x - x_c)^p (y - y_c)^q I(x,y) \tag{4.5}$$

and are invariant to translation. They may be computed from the moments m_{pq} by

$$\mu_{10} = 0 \tag{4.6}$$

$$\mu_{01} = 0 \tag{4.7}$$

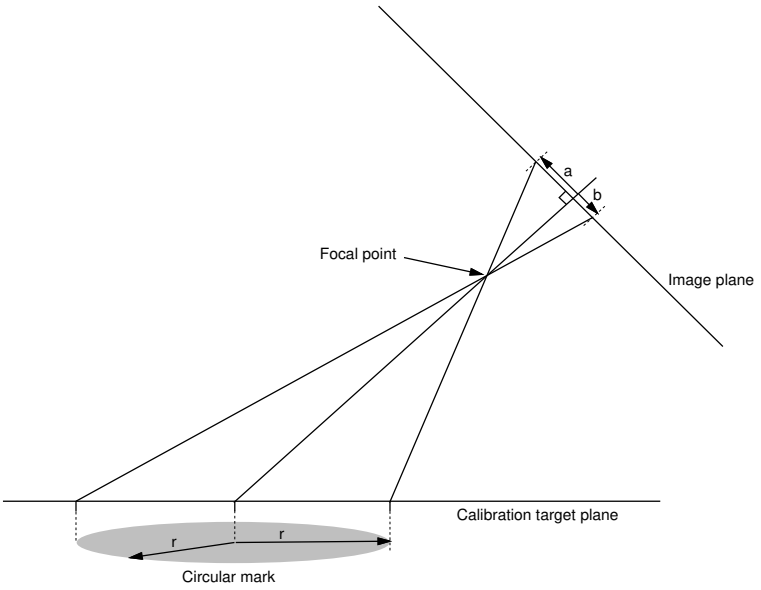


Figure 4.3: Exaggerated view showing circle centroid offset in the image plane.

$$\mu_{20} = m_{20} - \frac{m_{10}^2}{m_{00}} \tag{4.8}$$

$$\mu_{02} = m_{02} - \frac{m_{01}^2}{m_{00}} \tag{4.9}$$

$$\mu_{11} = m_{11} - \frac{m_{10}m_{01}}{m_{00}} \tag{4.10}$$

A commonly used but simple shape metric is *circularity*, defined as

$$\rho = \frac{4\pi m_{00}}{p^2} \tag{4.11}$$

where p is the region's perimeter. Circularity has a maximum value of $\rho \equiv 1$ for a circle, and a square can be shown to have $\rho = \pi/4$.

The second moments of area μ_{20}, μ_{02} and μ_{11} may be considered the moments of inertia about the centroid

$$I = \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix} \tag{4.12}$$

The eigenvalues are the principal moments of the region, and the eigenvectors of this matrix are the principal axes of the region, the directions about which the region has

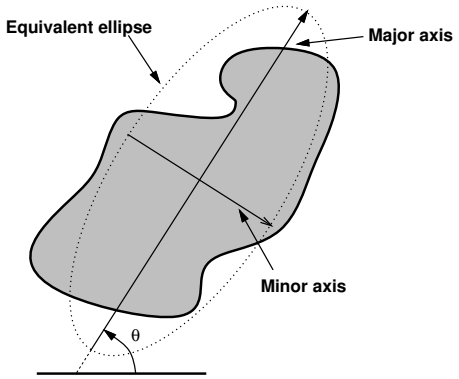


Figure 4.4: Equivalent ellipse for an arbitrary region.

maximum and minimum moments of inertia. From the eigenvector corresponding to the maximum eigenvalue we can determine the orientation of the principal axis as

$$\tan\theta = \frac{\mu_{20} - \mu_{02} - \sqrt{\mu_{20}^2 - 2\mu_{20}\mu_{02} + \mu_{02}^2 + 4\mu_{11}^2}}{2\mu_{11}} \tag{4.13}$$

or more simply

$$\tan 2\theta = \frac{2\mu_{11}}{\mu_{20} - \mu_{02}} \tag{4.14}$$

which is the same as Hu's equation (59). Many machine vision systems compute the so called 'equi valent ellipse' parameters. These are the major and minor radii of an ellipse with the same area moments as the region, see Figure 4.4. The principal moments are given by the eigenvalues of (4.12)

$$\lambda_1, \lambda_2 = \frac{\mu_{20} + \mu_{02} \pm \sqrt{(\mu_{20}\mu_{02})^2 + 4\mu_{11}^2}}{2} \tag{4.15}$$

and the area moments of an ellipse about the major and minor axes are given respectively by

$$I_{maj} = \frac{Aa^2}{4}, \quad I_{min} = \frac{Ab^2}{4} \tag{4.16}$$

where A is the area of the region, and a and b are the major and minor radii. This can be rewritten in the form

$$a = 2\sqrt{\frac{\lambda_1}{m_{00}}}, \quad b = 2\sqrt{\frac{\lambda_2}{m_{00}}} \tag{4.17}$$

The normalized moments

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}},\ \gamma = \frac{1}{2}(p+q)+1 \text{ for } p+q = 2,3\cdots \tag{4.18}$$

are invariant to scale. Third-order moments allow for the creation of quantities that are invariant with respect to translation, scaling and orientation within a plane. Hu [125] describes a set of seven moment invariants

$$\begin{aligned}
\phi_1 &= \eta_{20}+\eta_{02} && (4.19)\\
\phi_2 &= (\eta_{20}-\eta_{02})^2+4\eta_{11}^2 && (4.20)\\
\phi_3 &= (\eta_{30}-3\eta_{12})^2+(3\eta_{21}-\eta_{03})^2 && (4.21)\\
\phi_4 &= (\eta_{30}+\eta_{12})^2+(\eta_{21}+\eta_{03})^2 && (4.22)\\
\phi_5 &= (\eta_{30}-3\eta_{12})(\eta_{30}+\eta_{12})\left[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2\right]+ && (4.23)\\
&\quad (3\eta_{21}-\eta_{03})(\eta_{21}+\eta_{03})\left[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2\right]\\
\phi_6 &= (\eta_{20}-\eta_{02})\left[(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2\right]+ && (4.24)\\
&\quad 4\eta_{11}(\eta_{30}+\eta_{12})(\eta_{21}+\eta_{03})\\
\phi_7 &= (3\eta_{21}-\eta_{03})(\eta_{30}+\eta_{12})\left[(\eta_{30}+\eta_{12})^2-3(\eta_{21}+\eta_{03})^2\right]+ && (4.25)\\
&\quad (3\eta_{12}-\eta_{30})(\eta_{21}+\eta_{03})\left[3(\eta_{30}+\eta_{12})^2-(\eta_{21}+\eta_{03})^2\right]
\end{aligned}$$

and Hall [105] demonstrates this invariance for a number of real digital images that have been scaled, translated and rotated. Small differences are observed in the computed invariants and these are attributed to the discrete nature of the data.

Region moments can also be determined from the vertices of a polygon or the perimeter points of a boundary representation [278]. For n boundary points labelled $1\cdots n$ where point $P_0 \equiv P_n$

$$m_{pq} = \frac{1}{p+q+2}\sum_{\ell=1}^{n} A_\ell \sum_{i=0}^{p}\sum_{j=0}^{q}\frac{(-1)^{i+j}}{i+j+1}\binom{p}{i}\binom{q}{j} x_\ell^{p-i} y_\ell^{q-j}\Delta x_\ell^{i}\Delta y_\ell^{j} \tag{4.26}$$

where $A_\ell = x_\ell\Delta y_\ell - y_\ell\Delta x_\ell$, $\Delta x_\ell = x_\ell - x_{\ell-1}$ and $\Delta y_\ell = y_\ell - y_{\ell-1}$. An alternative formulation [278] is more suitable for computing a fixed set of moments during traversal of a chain-coded boundary.

4.1.3 Binary region features

Binary image processing and centroid determination is commonly used in field or frame rate vision systems for visual control. Here we look at how the estimation of object centroid and width are affected by threshold, noise, edge gradient and pixel fill factor. The analytic approach developed here differs from that given by Haralick and Shapiro [107] and Ho [118] in its explicit modelling of the thresholding process.

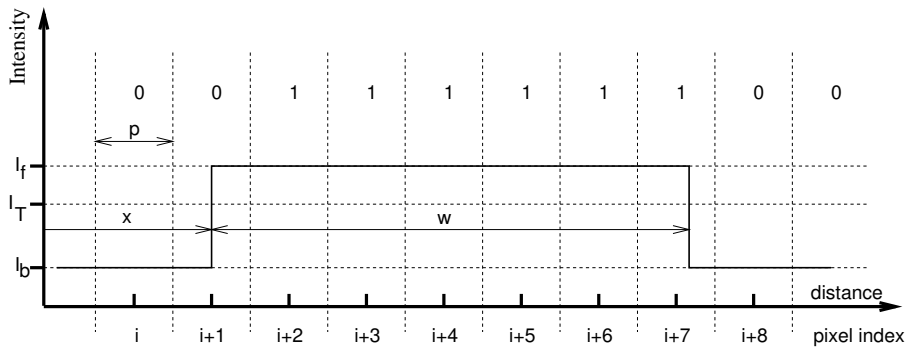


Figure 4.5: The ideal sensor array showing rectangular image and notation.

4.1.3.1 Effect on width measurement

Consider the sensor as a one dimensional array of light sensitive areas, each of width p, with no gap between them as shown in Figure 4.5. The image of an object of width w and intensity I_f is formed on the array. The background intensity is I_b. The output of a sensing site is a binary function of the average illumination over its area and the threshold I_T such that $I_b < I_T < I_f$. The pixel is set if the fraction of its sensing area covered by the object exceeds

$$T = \frac{I_f - I_T}{I_f - I_b} \tag{4.27}$$

The i^{th} pixel is centered about ip where i is an integer, and spans the range $(i-1/2)p$ to $(i+1/2)p$. It can be shown that the binary output of the i^{th} pixel for an edge at position x is given by

$$L_i(x) = H\left(ip - x + pT - \frac{p}{2}\right) \tag{4.28}$$

$$R_i(x) = H\left(x - ip + pT - \frac{p}{2}\right) \tag{4.29}$$

for the left and right hand sides of the object respectively, and where $H(x)$ is the Heaviside unit-step function. Substituting $T' = T - 1/2$, the index of the leftmost and rightmost set pixel are given by

$$i_L(x) = \text{ceil}\left(\frac{x}{p} - T'\right) \tag{4.30}$$

$$i_R(x) = \text{floor}\left(\frac{x}{p} + T'\right) \tag{4.31}$$

where ceil(x) is the smallest integer such that ceil$(x) \geq x$, and floor(x) is the largest integer such that floor$(x) \leq x$. The edge pixel positions are a function of the threshold used, as well as the background and foreground intensities.

The measured, quantized, width is

$$\begin{aligned} W^*(x) &= i_R(x+w) - i_L(x) + 1 && (4.32) \\ &= \text{floor}\left(\frac{x+w}{p} + T'\right) - \text{ceil}\left(\frac{x}{p} - T'\right) + 1 && (4.33) \end{aligned}$$

which can be simplified by substituting for distances normalized to the pixel width

$$x' = \frac{x}{p}, \;\; w' = \frac{w}{p} \tag{4.34}$$

so that (4.33) becomes

$$W^*(x') = \text{floor}(x' + w' + T') - \text{ceil}(x' - T') + 1 \tag{4.35}$$

which is a two-valued function with a period of one pixel width, as the object moves across the sensing array. In order to understand the distribution of quantized width estimates it is useful to substitute

$$\begin{aligned} x' &= x'^* + x && (4.36) \\ w' &= w'^* + w && (4.37) \end{aligned}$$

where $x'^* = \text{floor}(x')$, $w'^* = \text{floor}(w')$, and $x, w \in [0\ 1)$. By inspection it is clear that for integer values of θ^*

$$\begin{aligned} \text{floor}(\theta^* + x) &\equiv \theta^* + \text{floor}(x) && (4.38) \\ \text{ceil}(\theta^* + x) &\equiv \theta^* + \text{ceil}(x) && (4.39) \end{aligned}$$

so equation (4.35) can be rewritten as

$$\begin{aligned} W^*(x') &= \text{floor}(x' + w' + T') - \text{ceil}(x' - T') + 1 && (4.40) \\ &= \text{floor}(x'^* + x + w'^* + w + T') - \text{ceil}(x'^* + x - T') + 1 && (4.41) \\ &= w'^* + 1 + \text{floor}(x + w + T') - \text{ceil}(x - T') && (4.42) \end{aligned}$$

which is a periodic two-valued function of x. That is, a single quantized width measurement W^* switches between two values that bracket the actual width w'. For many measurements, and assuming a uniform distribution for x, the expected value of $W^*(x)$ can be shown to be

$$\begin{aligned} E(W^*) &= w'^* + w + 2T' && (4.43) \\ &= w' + 2T' && (4.44) \end{aligned}$$

The average width measurement as the object moves with respect to the sensing array is dependent upon the threshold selected. It will give an unbiased estimate of width only when $T' = 0$, or $T = 1/2$. From (4.27) this situation occurs only when the intensity threshold T is midway between the foreground and background intensities.

4.1.3.2 Accuracy of centroid estimate

Using a similar approach the accuracy of centroid determination in the horizontal direction can be derived. The quantized centroid of the object in Figure 4.5 is

$$\begin{aligned}
\overline{x}^*(x') &= \frac{i_R(x'+w')+i_L(x')}{2} && (4.45)\\
&= \frac{1}{2}\left\{\text{floor}(x'+w'+T')+\text{ceil}(x'-T')\right\} && (4.46)\\
&= \frac{1}{2}\left\{\text{floor}(x'^*+x+w'^*+w+T')+\text{ceil}(x'^*+x-T')\right\} && (4.47)\\
&= \frac{1}{2}\left\{2x'^*+w'^*+\text{floor}(x+w+T')+\text{ceil}(x-T')\right\} && (4.48)
\end{aligned}$$

which is again a periodic two-valued function. The expected value of $\overline{x}^*(x)$ is

$$\begin{aligned}
E(\overline{x}^*) &= \frac{1}{2}\left(2x'^*+w'^*+w+1\right) && (4.49)\\
&= x'^*+\frac{w'}{2}+\frac{1}{2} && (4.50)
\end{aligned}$$

and the true centroid is

$$\overline{x} = x' + \frac{w'}{2}, \tag{4.51}$$

so the expected value of error between them is

$$E(\overline{x}-\overline{x}^*) = 0 \tag{4.52}$$

indicating that the centroid estimate is an unbiased estimate of the centroid and unlike the width estimate is independent of threshold.

4.1.3.3 Centroid of a disk

The centroid of a disk in the horizontal direction can be considered the weighted mean of the centroids of each horizontal line segment comprising the image of the disk. Intuitively we would expect that the larger the disk and the greater the number of line segment centroids averaged, the closer that average would approach the expected value from (4.50). Ho [118] discusses this issue and derives the approximation

$$\sigma^2_{X_C} = \frac{1}{9\pi^2}\left(\frac{4}{d}+\frac{1}{d^3}\right) \tag{4.53}$$

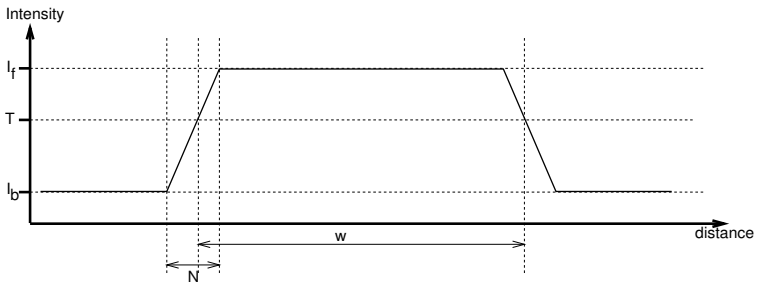


Figure 4.6: The effect of edge gradients on binarized width.

where d is the disk's diameter. Ravn et al. [13] show a simulation of centroid variance as a function of disk diameter which has approximately this characteristic.

4.1.3.4 Effect of edge intensity gradients

So far the the discussion has concentrated on the case where the image has a rectangular intensity profile but previous sections of this chapter have discussed mechanisms which reduce edge sharpness as shown in Figures 3.26 and 3.27. In a visual servo system edge gradient may be further reduced by focus errors as the object distance varies. The resulting edge may be more accurately modelled as a trapezoidal intensity profile as shown in Figure 4.6. Changes in threshold, ΔT, or ambient illumination, ΔI_b, will have a marked effect on the width of the binarized object,

$$\Delta W = 2\frac{\Delta I_b - \Delta T}{\rho} \tag{4.54}$$

where ρ is the edge gradient in units of greylevel/pixel. In practice, with edge widths of up to 5 pixels, this effect will dominate errors in the measured width of the binary object.

4.1.3.5 Effect of signal to noise ratio

Image noise sources were discussed in Section 3.3.7. The foreground and background image intensities should accurately be considered as random variables with known distributions. Provided that the two means are well separated it is possible to segment them using a fixed threshold.

Using the noise measurement data for the Pulnix camera, shown in Figure 3.18, the maximum variance can be taken as 4 greylevels^2. If a normal distribution is assumed then 99% of all pixels will be within ± 6 greylevels of the mean. The probability of

a pixel being greater than ± 10 greylevels from the mean is extremely low, around 1 pixel from the entire 512×512 pixel image.

In a scene with high contrast and a well chosen threshold, camera noise is unlikely to be a problem. However practical issues such as uneven scene illumination or the $\cos^4$ effect in (3.12) may bring some parts of the image close to the threshold resulting in patches of binary pixel noise. A conservative rule of thumb would be to keep the threshold at least 3σ above the background greylevel and at least 3σ below the foreground greylevel, keeping in mind that σ is a function of intensity.

Camera noise on the pixels where the edge intensity crosses the threshold will add uncertainty to the edge of the binary image and may even result in isolated noise pixels adjacent to the edge. The probability of noise changing the threshold outcome is inversely proportional to the edge gradient. Simple geometry results in another rule of thumb that $\rho > 3\sigma$. For worst-case observed camera variance of 4, this would require $\rho > 6$ greylevel/pixel which is easily satisfied. Figure 3.26 for example shows edge gradients of at least 50 greylevel/pixel.

Noise is manifested as single pixels of opposite color to their neighbours which can be readily eliminated by a median filter. Such a filter sets the pixel to the median value of all its neighbours. For a binary image such a filter can be implemented using simple logic.

4.1.3.6 Numerical simulation

The analytic approaches of Sections 4.1.3.1 and 4.1.3.2 become intractable when trying to model edge gradient, pixel fill factor and additive noise. Instead a numerical approach has been used which models a trapezoidal intensity profile moving one pixel across the sensing array in a large number of small steps while statistics are gathered about the mean width and centroid error. Conclusions which can be drawn from the simulations are:

- The threshold dependence of mean width error, (4.44), is verified for a rectangular intensity profile.
- The effect of threshold and intensity change on mean width error, (4.54), for the case of trapezoidal intensity profile is verified.
- The mean centroid error, (4.52), is 0 and independent of threshold.
- Reducing pixel fill factor, $h < 1$, has no apparent effect on mean error or variance of width or centroid.
- Additive noise has very little effect on mean centroid and width error for a well chosen threshold. As the threshold approaches either foreground or background intensity, mean width error increases by several pixels, but mean centroid error is unaffected.

- As edge gradient is reduced the mean width and centroid error are unchanged, but the variance increases.

Simulations have also been performed to examine the effect of object motion and finite exposure time, T_e, on an object with significant edge gradient or an asymmetric intensity profile. It was found that, irrespective of intensity profile, the centroid of the binarized blurred image is the centroid of the binarized image halfway through the exposure interval, as derived earlier in (3.47). It should be noted in this case that the centroid of the binary image does not correspond to the centroid of the object if the intensity profile is asymmetric.

4.1.3.7 Summary

For accurate determination of distance with a possibly unfocussed camera it is preferable to use the distance between centroids of features rather than the width of a feature. The latter is significantly affected by variation in threshold and illumination when edges are wide. Alexander [6] has shown, in the context of camera calibration, that spatial aliasing due to image sharpness can in fact introduce a small but systematic error in centroid determination. He shows how this may be minimized by introducing diffraction blur. Given the poor edge response of the lens used in this work problems due to excessive image sharpness are not expected to be significant. It can be concluded that if centroids of image features are used it is not necessary to have a sharp image since the centroid estimate is unbiased.

4.1.4 Feature tracking

Software computation of image moments is one to two orders of magnitude slower than specialized hardware. However the computation time can be greatly reduced if only a small image window, whose location is predicted from the previous centroid, is processed [77, 92, 99, 108, 212, 274]. The task of locating features in sequential scenes is relatively easy since there will be only small changes from one scene to the next [77, 193] and total scene interpretation is not required. This is the principle of verification vision proposed by Bolles [34] in which the system has considerable prior knowledge of the scene, and the goal is to verify and refine the location of one or more features in the scene. Determining the initial location of features requires the entire image to be searched, but this need only be done once. Papanikolopoulos et al. [197] use a sum-of-squared differences approach to match features between consecutive frames. Features are chosen on the basis of a confidence measure computed from the feature window, and the search is performed in software. The TRIAX system [19] is an extremely high-performance multiprocessor system for low latency six-dimensional object tracking. It can determine the pose of a cube by searching short check lines normal to the expected edges of the cube.

When the software feature search is limited to only a small window into the image, it becomes important to know the expected position of the feature in the image. This is the target tracking problem; the use of a filtering process to generate target state estimates and predictions based on noisy observations of the target's position and a dynamic model of the target's motion. Target maneuvers are generated by acceleration controls unknown to the tracker. Kalata [141] introduces tracking filters and discusses the similarities to Kalman filtering. Visual servoing systems have been reported using tracking filters [8], Kalman filters [77, 274], AR (auto regressive) or ARX (auto regressive with exogenous inputs) models [91, 124]. Papanikolopoulos et al. [197] use an ARMAX model and consider tracking as the design of a second-order controller of image plane position. The distance to the target is assumed constant and a number of different controllers such as PI, pole-assignment and LQG are investigated. For the case where target distance is unknown or time-varying adaptive control is proposed [196]. The prediction used for search window placement can also be used to overcome latency in the vision system and robot controller.

Dickmanns [77] and Inoue [128] have built multiprocessor systems where each processor is dedicated to tracking a single distinctive feature within the image. More recently, Inoue [129] has demonstrated the use of a specialized VLSI motion estimation device for fast feature tracking.

4.2 Perspective and photogrammetry

The perspective transform involved in imaging was introduced in Section 3.2.4 and the lens equations in pixel coordinates as (3.66) and (3.67). Using homogeneous coordinates this non-linear transformation may be expressed in linear form for an arbitrary camera location. Using matrix representation the overall camera transformation is:

$$\begin{bmatrix} {}^{i}x \\ {}^{i}y \\ {}^{i}z \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & X_0 & 0 \\ 0 & \alpha_y & Y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/f & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \left({}^{0}\mathbf{T}_c\right)^{-1} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.55}$$

where

α_x	X-axis scaling factor in pixels/mm (intrinsic)
α_y	Y-axis scaling factor in pixels/mm (intrinsic)
X_0	image plane offset in pixels (intrinsic)
Y_0	image plane offset in pixels (intrinsic)
f	focal length (intrinsic)
${}^{0}\mathbf{T}_c$	camera position in world coordinates (extrinsic), see Figure 4.7.

The image plane coordinates in pixels are then expressed in terms of homogeneous

coordinates as

$$ {}^{i}X = \frac{{}^{i}x}{{}^{i}z} \tag{4.56} $$

$$ {}^{i}Y = \frac{{}^{i}y}{{}^{i}z} \tag{4.57} $$

in units of pixels.

Intrinsic parameters are innate characteristics of the camera and sensor, while *extrinsic* parameters are characteristics only of the position and orientation of the camera. The *principal point* is the intersection of the optical axis and the CCD sensor plane, at pixel coordinates (X_0, Y_0).

The relevant transformations and coordinate frames are shown in Figure 4.7. ${}^{c}\mathbf{T}_P$ is the position of the object with respect to the camera. Other relationships include

$$ {}^{0}\mathbf{T}_P = {}^{0}\mathbf{T}_c \, {}^{c}\mathbf{T}_P \tag{4.58} $$

$$ {}^{c}\mathbf{T}_P = \left({}^{0}\mathbf{T}_c\right)^{-1} {}^{0}\mathbf{T}_P \tag{4.59} $$

which allow (4.55) to be written compactly as

$$ \begin{bmatrix} {}^{i}x \\ {}^{i}y \\ {}^{i}z \end{bmatrix} = \mathbf{C} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{4.60} $$

where $\mathbf{C}$ is the *camera calibration matrix*, a 3×4 homogeneous transform which performs scaling, translation and perspective. $\mathbf{C}$ represents the relationship between 3-D world coordinates and their corresponding 2-D image coordinates as seen by the computer.

Equation (4.55) may be expanded symbolically. With the inverse camera position transform represented in direction vector form ${}^{c}\mathbf{T}_0 = \left[\underline{n}\ \underline{o}\ \underline{a}\ \underline{p}\right]$ we can derive

$$ \mathbf{C} = \begin{bmatrix} \frac{\alpha_x n_x f - X_0 n_z}{f - c_z} & \frac{\alpha_x o_x f - X_0 o_z}{f - c_z} & \frac{\alpha_x a_x f - X_0 a_z}{f - c_z} & \frac{\alpha_x c_x f - X_0 c_z + X_0 f}{f - c_z} \\ \frac{\alpha_y n_y f - Y_0 n_z}{f - c_z} & \frac{\alpha_y o_y f - Y_0 o_z}{f - c_z} & \frac{\alpha_y a_y f - Y_0 a_z}{f - c_z} & \frac{\alpha_y c_y f - Y_0 c_z + Y_0 f}{f - c_z} \\ -\frac{n_z}{f - c_z} & -\frac{o_z}{f - c_z} & -\frac{a_z}{f - c_z} & 1 \end{bmatrix} \tag{4.61} $$

4.2.1 Close-range photogrammetry

Photogrammetry is the science of obtaining information about physical objects via photographic images, and is commonly used for making maps from aerial photographs [282]. Close-range, or terrestrial, photogrammetry is concerned with object distances less than 100 m from the camera. Much nomenclature from this discipline is used in the literature related to camera calibration and 3D vision techniques.

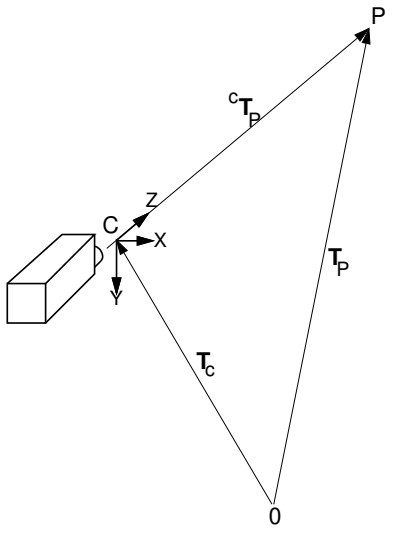


Figure 4.7: Relevant coordinate frames.

A *metric* camera is purpose built for photogrammetric use and is both stable and calibrated. The calibration parameters include the focal length, lens distortion, and the coordinates of the principal point. A metric camera has fiducial marks which are recorded on the photograph such that lines joining opposite fiducial marks intersect at the principal point.

A *nonmetric* camera is one manufactured for photographic use, where picture quality, not geometry, is important. Nonmetric cameras can be calibrated and used for less demanding photogrammetric applications such as machine vision.

4.2.2 Camera calibration techniques

Camera calibration is the process of determining the internal camera geometric and optical characteristics (intrinsic parameters) and the 3-D position and orientation of the camera frame relative to a certain world coordinate system (extrinsic parameters). Frequently it is useful to empirically determine the camera calibration matrix which relates a world coordinate system to the image plane coordinates [27] for a given camera position and orientation.

The important characteristics of a number of calibration techniques are summarized in Table 4.1. They differ in their ability to calibrate lens distortion, and the type of calibration target required. Charts with coplanar points can be generated conve-

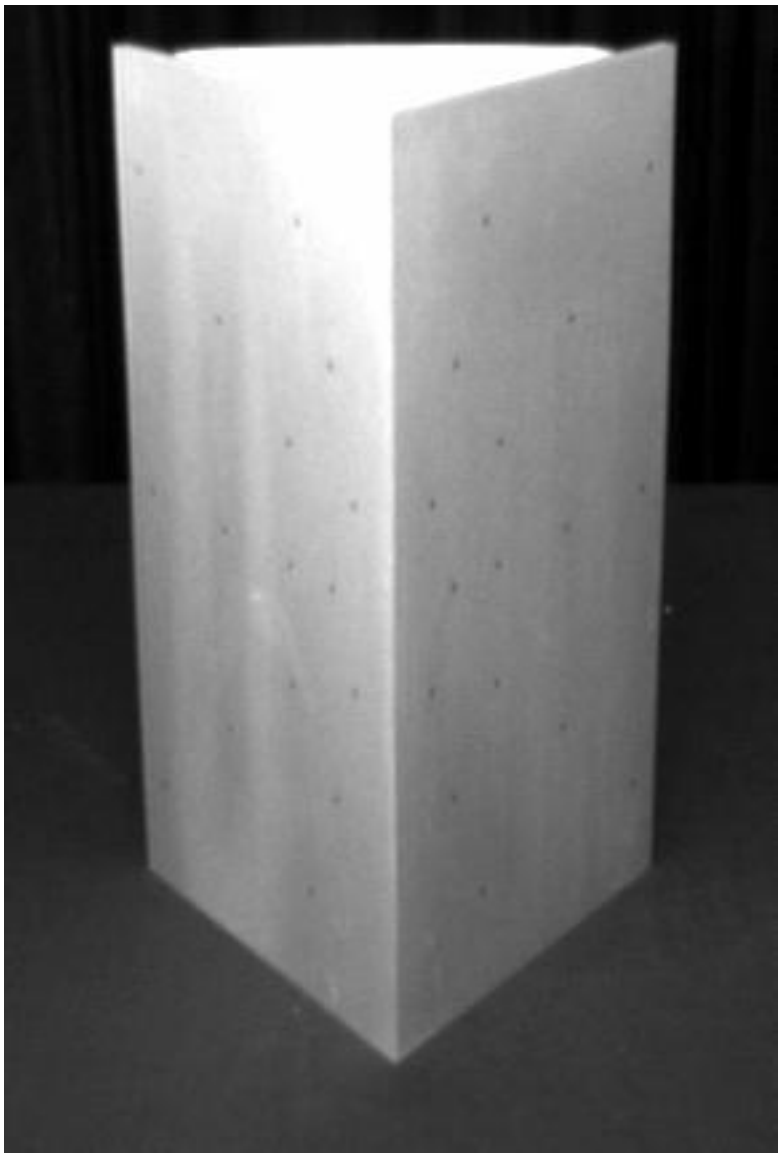


Figure 4.8: The SHAPE calibration target used for intrinsic parameter determination. The surface is brushed aluminium with black marker dots at accurately known locations on the two surfaces.

Method	Target type		Distortion
	Coplanar	Non-coplanar	
Classic nonlinear		•	•
Homogeneous transform		•	
2 planes	•		•
2 stage	•		•

Table 4.1: Comparison of the major features of different camera calibration techniques.

niently and accurately[1] using a laser printer. Circular marks are frequently used for centroid determination but errors will be introduced if these are not viewed along the surface normal. In that case, as shown earlier in Figure 4.3, the centroid of the image (which will be an ellipse) will not correspond with the centroid of the marker. This effect can be minimized by keeping the mark size small (image size of a few pixels), using a cross shaped target, or using the corners of rectangular markers. Calibration using non-coplanar points requires a mechanically complex calibration frame such as that shown in Figure 4.8, or motion of a robot to position a calibration mark with respect to the camera. Experiments conducted using the robot resulted in large residuals and this is suspected to be due to low positioning accuracy. The next sections briefly describe the different approaches to camera calibration and present some experimental results.

4.2.2.1 Classical non-linear approach

Techniques in this category have been developed in the photogrammetric community and are amongst the oldest published references [87, 239, 282, 283]. The camera is described by a detailed model, with in some cases up to 18 parameters. Non-linear optimization techniques use the calibration data to iteratively adjust the model parameters.

4.2.2.2 Homogeneous transformation approach

Approaches based on the homogeneous transformation [27, 243] allow direct estimation of the calibration matrix $\mathbf{C}$ in (4.60). The elements of this matrix are composites of the intrinsic and extrinsic parameters. Lens distortion cannot be represented in linear homogeneous equations, but this can be corrected in a separate computational step.

Expanding equations (4.60)-(4.57) we may write

$$C_{11}x + C_{12}y + C_{13}z + C_{14} - C_{31}{}^{i}Xx - C_{32}{}^{i}Xy - C_{33}{}^{i}Xz - C_{34}{}^{i}X = 0 \qquad (4.62)$$

[1] Less than 1% scale error measured in the paper feed direction.

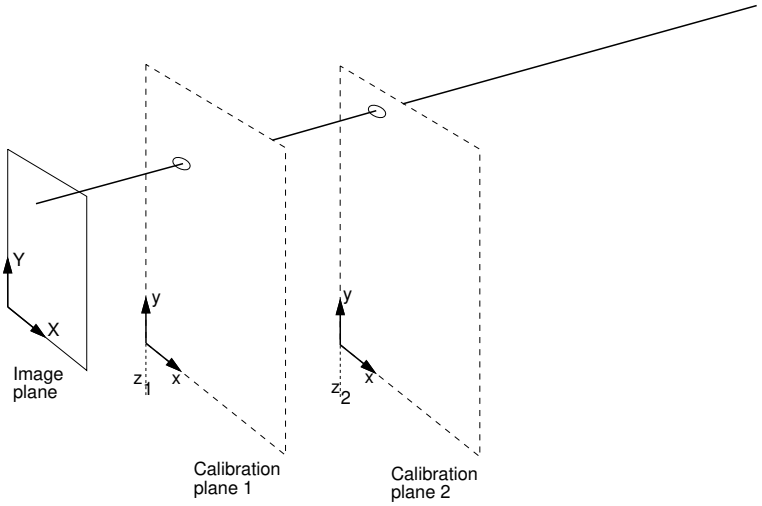


Figure 4.9: The two-plane camera model.

$$C_{21}x + C_{22}y + C_{23}z + C_{24} - C_{31}{}^iYx - C_{32}{}^iYy - C_{33}{}^iYz - C_{34}{}^iY = 0 \tag{4.63}$$

which relate an observed image coordinate $({}^iX, {}^iY)$ to a world coordinate (x, y, z). For n observations this may be expressed in matrix form as a homogeneous equation

$$\begin{bmatrix} x_1 & y_1 & z_1 & 1 & 0 & 0 & 0 & 0 & -{}^iX_1x_1 & -{}^iX_1y_1 & -{}^iX_1z_1 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & z_1 & 1 & -{}^iY_1x_1 & -{}^iY_1y_1 & -{}^iY_1z_1 \\ & & & & & \vdots & & & & & \\ x_n & y_n & z_n & n & 0 & 0 & 0 & 0 & -{}^iX_nx_n & -{}^iX_ny_n & -{}^iX_nz_n \\ 0 & 0 & 0 & 0 & x_n & y_n & z_n & n & -{}^iY_nx_n & -{}^iY_ny_n & -{}^iY_nz_n \end{bmatrix} \begin{bmatrix} C_{11} \\ C_{12} \\ \vdots \\ C_{33} \end{bmatrix} = \begin{bmatrix} {}^iX_1 \\ {}^iY_1 \\ \vdots \\ {}^iX_n \\ {}^iY_n \end{bmatrix} \tag{4.64}$$

A non-trivial solution can only be obtained to within a scale factor and by convention C_{34} is set equal to 1. Equation (4.64) has 11 unknowns and for solution requires at least 5.5 observations (pairs of $({}^iX, {}^iY)$ and (x, y, z) of non-coplanar points for solution[2]. This system of equations will generally be over determined, and a least squares solution may be obtained using a technique such as singular value decomposition.

[2]Coplanar points result in the left-hand matrix of (4.64) becoming rank deficient.

4.2.2.3 Two-plane calibration

The homogeneous transform approach is based on the pinhole camera model, and cannot be readily extended to deal with lens distortions. The two plane approach has been suggested [180] to overcome this limitation. As shown in Figure 4.9, a line in space is defined by a point on each of two calibration planes, given by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_k = \mathbf{A}_k \left[{}^iX^n \; {}^iX^{n-1} {}^iY \ldots {}^iY^n \ldots {}^iX \ldots {}^iY \; 1 \right]^T \tag{4.65}$$

where $\mathbf{A}_k$ is the interpolation matrix for calibration plane k. The two plane method allows for rapid computation of inverse perspective, that is, 3D lines from image plane points. The line is obtained by interpolating the 3D world coordinates corresponding to the image plane coordinate for the two calibration planes.

The interpolation matrix for each plane may be determined by placing a test pattern containing an array of dots at a known distance from the camera. The order of the interpolation should be chosen to balance computation time, accuracy and stability — in practice second or third order is sufficient [130]. The interpolation matrix implicitly corrects for lens distortion.

4.2.2.4 Two-stage calibration

The two-stage calibration scheme of Tsai [252] can be used to determine the intrinsic and extrinsic camera parameters from a single view of a planar calibration target. It relies on the so-called 'radial alignment constraint' which embodies the fact that lens distortion acts along radial lines from the principal point. The algorithm is moderately complex, but is claimed to execute very quickly. It does require prior knowledge of the digitizer to camera clock ratio, β, and the pixel scaling factors α_x and α_y, as well as the coordinates of the principal point. Tsai considers this latter parameter unimportant and sets it arbitrarily to the coordinate of the center of the framestore. As found later, the principal point for this camera and digitizer is some distance from the center.

Considerable difficulty was experienced using Tsai's method, and in particular it proved to very sensitive to slight changes in the parameter β and principal point coordinate. Anecdotal evidence from other researchers and also [210] suggest that this experience is not uncommon. One possible reason for the difficulty encountered is that the lens has insufficient radial distortion, though this would be surprising for such a short focal length lens.

4.2.2.5 Decomposing the camera calibration matrix

The homogeneous camera calibration matrix, $\mathbf{C}$, contains information about the position of the camera with respect to the world coordinate frame, as well as intrinsic

parameters of the camera. Determining the extrinsic parameters is also referred to as the *camera location determination problem*.

The line of sight of the camera in world coordinates can be determined as follows. For a given image plane point $({}^iX, {}^iY)$ we can write

$$(C_{11} - {}^iXC_{31})x + (C_{12} - {}^iXC_{32})y + (C_{13} - {}^iXC_{31})z = {}^iXC_{34} - C_{14} \tag{4.66}$$

$$(C_{21} - {}^iYC_{31})x + (C_{22} - {}^iYC_{32})y + (C_{23} - {}^iYC_{31})z = {}^iYC_{34} - C_{24} \tag{4.67}$$

which are the equations of two planes in world space, one vertical corresponding to constant ${}^iX$ in the image plane, and one horizontal. The intersection of these planes is the line in 3D space that is mapped to that image plane point[3]. A direction vector parallel to this intersection line is found via cross product

$$\begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ (C_{11} - {}^iXC_{31}) & (C_{12} - {}^iXC_{32}) & (C_{13} - {}^iXC_{31}) \\ (C_{21} - {}^iYC_{31}) & (C_{22} - {}^iYC_{32}) & (C_{23} - {}^iYC_{31}) \end{vmatrix} \tag{4.68}$$

Substituting ${}^iX = X_0$ and ${}^iY = Y_0$ gives a direction vector parallel to the camera's optical axis. However X_0 and Y_0 are intrinsic calibration parameters and cannot be assumed to be the center of the sensing array [169].

Alternatively the lens law singularity may be used to define the camera's focal plane, that is $z = f$. The focal plane is given by

$$C_{31}x + C_{32}y + C_{33}z = 0 \tag{4.69}$$

and the camera axis direction vector is normal to this plane, parallel to the vector

$$C_{31}\hat{x} + C_{32}\hat{y} + C_{33}\hat{z} \tag{4.70}$$

Strat [242] extends the approach of (4.69) to also determine camera position and roll about the optical axis.

The calibration matrix, **C**, encodes the 6 extrinsic parameters[4] and 5 intrinsic parameters: f, α_x, α_y, X_0 and Y_0. However it is not possible to uniquely determine all the intrinsic parameters, only the products $f\alpha_x$ and $f\alpha_y$[5]. This leaves 10 parameters to be determined from 11 elements of the camera calibration matrix which is over determined, leading to difficulties in solution. Ganapathy [97] describes a sophisticated procedure for determining the intrinsic and extrinsic parameters by introducing another parameter, δ, a quality measure in units of degrees which is ideally zero. This is interpreted as error in the orthogonality of the image plane X and Y axes. The approach is robust to inaccuracies in the calibration matrix.

[3] Equations (4.66) and (4.67) represent 2 equations in 3 unknowns. From two camera views of the same point, we are then able to solve for the 3D location of the point — the basis of stereo vision.

[4] 3D position vector and 3 angles for camera pose.

[5] From the symbolic expansion of (4.61) it can be seen that only the products $f\alpha_x$ and $f\alpha_y$ appear.

Given a numerical value for **C** it is also possible using gradient search techniques to find the values of the intrinsic and extrinsic parameters. In practice this approach is found to be quite sensitive to choice of initial values and can settle in local minima.

4.2.2.6 Experimental results

Four images of the SHAPE calibration target, shown in Figure 4.8, were taken from different viewpoints. Each time the focus and aperture were adjusted for image sharpness and brightness. The centroid of each marker was determined to sub-pixel precision by computing an intensity weighted average of pixel coordinates in the region of the marker. As shown in Figure 4.10 the intensity profile of a marker is a broad flat hill not a step function, and this would be expected from the earlier discussion regarding image MTF. The centroid determination procedure used is as follows:

1. An image display utility was used to manually determine the location of each mark to within ± 5 pixels. The target was difficult to light evenly due to its complex shape, making it infeasible to automatically and robustly find the markers.
2. The lowest[6] intensity pixel within a 10 x 10 region of each manually selected point was located. This point will be referred to as the center point (x_c, y_c).
3. The average background intensity, I_b, was computed from the intensity at the four corners of a square window about the center point.
4. Average background intensity was subtracted from all pixels in the region and the weighted average coordinates computed

$$\overline{x} = \frac{\Sigma_i \Sigma_j \; i(I_{ij} - I_b)}{\Sigma_i \Sigma_j \; (I_{ij} - I_b)} \tag{4.71}$$

$$\overline{y} = \frac{\Sigma_i \Sigma_j \; j(I_{ij} - I_b)}{\Sigma_i \Sigma_j \; (I_{ij} - I_b)} \tag{4.72}$$

A program was written to automate these last three steps.

Using known 3D marker location and the corresponding image plane coordinates the calibration matrix was computed by solving (4.64) using singular value decomposition. The resulting matrix was then decomposed using Ganapathy's method and the results are summarized in Table 4.2. *Residual* is the maximum residual after substituting the calibration matrix parameters back into (4.64). δ is the quality factor determined by Ganapathy's algorithm and should ideally be zero. The remaining columns show intrinsic parameters derived by Ganapathy's method. There is some variation in the principal point coordinate and also the scale factors. However the ratio of the scale

[6] The calibration marks were black on white.

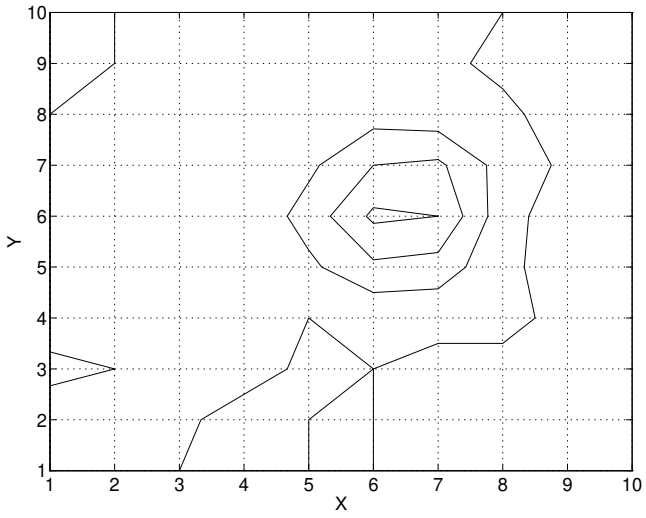


Figure 4.10: Contour plot of intensity profile around a typical calibration marker. Due to overall MTF the edges of the marker dot are not sharp.

Trial	Residual (pixel)	δ (deg)	X_0 (pixel)	Y_0 (pixel)	$\alpha_x f$ (pixel)	$\alpha_y f$ (pixel)	α_y/α_x
1	0.552	-0.0277	270.1	213.7	625.7	948.9	1.52
2	0.898	0.0672	271.4	223.1	614.0	933.0	1.52
3	0.605	0.0685	278.0	205.6	617.3	938.2	1.52
4	0.639	-0.1198	275.8	196.8	616.8	939.2	1.52
mean			273.8	209.8	615.5	939.8	
σ			3.2	9.7	5.0	6.6	

Table 4.2: Summary of calibration experiment results.

factors is constant at 1.52 which is also the value determined in Section 3.5.3. Using the pixel scale factor data from Table 3.8 the focal length is estimated as 7.8 mm. This is marginally lower than 8 mm, the nominal focal length of the lens, but within the 4% tolerance of ANSI Standard PH3.13-1958 "Focal Length Marking of Lenses". The Y-coordinate of the principal point is found to be well above the center of the pixel array. This effect could be explained by the placement of the CCD sensor within the camera at the appropriate point for an RS170 rather than CCIR image.

Many researchers have observed extreme sensitivity in the camera calibration to changes in focus and aperture setting. Changes in these settings cause lens elements to rotate, which compounded with asymmetries in the lens alter the scaling and position of the image. This may account for some of the variation observed since the lens was re-adjusted for each trial.

The procedure presented is not capable of explicitly modelling geometric lens distortion, and its effect will introduce errors into the other camera parameters. An experiment was conducted in which an array of dots, evenly spaced across the field of view, was imaged and their centroids computed. The magnitude of centroid deviation from the line of best fit was less than one pixel with the largest errors occurring at the edge of the field of view.

4.2.3 Eye-hand calibration

Robot eye to hand calibration is the process of determining the fixed transformation between the gripper and the camera coordinate system. A number of approaches have been described in the literature [132, 233, 253]. Generally they involve the robot making a number of moves and measuring the change in image plane coordinates of a fixed target. A fairly complex algorithm is then applied to determine the camera transform. Tsai's method [253] again relies on use of the planar calibration target, and given the difficulties above, was not tried.

A more pragmatic approach is to determine the transform from the known geometry of the camera, robot wrist and lens, as shown in Figure 4.11. The location of the CCD sensor plane within the camera is not directly measurable. However the lens manufacturer's data shows that the focal point of the lens is 17.5 mm behind the mating surface, and the plane of an equivalent simple lens will be located the focal length in front of that, see Figure 4.12[7]. From this data the distance d_2 can be inferred as 20.0 mm.

The coordinate frame of the camera is also shown in Figure 4.11. The X-axis is out of the page. The transform can be expressed in terms of elementary rotations and translations as

$$ {}^{6}\mathbf{T}_c = T_Z(d_1)\, R_X(-90^\circ) T_Z(d_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 20 \\ 0 & -1 & 0 & 121 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.73} $$

[7] A lens such as this, in which the lens plane is not within the body of the lens, and close to the image plane, is referred to as a *retrofocus* lens.

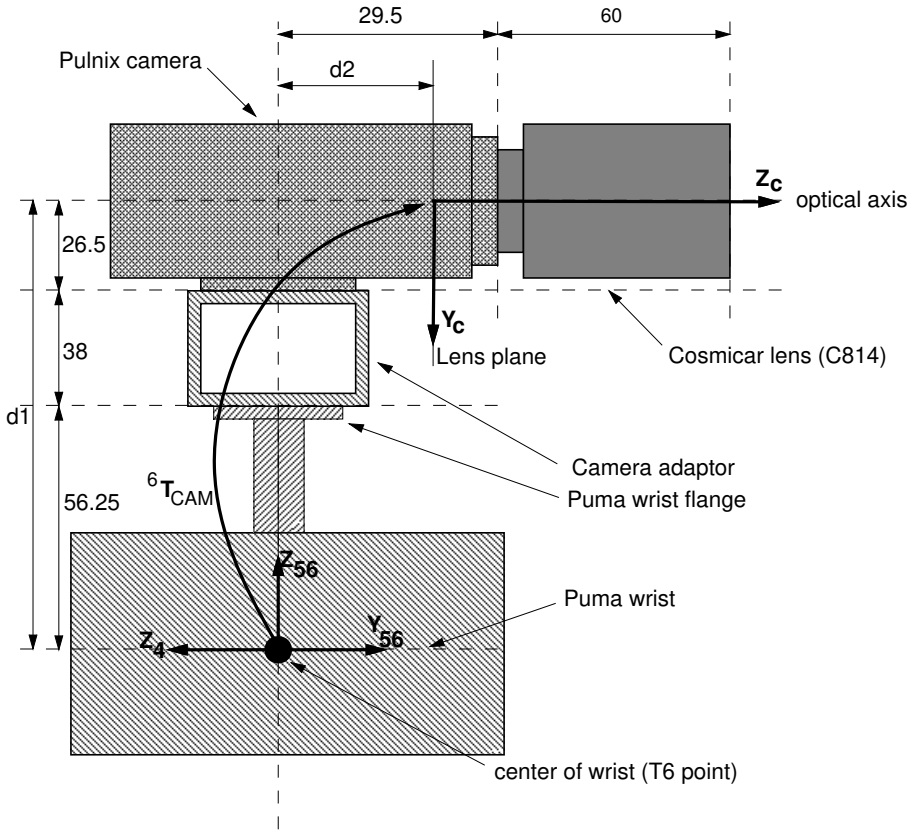


Figure 4.11: Details of camera mounting (not to scale, dimensions in mm). Robot wrist is in the zero angle pose.

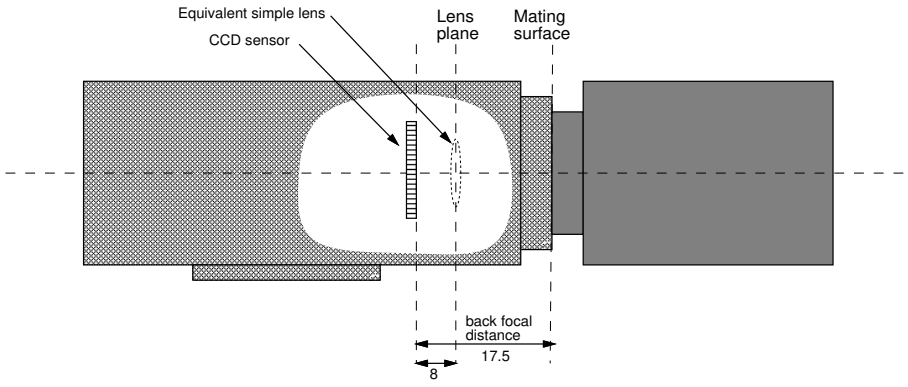


Figure 4.12: Details of camera, lens and sensor placement.

Chapter 5

Visual servoing*

With this chapter we start the discussion about visual servoing, that is, how visual features may be used to guide a robot manipulator or mechanism. The reported use of visual information to guide robots, or more generally mechanisms, is quite extensive and encompasses applications as diverse as manufacturing, teleoperation, missile tracking cameras and fruit picking as well as robotic ping-pong, juggling, catching and balancing.

Section 5.1 introduces the topic of visual servoing and introduces a consistent terminology that will be employed throughout, allowing discussion of papers despite the different nomenclature used by the authors. The section also introduces the two well known approaches to visual servoing: position-based and image-based. A more formal treatment of the fundamentals is given by Hager et al. [103].

The majority of prior work concentrates on the 'classical' issues in visual servoing, referred to in [52] as *visual servo kinematics*. This is concerned with the kinematic relationship between object pose, robot pose and image plane features without regard to dynamic effects. However control strategies based purely on kinematic considerations are only adequate for low-performance applications. *Visual servo dynamics*, on the other hand, is concerned with the dynamics or motion of the visual servo system and issues such as stability, settling time and tracking lags. This chapter concentrates largely on kinematic issues, while the following chapters address dynamics, modelling and control.

Section 5.2 provides a comprehensive review of prior work on the topic of visual servoing. Sections 5.3 and 5.4 discuss respectively the details of position- and image-based techniques. Miscellaneous issues relating to implementation and archi-

*An early version of this chapter was published as [59] "Visual control of robot manipulators — a review" in K. Hashimoto, editor, *Visual Servoing*, volume 7 of *Robotics and Automated Systems*, pages 1–31. World Scientific, 1993.

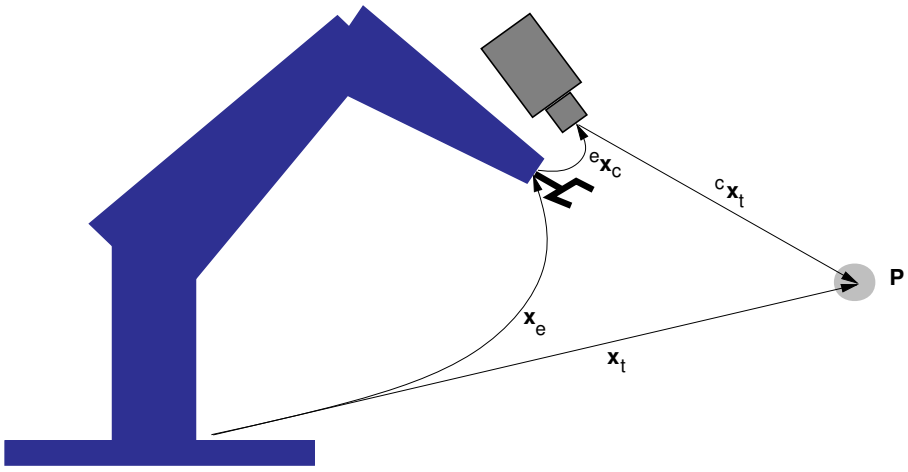


Figure 5.1: Relevant coordinate frames; world, end-effector, camera and target.

tectures are reviewed in Section 5.5. Necessarily with this structure the work of some researchers will be referred to several times, but in different contexts.

5.1 Fundamentals

The task in visual servoing is to control the *pose* of the robot's end-effector, $\underline{x}_{t6}$, using visual information, *features*, extracted from the image[2]. Pose, $\underline{x}$, is represented by a six element vector encoding position and orientation in 3D space. The camera may be fixed, or mounted on the robot's end-effector in which case there exists a constant relationship, ${}^{t6}\underline{x}_c$, between the pose of the camera and the pose of the end-effector. The image of the target[3] is a function of the relative pose between the camera and the target, ${}^{c}\underline{x}_t$. The relationship between these poses is shown in Figure 5.1. The distance between the camera and target is frequently referred to as *depth* or *range*.

The camera contains a lens which forms a 2D projection of the scene on the image plane where the sensor is located. This projection causes direct depth information to be lost, and each point on the image plane corresponds to a ray in 3D space. Some additional information is needed to determine the 3D coordinate corresponding to an image plane point. This information may come from multiple views, or knowledge of the geometric relationship between several feature points on the target.

[2]The task can also be defined for mobile robots, where it becomes the control of the vehicle's pose with respect to some landmarks.

[3]The word *target* will be used to refer to the object of interest, that is, the object which will be tracked.

An *image feature*, Section 4.1, is a scalar or vector quantity derived from some visual feature or features in the image. Commonly the coordinate of some easily distinguishable point, or a region centroid is used. A *feature vector*, $\underline{f}$, is a one dimensional vector containing feature information as described above. A good visual feature is one that can be located unambiguously in different views of the scene, such as a hole in a gasket [91,92] or a contrived pattern [71,211]. The image plane coordinates of three or more visual features can be used to determine the pose (not necessarily uniquely, see Section 5.3.1) of the target relative to the camera, given knowledge of the geometric relationship between the feature points.

Robots typically have 6 degrees of freedom (DOF), allowing the end-effector to achieve, within limits, any pose in 3D space. Visual servoing systems may control 6 or fewer DOF. Planar positioning involves only 2-DOF control and may be sufficient for some applications. Motion so as to keep one point in the scene, the interest point, at the same location in the image plane is referred to as *fixation*. Animals use fixation to direct the high resolution fovea of the eye toward regions of interest in the scene. In humans this low-level, unconscious, fixation motion is controlled by the brain's medulla region using visual feedback from the retina [5]. Keeping the target centred in the field of view has a number of advantages that include:

- eliminating motion blur since the target is not moving with respect to the camera;
- reducing the effect of geometric distortion in the lens by keeping the optical axis pointed at the target;
- minimizing the effect of the $\cos^4$ term in (3.12) since the angle concerned is close to zero.

Fixation may be achieved by controlling the pan/tilt angles of the camera like a human eye, or by moving the camera in a plane normal to the optical axis. High performance fixation control is an important component of many active vision strategies.

In 1980 Sanderson and Weiss [223] introduced an important classification of visual servo structures, and these are shown schematically in Figures 5.2 to 5.5. In *position-based* control, features are extracted from the image and used in conjunction with a geometric model of the target to determine the pose of the target with respect to the camera. In *image-based* servoing the last step is omitted, and servoing is done on the basis of image features directly. The structures referred to as *dynamic look and move* make use of joint feedback, whereas the PBVS and IBVS structures use no joint position information at all. It is important to note that almost none of the visual servo systems reported in the literatue are 'visual servo' systems according to the Weiss taxonomy, but rather are of the 'dynamic look and move' structure. However the term 'visual servoing' is widely used for any system that uses a machine vision system to close a position-control loop.

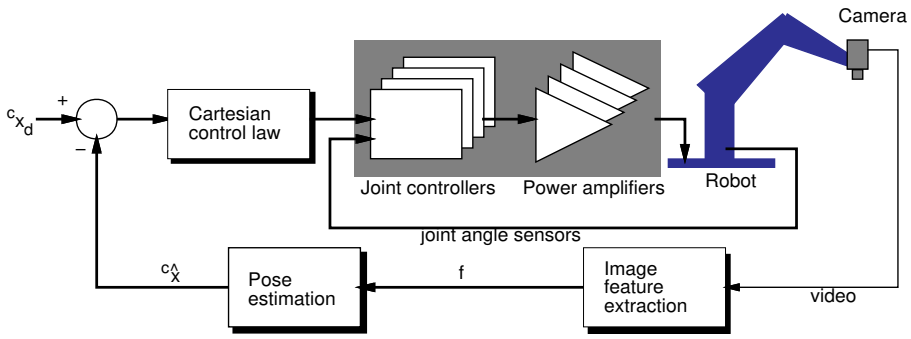


Figure 5.2: Dynamic position-based look-and-move structure.

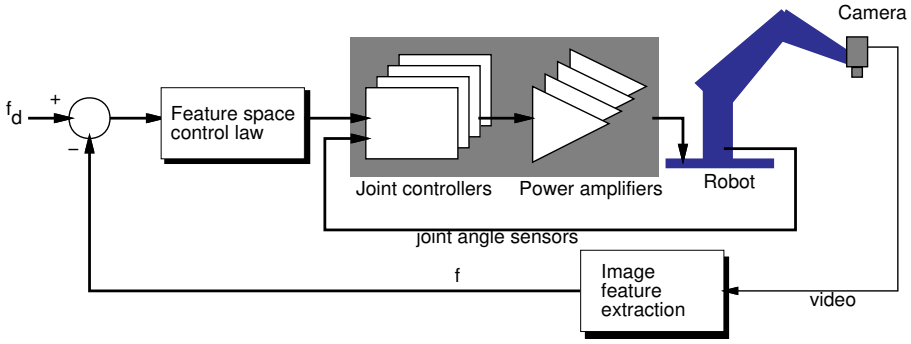


Figure 5.3: Dynamic image-based look-and-move structure.

The image-based approach may reduce computational delay, eliminate the necessity for image interpretation and eliminate errors in sensor modelling and camera calibration. However it does present a significant challenge to controller design since the plant is non-linear and highly coupled.

5.2 Prior work

This section summarizes research and applications of visual servoing, from the pioneering work of the early 1970s to the present day. The discussion is generally chronological, but related applications or approaches will be grouped together. The reported applications are quite extensive, encompassing manufacturing, teleoperation,

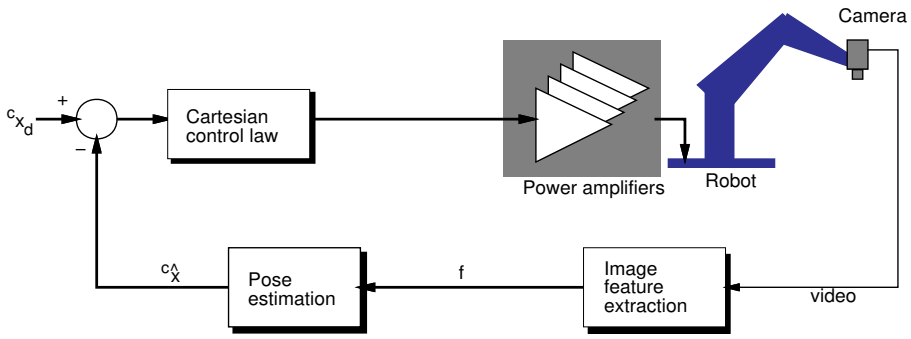


Figure 5.4: Position-based visual servo (PBVS) structure as per Weiss.

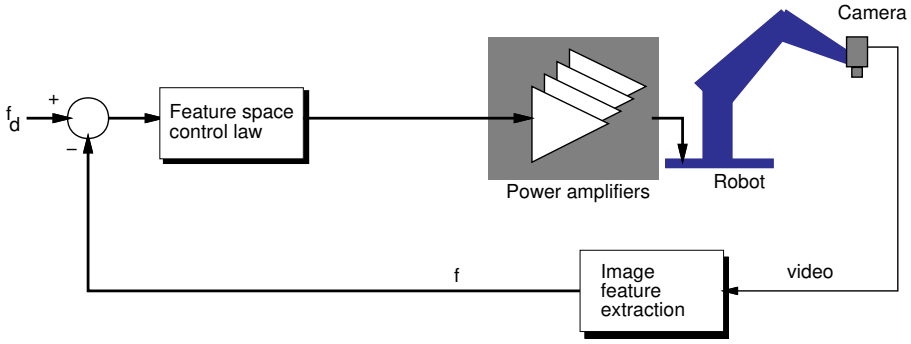


Figure 5.5: Image-based visual servo (IBVS) structure as per Weiss.

missile tracking cameras and fruit picking as well as robotic ping-pong, juggling, and balancing. Due to technological limitations of the time some of the significant early work fails to meet the strict definition of visual servoing given earlier, and would now be classed as *look-then-move* robot control. Progress has however been rapid, and by the end of the 1970s systems had been demonstrated which were capable of 10 Hz servoing and 3D position control for tracking, seam welding and grasping moving targets.

One of the earliest references is by Shirai and Inoue [232] in 1973 who describe how a visual feedback loop can be used to correct the position of a robot to increase task accuracy. A system is described which allows a robot to grasp a square prism and place it in a box using visual servoing. Edge extraction and line fitting are used to determine the position and orientation of the box. The camera is fixed, and a servo

cycle time of 10 s is reported.

Considerable work on the use of visual servoing was conducted at SRI International during the late 1970s. Early work [215, 216] describes the use of visual feedback for bolt-insertion and picking moving parts from a conveyor. Hill and Park [116] describe visual servoing of a Unimate robot in 1979. Binary image processing is used for speed and reliability, providing planar position as well as simple depth estimation based on the apparent distance between known features. Experiments were also conducted using a projected light stripe to provide more robust depth determination as well as surface orientation. These experiments demonstrated planar and 3D visually-guided motion, as well as tracking and grasping of moving parts. They also investigated some of the dynamic issues involved in closed-loop visual control. Similar work on a Unimate-based visual-servo system is discussed later by Makhlin [179]. Prajoux [207] demonstrated visual servoing of a 2-DOF mechanism for following a swinging hook. The system used a predictor to estimate the future position of the hook, and achieved settling times of the order of 1 s. Coulon and Nougaret [68] address similar issues and also provide a detailed imaging model for the vidicon sensor's memory effect. They describe a digital video processing system for determining the location of one target within a processing window, and use this information for closed-loop position control of an XY mechanism to achieve a settling time of around 0.2 s to a step demand.

Simple hand-held light stripers of the type proposed by Agin [3] have been used in planar applications such as connector acquisition [187], weld seam tracking [49], and sealant application [227]. The last lays a bead at 400 mm/s with respect to a moving car-body, and shows a closed-loop bandwidth of 4.5 Hz. More recently Venkatesan and Archibald [258] describe the use of two hand-held laser scanners for real-time 5-DOF robot control.

Gilbert [101] describes an automatic rocket-tracking camera which keeps the target centred in the camera's image plane by means of pan/tilt controls. The system uses video-rate image processing hardware to identify the target and update the camera orientation at 60 Hz. Dzialo and Schalkoff [81] discuss the effects of perspective on the control of a pan-tilt camera head for tracking.

Weiss [273] proposed the use of adaptive control for the non-linear time varying relationship between robot pose and image features in image-based servoing. Detailed simulations of image-based visual servoing are described for a variety of manipulator structures of up to 3-DOF.

Weber and Hollis [271] developed a high-bandwidth planar-position controlled micro-manipulator. It is required to counter room and robot motor vibration effects with respect to the workpiece in a precision manufacturing task. Correlation is used to track workpiece texture. To achieve a high sample rate of 300 Hz, yet maintain resolution, two orthogonal linear CCDs are used to observe projections of the image. Since the sample rate is high the image shift between samples is small which reduces the size

of the correlation window needed. Image projections are also used by Kabuka [137]. Fourier phase differences in the vertical and horizontal binary image projections are used for centering a target in the image plane and determining its rotation. This is applied to the control of a two-axis camera platform [137] which takes 30 s to settle on a target. An extension to this approach [139] uses adaptive control techniques to minimize performance indices on grey-scale projections. The approach is presented generally but with simulations for planar positioning only.

An application to road vehicle guidance is described by Dickmanns [76]. Real-time feature tracking and gaze controlled cameras guide a 5 tonne experimental road vehicle at speeds of up to 96 km/h along a test track. Later work by Dickmanns [78] investigates the application of dynamic vision to aircraft landing. Control of underwater robots using visual reference points has been proposed by Negahdaripour and Fox [191].

Visually guided machines have been built to emulate human skills at ping-pong [17,88], juggling [212], inverted pendulum balancing [13,76], catching [41,220], and controlling a labyrinth game [13]. The latter is a wooden board mounted on gimbals on which a ball bearing rolls, the aim being to move the ball through a maze and not fall into a hole. The ping-pong playing robot [17] does not use visual servoing, rather a model of the ball's trajectory is built and input to a dynamic path planning algorithm which attempts to strike the ball.

Visual servoing has also been proposed for catching flying objects on Earth or in space. Bukowski et al. [39] report the use of a Puma 560 to catch a ball with an end-effector mounted net. The robot is guided by a fixed-camera stereo-vision system and a 386 PC. Skofteland et al. [237] discuss capture of a free-flying polyhedron in space with a vision guided robot. Skaar et al. [236] use as an example a 1-DOF robot to catch a ball. Lin et al. [171] propose a two-stage algorithm for catching moving targets; coarse positioning to approach the target in near-minimum time and 'fine tuning' to match robot acceleration and velocity with the target.

There have been several reports of the use of visual servoing for grasping moving targets. The earliest work appears to have been at SRI in 1978 [216]. Recently Zhang et al. [290] presented a tracking controller for visually servoing a robot to pick items from a fast moving conveyor belt (300 mm/s). The camera is hand-held and the visual update interval used is 140 ms. Allen et al. [8] use a 60 Hz fixed-camera stereo vision system to track a target moving at 250 mm/s. Later work [7] extends this to grasping a toy train moving on a circular track. Houshangi [124] uses a fixed overhead camera and a visual sample interval of 196 ms to enable a Puma 600 robot to grasp a moving target.

Fruit picking is a non-manufacturing application of visually guided grasping where the target may be moving. Harrell [108] describes a hydraulic fruit-picking robot which uses visual servoing to control 2-DOF as the robot reaches toward the fruit prior to picking. The visual information is augmented by ultrasonic sensors to de-

termine distance during the final phase of fruit grasping. The visual servo gains are continuously adjusted to account for changing camera target distance. This last point is significant but mentioned by few authors [62, 81].

Part mating has also been investigated using visual servoing. Geschke [99] described a bolt-insertion task using stereo vision and a Stanford arm. The system features automated threshold setting, software image feature searching at 10 Hz, and setting of position loop gains according to the visual sample rate. Stereo vision is achieved with a single camera and a novel mirror arrangement. Ahluwalia and Fogwell [4] describe a system for mating two parts, each held by a robot and observed by a fixed camera. Only 2-DOF for the mating are controlled and a Jacobian approximation is used to relate image-plane corrections to robot joint-space actions. On a larger scale, visually servoed robots have been proposed for aircraft refuelling [163] and demonstrated for mating an umbilical connector to the US Space Shuttle from its service gantry [71].

The image-based servo approach has been investigated experimentally by a number of researchers, but unlike Weiss they use closed-loop joint control as shown in Figure 5.3. Feddema [90–92] uses an explicit feature-space trajectory generator and closed-loop joint control to overcome problems due to low visual sampling rate. Experimental work demonstrates image-based visual servoing for 4-DOF. Rives et al. [48, 211] describe a similar approach using the task function method [222] and show experimental results for robot positioning using a target with four circle features. Hashimoto et al. [111] present simulations to compare position-based and image-based approaches. Experiments, with a visual servo interval of 250 ms, demonstrate image-based servoing of a Puma 560 tracking a target moving in a circle at 30 mm/s. Jang et al. [134] describe a generalized approach to servoing on image features with trajectories specified in feature space, leading to trajectories (tasks) that are independent of target geometry.

Westmore and Wilson [274] demonstrate 3-DOF planar tracking and achieve a settling time of around 0.5 s to a step input. This is extended [269] to full 3D target pose determination using extended Kalman filtering and then to 3D closed-loop robot pose control [280]. Papanikolopoulos et al. [197] demonstrate tracking of a target undergoing planar motion with the CMU DD-II robot system. Later work [198] demonstrates 3D tracking of static and moving targets, and adaptive control is used to estimate the target distance.

The use of visual servoing in a telerobotic environment has been discussed by Yuan et al. [289], Papanikolopoulos et al. [198] and Tendick et al. [249]. Visual servoing can allow the task to be specified by the human operator in terms of selected visual features and their desired configuration.

Approaches based on neural networks [110, 158, 183] and general learning algorithms [185] have also been used to achieve robot hand-eye coordination. A fixed camera observes objects and the robot within the workspace and can learn the relation-

ship between robot joint angles and the 3D end-effector pose. Such systems require training, but the need for complex analytic relationships between image features and joint angles is eliminated.

5.3 Position-based visual servoing

A broad definition of position-based servoing will be adopted that includes methods based on analysis of 2D features or direct pose determination using 3D sensors. The simplest form of visual servoing involves robot motion in a plane orthogonal to the optical axis of the camera, and can be used for tracking planar motion such as a conveyor belt. However tasks such as grasping and part mating require control over the relative distance and orientation of the target.

Humans use a variety of vision-based depth cues including texture, perspective, stereo disparity, parallax, occlusion and shading. For a moving observer, apparent motion of features is an important depth cue. The use of multiple cues, selected according to visual circumstance, helps to resolve ambiguity. Approaches suitable for computer vision are reviewed by Jarvis [136]. However non-anthropomorphic approaches to sensing may offer some advantages. Active range sensors, for example, project a controlled energy beam, generally ultrasonic or optical, and detect the reflected energy. Commonly a pattern of light is projected on the scene which a vision system interprets to determine depth and orientation of the surface. Such sensors usually determine depth along a single stripe of light, multiple stripes or a dense grid of points. If the sensor is small and mounted on the robot [3, 79, 258] the depth and orientation information can be used for servoing. The operation and capability of many commercially available active range sensors are surveyed in [33, 45].

In contrast, passive techniques rely only on observation under ambient illumination to determine the depth or pose of the object. Some approaches relevant to visual servoing will be discussed in the following sections.

5.3.1 Photogrammetric techniques

Close-range photogrammetry, introduced in Section 4.2.1, is highly relevant to the task of determining the relative pose of a target. In order to determine the 3D relative pose of an object, ${}^{c}\underline{x}$, from 2D image plane coordinates, $\underline{f}$, some additional information is needed. This data includes knowledge of the relationship between the observed feature points (perhaps from a CAD model) and also the camera's intrinsic parameters. It can be shown that at least three feature points are needed to solve for pose. Intuitively, the coordinate of each feature point yields two equations and six equations are needed to solve for the six elements of the pose vector. In fact three feature points will yield multiple solutions, but four coplanar points yield a unique solution. Analytic solutions for three and four points are given by Fischler and Bolles [93] and

Ganapathy [97]. Unique solutions exist for four coplanar, but not collinear, points (even for partially known intrinsic parameters) [97]. Six or more points always yield unique solutions, as well as the intrinsic camera calibration parameters.

Yuan [288] describes a general iterative solution independent of the number or distribution of feature points. For tracking moving targets the previous solution can be used as the initial estimate for iteration. Wang and Wilson [269] use an extended Kalman filter to update the pose estimate given measured image plane feature locations. The filter convergence is analogous to the iterative solution.

Once the target pose relative to the camera is determined, it is then necessary to determine the target pose relative to the robot's end-effector. This requires robot eye-hand calibration as discussed in Section 4.2.3.

The commonly cited drawbacks of the photogrammetric approach are the complex computation, and the necessity for camera calibration and a model of the target. None of these objections are overwhelming, and systems based on this principle have been demonstrated using iteration [289], Kalman filtering [274], and analytic solution [98].

5.3.2 Stereo vision

Stereo vision [285] is the interpretation of two views of the scene taken from known different viewpoints in order to resolve depth ambiguity. The location of feature points in one view must be matched with the location of the same feature points in the other view. This matching, or correspondence, problem is not trivial and is subject to error. Another difficulty is the missing parts problem where a feature point is visible in only one of the views and therefore its depth cannot be determined.

Matching may be done on a few feature points such as region centroids or corner features, or on fine feature detail such as surface texture. In the absence of significant surface texture a random texture pattern could be projected onto the scene.

Implementations of 60 Hz stereo-vision systems have been described by Andersson [17], Rizzi et al. [212], Allen et al. [8] and Bukowski et al. [39]. The first two operate in a simple contrived environment with a single white target against a black background. The last two use optical flow or image differencing to eliminate static background detail. All use fixed rather than end-effector-mounted cameras.

5.3.3 Depth from motion

Closely related to stereo vision is *monocular* or *motion stereo* [193] also known as *depth from motion*. Sequential monocular views, taken from different viewpoints, are interpreted to derive depth information. Such a sequence may be obtained from a robot hand-mounted camera during robot motion. It must be assumed that targets in the scene do not move significantly between the views. Vernon and Tistarelli [259] use two contrived trajectories — one along the optical axis, and one rotation about a

fixation point — to construct a depth map of a bin of components. The AIS visual-servoing scheme of Jang et al. [135] reportedly uses motion stereo to determine depth of feature points.

Self motion, or egomotion, produces rich depth cues from the apparent motion of features, and is important in biological vision [260]. Dickmanns [77] proposes an integrated spatio-temporal approach to analyzing scenes with relative motion so as to determine depth and structure. Based on tracking features between sequential frames, he terms it *4D vision*.

Research into insect vision [240] indicates that insects use self-motion to infer distances to targets for navigation and obstacle avoidance. Compared to mammals, insects have effective but simple visual systems and may offer a practical alternative model upon which to base future robot-vision systems [241].

Fixation occurs when a sequence of images is taken with the camera moving so as to keep one point in the scene, the interest point, at the same location in the image plane. Knowledge of camera motion during fixation can be used to determine the 3D position of the target. Stabilizing one point in a scene that is moving relative to the observer also induces the target to stand out from the non-stabilized and blurred parts of the scene, and is thus a basis for scene segmentation. Coombs and Brown [51] describe a binocular system capable of fixating on a target and some of the issues involved in control of the camera motion.

5.4 Image based servoing

Image-based visual servo control uses the location of features on the image plane directly for feedback. For example, consider Figure 5.6, where it is desired to move the robot so that the camera's view changes from initial to final view, and the feature vector from $\underline{f}_0$ to $\underline{f}_d$. The feature vector may comprise coordinates of vertices, or areas of the faces. Implicit in $\underline{f}_d$ is that the robot is normal to, and centered over the face of the cube, at the desired distance. Elements of the task are thus specified in image space, not world space. Skaar et al. [236] propose that many real-world tasks may be described by one or more camera-space tasks.

For a robot with an end-effector-mounted camera, the viewpoint, and hence the features will be a function of the relative pose of the camera with respect to the target, ${}^c\underline{x}_t$. In general this function is non-linear and cross-coupled such that motion of one end-effector DOF will result in the complex motion of many features. For example, camera rotation can cause features to translate horizontally and vertically on the image plane. This relationship

$$\underline{f} = f({}^c\underline{x}_t) \tag{5.1}$$

may be linearized about the operating point

$$\delta\underline{f} = {}^f\mathbf{J}_c({}^c\underline{x}_t)\,\delta{}^c\underline{x}_t \tag{5.2}$$

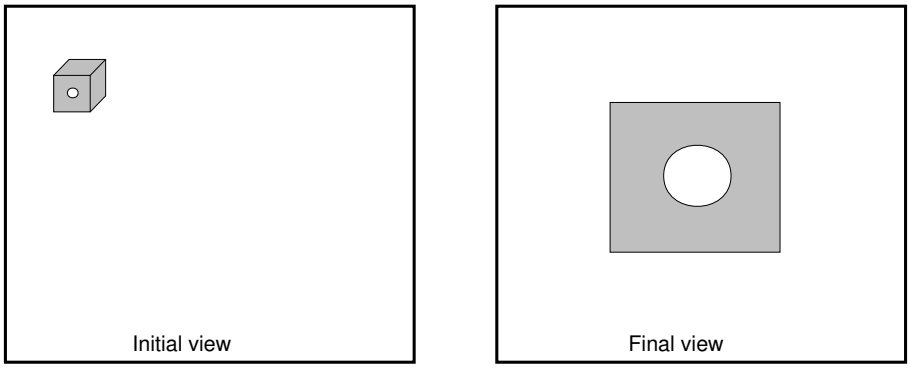


Figure 5.6: Example of initial and desired view of a cube

where

$$ {}^{f}\mathbf{J}_{c}({}^{c}\underline{x}_{t}) = \frac{\partial \underline{f}}{\partial {}^{c}\underline{x}_{t}} \tag{5.3} $$

is a Jacobian matrix, relating rate of change in pose to rate of change in feature space. This Jacobian is referred to variously as the *feature Jacobian*, *image Jacobian*, *feature sensitivity matrix*, or *interaction matrix*. Assume for the moment that the Jacobian is square and non-singular, then

$$ {}^{c}\dot{\underline{x}}_{t} = {}^{f}\mathbf{J}_{c}^{-1}({}^{c}\underline{x}_{t})\,\dot{\underline{f}} \tag{5.4} $$

and a simple proportional control law

$$ {}^{c}\dot{\underline{x}}_{t}(t) = \mathbf{k}\,{}^{f}\mathbf{J}_{c}^{-1}({}^{c}\underline{x}_{t})\,\left(\underline{f}_{d} - \underline{f}(t)\right) \tag{5.5} $$

will tend to move the robot towards the desired feature vector. $\mathbf{k}$ is a diagonal gain matrix, and (t) indicates a time varying quantity.

Pose rates ${}^{c}\dot{\underline{x}}_{t}$ may be transformed to robot end-effector rates via a constant Jacobian, ${}^{t6}\mathbf{J}_{c}$, derived from the relative pose between the end-effector and camera by (2.10). In turn, the end-effector rates may be transformed to manipulator joint rates using the manipulator's Jacobian [277]

$$ \dot{\underline{\theta}} = {}^{t6}\mathbf{J}_{\theta}^{-1}(\underline{\theta})\,{}^{t6}\dot{\underline{x}}_{t} \tag{5.6} $$

where $\underline{\theta}$ represents the joint angles of the robot. The complete equation is

$$ \dot{\underline{\theta}}(t) = \mathbf{k}\,{}^{t6}\mathbf{J}_{\theta}^{-1}(\underline{\theta})\,{}^{t6}\mathbf{J}_{c}\,{}^{f}\mathbf{J}_{c}^{-1}({}^{c}\underline{x})\,\left(\underline{f}_{d} - \underline{f}(t)\right) \tag{5.7} $$

Such a closed-loop system is relatively robust in the presence of image distortions [68] and kinematic parameter variations in the manipulator Jacobian [186].

A number of researchers have demonstrated results using this image-based approach to visual servoing. The significant problem is computing or estimating the feature Jacobian and a variety of approaches will be described next.

5.4.1 Approaches to image-based visual servoing

The proposed IBVS structure of Weiss, Figure 5.5, controls robot joint angles directly using measured image features. The non-linearities include the manipulator kinematics and dynamics as well as the perspective imaging model. Adaptive control is proposed since the gain, ${}^{f}\mathbf{J}_{c}^{-1}(\underline{\theta})$, is pose dependent and $\underline{\theta}$ is not measured. The changing relationship between robot pose and image feature change is learned during the motion. Weiss uses independent single-input single-output (SISO) model-reference adaptive control (MRAC) loops for each DOF, citing the advantages of modularity and reduced complexity compared to multi-input multi-output (MIMO) controllers. The proposed SISO MRAC requires one feature to control each joint and no coupling between features, and a scheme is introduced to select features so as to minimize coupling. In practice this last constraint is difficult to meet since camera rotation inevitably results in image feature translation.

Weiss [273] presents detailed simulations of various forms of image-based visual servoing with a variety of manipulator structures of up to 3-DOF. Sample intervals of 33 ms and 3 ms are investigated, as is control with measurement delay. With non-linear kinematics (revolute robot structure) the SISO MRAC scheme has difficulties. Solutions proposed, but not investigated, include MIMO control and a higher sample rate, or the dynamic-look-and-move structure, Figure 5.3.

Weiss found that even for a 2-DOF revolute mechanism a sample interval less than 33 ms was needed to achieve satisfactory plant identification. For manipulator control Paul [199] suggests that the sample rate should be at least 15 times the link structural frequency. Since the highest sample frequency achievable with standard cameras and image processing hardware is 60 Hz, the IBVS structure is not currently practical for visual servoing. The so called dynamic look and move structure, Figure 5.3, is more suitable for control of 6-DOF manipulators, by combining high-bandwidth joint level control in conjunction with a lower rate visual position control loop. The need for such a control structure is hinted at by Weiss and is used by Feddema [91] and others [65, 111, 134].

Feddema extends the work of Weiss in many important ways, particularly by experimentation [90–92] and cites the difficulties of Weiss's approach as being the assumption that vision update interval, T, is constant, and significantly greater than the sub millisecond period needed to control robot dynamics. Due to the low speed feature extraction achievable (every 70 ms) an explicit trajectory generator operating in

feature space is used rather than the pure control loop approach of Weiss. Feature velocities from the trajectory generator are resolved to manipulator configuration space for individual closed-loop joint PID control.

Feddema [91, 92] describes a 4-DOF servoing experiment where the target was a gasket containing a number of circular holes. Binary image processing and Fourier descriptors of perimeter chain codes were used to describe each hole feature. From the two most unique circles four features are derived; the coordinates of the midpoint between the two circles, the angle of the midpoint line with respect to image coordinates, and the distance between circle centers. It is possible to write the feature Jacobian in terms of these features, that is ${}^{f}\mathbf{J}_c(\underline{f})$, though this is not generally the case. The experimental system could track the gasket moving on a turntable at up to 1 rad/s. The actual position lags the desired position, and 'some oscillation' is reported due to time delays in the closed-loop system.

A similar experimental setup [91] used the centroid coordinates of three gasket holes as features. This more typical case does not allow the Jacobian to be formulated directly in terms of the measured features. Two approaches to evaluating the Jacobian are described. Firstly [90] the desired pose is used to compute the Jacobian, which is then assumed constant. This is satisfactory as long as the initial pose is close to that desired. Secondly [90, 273], the pose is explicitly solved using photogrammetric techniques and used to compute the Jacobian. Simulation of 6-DOF image based servoing [91] required determination of pose at each step to update the Jacobian. This appears more involved than pure position-based servoing.

Rives et al. [48,211] describe an approach that computes the camera velocity screw as a function of feature values based on the task function approach. The task is defined as the problem of minimizing $\| e({}^{c}\underline{x}_{t6}(t)) \|$ where $e()$ is the task function. For visual servoing the task function is written in terms of image features $\underline{f}$ which in turn are a function of robot pose

$$e(\underline{x}_{t6}(t)) = \mathbf{C}\left(\underline{f}(\underline{x}_{t6}(t)) - \underline{f}_d\right) \tag{5.8}$$

To ensure convergence $\mathbf{C}$ is chosen as $\mathbf{C} = \mathbf{L}^{T+}$, where $\mathbf{L}^T = \partial \underline{f} / \partial {}^{c}\underline{x}_{t6}(t)$ is referred to as the *interaction matrix*, and $^+$ denotes the generalized inverse. As previously it is necessary to know the model of the interaction matrix for the visual features selected, and interaction matrices for point clusters, lines and circles are derived. Experimental results for robot positioning using four point features are presented [48].

Frequently the feature Jacobian can be formulated in terms of features plus depth. Hashimoto et al. [111] estimate depth explicitly based on analysis of features. Papanikolopoulos [198] estimates depth of each feature point in a cluster using an adaptive control scheme. Rives et al. [48, 211] set the desired distance rather than update or estimate it continuously.

Feddema describes an algorithm [90] to select which three of the seven measurable features gives best control. Features are selected so as to achieve a balance between

controllability and sensitivity with respect to changing features. The generalized inverse of the feature Jacobian [111,134,211] allows more than 3 features to be used, and has been shown to increase robustness, particularly with respect to singularities [134].

Jang et al. [135] introduce the concepts of augmented image space (AIS) and transformed feature space (TFS). AIS is a 3D space whose coordinates are image plane coordinates plus distance from camera, determined from motion stereo. In a similar way to Cartesian space, trajectories may be specified in AIS. A Jacobian may be formulated to map differential changes from AIS to Cartesian space and then to manipulator joint space. The TFS approach appears to be very similar to the image-based servoing approach of Weiss and Feddema.

Bowman and Forrest [36] describe how small changes in image plane coordinates can be used to determine differential change in Cartesian camera position and this is used for visual servoing a small robot. No feature Jacobian is required, but the camera calibration matrix is needed.

Most of the above approaches require analytic formulation of the feature Jacobian given knowledge of the target model. This process could be automated, but there is attraction in the idea of a system that can 'learn' the non-linear relationship automatically as originally envisaged by Sanderson and Weiss. Some recent results [123, 133] demonstrate the feasibility of online image Jacobian estimation.

Skaar et al. [236] describe the example of a 1-DOF robot catching a ball. By observing visual cues such as the ball, the arm's pivot point, and another point on the arm, the interception task can be specified even if the relationship between camera and arm is not known *a priori*. This is then extended to a multi-DOF robot where cues on each link and the payload are observed. After a number of trajectories the system 'learns' the relationship between image-plane motion and joint-space motion, effectively estimating a feature Jacobian. Tendick et al. [249] describe the use of a vision system to close the position loop on a remote slave arm with no joint position sensors. A fixed camera observes markers on the arm's links and a numerical optimization is performed to determine the robot' s pose.

Miller [185] presents a generalized learning algorithm based on the CMAC structure proposed by Albus [5] for complex or multi-sensor systems. The CMAC structure is table driven, indexed by sensor value to determine the system command. The modified CMAC is indexed by sensor value as well as the desired goal state. Experimental results are given for control of a 3-DOF robot with a hand-held camera. More than 100 trials were required for training, and good positioning and tracking capability were demonstrated. Artificial neural techniques can also be used to learn the non-linear relationships between features and manipulator joint angles as discussed by Kuperstein [158], Hashimoto [110] and Mel [183].

5.5 Implementation issues

Progress in visual servoing has been facilitated by technological advances in diverse areas including sensors, image processing and robot control. This section summarizes some of the implementation details reported in the literature.

5.5.1 Cameras

The earliest reports used thermionic tube, or vidicon, image sensors. These devices had a number of undesirable characteristics such as physical weight and bulk, fragility, poor image stability and memory effect [68].

Since the mid 1980s most researchers have used some form of solid state camera based on an NMOS, CCD or CID sensor. The only reference to color vision for visual servoing is the fruit picking robot [108] where color is used to differentiate fruit from the leaves. Given real-time constraints the advantages of color vision for object recognition may be offset by the increased cost and high processing requirements of up to three times the monochrome data rate. Almost all reports are based on the use of area sensors, but line-scan sensors have been used by Webber et al. [271] for very high-rate visual servoing. Cameras used generally conform to either of the two main broadcast standards RS170 or CCIR.

Motion blur can be a significant problem when there is relative motion between target and camera. In conjunction with a simple binary vision system this can result in the target being 'lost from sight' which in turn leads to 'rough' motion [17, 65]. The electronic shuttering facility common to many CCD sensors can be used to overcome this problem, but with some caveats, as discussed previously in Section 3.4.1.

Cameras can be either fixed or mounted on the robot's end-effector. The benefits of an end-effector-mounted camera include the ability to avoid occlusion, resolve ambiguity and increase accuracy, by directing its attention. Tani [244] describes a system where a bulky vidicon camera is mounted remotely and a fibre optic bundle used to carry the image from near the end-effector. Given the small size and cost of modern CCD cameras such an approach is no longer necessary. All reported stereo-based systems use fixed cameras although there is no reason a stereo-camera cannot be mounted on the end-effector, apart from practical considerations such as payload limitation or lack of camera system robustness.

Zhang et al. [290] observe that, for most useful tasks, an overhead camera will be obscured by the gripper, and a gripper mounted camera will be out of focus during the final phase of part acquisition. This may imply the necessity for switching between several views of the scene, or using hybrid control strategies which utilize vision and conventional joint-based control for different phases of the task. Nelson et al. [154] discuss the issue of focus control for a camera mounted on a 'looking' robot observing another robot which is performing the task using visual-servoing.

5.5.2 Image processing

Vision has not been widely exploited as a sensing technology when high measurement rates are required due to the enormous amount of data produced by vision sensors, and the technological limitations in processing that data rapidly. A vision sensor's output data rate (typically 6 Mbyte/s) is several orders of magnitude greater than that of a force sensor for the same sample rate. This necessitates special-purpose hardware for the early stages of image processing, in order to reduce the data rate to something manageable by a conventional computer.

The highest speed systems reported in the literature generally perform minimal image processing and the scenes are contrived to be simple to interpret, typically a single white target on a black background [17, 65, 92, 212]. Image processing typically comprises thresholding followed by centroid determination. Selection of a threshold is a practical issue that must be addressed and a number of techniques are reviewed by Weszka [275].

General scenes have too much 'clutter' and are difficult to interpret at video rates. Harrell [108] describes a vision system which uses software to classify pixels by color so as to segment citrus fruit from the surrounding leaves. Allen et al. [7, 8] use optical flow calculation to extract only moving targets in the scene, thus eliminating stationary background detail. They use the Horn and Schunk optical flow algorithm [122], implemented on a NIST PIPE [147] processor, for each camera in the stereo pair. The stereo flow fields are thresholded, the centroid of the motion energy regions determined, and then triangulation gives the 3D position of the moving body. Haynes [39] also uses a PIPE processor for stereo vision. Sequential frame differencing or background subtraction is proposed to eliminate static image detail. These approaches are only appropriate if the camera is stationary but Dickmanns [77] suggests that the use of foveation and feature tracking can be used in the general case of moving targets and observer.

Datacube processing modules [74] have also been used for video-rate visual servoing [65, 71] and active vision camera control [51].

5.5.3 Feature extraction

A very important operation in most visual servoing systems is determining the coordinate of an image feature point, frequently the centroid of a region. The centroid can be determined to sub-pixel accuracy, even in a binary image, and for a circle that accuracy was shown, (4.53), to increase with region diameter. The calculation of centroid can be achieved by software or specialized moment generation hardware.

There are many reports of the use of moment generation hardware to compute the centroid of objects within the scene at video data rates [16, 94, 114, 261, 264]. Andersson's system [16] computed grey-scale second order moments but was incapable of connected-region analysis, making it unsuitable for scenes with more than one object.

Hatamian's system [114] computed grey-scale third order moments using cascaded single pole digital filters and was also incapable of connected-region analysis. The APA-512 [261], described further in appendix C, combines single-pass connectivity with computation of moments up to second order, perimeter and bounding box for each connected region in a binary scene.

The centroid of a region can also be determined using multi-spectral spatial, or pyramid, decomposition of the image [14]. This reduces the complexity of the search problem, allowing the computer to localize the region in a coarse image and then refine the estimate by limited searching of progressively higher resolution images.

Software computation of image moments is one to two orders of magnitude slower than specialized hardware. However the computation time can be greatly reduced if only a small image window, whose location is predicted from the previous centroid, is processed [77, 92, 99, 108, 212, 274]. The task of locating features in sequential scenes is relatively easy since there will be only small changes from one scene to the next [77, 193] and total scene interpretation is not required. This is the principle of verification vision proposed by Bolles [34] in which the system has considerable prior knowledge of the scene, and the goal is to verify and refine the location of one or more features in the scene. Determining the initial location of features requires the entire image to be searched, but this need only be done once. Papanikolopoulos et al. [197] use a sum-of-squared differences approach to match features between consecutive frames. Features are chosen on the basis of a confidence measure computed from the feature window, and the search is performed in software. The TRIAX system [19] is an extremely high-performance multiprocessor system for low latency six-dimensional object tracking. It can determine the pose of a cube by searching short check lines normal to the expected edges of the cube.

When the software feature search is limited to only a small window into the image, it becomes important to know the expected position of the feature in the image. This is the target tracking problem; the use of a filtering process to generate target state estimates and predictions based on noisy observations of the target's position and a dynamic model of the target's motion. Target maneuvers are generated by acceleration controls unknown to the tracker. Kalata [141] introduces tracking filters and discusses the similarities to Kalman filtering. Visual servoing systems have been reported using tracking filters [8], Kalman filters [77, 274], AR (auto regressive) or ARX (auto regressive with exogenous inputs) models [91, 124]. Papanikolopoulos et al. [197] use an ARMAX model and consider tracking as the design of a second-order controller of image plane position. The distance to the target is assumed constant and a number of different controllers such as PI, pole-assignment and LQG are investigated. For the case where target distance is unknown or time-varying, adaptive control is proposed [196]. The prediction used for search window placement can also be used to overcome latency in the vision system and robot controller.

Dickmanns [77] and Inoue [128] have built multiprocessor systems where each

processor is dedicated to tracking a single distinctive feature within the image. More recently, Inoue [129] has demonstrated the use of a specialized VLSI motion estimation device for fast feature tracking.

Hager's XVision[4] is a portable software package for high-speed tracking of line and region features. Its object oriented structure allows for the creation of more complex features based on the inbuilt primitive features.

5.5.4 Visual task specification

Many of the visual servo systems reported are capable of performing only a single task. This is often due to the optimization necessary given the constraints of image processing. There has been little work on more general approaches to task description in terms of visual features. Jang [134] and Skaar et al. [236] have shown how tasks such as edge following or catching can be expressed in terms of image plane features and their desired trajectories which are described algorithmically. Commercial robot/vision systems have inbuilt language support for both visual feature extraction and robot motion; however, these are limited to look-then-move operation since motion is based on finite duration trajectories to known or computed destination points.

Geschke [100] describes the Robot Servo System (RSS) software which facilitates the development of applications based on sensory input. The software controls robot position, orientation, force and torque independently, and specifications for control of each may be given. The programming facilities are demonstrated with applications for vision-based peg in hole insertion and crank turning. Haynes et al. [39] describe a set of library routines to facilitate hand-eye programming for a PC connected to a Unimate robot controller and stereo-camera system.

The use of a table-driven state machine is proposed by Adsit [1] to control a visually-servoed fruit-picking robot. The state machine was seen to be advantageous in coping with the variety of error conditions possible in such an unstructured work environment. Another approach to error recovery is to include a human operator in the system. The operator could select image features and indicate their desired configuration allowing the task itself to be performed under closed-loop visual control. This would have particular advantage in situations, such as space or undersea, where there is considerable communications time delay. Teleoperation applications have been described by Yuan et al. [289], Papanikolopoulos et al. [198] and Tendick et al. [249].

[4] Available from http://WWW.CS.Yale.EDU/HTML/YALE/CS/AI/VisionRobotics/YaleTracking.html

Chapter 6

Modelling an experimental visual servo system

This and following chapters are concerned with the dynamic characteristics of closed-loop visual control systems, as distinct from the kinematic control issues discussed in the previous chapter. Researchers in robotic force control (e.g., Whitney [276], Eppinger and Seering [85]) have demonstrated that as the bandwidth of the sensor increases, dynamic effects in the sensor, robot and environment introduce stability problems as attempts are made to increase the closed-loop bandwidth. Similar problems exist with high-bandwidth visual control but to date reported sample rates and manipulator speeds have generally been so low that these problems have not been widely appreciated. This has largely been due to technological limitations in image processing. However in a few cases such effects have arisen but were only mentioned in passing and not adequately investigated.

This chapter starts by examining the architecture and dynamic performance of visual servo systems previously described in the literature, and draws some conclusions about system architectural characteristics that are prerequisites to achieving high-performance. Then the experimental hardware and software system used in this work is introduced: a general-purpose sensor-based robot controller and a low-latency 50 Hz vision system. The system described uses a single end-effector mounted camera (monocular eye-in-hand vision) to implement a target following or fixation behaviour by rotating the camera using the robot's wrist axes. It employs "simplistic computer vision" techniques and a specialised hardware architecture to provide a high sample-rate and minimum latency vision system. This results in an ideal testbed with which to bring closed-loop dynamic problems to the fore, conduct modelling, and experiment with various control strategies. The temporal characteristics of the controller hardware and software affect the visual closed-loop dynamics and must be understood in order

to explain the observed behaviour.

6.1 Architectures and dynamic performance

The earliest reference to a visual servo stability problem appears to be Hill and Park's [116] 1979 paper which describes visual servoing of a Unimate robot and some of the dynamics of visual closed-loop control. Stability, accuracy and tracking speed for another Unimate-based visual-servo system are discussed by Makhlin [179]. Coulon and Nougaret [68] describe closed-loop position control of an XY mechanism to achieve a settling time of around 0.2 s to a step demand. Andersen [13] describes the dynamic control of a labyrinth game — a wooden board mounted on gimbals on which a ball bearing rolls, the aim being to move the ball through a maze and not fall into a hole. The ball's position is observed at 40 ms intervals and a Kalman filter is used to reconstruct the ball's state. State-feedback control gives a closed loop bandwidth of 1.3 Hz.

Weiss [273] proposes *image based visual servoing*, a control strategy in which robot joint torques are controlled by visual feature data via SISO MRAC loops. The adaptive controllers 'track' the time-varying kinematic relationship between robot joint angles and target features, as well as compensating for the time-varying rigid-body manipulator dynamics. Weiss presents simulation results only, but for complex mechanisms such as 3-DOF revolute robots he finds that visual sample rates over 300 Hz are required for stable and accurate control. The axis models used in the simulation do not include any non-linear friction terms.

Pool [206] provides a detailed description of a motion control system for a citrus picking robot. The 2-DOF robot is hydraulically actuated and has three control modes: velocity control for scanning the tree, position control for return after fruit picking and visual control while reaching toward a fruit. In position control mode an inner velocity loop was found to be necessary to counter the effects of static friction in the actuator. The visual control mode uses visual error to control the hydraulic valve via a lead/lag compensator — there is no inner axis velocity or position loop despite the problems mentioned with static friction in position control mode. Acceptable static error of 6 pixels was achieved without any explicit gravity compensation. The controller is implemented digitally at 60 Hz and is synchronous with the RS170 field rate image processing system. Surprisingly, the controller design is performed using continuous time techniques with no modelling of latency in the vision system or controller. The design aim was to achieve a phase lag of 10° with a fruit swinging at 1.1 Hz but this was not achieved. At that frequency the desired phase lag is equivalent to a delay of 25 ms and the vision system alone would contribute nearly 17 ms of delay (one RS170 field time), requiring very high performance servos to meet this criterion.

Important prerequisites for high-performance visual servoing are:

- a vision system capable of a high sample rate with low latency;
- a high-bandwidth communications path between the vision system and the robot controller.

Most reports make use of standard video sensors and formats and are thus limited in sample rate to at most 60 Hz (the RS170 field rate). Weber and Hollis [271] developed a high-bandwidth planar position-controlled micro-manipulator with a 300 Hz sample rate by using linear rather than array sensors, dramatically reducing the number of pixels to be processed.

For visual control [68, 116] and force control [201] it has been observed that latency from sensed error to action is critical to performance. While pipelined computing systems are capable of high throughput rates (and are commonly used in image processing) they achieve this at the expense of latency. For sensor-based systems it may be preferable to employ parallel rather than pipelined processing. Many reports are based on the use of slow vision systems where the sample interval or latency is significantly greater than the video frame time. If the target motion is constant then prediction can be used to compensate for the latency, but the low sample rate results in poor disturbance rejection and long reaction time to target 'maneuvers'. Zhang et al. [290] have suggested that prediction is always required for the final phase of part grasping, since an overhead camera will be obscured by the gripper, and a gripper mounted camera will be out of focus.

Grasping objects on a conveyor belt is an ideal application for prediction since the target velocity is constant. Zhang et al. [290] have presented a tracking controller for visually servoing a robot to pick items from a fast moving conveyor belt (300 mm/s). The camera is hand-held and the visual update interval used is 140 ms. A Kalman filter is used for visual state estimation, and generalized predictive control is used for trajectory synthesis and long-range prediction along the axis of conveyor motion. Allen et al. [8] use a 60 Hz fixed-camera stereo vision system to track a target moving at 250 mm/s. Later work [7] extends this to grasping a toy train moving on a circular track. The vision sensor introduces considerable lag and noise and an $\alpha-\beta-\gamma$ filter [141] is used to smooth and predict the command signal to the robot. Houshangi [124] uses a fixed overhead camera, and a visual sample interval of 196 ms, to enable a Puma 600 robot to grasp a moving target. Prediction based on an auto-regressive model is used to estimate the unknown dynamics of the target and overcome vision system delays. Prajoux [207] describes a system to grasp a swinging object using 2-DOF visual servoing. A predictor is used to estimate the future position of the hook, enabling settling times of the order of 1s.

It is an unfortunate reality that most 'off-the-shelf' robot controllers have low-bandwidth communications facilities. In many reported visual servo systems the communications path between vision system and robot is typically a serial data communications link [91, 124, 258, 280]. The Unimate VAL-II controller's *ALTER* facility

used by some researchers [198, 258] allows trajectory modification requests to be received via a serial port at intervals of 28 ms. Tate [248] identified the transfer function between this input and the manipulator motion, and found the dominant dynamic characteristic below 10 Hz was a time delay of 0.1s. Bukowski et al. [39] describe a novel analog interconnect between a PC and the Unimate controller, so as to overcome the inherent latency associated with the Unimate *ALTER* facility.

To circumvent this communications problem it is necessary to follow the more difficult path of reverse engineering the existing controller or implementing a custom controller. One of the more common such approaches is to use RCCL [115] or similar [64, 65] to connect a 'foreign controller' to a Puma robot's servo system, bypassing the native VAL based control system. This provides direct application access to the Unimate joint servo controllers, at high sample rate, and with reduced communications latency. Visual servo systems based on RCCL have been described by Allen et al. [8], Houshangi [124] and Corke [66]. Allen et al. use a 'high speed interface' between the PIPE vision system and RCCL, while Corke uses an Ethernet to link a Datacube based vision system to a Cartesian velocity server based on RCCL. Houshangi uses RCCL as a platform for adaptive axis control, but uses a low visual sample rate and a vision system connected by serial link. Higher communications bandwidth can be achieved by means of a common computer backplane. Such an approach is used by Papanikolopoulos [197] with the Chimera multi-processor control architecture. Later work by Corke [65], including this work, is based on a VMEbus robot controller and Datacube vision system with a shared backplane. A similar structure is reported by Urban et al. [256].

Less tightly coupled systems based on transputer technology have been described by Hashimoto et al. [111], Rizzi et al. [212] and Westmore and Wilson [274]. The vision system and robot actuators are connected directly to elements of the computational network. Hashimoto's system closes the joint velocity control loops of a Puma 560 at 1 kHz, but the visual servo sample interval varies from 85 ms to 150 ms depending upon the control strategy used. Wilson's transputer-based vision system performs feature tracking and pose determination, but communicates with a CRS robot via a serial link.

The majority of reported experimental systems implement a visual feedback loop 'on top of' the robot's existing position-control loops. This has occurred, probably, for the pragmatic reason that most robot controllers present an abstraction of the robot as a position-controlled device. However some researchers in visual servoing who have 'built their own' robot controllers at the axis level have all chosen to implement axis-position control. Papanikolopoulos [197] implements custom Cartesian position controllers based on PD plus gravity or model-based torque-feedforward techniques. Sharkey et al. [231] control the 'Yorick' head by means of an inner high-rate position control loop. Houshangi [124] implements adaptive self-tuning axis control with an outer position-control loop. The redundant levels of control add to system complexity

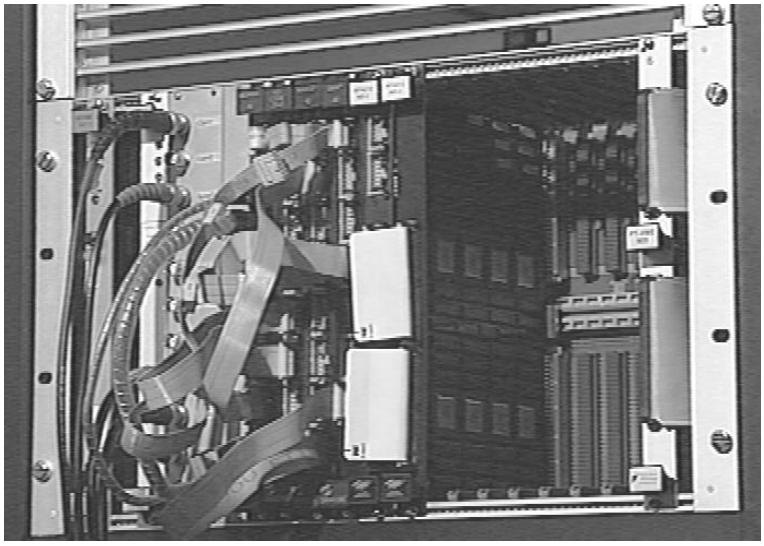


Figure 6.1: Photograph of VME rack. From left to right is CPU, Puma interface, video patch panel, Datacube boards, APA-512, a gap, then the VME to Multibus repeater.

and may impact on closed-loop performance, but this issue has not been investigated in the literature and is discussed in Section 7.4. Hashimoto [111] and Pool [206] provide the only known prior reports of visual servo systems based upon robot axis-velocity control.

6.2 Experimental hardware and software

The facility used for the present visual servoing work has evolved from that developed in the mid 1980s for investigation into robotic force control and deburring [67]. It provides general facilities for sensor-based robot control which proved well suited for early experimental work on visual servoing. The important components of the experimental system are:

- processor and operating system;
- robot control hardware;
- ARCL robot programming support software;
- high-performance low-latency vision system

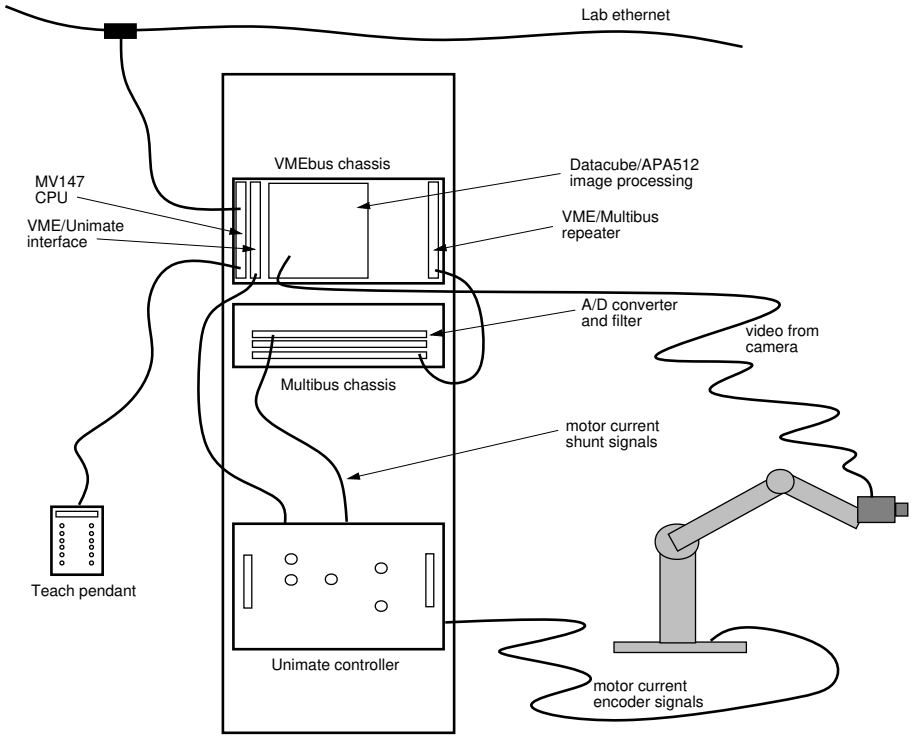


Figure 6.2: Overall view of the system used for visual servo experiments.

- RTVL visual servo support software.

The robot controller and vision system are individually capable of high performance, but the significant advantage of this system is that the robot controller and vision system share a common backplane, see Figure 6.1. This provides a high-bandwidth visual feedback path which is in contrast to the low-speed serial links reported in much of the visual servoing literature. The remainder of this section will briefly describe each of these components. Additional details can be found in the Appendices.

6.2.1 Processor and operating system

The controller is built on the VMEbus [188], an old but widely used high-performance 32-bit bus, for which a large range of computing, image processing and interface modules are available. The CPU is a Motorola MV147 68030 processor card [50] running

at 25 MHz[1]. It provides four serial communication lines, as well as an Ethernet interface. In contrast to some systems described in the literature which are based on multiple high-speed processors [17, 111, 274] a single processor is adequate for the task and is far easier to program and debug.

The VxWorks [281] real-time multi-tasking operating system is used to support software running in the target processor. VxWorks provides common Unix and POSIX library functions, as well as its own libraries for managing tasks, semaphores, timers and so on. It also has extensive networking functions allowing the target processor, for instance, to access remote files via NFS. An interactive shell, accessible via `rlogin` or `telnet`, allows task control, program loading, and inspection or modification of global program variables.

6.2.2 Robot control hardware

A schematic of the visual servo controller system is shown in Figure 6.2, and a detailed view of the principal robot controller modules is shown in Figure 6.3. A custom interface allows the VMEbus host computer to communicate with the attached Unimate servo system. The custom interface is connected directly to the Unimate *arm-interface board*, taking over the role of the low-powered LSI-11/2 microprocessor which runs the VAL language interpreter. The arm-interface is responsible for relaying commands and data between the host and the individual joint position-control cards which were described in Section 2.3.6. Controllers for Unimate robots built on similar principles have been reported by others [115, 184, 262]. In this system the host is able to read the present position of each axis, in encoder units, or specify a new setpoint. Setpoints may be either a position demand, in encoder units, or a motor current demand, in DAC units.

The significant point is that the position controller can only accept setpoints at intervals of 3.5×2^n ms, where $n = \{0, 1, \ldots 5\}$. Current demand setpoints may be issued at any time, and are passed on to the current loop in less than 1 ms. Details of the Unimate controller structure are given in Corke [56][2].

Robot motor currents are also accessible to the host computer. The motor current shunt voltages have been tapped, and brought out to a connector on the back of the Unimate control box. This may be connected to an 8-channel 12-bit simultaneous-capture ADC board [72] via a bank of custom anti-aliasing filters. The filters are 4th order Butterworth with a break frequency of 40 Hz.

The Unimate teach pendant has also been disconnected from the VAL processor and connected to a serial port on the MV147 CPU. A small software library makes the pendant available to application programs, and is a more convenient input device for manual robot control than graphical jog buttons or sliders on a workstation screen.

[1] For some of the work described in Chapter 8 a 33 MHz processor was used instead.

[2] Available in an online document, see Appendix B.

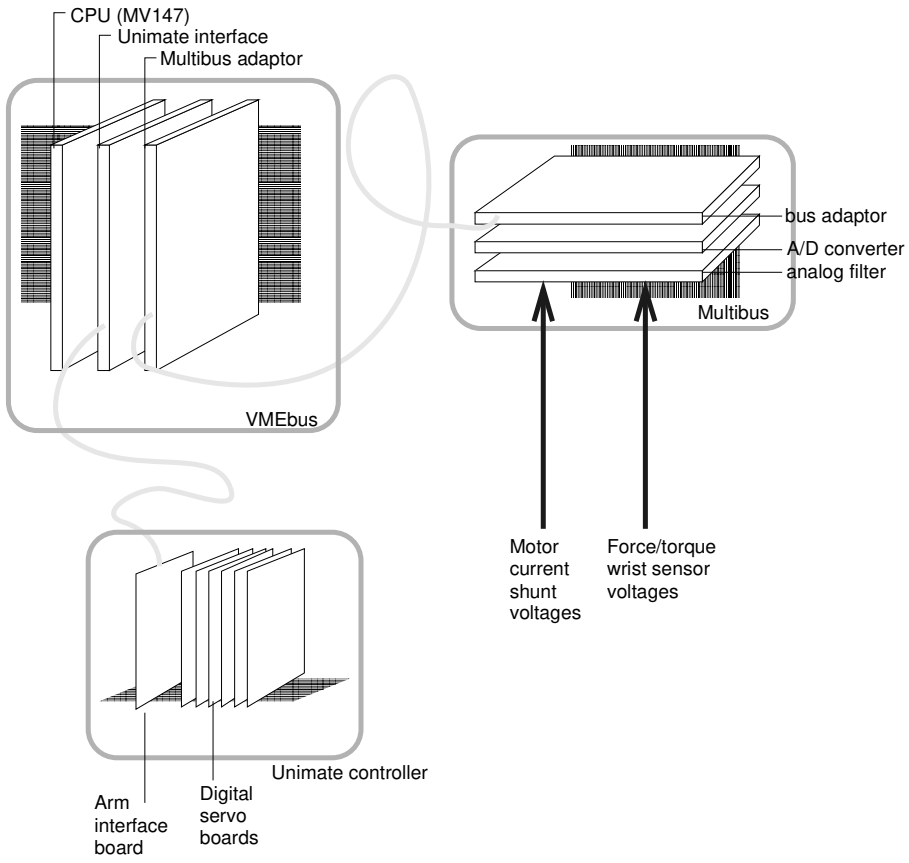


Figure 6.3: Robot controller hardware architecture showing datapaths from the main VME controller to the Multibus I/O box and Unimate servo controller.

6.2.3 ARCL

ARCL [55] is a software system for manipulator control, based on the concepts of RCCL [115]. The programmer's model is based on Cartesian coordinates represented by homogeneous transformations. ARCL is however modular and designed for ease of porting to different computer platforms, operating systems and robot manipulators [64]. Portability is achieved by a number of clearly defined interfaces to, and within, the ARCL software.

The user's application program executes concurrently with a real-time periodic trajectory generation process which computes the next set of manipulator joint angle

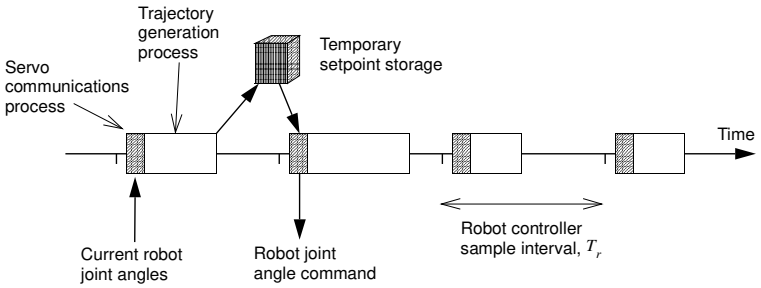


Figure 6.4: Timing of ARCL setpoint generator and servo communication processes.

setpoints. The user's program communicates with the trajectory generator by means of a motion request queue. Various mechanisms exist to allow the user's program to synchronize with the motion of the manipulator, and to allow sensory data to modify the manipulator's path. One conceptually simple, but powerful, approach to sensor-based control is implemented by defining a position in terms of a transformation bound to a function which reflects the value obtained by the vision system. Moving to a position defined in terms of that transformation results in motion controlled by the vision system.

The processing sequence of the ARCL software influences the closed-loop visual servo model and warrants discussion at this point. The principal timing, shown in Figure 6.4, is controlled by a periodic interrupt from a timer on the CPU card which activates the servo communications task. That task communicates with the six digital servo boards in the Unimate controller chassis via the arm interface board. It reads the current encoder values, transmits the previously computed setpoints, and then activates the trajectory generator task to compute the next position setpoint. This 'double handling' of the position setpoint data is necessary to accommodate the variable execution time of the trajectory generation algorithm, but at the expense of increased latency in robot response. An irregular sequence of setpoints results in undesirably rough joint motion.

6.2.4 Vision system

The image processing subsystem, see Figure 6.1, is based on Datacube video-rate pixel-pipelined image processing modules — VMEbus boards that perform simple arithmetic operations on digital video data. Digital video data would exceed the bandwidth of the VMEbus so separate video datapaths are established via sockets along the front edge of each board. The user-installed patch cables between these sockets create the video datapaths required by the application, see Figure 6.5. These inter-

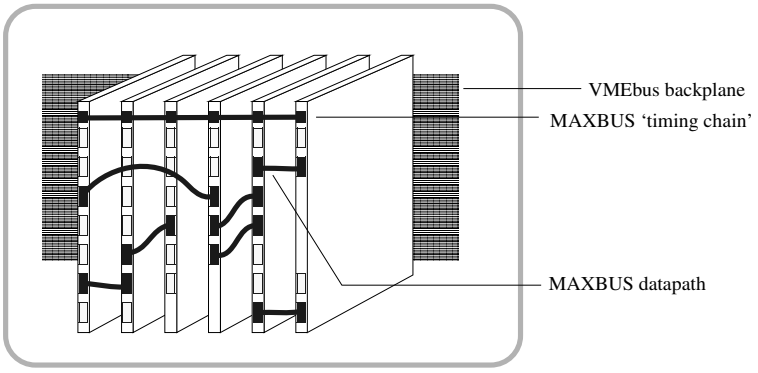


Figure 6.5: MAXBUS and VMEbus datapaths. The MAXBUS 10 Mpixel/s datapaths are established by user-installed patch cables between board front panel sockets.

module video datapaths are known as MAXBUS[3] and carry byte-parallel pixel data at 10 Mpixel/s. Operating parameters are set, and board status monitored, by the host computer via the VMEbus. All Datacube modules are linked by a common timing bus, providing pixel, line and frame clocks. In this configuration the timing bus is driven by the DIGIMAX digitizer which is synchronized to the video signal from the Pulnix TM-6 camera which was modelled in Chapter 3. This camera employs field shuttering and is operated with a shutter interval of 2 ms unless stated otherwise.

The image processing flow, implemented using Datacube modules, is shown schematically in Figure 6.6. The incoming analog video signal is digitized to form a 512×512 pixel digital image. This is thresholded by a lookup table which maps pixels to one of two grey levels, corresponding to the binary values *black* or *white*. Binary median filtering on a 3×3 neighbourhood is then performed to eliminate one or two pixel noise regions which would overwhelm downstream processing elements. Another framestore is used by the run-time software to display real-time color graphics and performance data which are overlaid on the live or binarized camera image.

The APA-512+ [25] (for Area Parameter Accelerator) is a hardware unit designed to accelerate the computation of area parameters of objects in a scene. Digital video data input via MAXBUS is binarized[4] and single pass connectivity analysis is performed. For each region, black or white, a group of parallel ALUs compute moments up to second order, perimeter and bounding box. The APA notifies the host by interrupt or poll-able status flag when a region is completed, and the results for that region

[3] Trademark of Datacube Inc., Danvers MA, USA, who developed the standard and a range of conforming image processing modules.

[4] In this system the image is binarized in the processing stages prior to the APA.

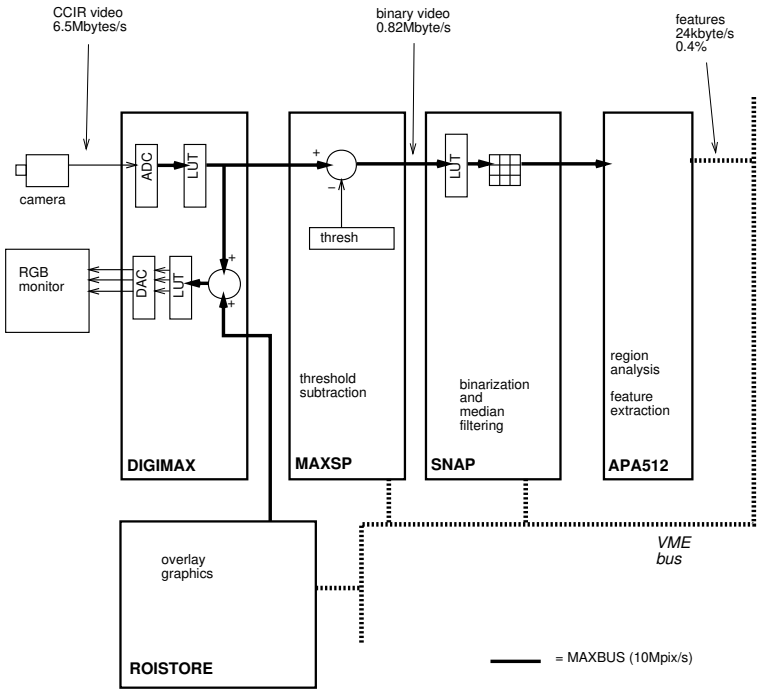


Figure 6.6: Schematic of image processing data flow. Video data is digitized, thresholded and median filtered prior to region feature extraction. The final data rate is a small fraction of that from the camera and can be readily handled by a moderately powered microprocessor. Note also the framestore used for rendering overlay graphics.

are then available from onboard shared memory.

The APA performs very effective data reduction, reducing a 10 Mpixel/s stream of grey-scale video data input via MAXBUS, to a stream of feature vectors representing objects in the scene. Each region is described by 48 bytes of data, so a scene with 10 regions results in a feature data rate of 24 kbyte/s, or 0.4% of the incoming pixel rate. The host processor screens the feature vectors according to their parameters (size, shape, 'color'), and thus finds the objects of interest. Appendix C provides additional details of APA operation.

Many of the reported visual servoing systems process complete video frames. As discussed in Section 3.4.1 full-frame processing requires deinterlacing which increases latency and can result in a ragged image of a fast moving target. Figure 6.7 compares the latency for field and frame rate processing. In this work field processing

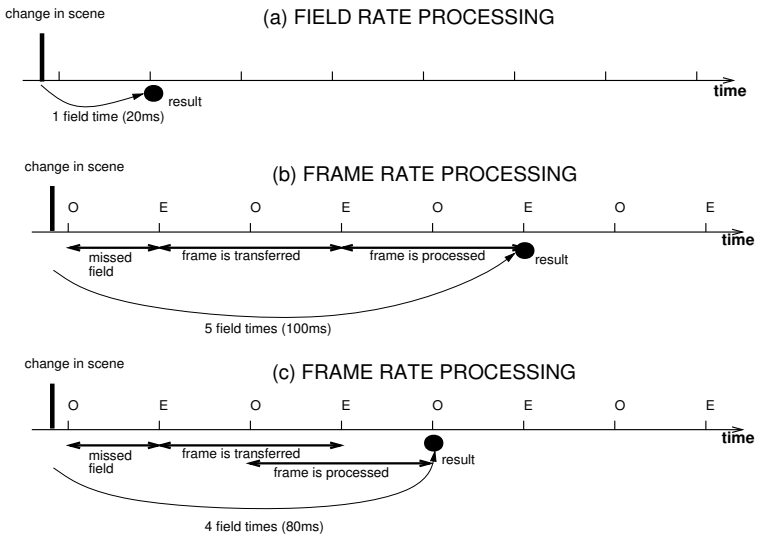


Figure 6.7: Comparison of latency for frame and field-rate processing. 'E' and 'O' designate even and odd fields respectively. Approach (a), as used in this work allows the pixel transfer and processing to be overlapped, giving results within a maximum of 20 ms after pixel exposure. Approach (b) is a straightforward frame processing strategy where both fields must be loaded into the framestore before processing can commence. Since a CCIR frame by definition starts with an even field an extra field delay, shown as *missed field* is introduced. Approach (c) is a modification of (b) which recognizes that frame processing can commence with the second, odd, field.

is used, allowing a 50 Hz visual sample rate by treating the interlaced camera output as two consecutive frames of half vertical resolution.

6.2.5 Visual servo support software — RTVL

Early experimental work in visual servoing showed that quite simple applications rapidly became bloated with detailed code to handle the requirements of vision and robot control, graphical display, diagnostics, data logging and so on [57]. Considerable work has gone into the design of the software system known as RTVL for Real-Time Vision Library. RTVL provides extensive functions to the user's application program encompassing visual-feature extraction, data-logging, remote variable setting, and graphical display. The RTVL software is described in detail in Appendix D, and a typical display is shown in Figure 6.8.

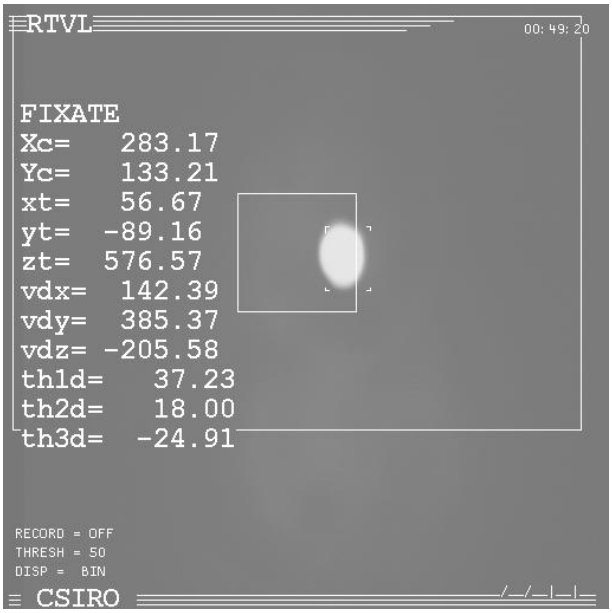


Figure 6.8: Typical RTVL display as seen on the attached color monitor. On the left of the display are 'watched' variables while at lower left various status items are displayed. Top right is the current time and bottom right are activity indicators which appear to spin, allowing a quick appraisal of system operation. In the center can be seen the grey-scale target image with a tracking cursor. The white square almost surrounding the target is the region in which integral action control is enabled.

RTVL provides visual-servo-specific extensions to VxWorks and is loaded into memory at system boot time. Visual servo applications programs are loaded subsequently and access RTVL via a well-defined function call interface. Internally it comprises a number of concurrent tasks and shared data structures. The feature extraction process is simplistic and reports the first (in raster order) n regions which meet the application program's acceptance criteria. These are expressed in terms of a boolean screening function applied to all extracted regions in the scene. Typically screening is on the basis of object color (black or white), upper and lower bounds on area, and perhaps circularity (4.11). Functions exist to compute useful measures such as centroid, circularity and central moments from the returned region datastructures.

Convenient control of applications is facilitated by a mechanism that allows program variables to be registered with a remote procedure call (RPC) server. The client is an interactive control tool running under OpenWindows on an attached workstation

computer. A list of variables registered under the real-time system can be popped up, and for user selected variables a value slider is created which allows that variable to be adjusted. Variables can be boolean, integer, floating point scalar or vector. For debugging and analysis RTVL provides two mechanisms for monitoring variables. One continuously displays the values of a nominated list of variables in the graphics plane which is superimposed on the live camera image, see Figure 6.8. Another allows variables to be timestamped and logged with low overhead into a large memory buffer. The buffer can be written to a file via NFS and a postprocessor used to extract selected variables. These may be displayed in tabular form or imported into MATLAB for analysis and plotting. Most of the experimental data in this book has been obtained using this facility.

Timestamping events within the controller is essential in understanding the temporal behavior of such a complex multi-tasking sensor-based system. It is also desirable that the mechanism has low overhead so as not to impact on system performance. A novel timing board has been developed that provides, via one 32-bit register, a count of video lines since midnight. Each video line is 64μs and this provides adequate timing resolution, but more importantly since the count is derived from MAXBUS horizontal synchronization signals it gives a time value which can be directly related to the video waveform. The time value can be readily converted into frame number, field number, field type (odd or even) and line number within the field or frame, as well as into conventional units such as time of day in hours, minutes and seconds. A comprehensive group of macros and functions is provided to perform these conversions. For debugging and performance analysis this allows the timing of events with respect to the video waveform to be precisely determined.

6.3 Kinematics of camera mount and lens

This section analyses two important mappings in the experimental visual servo system. The first is the the mapping between robot wrist motion and motion of the camera which is a function of the kinematics of the camera mounting on the end-effector. The second is the mapping between camera motion or target motion and perceived motion on the image plane.

6.3.1 Camera mount kinematics

To simplify analysis it is desirable to mount the camera such that camera DOF are controlled individually by robot wrist axes. The fundamental requirement is to control the camera's rotational velocity in the camera's coordinate frame. Traditionally the terms camera *pan* and *tilt* are used to describe camera rotational motion. In robotic terminology these are respectively yaw and pitch motion. Using the standard camera

coordinate frame of Figure 3.25 the rotational rates may be expressed as $\omega_{tilt} = {}^{c}\omega_{x}$ and $\omega_{pan} = {}^{c}\omega_{y}$.

Paul and Zhang [202] give the manipulator's Jacobian, in the T_6 or wrist frame, as

$$ {}^{t_6}\mathbf{J}_{\theta}(\underline{\theta}) = \begin{bmatrix} \mathbf{J}_{11} & \mathbf{0} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix} \tag{6.1} $$

where $\mathbf{J}_{22}$ describes end-effector rotation due to wrist axis motion, and $\mathbf{J}_{21}$ describes rotation of the end-effector due to base axis motion which is assumed zero in this case. Consider the camera mount arrangement shown in Figure 6.9 where the camera frame has simply been translated along the Z_6 axis. Rotational velocity of the camera is equal to the wrist's rotational velocity. For given pan/tilt motion, and using the inverse Jacobian given by Paul and Zhang, the required joint rates can be shown to be

$$ \begin{bmatrix} \dot{\theta}_4 \\ \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} = -\frac{1}{S5} \begin{bmatrix} -S_6 & -C_6 \\ -S_5C_6 & S_5C_6 \\ C_5S_6 & C_5C_6 \end{bmatrix} \begin{bmatrix} \omega_{tilt} \\ \omega_{pan} \end{bmatrix} \tag{6.2} $$

which has two undesirable characteristics. Firstly, it is singular for $\theta_5 = 0$ which is in the middle of the desired workspace, and will be poorly conditioned for camera gaze directions in a cone around the joint 4 axis, Z_3. Secondly, there is considerable cross-coupling which requires the motion of 3 wrist axes to control the required 2-DOF of the camera.

The camera mounting arrangement shown in Figures 6.10 and 6.11 can be seen intuitively to couple camera pan and tilt motion to wrist axes 6 and 5 respectively. The transformation from the $\mathbf{T}_6$ frame to camera frame was given in (4.73) and allows the camera pose rate to be written in terms of a constant Jacobian and the manipulator Jacobian

$$ {}^{c}\dot{\underline{x}} = {}^{c}\mathbf{J}_{t_6}{}^{6}\mathbf{J}_{\theta}(\underline{\theta})\dot{\underline{\theta}} \tag{6.3} $$

$$ = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 121 & -20 \\ 0 & 0 & -1 & -20 & 0 & 0 \\ 0 & 1 & 0 & -121 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right] \begin{bmatrix} \mathbf{J}_{11} & \mathbf{0} \\ \mathbf{J}_{21} & \mathbf{J}_{22} \end{bmatrix} \dot{\underline{\theta}} \tag{6.4} $$

With joints 1–4 stationary, $\dot{\theta}_4 = 0$, this results in

$$ \begin{bmatrix} \omega_{tilt} \\ \omega_{pan} \end{bmatrix} = - \begin{bmatrix} S_6 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_5 \\ \dot{\theta}_6 \end{bmatrix} \tag{6.5} $$

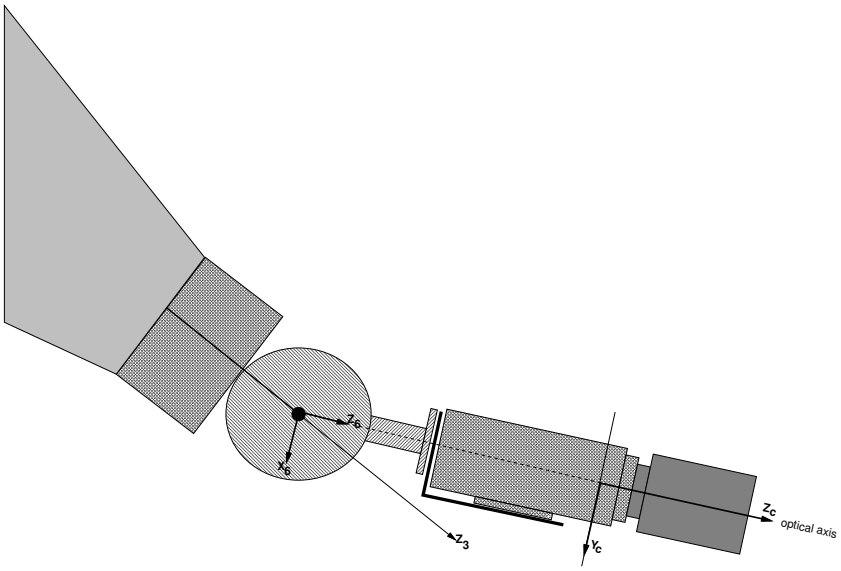


Figure 6.9: A simple camera mount.

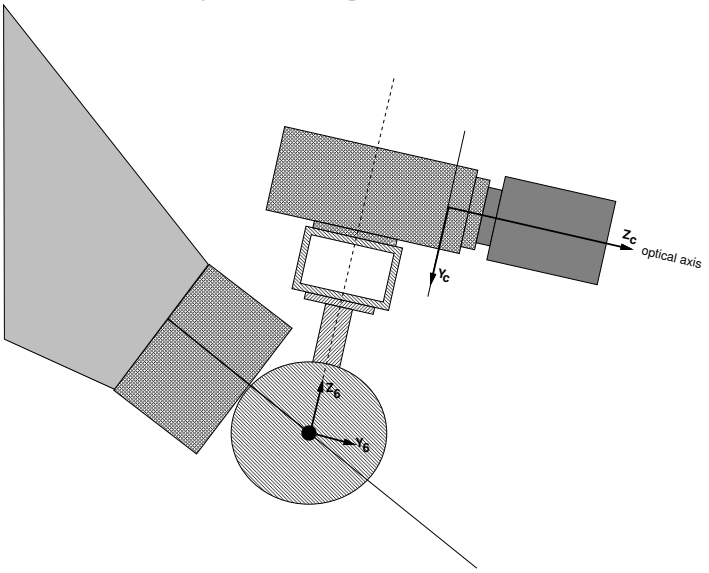


Figure 6.10: Camera mount used in this work. In this pose $\theta_6 \approx -\pi/2$.

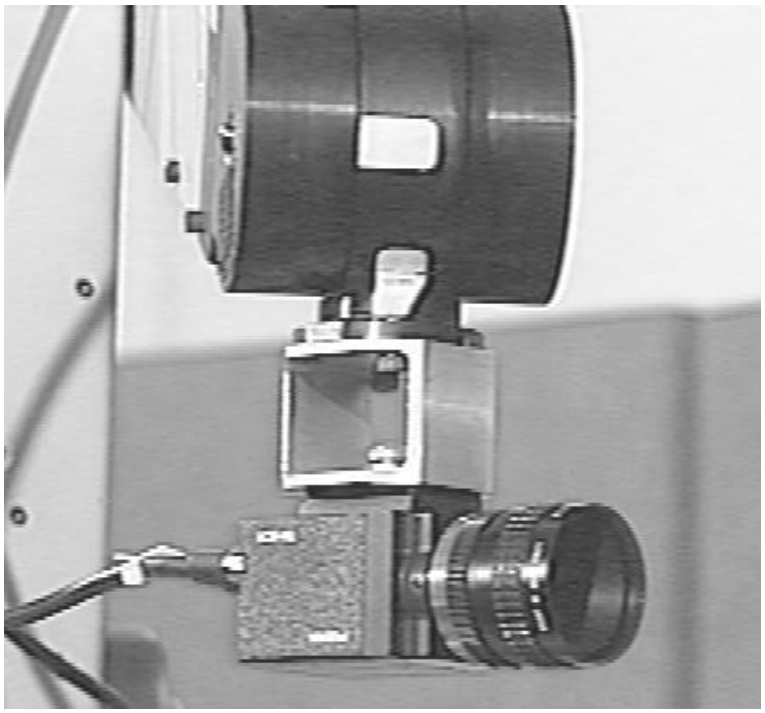


Figure 6.11: Photograph of camera mounting arrangement.

which shows that the camera DOF are indeed independently controlled by the wrist axes 5 and 6. Rearranging to the form

$$\begin{bmatrix} \dot{\theta_5} \\ \dot{\theta_6} \end{bmatrix} = -\begin{bmatrix} \frac{1}{S_6} & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \omega_{tilt} \\ \omega_{pan} \end{bmatrix} \tag{6.6}$$

shows that a singularity occurs when $S_6 = 0$ but this is at the edge of the work envelope and is not a significant problem. As shown in Figure 6.10 the normal working pose is with $\theta_6 \approx \pi/2$. Camera roll, that is rotation about the optical axis, is not controlled and the roll rate can be shown to be

$$\omega_{roll} = -C_6\dot{\theta}_5 \tag{6.7}$$

For fixation motion where the target's centroid is maintained at the principal point on the image plane, camera roll is of no significance. The only disadvantage of this mounting scheme is that it is possible for the camera and its mount to collide with links 3 and 4, particularly when the camera is tilted upward.

6.3.2 Modelling the lens

In a visual servoing system the lens introduces a gain relating changes in target pose to changes in image plane displacement. Changes in this gain will affect the performance and stability of the closed-loop system.

For fixation by translation the lens gain relates camera translation to image plane translation. The small-signal gain of the lens is given by the Jacobian matrix[5], $\partial {}^i\underline{X}/\partial {}^c\underline{x}_t$ where ${}^i\underline{X} = ({}^iX, {}^iY)$ and ${}^c\underline{x}_t$ is the target's relative pose. Recalling the lens equations expressed in pixels (3.66) and (3.67)

$$ {}^iX = \frac{\alpha_x f {}^cx_t}{{}^cz_t - f} + X_0, \quad {}^iY = \frac{\alpha_y f {}^cy_t}{{}^cz_t - f} + Y_0 \tag{6.8} $$

the lens gain is such that

$$ \begin{bmatrix} \delta {}^iX \\ \delta {}^iY \end{bmatrix} = \frac{\partial {}^i\underline{X}}{\partial {}^c\underline{x}_t} \begin{bmatrix} \delta {}^cx_t \\ \delta {}^cy_t \\ \delta {}^cz_t \end{bmatrix} \tag{6.9} $$

$$ = \begin{bmatrix} \frac{\alpha_x f}{{}^cz_t - f} & 0 & -\frac{\alpha_x f {}^cx_t}{({}^cz_t - f)^2} \\ 0 & \frac{\alpha_y f}{{}^cz_t - f} & -\frac{\alpha_y f {}^cy_t}{({}^cz_t - f)^2} \end{bmatrix} \begin{bmatrix} \delta {}^cx_t \\ \delta {}^cy_t \\ \delta {}^cz_t \end{bmatrix} \tag{6.10} $$

The rightmost column of the Jacobian gives the component of image plane motion due to perspective dilation and contraction which will be ignored here since motion in a plane normal to the optical axis is assumed. The lens can thus be modelled in terms of two, distance dependent, gain terms

$$ \begin{bmatrix} \delta {}^iX \\ \delta {}^iY \end{bmatrix} = \begin{bmatrix} K_{lens_x} & 0 \\ 0 & K_{lens_y} \end{bmatrix} \begin{bmatrix} \delta {}^cx_t \\ \delta {}^cy_t \end{bmatrix} \tag{6.11} $$

where

$$ K_{lens_x} = \frac{\alpha_x f}{{}^cz_t - f}, \quad K_{lens_y} = \frac{\alpha_y f}{{}^cz_t - f} \tag{6.12} $$

For brevity, the remainder of this development will be in terms of the X-direction only. If the target distance is large, ${}^cz_t \gg f$, the lens gain

$$ K_{lens_x} \approx \frac{\alpha_x f}{{}^cz_t} \tag{6.13} $$

is strongly dependent on target distance.

An alternative fixation strategy is to control camera orientation, so that lens gain relates relative camera rotation, or bearing angle, to image plane translation. The

[5] A simple image Jacobian as introduced in Section 5.4.

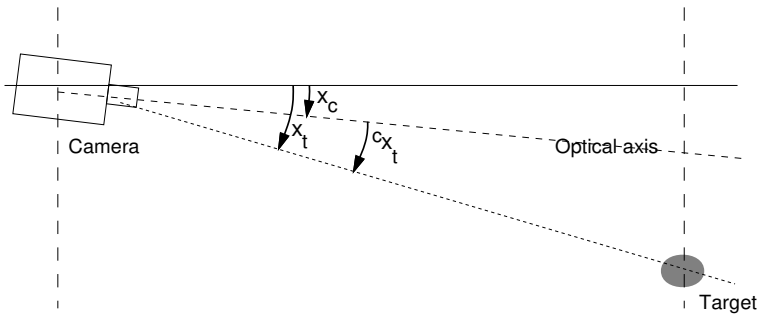


Figure 6.12: Target location in terms of bearing angle. For rotational fixation it is advantageous to consider the target pose, x_t as a bearing angle $\theta_x \approx x/z$.

small-signal gain of the lens in this case is given by the Jacobian matrix, $\partial^i \underline{X} / \partial^c \underline{\theta}_t$ where ${}^c\underline{\theta}_t$ is the target's relative bearing as shown in Figure 6.12. From (3.66) we may write

$$ {}^iX = \frac{\alpha_x f {}^c x_t \tan {}^c\theta_t}{{}^c z_t - f} \tag{6.14}$$

For small ${}^c\theta_t$, as it would be during fixation, $\tan {}^c\theta_t \approx {}^c\theta_t$ and the lens gain is

$$ K_{lens_x} = \frac{\partial^i X}{\partial^c \theta_t} = \frac{\alpha_x f {}^c z_t}{{}^c z_t - f} \tag{6.15}$$

For a large target distance, ${}^c z_t \gg f$, the lens gain

$$ K_{lens_x} \approx \alpha_x f \tag{6.16}$$

is independent of the target distance and the image plane coordinate is proportional to the bearing angle.

With the simple lens shown in Figure 3.5 it is possible to imagine the lens being rotated so as to keep the optical axis, for instance, pointed at the target. However for the compound lens used in practice the situation is not so simple. It is not possible to simply replace the compound lens with an equivalent simple lens at its lens plane. With the mechanical camera mounting shown in Figure 4.11 the axis of rotation passes through the body of the camera at a point determined by the camera mounting hole. An exaggerated representation is shown in Figure 6.13 for a target at distance l which subtends an angle ${}^c\theta_t$ at the lens, and where θ_c the angle of camera rotation. What is unknown is the point in the lens system, distance r from the camera rotation axis, about which the target appears to rotate. This point is known as the *nodal point* of the lens and is not generally given by the lens manufacturer. It may be determined experimentally using a nodal slide.

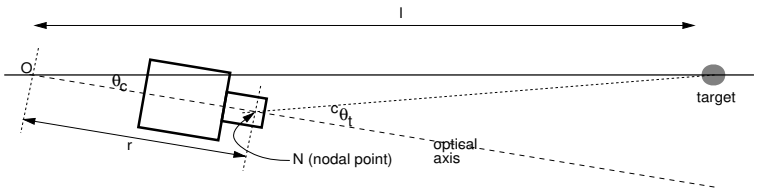


Figure 6.13: Lens center of rotation. The camera rotates about the point O, but the image of the target rotates about the point N. θ_c and ${}^c\theta_t$ are equivalent to x_r and x_t respectively.

Application of the sine rule to the geometry shown in Figure 6.13 results in the relationship

$$\tan {}^c\theta_t = \frac{l \sin\theta_c}{l \cos\theta_c - r} \tag{6.17}$$

between the two angles and it is clear that ${}^c\theta_t > \theta_c$ if $r > 0$.

For rotational fixation the lens gain must be redefined as the apparent shift of the target to *rotation of the camera* about its axis since that is the variable being controlled. Thus

$$K_{lens_x} = \frac{d^i X}{d\theta_c} = \frac{d^i X}{d^c\theta_t} \frac{d^c\theta_t}{d\theta_c} \tag{6.18}$$

and substituting from (6.16) gives

$$K_{lens_x} = f\alpha_x \frac{d^c\theta_t}{d\theta_c} \tag{6.19}$$

$$= \frac{l\,(l - r\cos\theta_c)}{r^2 + l^2 - 2\,l\,r\cos\theta_c} \tag{6.20}$$

For small camera rotations, $\cos\theta_c \approx 1$, the lens gain

$$K_{lens_x} \approx f\alpha_x \frac{l}{l - r} \tag{6.21}$$

is dependent on target distance, but this dependence is weaker than the case for pure translational motion, (6.13). In experiments the lens gain has been observed to lie in the range 700 to 730 pixel/rad depending on camera distance from the target. This is considerably higher than the value of $K_{lens_x} = \alpha_x f = 634$ pixel/rad expected from (6.16), suggesting that the effect described by (6.21) is significant for these experiments in which the target is relatively close to the camera. Such experiments can be used to estimate the nodal point: an experiment measured $K_{lens} = 724$ pixel/rad and $l = 610$ mm, from which the effective radius is determined to be $r = 86$ mm. This

places the nodal point at approximately the glass surface on the front of the lens. It is possible to rotate the camera about its nodal point — this simply requires coordinated motion of all six manipulator axes. However doing so would cloud the central investigation since the resulting motion would be a complex function of the dynamics of all six axes.

6.4 Visual feedback control

The underlying robot controller accepts absolute position setpoint commands but the vision sensor provides relative position information. In general, due to limited torque capabilities, the robot will be unable to reach an arbitrary target position within one setpoint interval and thus requires some 'planned' motion over many intervals. In robotics this is the well known trajectory generation problem [199], but that approach has a number of disadvantages in this situation:

1. to ensure 'smooth' motion over many setpoint intervals it is necessary to explicitly control robot acceleration and deceleration. Techniques based on polynomial trajectories are well known, but add complexity to the control.

2. the target may move while the robot is moving toward its previous goal. This necessitates evaluation of a new path while maintaining continuity of position and its time derivatives. Such an approach has been used by Feddema [91] and Houshangi [124].

A far simpler approach is to consider the positioning task as a control problem: perceived error generates a velocity demand which moves the robot toward the target. This results automatically in target following behaviour, but without the complexities of an explicit trajectory generator. This behaviour is analogous to the way we control cars and boats within an unstructured environment: we do not compute a sequence of precise spatial locations, but rather a velocity which is continually corrected (steering) on the basis of sensory inputs until the goal is achieved.

A simple visual control strategy such as fixation can be based on feedback of the centroid of a single visual feature

$$\dot{\underline{x}}_d = \mathbf{F} \begin{bmatrix} {}^iX - {}^iX_d \\ {}^iY - {}^iY_d \end{bmatrix} \tag{6.22}$$

where $\mathbf{F}$ is a 6×2 matrix which selects how the robot's pose, $\underline{x}$, is adjusted based on observed, $({}^iX, {}^iY)$, and desired $({}^iX_d, {}^iY_d)$ image plane centroid. A single centroid provides only two 'pieces' of information which can be used to control two robot DOF. As described in the previous chapter additional image features can be used to control a greater number of Cartesian DOF. In some earlier work by the author [62,65]

the camera was translated so as to keep the target centered in the field of view. In this chapter, in order to achieve higher performance, the last two axes of the robot are treated as an autonomous camera 'pan/tilt' mechanism since:

- relatively small rotational motion is required to follow large object excursions;
- the effective field of view is much greater;
- from Table 2.21 it is clear that for the Puma 560 the wrist axes have significantly greater velocity and acceleration capability;
- it is more efficient to accelerate only the camera, not the massive links of the robot.

Visual fixation by rotation may be achieved by a proportional control strategy, where (6.22) is written as

$$ {}^{c}\dot{\underline{x}}_d = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & K_{p_y} \\ K_{p_x} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} {}^{i}X - {}^{i}X_d \\ {}^{i}Y - {}^{i}Y_d \end{bmatrix} \tag{6.23} $$

This control approach produces a pose rate demand that, for a position-controlled robot, must be integrated to determine the robot joint angle setpoints. The integration may be performed in Cartesian or joint space. In the former, desired Cartesian velocity is integrated

$$ \underline{x}_d = \int \dot{\underline{x}}_d \, dt \tag{6.24} $$

and the corresponding joint positions obtained using inverse kinematics

$$ \underline{q}_d = \mathrm{K}^{-1}(\underline{x}_d) \tag{6.25} $$

This technique has been used to implement fixation based on camera translation [62, 65], where the first three robot axes control motion of the camera in a plane normal to the optical axis.

Alternatively the Cartesian velocity demand can be *resolved* to joint velocity [277] demand

$$ \dot{\underline{q}}_d = {}^{t_6}\mathbf{J}_{\theta}^{-1}(\underline{q})\, \dot{\underline{x}}_d \tag{6.26} $$

and then integrated

$$ \underline{q}_d = \int \dot{\underline{q}}_d \, dt \tag{6.27} $$

The latter form (6.26) is less robust since numerical errors in the computed Jacobian result in a Cartesian velocity slightly different from that demanded, causing the robot's

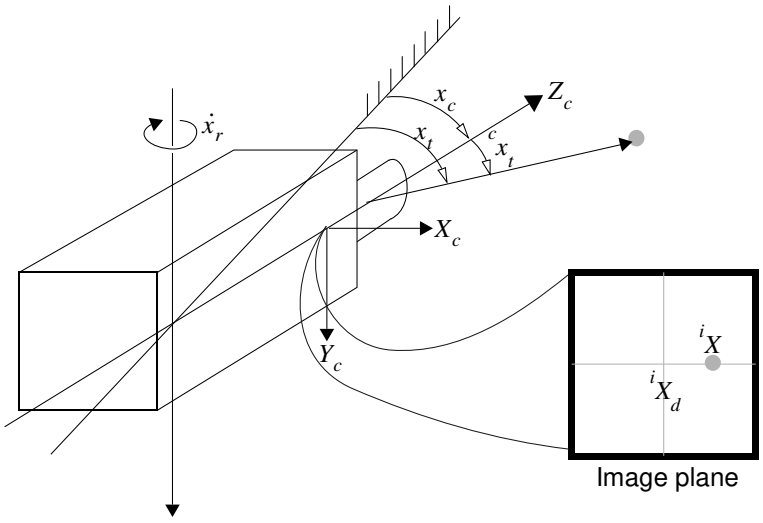


Figure 6.14: Coordinate and sign conventions. Note that camera angle, x_c, is equal to robot angle x_r.

pose to drift slowly with time. For those Cartesian DOF with visual feedback the drift will be controlled, but as observed experimentally in planar visual servoing, the camera drifts slowly normal to the control plane and changes orientation. Those DOF not visually controlled would require some form of Cartesian position control loop to arrest the drift, whereas the approach (6.24) is free from drift. Visual control using (6.26) and (6.27) is described in Section 8.2.

The principal objective of this section is to understand the interaction of vision system and robot electro-mechanical dynamics. To simplify this task only one camera DOF will be controlled and it is desirable that the DOF be driven by only a single robot axis, preferably one whose dynamics characteristics are well understood. Joint 6 was modelled in some detail in Chapter 2 and will be used to actuate the camera in the experiments described. For 2-DOF camera control described in Section 7.5 it will be assumed that the joint 5 dynamics are similar to those of joint 6.

Throughout this chapter the symbols x_r and x_t will be used to denote robot and target pose respectively. However for 1-DOF control these will be scalars and may be interpreted as bearing angles, see Figure 6.12.

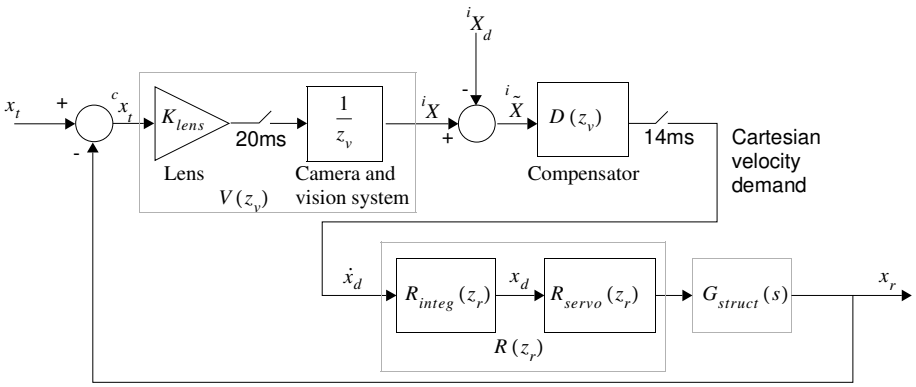


Figure 6.15: Block diagram of the 1-DOF visual feedback system. x_t is the world coordinate location of the target, and $^{i}X_d$ is the desired location of the target on the image plane. Note the two samplers in this system.

6.4.1 Control structure

The structure of the 1-DOF visual feedback controller is shown in Figure 6.15. The target and robot pose, x_t and x_r respectively, are angular displacements in this experiment and Figure 6.14 shows the coordinate and sign conventions used. The leftmost summing junction represents the action of the end-effector mounted sensor which measures relative position, that is, the target position with respect to the end-effector. The second summing junction allows a reference input to set the desired image-plane location of the target's centroid. The integrator, (6.24), converts the visually generated velocity demand into a position setpoint suitable for transmission to the robot's digital position loops. The inverse kinematics required by (6.25) are trivial in this case, (6.6), due to the camera mounting arrangement used. The block labelled $G_{struct}(s)$ represents the mechanical dynamics of the transmission and manipulator link. Discussion of this issue will be deferred until Section 8.1.4.

The system is multi-rate since the robot tasks and the axis controller are constrained by the Unimation position servos to operate at a $T_r = 14\,\text{ms}$ period, while vision processing is constrained to a $T_v = 20\,\text{ms}$ period which is the CCIR video field time. The notation $z_r = e^{sT_r}$ and $z_v = e^{sT_v}$ will be used throughout this section.

The controller is implemented by a short program which makes use of the RTVL and ARCL libraries. In terms of implementation $\dot{x}_d$ is a scalar global variable shared between the vision and robot code modules. This represents a simple but effective way of linking the subsystems running at different sample rates.

Figure 6.16: Photograph of square wave response test configuration. Small panel on the far wall contains the two LEDs.

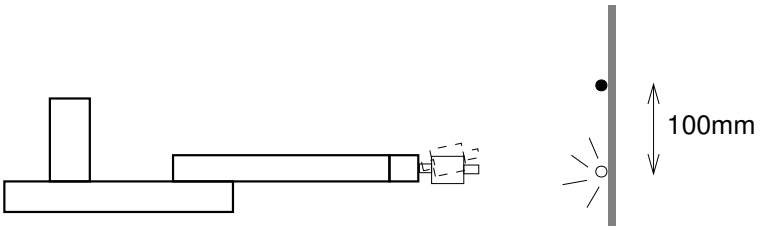


Figure 6.17: Experimental setup for step response tests.

6.4.2 "Black box" experiments

In this experiment the target position is a 'visual square wave' contrived by alternating two LEDs as shown in Figures 6.17 and 6.16. The LEDs are spaced 100 mm apart and alternate at a low frequency, below 1 Hz. The visual servoing system controls the orientation of the end-effector mounted camera so as to keep the image of the illuminated LED at the desired coordinate on the image plane. Only one axis, joint 6, is controlled and the magnitude of the rotation is approximately 0.17 radl. The resultant square wave response provides a useful measure of the closed-loop system performance. The controller proportional gain was set empirically on the basis of

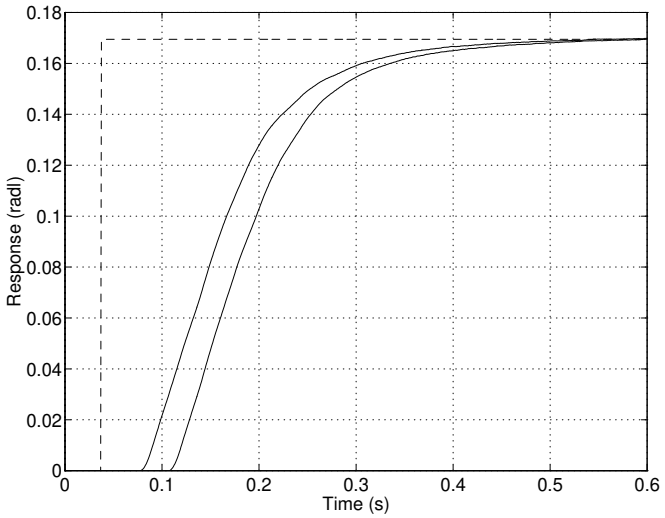


Figure 6.18: Measured response to 'visual step' demand (dashed) showing the earliest and latest responses observed. Variation is due to the the multi-rate discrete time nature of the controller. ($K_p = 1.3 \times 10^{-4}$, $K_{lens} = 708$)

observed step response.

The visual servoing system may be considered as a 'black box', whose input is the square wave signal to the LEDs and output is the robot's motor angle measured by means of its quadrature encoder signals. The input and output signals were fed to an FFT analyzer to examine the closed-loop response. Figure 6.18 shows experimental results for motor position response to the square wave visual input. The two curves show the earliest and latest responses observed. It can be seen that there is considerable delay between the change in visual demand and the onset of robot motion and that this delay is not constant. This variation is an artifact of the multi-rate discrete-time nature of the system. The delay depends upon when the excitation occurs with respect to the two samplers shown in Figure 6.15. Detailed examination of the data for Figure 6.18 shows that the earliest response is delayed by 42 ms, and the latest by 78 ms. Assuming a uniform distribution of delay the mean is 60 ms.

The FFT analyzer can also be used to estimate the transfer function of the closed-loop system, and this is shown in Figure 6.19. The transfer function is computed over many cycles and thus reflects the 'average' characteristics of the system rather than the cycle by cycle variation observed in the time response. There is evidence of a low

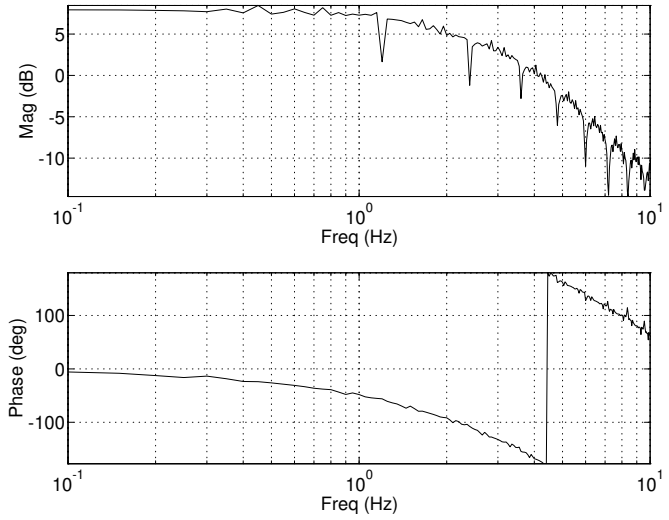


Figure 6.19: Measured closed-loop frequency response of single-axis visual servo system, $X_r(j\omega)/X_t(j\omega)$ for 'visual square wave' excitation. The periodic dips in the magnitude response are due to the lack of excitation at these frequencies as a result of the square wave system input. ($K_p = 1.3 \times 10^{-4}$, $K_{lens} = 708$)

frequency pole at approximately 2 Hz, and the phase response shows evidence of time delay. A linear frequency phase plot is shown in Figure 6.20 along with a regression fit line. This line corresponds to a delay of 58.0 ms which agrees well with the mean delay estimated from the step response measurements.

6.4.3 Modelling system dynamics

Given the experimental results just described, this section develops a detailed model of the closed-loop system which can predict the observed responses. This exercise will build upon the previously developed axis dynamic models, as well as knowledge of the lens, vision system and controller architecture. The results of the modelling exercise will be used to identify shortcomings in the present system and lead to the development of more sophisticated visual servo controllers. A block diagram of the visual servo controller for one axis was given in Figure 6.15. Figure 6.21 shows the overall timing of the two discrete-time subsystems, and the communications between them.

The effective gain of the optical system, K_{lens}, for rotational visual servoing is

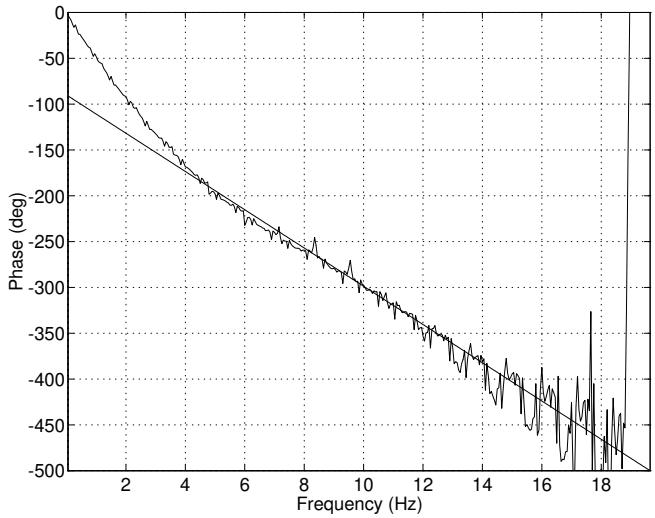


Figure 6.20: Measured closed-loop phase response on linear frequency scale. The superimposed regression fit has a slope of 20.9 deg/Hz corresponding to a delay of 58.0 ms. The line is fitted over the range 0 to 14 Hz and assumes that the line passes through -90° as it would for a single pole.

given by (6.21) and was measured for this experiment as 708 pixel/rad. A CCD camera typically responds to the integral of scene intensity over the sampling period, whereas an ideal temporal sampler, as assumed in discrete time control design, reflects the state of the observed system only at the sampling instant. With a sufficiently short exposure time a CCD camera may be used to approximate ideal temporal sampling. Figure 3.14 shows details of the exposure timing with respect to the video waveform which is an effective timing datum.

The image is transmitted pixel serially to the vision system which processes image data 'on the fly'. Overlapping data transfer and processing reduces latency to the minimum possible. Centroid data is available as soon as the region has been completely raster scanned into the vision system at a time which is dependent on the Y coordinate of the region. Such variable timing is not desirable in a control system so the region centroid data is held until the following video blanking interval. Thus the camera and vision processing may be modelled as a single time step delay,

$$V(z_v) = \frac{K_{lens}}{z_v} \tag{6.28}$$

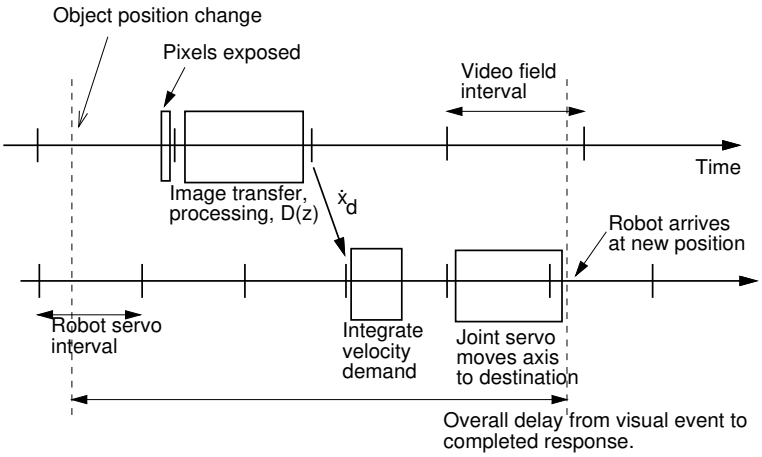


Figure 6.21: Temporal relationships in image processing and robot control. The parallel timelines represent the two discrete-time processes involved in image processing and robot control.

as indicated in Figure 6.15.

The output of the vision subsystem is the image plane pixel error which is input to a compensator, $D(z_v)$, whose output is the task-space (Cartesian) velocity command, $\dot{\underline{x}}_d$. In this section simple proportional control is used

$$D(z_v) = K_p \tag{6.29}$$

A variable delay, from 0 to 14 ms, is introduced between the output of the vision subsystem and action being taken by the robot control process, due to the asynchronous nature of the two processes.

Figure 6.4 shows that the robot control software implements a synchronous pipeline passing data from trajectory generator to the axis controller. The integration and inverse kinematics occupy one time 'slot' and the position demand is forwarded to the axis controller at the next time step. This 'double handling' of the data was discussed in Section 6.2.3 and introduces an additional delay. The combined delay and integration, implemented by the 'C' program, can be written in difference equation form as

$$x_{d_k} = x_{d_{k-1}} + \dot{x}_{d_{k-1}} \tag{6.30}$$

or as the transfer function

$$R_{integ}(z_r) = \frac{1}{z_r - 1} \tag{6.31}$$

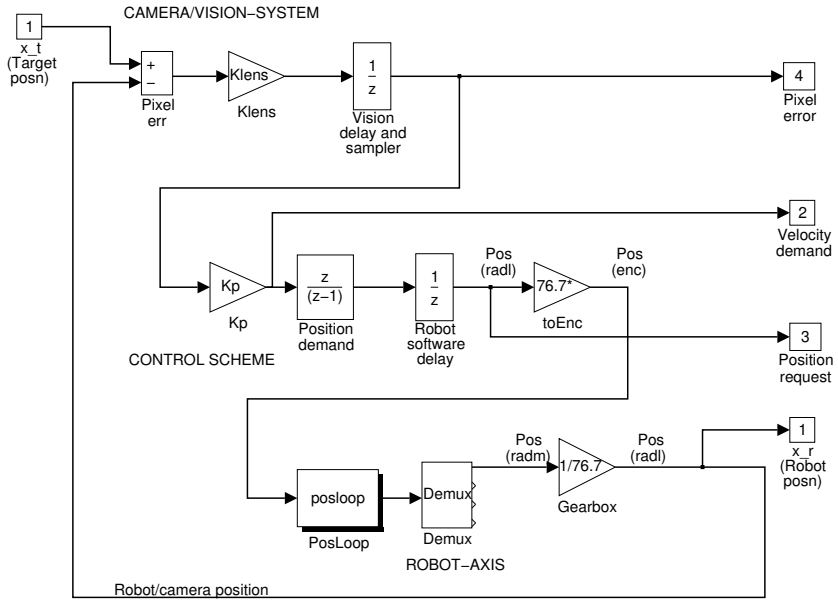


Figure 6.22: SIMULINK model of the 1-DOF visual feedback controller.

Note that the integrator as implemented is scaled when compared to the transfer function of a 'Forward Rectangular Rule' [95] integrator $T/(z-1)$ — this is an historical anomaly.

From Section 2.3.6 we know that the axis controller will always endeavour to move to the demanded position within one setpoint time interval, due to the action of the position interpolator. For 'small' motion requests, that is those within the velocity and acceleration limits of the axis, the axis behaviour may be modelled as a pure delay

$$R_{servo}(z_r) = \frac{1}{z_r} \tag{6.32}$$

allowing the robot and integrator to be modelled as

$$R(z_r) = \frac{1}{z_r(z_r-1)} \tag{6.33}$$

The SIMULINK model of Figure 6.22 is a detailed representation of the multi-rate system given in Figure 6.15, but ignoring manipulator structural dynamic effects by

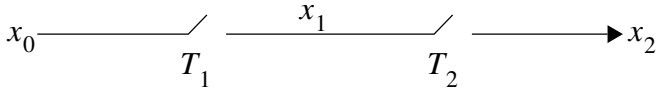


Figure 6.23: Multi-rate sampling example, $T_1 = 20$ and $T_2 = 14$.

assuming that $G_{struct}(s) = 1$. This model is based on the models POSLOOP, VLOOP and LMOTOR and parameter values established in Section 2.3. The step responses of the measured system and the model are shown together in Figure 6.26 along with that of a single-rate approximation developed later. The model response is bounded by the measured early and late response, and up to 0.16 s is parallel to the early response but lagging by around 9 ms. This lag is due to the samplers in the simulation not being synchronized with those in the measured system. The model response is slightly more damped than the measured responses.

6.4.4 The effect of multi-rate sampling

Multi-rate systems in which the rates are not integrally related are difficult to analyze, and this topic is generally given only cursory treatment in textbooks. In this section a rather pragmatic approach will be taken to modelling the multi-rate samplers as a single rate sampler plus a time delay. While not a precise analysis it does captures the dominant or 'first order' characteristics of the system.

A sampler with interval T will sample at times iT where i is an integer. An edge at time t will propagate through a single sampler at time

$$t_T = T\,\text{ceil}\left(\frac{t}{T}\right) \tag{6.34}$$

where $\text{ceil}(x)$ returns an integer i such that $i \geq x$. This implies that an edge occurring at the actual sampling instant will be propagated immediately.

Consider the example shown in Figure 6.23 which comprises two cascaded samplers operating at intervals of T_1 and T_2 respectively. An edge occurring at time t_0 propagates through the first and second samplers at times t_1 and t_2 respectively. From (6.34) we may write

$$t_1 = T_1\,\text{ceil}\left(\frac{t_0}{T_1}\right) \tag{6.35}$$

$$t_2 = T_2\,\text{ceil}\left(\frac{t_1}{T_2}\right) \tag{6.36}$$

$$\Delta_{20} = t_2 - t_0 \tag{6.37}$$

$$\Delta_{21} = t_2 - t_1 \tag{6.38}$$

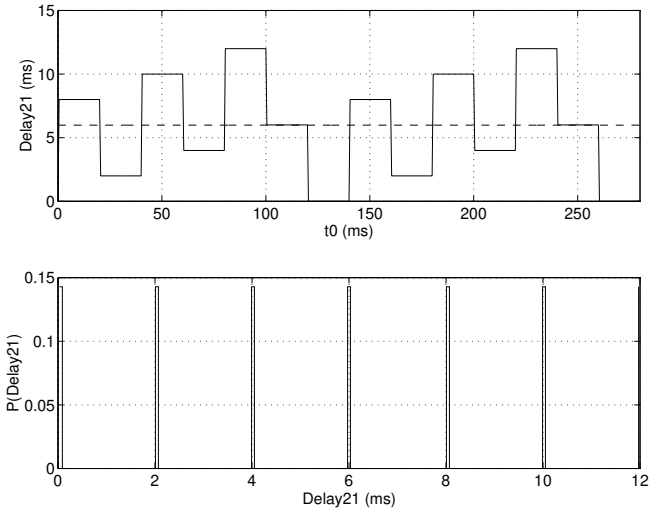


Figure 6.24: Analysis of sampling time delay Δ_{21} for the case where $T_1 = 20\,\text{ms}$ and $T_2 = 14\,\text{ms}$. Top shows variation in delay versus sampling time (mean of 6 ms shown dashed), while bottom shows probability distribution.

where Δ_{20} is the delay of the edge from input to output, and Δ_{21} is the delay of the edge through the second sampler. A numerical simulation of the delay for the visual servo control case where $T_1 = 20\,\text{ms}$ and $T_2 = 14\,\text{ms}$ has been conducted. Figures 6.24 and 6.25 show Δ_{21} and Δ_{20} respectively, along with the corresponding probability distributions of the delays. Both delay functions are periodic, with a period of 140 ms, the least common multiple of 14 and 20. The mean delay values, marked on the delay plots, are $\bar{\Delta}_{20} = 16\,\text{ms}$ and $\bar{\Delta}_{21} = 6\,\text{ms}$. However the mean delay is sensitive to the relative 'phasing' of the samplers, and further simulation shows that $\bar{\Delta}_{21}$ varies periodically between 6 and 8 ms as phase shift between the samplers is increased. The mean of $\bar{\Delta}_{21}$ over all phase shifts is one-half the sampler interval, or 7 ms.

Clearly the principal effect of multi-rate sampling is to introduce a small time delay. The two samplers may be approximated by a single sampler and a delay in the range 6 to 8 ms depending on phasing which is unknown[6]. Assuming a sampler delay of 7 ms combined with the already identified delays such as 20 ms for pixel transport, 14 ms servo communications and 14 ms robot motion delay, the total latency is 55 ms.

[6] Since the two sampling processes are controlled by separate and unsynchronized clocks it is likely that the phasing will vary with time.

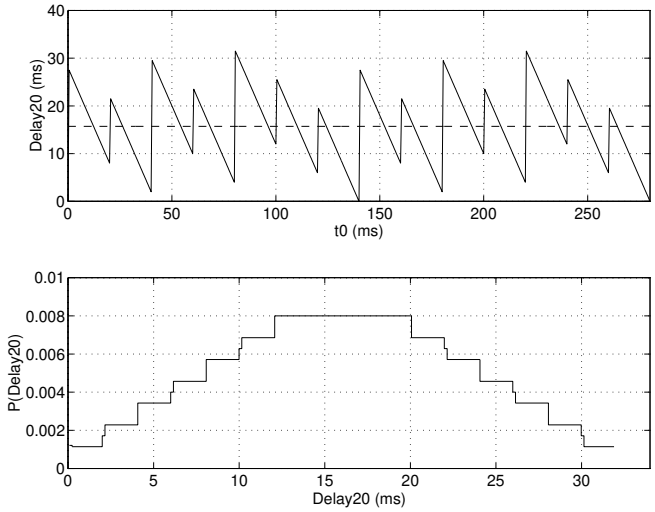


Figure 6.25: Analysis of sampling time delay Δ_{20} for the case where $T_1 = 20\,\text{ms}$ and $T_2 = 14\,\text{ms}$. Top shows variation in delay versus sampling time (mean of 16 ms shown dashed), while bottom shows probability distribution.

This is consistent with earlier estimates of 58 ms and 60 ms delay. Further refinement is possible by accounting for further multi-rate effects between the 14 ms robot control process and the 980 μs digital servo board process, but this will not be pursued further.

6.4.5 A single-rate model

To simplify analysis the system may be approximated by a single-rate model operating with a sampling interval of T_v. As implemented $V(z)$ and $D(z)$ already operate at T_v, but it is necessary to express the robot dynamics, $R(z)$, at this sampling interval

$$R(z_r) = \frac{1}{z_r(z_r-1)} \tag{6.39}$$

$$= \frac{1}{T_r}\frac{1}{z_r}\frac{T_r}{(z_r-1)} \tag{6.40}$$

$$\approx \frac{1}{T_r}\frac{1}{z_v}\frac{T_v}{(z_v-1)} \tag{6.41}$$

where the rightmost term, the integrator, has been changed from the robot sample interval to the visual sample interval, and the unit delay T_r has been approximated by T_v. Introducing the notation $z = z_v$ the open-loop transfer function may be written as

$$V(z)D(z)R(z) = \frac{K_{lens}}{z} K_p \frac{1}{T_r} \frac{1}{z} \frac{T_v}{(z-1)} \tag{6.42}$$

$$= \frac{K_p K_{lens}}{z^2(z-1)} \frac{T_v}{T_r} \tag{6.43}$$

The resulting single-rate closed-loop transfer function is

$$\frac{x_r(z)}{x_t(z)} = \frac{K}{z^3 - z^2 + K} \tag{6.44}$$

where

$$K = \frac{T_v}{T_r} K_p K_{lens} \tag{6.45}$$

The overall delay is $\mathrm{Ord}(\text{denominator}) - \mathrm{Ord}(\text{numerator})$, in this case three sample periods or 60 ms. From Figure 6.21 it can be seen that these are due to pixel transport, velocity computation and servo response. This 60 ms delay agrees with previous observations of latency and the step response of this model, shown in Figure 6.26, is seen to be bounded by the measured early and late responses. The single rate response starts close to the early response and finishes closer to the late response, showing that it has captured the 'average' characteristic of the real system. This model is also slightly more damped than the measured responses. The model (6.44) is the same as that determined by Corke and Good [62] for the case of multi-axis translational fixation.

The single-rate open-loop model (6.43) can be used for stability analysis as well as state-estimator and controller design using classical digital control theory. The root-locus diagram in Figure 6.27 shows that for the known gain setting, $K = 0.132$, the poles are located at $z = 0.789$, 0.528 and -0.316. The dominant pole corresponds to a frequency of 1.9 Hz which is in good agreement with the estimate of 2 Hz made earlier. This pole is very sensitive to changes in loop gain such as those caused by variation in target distance. Figure 6.28 shows the Bode plot of the single rate model overlaid on the experimentally determined frequency response function from Section 6.4.2.

6.4.6 The effect of camera shutter interval

Section 3.3.3 discusses the effect of motion blur in terms of shape distortion and lagging centroid estimate. Figure 6.29 shows very clearly that with an exposure interval of 20 ms the apparent area of the LED target is increased during the high velocity

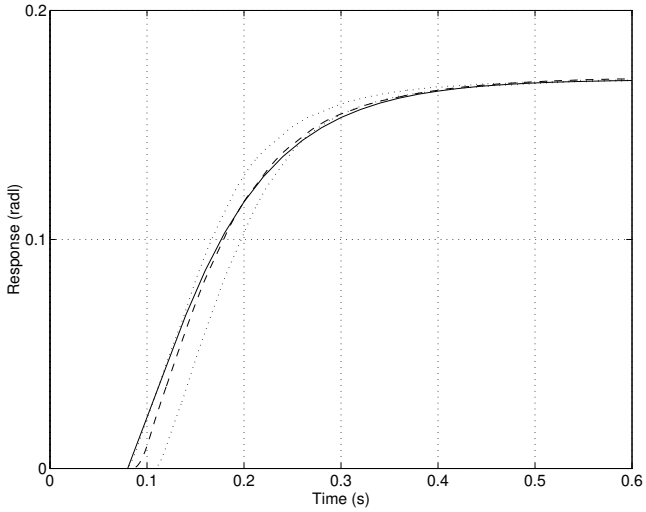


Figure 6.26: Comparison of measured and simulated step response. Measured early and late step responses (dotted), single-rate model (solid), and the multi-rate model (dashed). ($f = 8\,\text{mm}$, $K_{lens} = 724$, and $K_p = 1.3 \times 10^{-4}$)

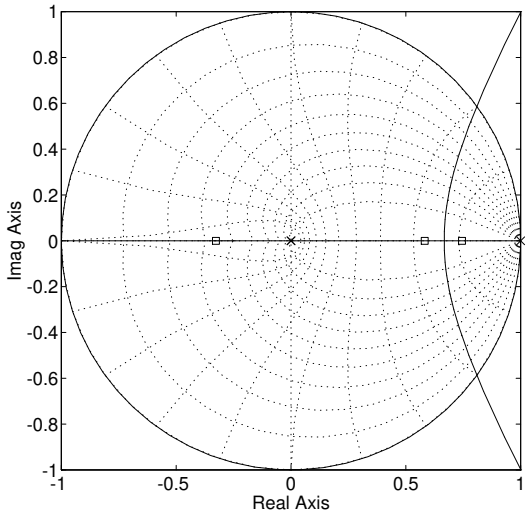


Figure 6.27: Root-locus of single-rate model. Closed-loop poles for gain $K_p = 1.3 \times 10^{-4}$ are marked ($K_{lens} = 708$).

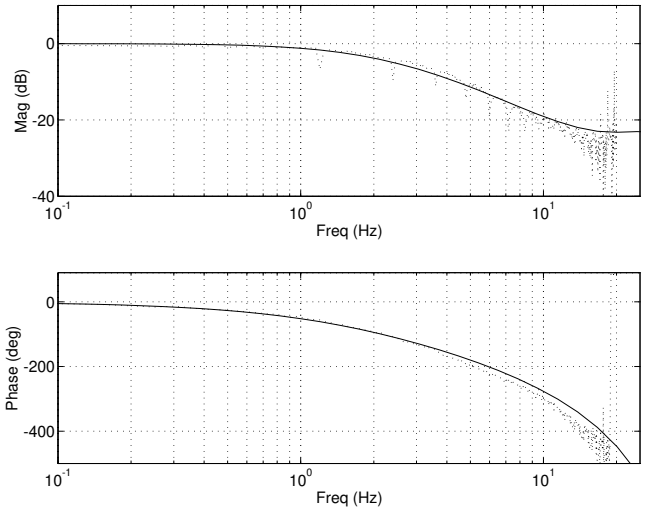


Figure 6.28: Bode plot of single-rate model closed-loop transfer function (solid) with measured frequency response (dotted) overlaid. Note the considerable phase error for moderate frequency demands, eg. 1 Hz. ($K_p = 1.3 \times 10^{-4}$, $K_{lens} = 708$)

phase of the fixation motion. The camera rotation rate is 1.5 radl/s and has increased the apparent area by a factor of 2.6. The circularity, (4.11), also drops to 0.76 from its normal value of 1.0 as the target image becomes elongated. The experimental visual servo system uses simple criteria to identify the target region from other regions in the scene, usually on the basis of area and circularity. These criteria must be relaxed in order that the controller does not 'lose sight' of the target when its apparent shape changes so dramatically.

The other effect of motion blur is to introduce a lagging centroid estimate. This will alter the dynamics of the closed-loop system by increasing open-loop latency. Figure 6.30 shows that the increased latency has had the expected 'destabilizing' effect, resulting in increased overshoot.

6.4.7 The effect of target range

Section 6.3.2 showed that a visual servoing system introduces a target distance dependent gain due to the lens, which will affect the loop gain and stability of the closed-loop system. Few reports on visual servoing discuss this effect, but much work is performed with fixed camera-target distances — perhaps due to focus or depth of field

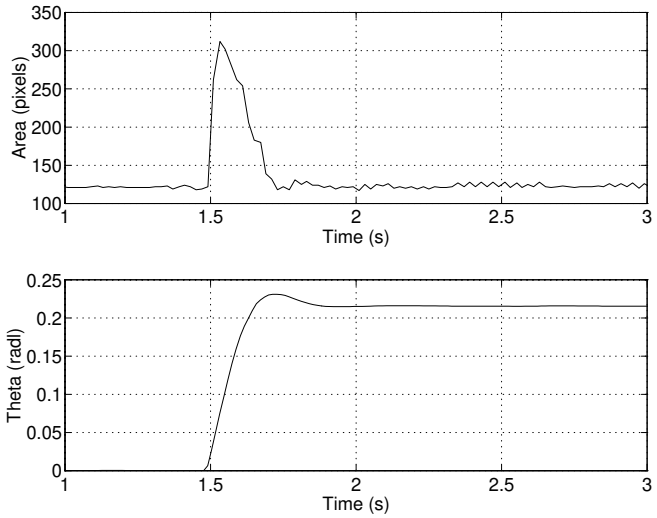


Figure 6.29: Measured effect of motion blur on apparent target area for 'long' exposure interval ($T_e = 20$ ms). The lower plot shows the camera pan angle in response to a step change in target position; the apparent target area at corresponding times is shown in the upper plot. During the highest-velocity motion apparent area of the LED target has increased by a factor of 2.6.

problems. This issue is mentioned by Dzialo and Schalkoff [81] in the control of a pan-tilt camera head. Experimental results, for the case of translational fixation, have been reported by Corke and Good [62]. Pool [206] recognized this problem in the citrus picking application, as the robot approaches the fruit, and used an ultrasonic sensor to measure range in order to modify the visual loop gain during approach motion.

Measured step responses for different camera-object distances are shown in Figure 6.31. The increased loop gain for low target range results in increased overshoot. In this experiment, with rotational fixation motion, the distance dependence is weak. With translational fixation this effect is much more pronounced [62]. An adaptive controller or a self-tuning regulator could maintain the desired dynamic response as target distance changed, and the parameter values would be a function of target distance. This introduces the possibility of determining object distance from closed-loop dynamics using a single camera and without prior knowledge of object size.

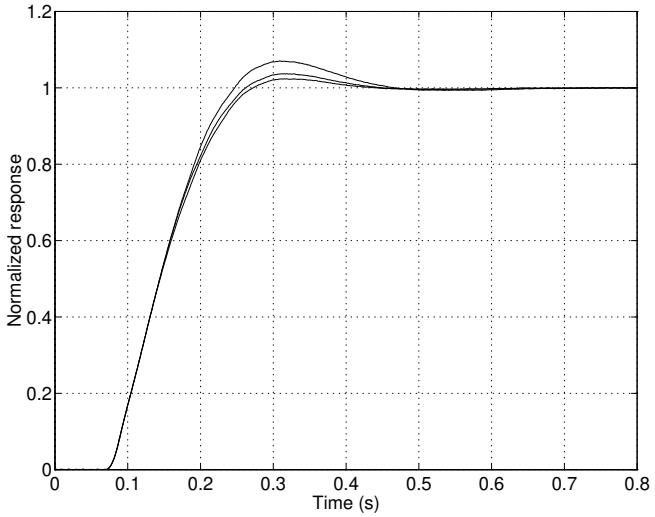


Figure 6.30: Measured step responses for exposure intervals of 2 ms, 8 ms and 20 ms. The response becomes more oscillatory as the exposure interval is increased. Conditions are object range 460 mm and $K_p = 1.4 \times 10^{-4}$.

6.4.8 Comparison with joint control schemes

The visual servo controller and the conventional axis position controller described in Section 2.3.6 both employ a sensor which quantizes position into unit steps: either pixels or encoder counts. For the joint 6 visual servo experiment described in this section, and ignoring center of rotation issues, one pixel corresponds to

$$1\,\text{pixel} = \frac{1}{K_{lens}} \tag{6.46}$$

$$= \frac{1}{\alpha_x f} \tag{6.47}$$

$$= 1.6 \times 10^{-3}\,\text{radl} \tag{6.48}$$

Increased angular resolution can be obtained by increasing the focal length, f, of the lens, but at the expense of reduced field of view. The angular resolution of the axis position controller is determined by the encoder pitch which is summarized in Table 2.19.

$$1\,\text{encoder} = \frac{2\pi}{N_{enc}} \frac{1}{G_6} \tag{6.49}$$

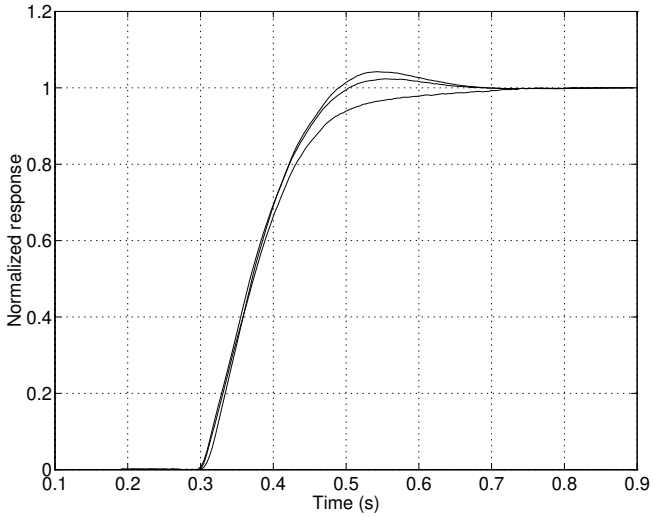


Figure 6.31: Measured step response for target ranges of 395 mm, 470 mm and 890 mm. The response becomes more oscillatory as the target range is decreased.

$$= \quad 1.6 \times 10^{-4}\,\text{radl} \tag{6.50}$$

which is one order of magnitude better than the visual servo, and this axis has the poorest encoder resolution. However the vision system can achieve some measure of sub-pixel precision as was discussed in Section 4.1.3. Comparable accuracy with the axis position controller would be obtained if the centroid estimate were accurate to 0.1 pixel. At the 3σ confidence level

$$3\sigma_{X_c} = 0.1 \tag{6.51}$$

which from (4.53) would be achieved for a circular target with a diameter greater than 40 pixels.

6.4.9 Summary

This section has investigated in considerable detail the operation of a single DOF visual-feedback control system. With simple proportional control it has been possible to demonstrate stable and high-speed motion control at rates approaching the limits established in Chapter 2. This has provided insights into the interaction between the

dynamic characteristics of the robot, its controller, and the vision system. The visual servo system has been modelled in some detail and the model validated against experimental results. The actual system is multi-rate, which complicates analysis, but an effective single-rate approximation has been developed that compares well in terms of time and frequency response. The multi-rate sampling process was analyzed and shown to have a first-order characteristic that is a constant delay. A number of system characteristics that affect the dynamic response have been investigated including lens center of rotation, exposure interval and target range.

The experimental system described is well suited for this application and is able to minimize delay by overlapping serial pixel transport and image processing. Despite this, the dominant dynamic characteristic remains delay due to pixel transport, computation and finite motion time. Some enhancements to the architecture are possible which will reduce delay but it cannot be entirely eliminated. The next chapter will investigate more sophisticated controllers and evaluate them in terms of a defined performance metric.

Chapter 7

Control design and performance

Most reported visual servo systems do not perform as well as would be expected or desired. The most obvious characteristics are slowness of motion, lag with respect to target motion, and often significant jitter or shakiness. All of these characteristics are indicative of poorly designed control systems, inappropriate control architectures or both. This chapter focusses on issues of control system performance and design: defining the control systems problem, defining performance measures, and comparing a number of different control techniques for challenging target motion.

As shown in the previous chapters machine vision has a number of significant disadvantages when used as a feedback sensor: a relatively low sample rate, significant latency (one or more sample intervals) and coarse quantization. While these characteristics present a challenge for the design of high-performance motion control systems they are not insurmountable.

Latency is the most significant dynamic characteristic and its sources, as discussed previously, include: transport delay of pixels from camera to vision system, image processing algorithms, communications between vision system and control computer, control algorithm software, and communications with the robot. In fact problems of delay in visual servo systems were first noted over 15 years ago [116]. If the target motion is constant then prediction can be used to compensate for latency, but combined with a low sample rate results in poor disturbance rejection and long reaction time to target 'maneuvers', that is, unmodeled motion. Grasping objects on a conveyor belt [290] or a moving train [7] are however ideal applications for prediction.

Franklin [95] suggests that the sample rate of a digital control system be between 4 and 20 times the desired closed-loop bandwidth. For the case of a 50 Hz vision system this implies that a closed-loop bandwidth between 2.5 Hz and 12 Hz is achievable.

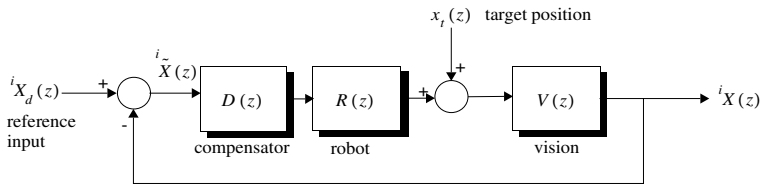


Figure 7.1: System block diagram of of a visual feedback control system showing target motion as disturbance input.

That so few reported systems achieve this leads to the conclusion that the whole area of dynamic modelling and control design has, to date, been largely overlooked.

7.1 Control formulation

Two useful transfer functions can be written for the visual servo system of Figure 7.1. The response of image plane centroid to demanded centroid is given by the transfer function

$$\frac{^iX(z)}{^iX_d(z)} = \frac{V(z)R(z)D(z)}{1+V(z)R(z)D(z)} \tag{7.1}$$

and substituting (6.28), (6.33), and $D(z) = D_N(z)/D_D(z)$ results in

$$\frac{^iX(z)}{^iX_d(z)} = \frac{K_{lens}TD_N(z)}{z^2(z-1)D_D(z)+K_{lens}TD_N(z)} \tag{7.2}$$

Jang [134] has shown how tasks can be expressed in terms of image plane feature trajectories, $^iX_d(t)$, for which this image plane reference-following performance is significant.

For a fixation task, where ^iX_d is constant, $^iX(z)/X_t(z)$ describes the image plane error response to target motion, and is given by

$$\frac{^iX(z)}{X_t(z)} = \frac{V(z)}{1+V(z)R(z)D(z)} \tag{7.3}$$

$$= \frac{K_{lens}z(z-1)D_D(z)}{z^2(z-1)D_D(z)+K_{lens}D_N(z)} \tag{7.4}$$

where $^iX = {}^iX_d - {}^iX$ is the image plane error and ^iX_d is assumed, for convenience, to be zero. The motion of the target, x_t, can be considered a disturbance input and ideally the disturbance response, $|^iX(z)/X_t(z)|$, would be zero. However with the previously identified dynamics it is non zero and rises with frequency. To obtain better tracking

performance it is necessary to reduce $|{}^{i}X(z)/X_t(z)|$ over the target motion's frequency range.

The steady-state value of the image plane error for a given target motion, $X_t(z)$, can be found by the final-value theorem

$$\lim_{t\to\infty} {}^{i}X(t) = \lim_{z\to 1} (z-1){}^{i}X(z) \tag{7.5}$$

$$= \lim_{z\to 1} (z-1)\frac{V(z)}{1+V(z)R(z)D(z)}X_t(z) \tag{7.6}$$

Using proportional control and substituting previously identified models for $V(z)$ and $R(z)$ the steady-state error is

$${}^{i}X(\infty) = \lim_{z\to 1} (z-1)\frac{K_{lens}z(z-1)}{z^2(z-1)+K_pK_{lens}}X_t(z) \tag{7.7}$$

which may be evaluated with the target motion chosen as one of

$$X_t(z) = \begin{cases} \frac{z}{z-1} & \text{for a step input} \\ \frac{Tz}{(z-1)^2} & \text{for a ramp input} \\ \frac{T^2}{2}\frac{z(z+1)}{(z-1)^3} & \text{for a parabolic input} \end{cases} \tag{7.8}$$

For the steady-state tracking error to be zero, the numerator of (7.7) must cancel the poles at $z=1$ of $X_t(z)$ and retain a factor of $(z-1)$. This numerator comprises the poles of the robot and compensator transfer functions — the latter may be selected to give the desired steady-state error response by inclusion of an appropriate number of integrator terms, or $(z-1)$ factors. The number of such open-loop integrators is referred to as the Type of the system in classical control literature. Equation (7.7) has one open-loop integrator and is thus of Type 1 giving zero steady-state error to a step input. The addition of an integrator to raise system Type supports the intuitive argument of Section 6.4 which suggested that the control problem be considered as a 'steering problem', where the robot velocity, not position, be controlled.

A ramp input however will result in a finite error of T/K_p pixels which can be seen as the substantial lag in Figure 7.2. Also note that the numerator of the closed-loop transfer function (7.4) has a zero at $z=1$ which acts as a differentiator. From (7.4) compensator poles will appear as additional closed-loop zeros.

Common approaches to achieving improved tracking performance are:

1. Increasing the loop gain, K_p, which minimizes the magnitude of ramp-following error. However for the system, (6.44), this is not feasible without additional compensation as the root-locus plot of Figure 6.27 showed.

2. Increasing the Type of the system, by adding open-loop integrators. Type 1 and 2 systems will have zero steady-state error to a step and ramp demand respectively.

3. Introducing feedforward of the signal to be tracked. This is a commonly used technique in machine-tool control and has been discussed, for instance, by Tomizuka [251]. For the visual servo system, the signal to be tracked is the target position which is not directly measurable. It can however be estimated and this approach is investigated in Section 7.5.

7.2 Performance metrics

It is currently difficult to compare the temporal performance of various visual servo systems due to the lack of any agreed performance measures. At best only qualitative assessments can be made from examining the time axis scaling of published results (is it tracking at 1 mm/s or 1000 mm/s ?) or video tape presentations.

Traditionally the performance of a closed-loop system is measured in terms of bandwidth. While 'high bandwidth' is desirable to reduce error it can also lead to a system that is sensitive to noise and unmodeled dynamics. However the notion of bandwidth is not straightforward since it implies both magnitude (3 dB down) and phase (45° lag) characteristics. Time delay introduces a linear increase in phase with frequency, and the Bode plot of Figure 6.28 indicates 45° lag occurs at less than 1 Hz. For tracking applications phase is a particularly meaningful performance measure — for instance the citrus picking robot's tracking specification was in terms of phase.

Other commonly used performance metrics such as settling time, overshoot and polynomial signal following errors are appropriate. Note though that the feedback system developed in the previous section has a 'respectable' square wave response but experiments show poor ability to follow a sinusoid. Simulated tracking of sinusoidal motion in Figure 7.2 shows a robot lag of approximately 30° and the centroid error peaking at over 80 pixels.

In this work we choose to consider the magnitude of image plane error, either peak-to-peak or RMS, in order to quantitatively measure performance. The RMS error over the time interval $[t_1, t_2]$ is given by

$$\eta = \sqrt{\frac{\int_{t_1}^{t_2} \{{}^iX(t) - {}^iX_d(t)\}^2 \, dt}{t_2 - t_1}} \tag{7.9}$$

An image-plane error measure is appropriate since the fixation task itself is defined in terms of image plane error, and is ideally zero. Image plane error is computed by the controller and can easily be logged, and this is considerably simpler than estimating closed-loop bandwidth.

The choice of appropriate target motion with which to evaluate performance depends upon the application. For instance if tracking a falling or flying object it would be appropriate to use a parabolic input. A sinusoid is particularly challenging since

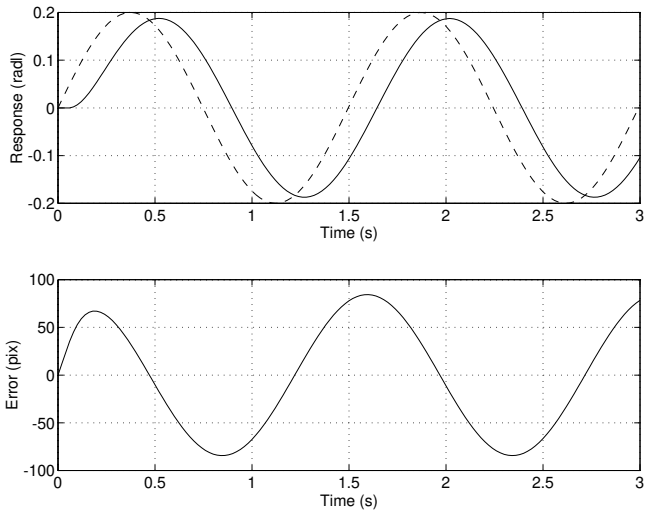


Figure 7.2: Simulated tracking performance of visual feedback controller, where the target is moving sinusoidally with a period of 1.5 s (0.67 Hz). Note the considerable lag of the response (solid) to the demand (dashed).

it is a non-polynomial with significant and persistent acceleration that can be readily created experimentally using a pendulum or turntable. Sinusoidal excitation clearly reveals phase error which is a consequence of open-loop latency. In the remainder of this work a fixed frequency sinusoid

$$x_t(t) = M \sin(4.2t) \tag{7.10}$$

will be used as the standard target motion. This corresponds to an object rotating on the laboratory turntable at its maximum rotational speed of 40 rpm (0.67 rev/s). The magnitude, M, of the target motion depends on the rotational radius of the target, and the distance of the camera. Peak angular target velocity is $4.2M$ rad/s and can be selected to challenge the axis capability.

7.3 Compensator design and evaluation

7.3.1 Addition of an extra integrator

A controller with improved tracking performance must incorporate at least one

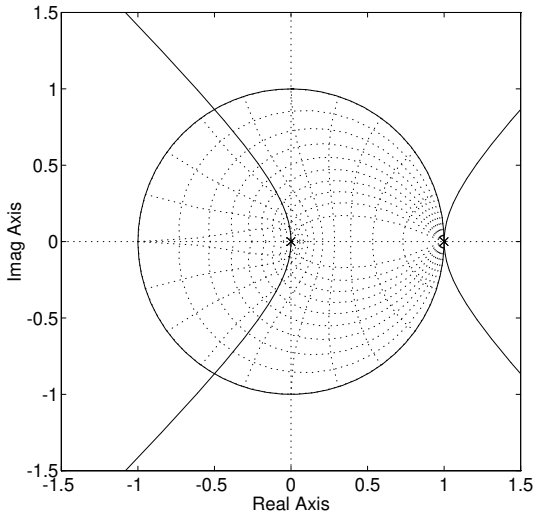


Figure 7.3: Root locus for visual feedback system with additional integrator and proportional control. Clearly the system is unstable for any finite loop gain.

additional integrator for Type 2 characteristics. The open-loop transfer function (6.43) becomes

$$\frac{K_p K_{lens}}{z^2(z-1)^2} = \frac{K_p K_{lens}}{z^4 - 2z^3 + z^2} \tag{7.11}$$

which, without additional compensation, is unstable for any finite loop gain as shown by the root-locus in Figure 7.3.

7.3.2 PID controller

A discrete time PID compensator can be written as

$$D(z) = P + \frac{Tz}{z-1} I + \frac{1-z^{-1}}{T} D \tag{7.12}$$

$$= \frac{z^2(PT + IT^2 + D) - z(PT + 2D) + D}{Tz(z-1)} \tag{7.13}$$

where P, I and D are the proportional, integral and derivative gains respectively, and T the sampling interval. The derivative is computed by means of a simple first-order difference. This compensator adds a pole at $z = 1$ which increases the Type of the

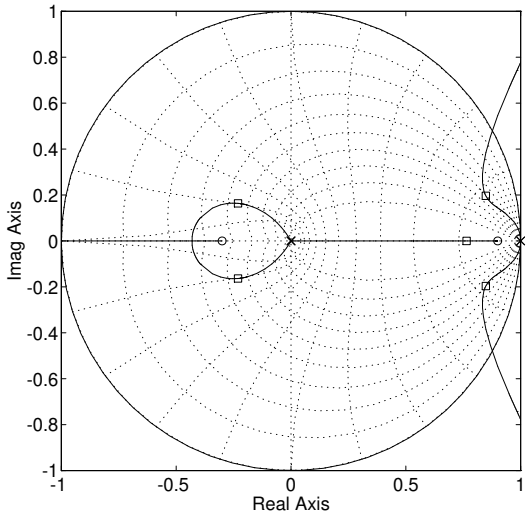


Figure 7.4: Root locus for visual feedback system with PID controller which places the zeros at $z = -0.3$ and $z = 0.9$. The closed-loop poles marked by squares are used in the simulation of Figure 7.5.

system as desired, and allows arbitrary placement of the two zeros to shape the root locus.

An example root locus is shown in Figure 7.4. One zero is placed close to $z = 1$ in order to 'bend' the arms of the root locus inside the unit circle, however it is not possible to move the dominant closed-loop poles so as to achieve a high natural frequency. The closed-loop transfer function is

$$\frac{{}^{i}X(z)}{x_t(z)} = \frac{K_{lens}Tz^2(z-1)^2}{z^3(z-1)^2T + K_{lens}T\left(z^2(PT+IT^2+D) - z(PT+2D)+D\right)} \tag{7.14}$$

which, since the system Type is 2, has a pair of closed-loop zeros at $z = 1$. This double differentiator causes significant overshoot in response to target acceleration — note the initial transient in Figure 7.5. The compensator zeros could be configured as a complex pair but must be placed close to $z = 1$ if they are to 'capture' the open-loop integrator poles, which would otherwise leave the unit circle rapidly.

In practice it is found that derivative gain must be limited, since the derivative term introduces undesirable 'roughness' to the control, and this constrains the locations of the open-loop zeros. This roughness is a consequence of numerically differentiating the quantized and noisy centroid estimates from the vision system. Åström and

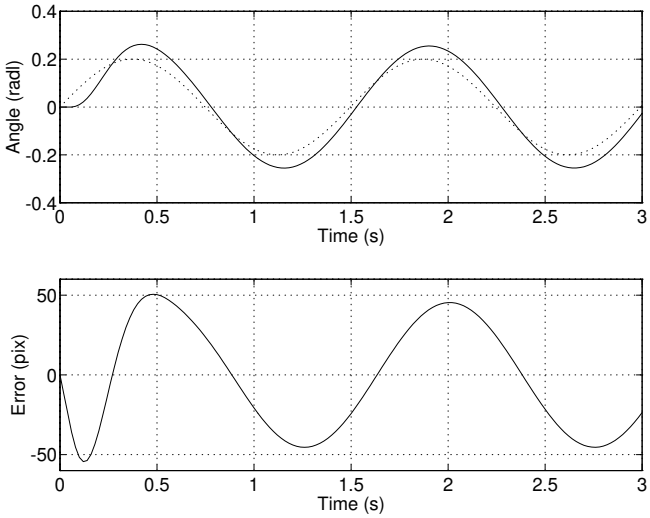


Figure 7.5: Simulated response for visual feedback system with PID controller which places the zeros at $z = -0.3$ and $z = 0.9$. Closed-loop poles at $z = 0.85 \pm 0.20j$, equivalent damping factor $\zeta = 0.52$.

Wittenmark [24] propose filtering the derivative estimate and while this may allow higher derivative gain this form of compensator has only limited ability to position the closed-loop poles[1]. The simulation results in Figure 7.5 show the time response of this compensator to the standard target trajectory when the closed-loop poles are placed as marked on Figure 7.4. The motion exhibits considerable overshoot and the peak-to-peak image plane error is approximately 90 pixels.

7.3.3 Smith's method

Smith's method, also known as Smith's predictor [24], was proposed in 1957 for the control of chemical processes with significant, and known, time delays [238] and there are some reports of its application to visual servoing [37, 229]. Consider a plant with dynamics

$$H(z) = \frac{1}{z^d} H'(z) = \frac{1}{z^d} \frac{B'(z)}{A'(z)} \tag{7.15}$$

[1] Three controller parameters (P, I and D) are insufficient to arbitrarily 'place' the five poles.

where $\mathrm{Ord}(A') = \mathrm{Ord}(B')$ and d is the time delay of the system. A compensator, $D'(z)$, is designed to give desired performance for the delay-free system $H'(z)$. Smith's control law gives the plant input as

$$U = D'\{Y_d - Y\} - D'H'\left\{1 - z^{-d}\right\}U \tag{7.16}$$

which can be expressed as a compensator transfer function

$$D(z) = \frac{z^d A'D'}{z^d(A' + B'D') - B'D'} \tag{7.17}$$

which clearly attempts to cancel the plant dynamics $A'(z)$. The visual servo open-loop model of (6.43) can be written in the form

$$H(z) = \frac{1}{z^2}\frac{K}{z-1} = \frac{1}{z^2}H'(z) \tag{7.18}$$

and a simple proportional compensator, $D' = K_p$, used to control the dynamics of the first-order system $H'(z)$. The resulting compensator by (7.17) is

$$D(z) = \frac{z^2(z-1)K_p}{z^2(z-1) + KK_p(z^2-1)} \tag{7.19}$$

When simulated with the linear single-rate model the compensator gives the expected performance and the pole can be positioned to give arbitrarily fast performance. However when simulated with the full non-linear multi-rate model similar to Figure 6.22 the effect of model errors became apparent. The closed-loop pole of $H'(z)$ could no longer be arbitrarily positioned and attempts to place the pole at $z < 0.6$ led to poor dynamic response and instability. Figure 7.6 shows the response for the case of $z = 0.6$ which shows a peak-to-peak error of nearly 70 pixels.

Sharkey et al. [229] report experimental gaze control results obtained with the 'Yorick' head. Using accurate knowledge of actuator and vision processing delays a target position and velocity predictor is used to counter the delay and command the position controlled actuators. They suggest that this scheme "naturally emulates the Smith Regulator" and block diagram manipulation is used to show the similarity of their scheme to that of Smith, however no quantitative performance data is given. Brown [37] used simulation to investigate the application of Smith's predictor to gaze control by means of simulation and thus did not have problems due to plant modelling error encountered here. He also assumed that all delay was in the actuator, not in the sensor or feedback path.

7.3.4 State feedback controller with integral action

State-feedback control can be used to arbitrarily position the closed-loop poles of a plant. For best tracking performance the poles should all be placed at the origin in

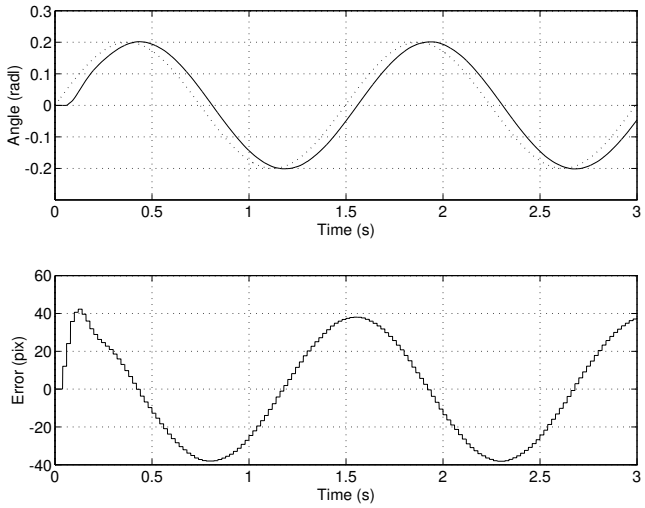


Figure 7.6: Simulation of Smith's predictive controller (7.23) and detailed non-linear multi-rate system model. Simulated response (solid) and demand (dotted).

order that the closed-loop system functions as a pure delay, but this may result in un-realizable control inputs to the plant.

The configuration of plant, state estimator and controller are shown in Figure 7.7. The open-loop system, $V(z)R(z)$ is represented in state-space form as

$$\underline{x}_{k+1} = \mathbf{\Phi}\underline{x}_k + \mathbf{\Delta}\underline{u}_k \tag{7.20}$$

$$\underline{y}_k = \mathbf{C}\underline{x}_k \tag{7.21}$$

The estimator shown is predictive, and allows one full sample interval in which to compute the state estimate, $\underline{\hat{x}}$, and the control input, $\underline{u}$. Within the labelled compensator block there is feedback of the computed plant input to the estimator. The closed-loop compensator poles are the eigenvalues of

$$\mathbf{\Phi} - \mathbf{K}_e\mathbf{C} - \mathbf{\Delta}\mathbf{K} \tag{7.22}$$

where $\mathbf{K}$ is the state-feedback gain matrix and $\mathbf{K}_e$ is the estimator gain matrix. The separation principle [95] states that the poles of a closed-loop system with an estimator is the union of the poles of the estimator and the controlled plant, and that these may be designed independently. However there is an additional consideration that the compensator itself must be stable, and this constrains the selection of plant and

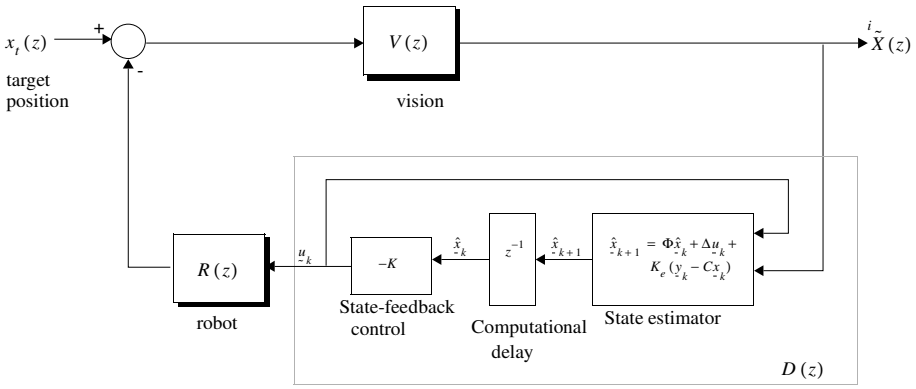


Figure 7.7: Visual feedback system with state-feedback control and estimator.

estimator pole locations. The requirement of compensator stability appears to be a matter of some debate and is not well covered in standard texts. Simple examples can be contrived in which an unstable plant and an unstable compensator result in a stable closed-loop system. However Maciejowski [178] states that there is a "theoretical justification for preferring (open-loop) stable controllers — but there are also powerful practical reasons for this preference". Simulations show that while an unstable compensator may give perfectly satisfactory performance in a linear discrete-time simulation, a more detailed simulation or the introduction of model error can induce instability. This is believed to be related to plant non-linearity, where 'internal' signals between the compensator and plant may result in saturation, changing the apparent dynamics of the plant and leading to instability.

Compensators have been designed for the single-rate approximation (6.43) using pole placement and LQG design procedures. Åström and Wittenmark [24] describe a procedure for designing pole placement compensators for SISO systems based on state feedback and state estimation. Considerable experimentation was required to choose closed-loop and estimator poles that result in a stable compensator. Franklin [95] suggests that the estimator poles be 4 times faster than the desired closed-loop poles. The upper limit on estimator response is based on noise rejection and modelling errors. To reduce the degrees of freedom in the control/estimator design process the estimator poles were placed at $z = 0,0,0$ giving a dead-beat response. In detailed simulation and experiment this has given satisfactory performance. Introducing an additional open-loop integrator, and placing the closed-loop poles at $z = 0.6 \pm 0.4j, 0.4$ results in the compensator

$$D(z) = \frac{z^2(7.01z - 5.65)}{z^3 - 0.6z^2 + 0.4z - 0.8} \tag{7.23}$$

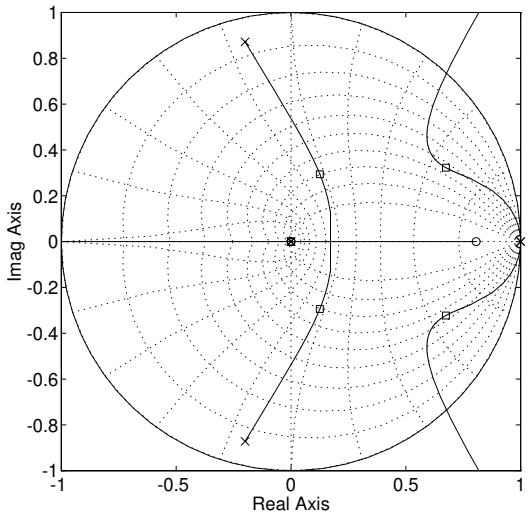


Figure 7.8: Root locus for pole-placement controller. Closed-loop poles at $z = 0.6 \pm 0.4j$ and $z = 0.4$. Note that a pair of poles and zeros have been cancelled at the origin.

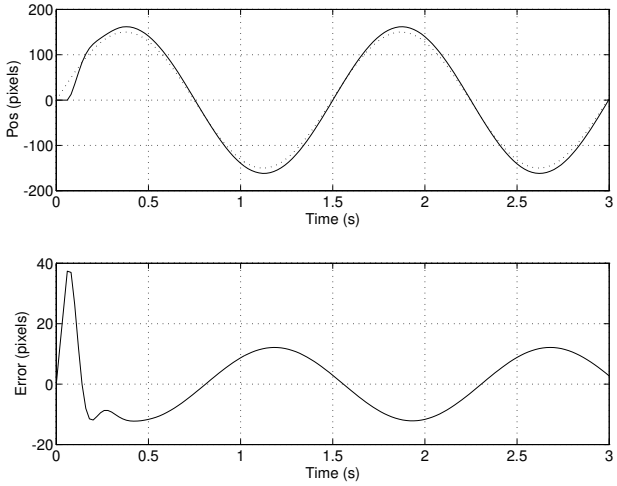
$$= \quad 7.01 \frac{z^2(z-0.806)}{(z-1)(z-\{-0.2 \pm 0.872j\})} \tag{7.24}$$

The root-locus diagram in Figure 7.8 shows that the compensator has introduced a complex pole pair near the unit circle and has been able to achieve faster closed-loop poles than the PID compensator of Figure 7.4. Figure 7.9 shows the simulated closed-loop response for the standard target motion for both the single-rate model and the detailed SIMULINK model. The simulation results are similar but the more detailed model is somewhat rougher due to the multi-rate effects. The peak-to-peak error in pixel space is approximately 28 pixels.

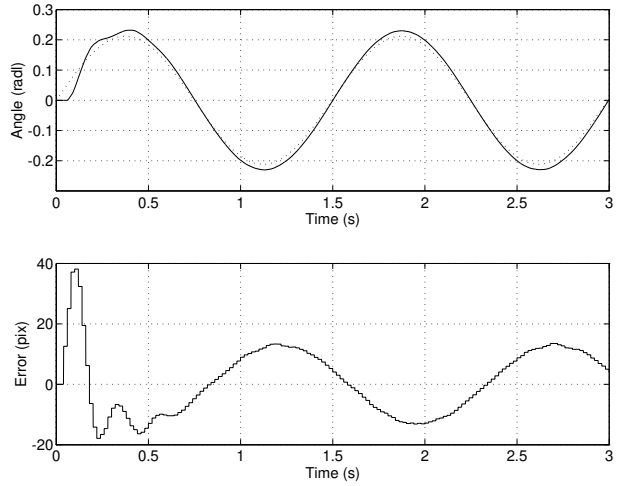
Clearly the tracking performance is greatly improved compared to simple proportional control but a number of characteristics of the response deserve comment:

- The error is greatest at the peaks of the sinusoid, which correspond to greatest acceleration. The Type 2 controller has zero error following a constant velocity target (which the slopes of the sinusoid approximate), but will have finite error for an accelerating target.

- There is considerable delay at $t = 0$ before the robot commences moving, due



Discrete-time linear single-rate simulation.



Non-linear multi-rate simulation.

Figure 7.9: Simulation of pole-placement fixation controller (7.23). Simulated response (solid) and demand (dotted). Closed-loop poles at $z = 0.6 \pm 0.4j, 0.4$.

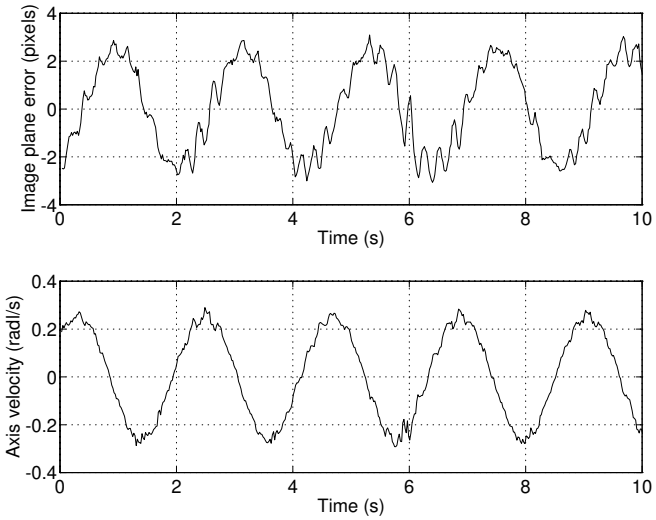


Figure 7.10: Experimental results with pole-placement compensator. Note that the target frequency in this experiment is only 0.5 Hz, less than the standard target motion frequency. Closed-loop poles at $z = 0.6 \pm 0.4\,j, 0.4$.

to delay in the vision system and controller.

- Once again this is a Type 2 system and the closed-loop response will include a double differentiator. This effect is evident in the initial transient shown in the simulated closed-loop response of Figure 7.9.

Experimental results given in Figure 7.10, for a peak axis velocity of 0.3 radl/s and target frequency of approximately 0.5 Hz, show a peak-to-peak pixel error of only 6 pixels. This corresponds very closely to simulation results for the same amplitude target motion, but some low-amplitude oscillation at around 10 Hz can be observed. In practice the compensator was 'temperamental' and difficult to initiate. For moderate initial error the two integrators wind up and saturate the axis which leads to rather violent oscillation. It was found best to switch to the pole-placement compensator from proportional control once fixation had been achieved. Table 7.1 compares the simulated RMS pixel error value of the model for the standard sinusoidal excitation at various amplitudes. The RMS error scales almost linearly with amplitude in the first three rows, but in the last case actuator saturation has occurred and the pixel error is much higher. High tracking performance is achieved but with a loss of robustness to plant saturation and some difficulty in start up.

θ_{pk} (radl)	$\dot{\theta}_{pk}$ $(radl/s)$	RMS pixel error
0.23	0.97	11.1
0.40	1.70	21.2
0.45	1.89	24.7
0.50	2.1	2100

Table 7.1: Simulated RMS pixel error for the pole-placement controller as a function of target peak velocity. Simulated using a SIMULINK model over the interval 0 to 3s, and including the initial transient. Note the very large error in the last row — the system performs very badly once actuator saturation occurs.

Design of a state estimator and state feedback controller using the linear quadratic Gaussian (LQG) approach has also been investigated, but is complicated by the state transition matrix, $\mathbf{\Phi}$, being singular since the delay states introduce zero eigenvalues[2]. The common LQG design approaches, implemented by MATLAB toolbox functions, are based on eigenvector decomposition and fail for the case of a singular state transition matrix. Instead, the recurrence relationship derived by Kalman [142] from dynamic programming

$$\mathbf{M}_{k+1} = \mathbf{Q} + \mathbf{K}_k' \mathbf{R} \mathbf{K}_k + (\mathbf{\Phi} + \mathbf{\Delta}\mathbf{K}_k)' \mathbf{M}_k (\mathbf{\Phi} + \mathbf{\Delta}\mathbf{K}_k) \quad (7.25)$$

$$\mathbf{K}_{k+1} = -\left(\mathbf{R} + \mathbf{\Delta}'\mathbf{M}_k\mathbf{\Delta}\right)^{-1} \mathbf{\Delta}'\mathbf{M}_k\mathbf{\Phi} \quad (7.26)$$

was used to compute the steady-state value of the state feedback gain matrix given the state and input weighting matrices $\mathbf{Q}$ and $\mathbf{R}$ respectively. This method is numerically less sound than other approaches but gives satisfactory results for the low order system in this case.

As in the previous case stable estimator and closed-loop poles do not necessarily result in a stable compensator. The design procedure requires adjustment of the elements of two weighting matrices for each of the controller and estimator, resulting in a high-dimensionality design space. Stable plants and estimators could be readily achieved for a Type 1 system, but not for the desired case with an additional integrator (Type 2).

7.3.5 Summary

The image plane errors for each compensator, to the same sinusoidally moving target, are compared in Figure 7.11. The simulations are based on detailed multi-rate nonlinear models of the position and velocity loops of the Unimate Puma servo system.

[2]The determinant of a matrix is the product of its eigenvalues, $det(\mathbf{A}) = \prod_i e_i$, so any zero eigenvalue results in a singular matrix.

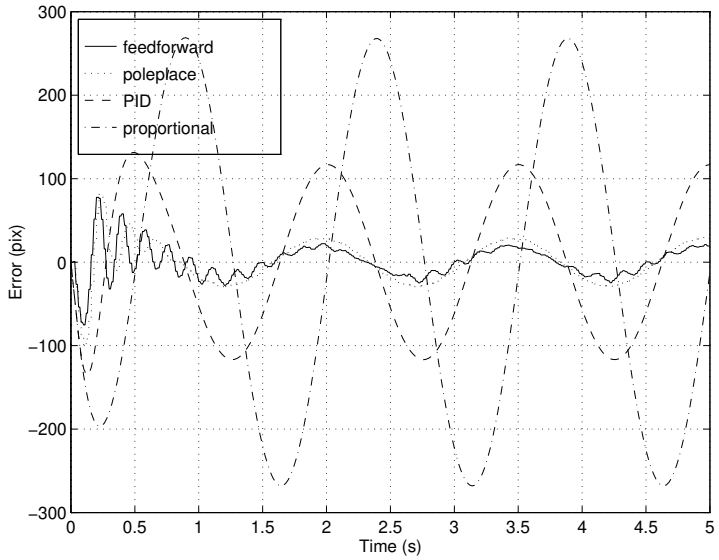


Figure 7.11: Comparison of image plane error for various visual feedback compensators from above. Target motion in all cases is the sinusoid $0.5\sin(4.2t)$ radm/s.

The simple proportional feedback controller, while demonstrating an adequate step response, has poor tracking performance and this may be improved by means of a more sophisticated compensator.

The compensator design approach has followed classical linear principles of increasing system Type and positioning the closed-loop poles by means of PID or pole-placement control. The system's Type is important to the steady-state tracking error and at least Type 2 is necessary to adequately track sinusoidal or parabolic motion. However in the short term, tracking performance is dependent on the closed-loop pole locations. This is illustrated quite clearly in Figure 7.12 which shows the Type 2 system from above following a target moving with triangular position profile. Ideal tracking behaviour is observed during the regions of constant velocity demand, but the settling time after each peak is around 300 ms, the time constant of the dominant closed-loop poles. These closed-loop poles cannot be made arbitrarily fast and are constrained by the practical requirement for compensator stability. Linear discrete-time simulations indicate no difficulty with an unstable compensator, but non-linear simulations show that the closed-loop system will be unstable. This is believed to be due to non-linearities in the plant which change the apparent dynamics at saturation.

The addition of integrators, while raising the system Type, places stringent con-

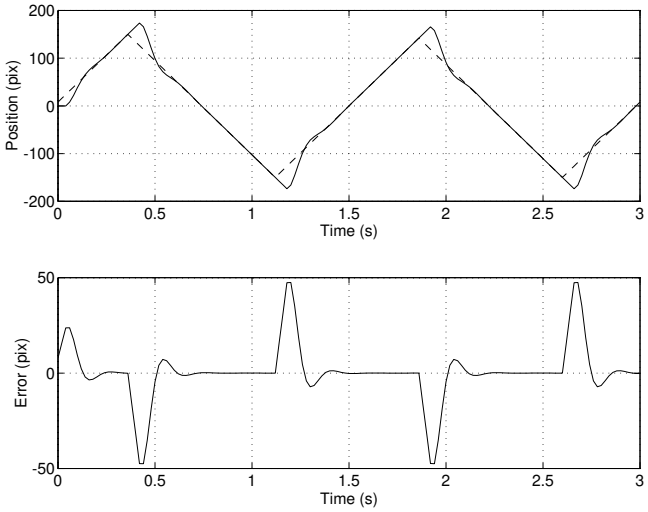


Figure 7.12: Simulation of pole-placement fixation controller (7.23) for triangular target motion. Closed-loop poles at $z = 0.6 \pm 0.4j, 0.4$.

straints on closed-loop and estimator pole locations.

Close agreement has been shown between experimental and simulation results using both linear single-rate and non-linear multi-rate models. The controller designs have been based on the single-rate model (6.44) and the success of the compensators is an implicit endorsement of that model's accuracy.

7.4 Axis control modes for visual servoing

All the experiments so far described, and those of many others, for example [8,91,134, 197,231,270], are based on an approach in which the visual servo is built 'on top of' an underlying position-controlled manipulator. This is probably for the pragmatic reason that most robot controllers present the abstraction of a robot as a position-controlled device. There are some reports of the use of axis velocity control [58, 63, 111, 206], and Weiss [273] proposed visual control of actuator torque.

This section will compare the performance of visual servo systems based around position, velocity and torque-controlled actuators. Motion of a robot manipulator is caused by actuators exerting torques at the joints. In conventional robot applications where end-point position control is required, it is common to establish an abstraction

whereby the axes are capable of following the joint angle demands of a trajectory generator. Such an abstraction is achieved by closing a position-control loop around the actuator.

Many robots employ a 'classic' nested control structure which is typical of machine tool controllers, commonly comprising current, velocity and position-control loops. The nested loop, or hierarchical, control structure has a number of advantages including:

1. The effective dynamics presented by the closed-loop system to the outer loop are simplified and ideally independent of parameter uncertainty.

2. The dynamics presented are generally first or second order, allowing the design to be carried out in a series of relatively simple steps using classical control techniques. This issue is not as important as it once was, given the power of modern control design tools and available computational hardware for implementing modern control algorithms.

The Puma's velocity loop provides a good example of reducing parameter uncertainty. This loop is closed around the actuator dynamics for which the pole and DC gain are functions of inertia and friction. The high-gain velocity loop minimizes the effect of variation in these parameters on the closed-loop dynamics. An alternative approach, now feasible due to low-cost high-performance computing, is to feed forward torque to compensate for changing inertia and friction. These estimates may be derived from a fixed parameter model identified experimentally or from online identification and adaptation.

The next sections will examine the dynamics of a visual servo system where the actuator is treated at various levels of abstraction: as a torque, velocity or position source. The notation is shown in Figure 7.13, and it will be assumed that the multirate effects and communications latencies of the earlier system have been eliminated by optimization of the control architecture. The dynamics will all be formulated at the visual sampling interval, $T = 20$ ms. The models of actuator dynamics used are those established earlier in Chapter 2, and the vision system is modelled as a single time step delay $V(z) = K_{lens}/z$.

7.4.1 Torque control

The action of the power amplifier's current loop allows motor current (proportional to motor torque) to be commanded directly. From (2.65) the transfer function of the actuator is

$$R(s) = \frac{\Theta_m(s)}{U(s)} = \frac{1}{s}\frac{K_i K_m}{J_m s + B_m} \tag{7.27}$$

where θ_m is the shaft angle, $u(t)$ the motor current demand v_{i_d}, J_m the inertia, B_m the viscous friction, K_i the amplifier transconductance and K_m the motor torque con-

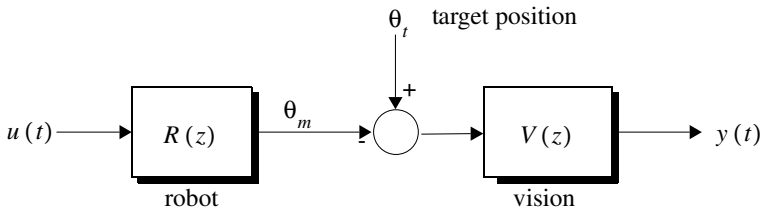


Figure 7.13: Block diagram of generalized actuator and vision system.

stant. The discrete-time transfer function is obtained by introducing a zero-order hold network and taking the Z-transform

$$\frac{\Theta_m(z)}{U(z)} = \mathrm{Z}\left\{\frac{1-e^{-sT}}{s}\frac{1}{s}\frac{K_i K_m}{Js+B}\right\} \tag{7.28}$$

Using the model of (2.64) the discrete-time dynamics are

$$\frac{\Theta_m(z)}{U(z)} = 0.126\frac{(z+0.81)}{(z-1)(z-0.534)} \tag{7.29}$$

where the zero is due to the zero-order hold and sampler. Adding the vision system delay results in the open-loop transfer function

$$\frac{{}^iX(z)}{U(z)} = 0.126K_{lens}\frac{(z+0.81)}{z(z-1)(z-0.534)} \tag{7.30}$$

and a root-locus plot is shown in Figure 7.14. Without any compensation the closed-loop poles are limited to having a very low natural frequency.

7.4.2 Velocity control

The transfer function of the Unimate analog velocity loop is

$$R(s) = \frac{\Theta_m(s)}{U(s)} = \frac{1}{s}\frac{K_v}{s+p_v} \tag{7.31}$$

where $u(t)$ is the velocity demand input ω_d and K_v the gain of the velocity loop which has a pole at $s = -p_v$. Using the model (2.78) and a similar procedure to above, the open-loop transfer function is

$$\frac{{}^iX(z)}{U(z)} = 0.162K_{lens}\frac{(z+0.39)}{z(z-1)(z-0.051)} \tag{7.32}$$

The velocity-loop pole is very fast with respect to the sampling interval and is thus very close to the origin on the Z-plane. Figure 7.15 shows a root-locus plot for this system.

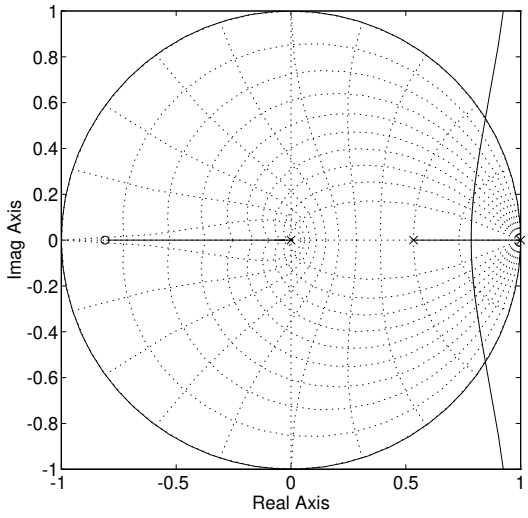


Figure 7.14: Root locus for visual servo with actuator torque controller.

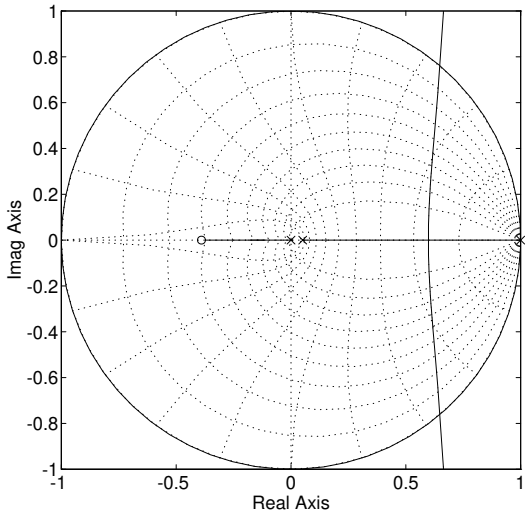


Figure 7.15: Root locus for visual servo with actuator velocity controller.

7.4.3 Position control

As explained in Section 2.3.6 the position loop contains an interpolator which moves the joint to the desired value over the setpoint interval of the trajectory generator, T_{tg}, provided velocity and acceleration limits are not exceeded. Generally the position-loop sample interval is an order of magnitude smaller than T_{tg} providing accurate position control and good disturbance rejection. In a discrete-time system with the same sampling interval, that is $T = T_{tg}$, the position controller acts as a unit delay, and the discrete-time dynamics are simply

$$\frac{\Theta_m(z)}{U(z)} = \frac{1}{z} \tag{7.33}$$

where $u(t)$ is the position demand input to the position-control loop. After introducing the vision system delay, the open-loop transfer function is

$$\frac{{}^{i}X(z)}{U(z)} = \frac{K_{lens}}{z^2} \tag{7.34}$$

and Figure 7.16 shows a root-locus plot for this system.

7.4.4 Discussion

All control modes result in a broadly similar discrete-time transfer function. In particular:

1. All root-loci have a pole-zero excess of two, thus the systems will become unstable as gain is increased.
2. All have a transmission delay, $\text{Ord}(\text{denom}) - \text{Ord}(\text{numer})$, of 2 sample times.
3. The previous model (6.44) had 3 poles and a 3 sample time delay due to the double handling of setpoint data and the multi-rate control.
4. Both torque- and velocity-control modes have one open-loop pole at $z = 1$ giving Type 1 characteristics, that is, zero steady-state error to a step input. The position-control mode results in a Type 0 system.

The robot's time response to a unit step target motion is shown in Figure 7.18 for a simple proportional controller with the gain chosen to achieve a damping factor of 0.7. The controller based on a position-mode axis has a large steady-state error since it is only of Type 0. The velocity-mode axis control results in a significantly faster step response than that for torque-mode.

If an integrator is added to the position-mode controller then the response becomes similar to, but slightly faster than, the velocity mode controller as shown in Figure

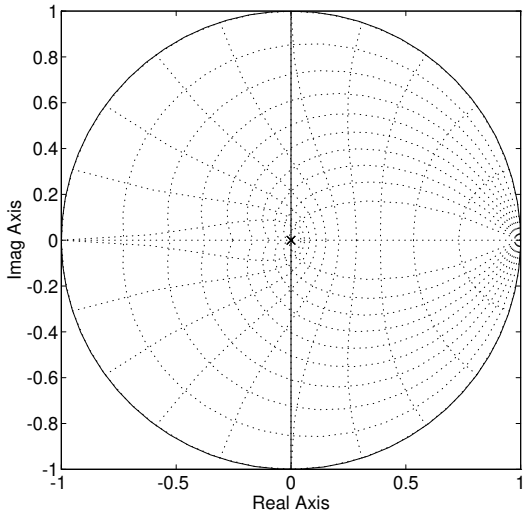


Figure 7.16: Root locus for visual servo with actuator position controller.

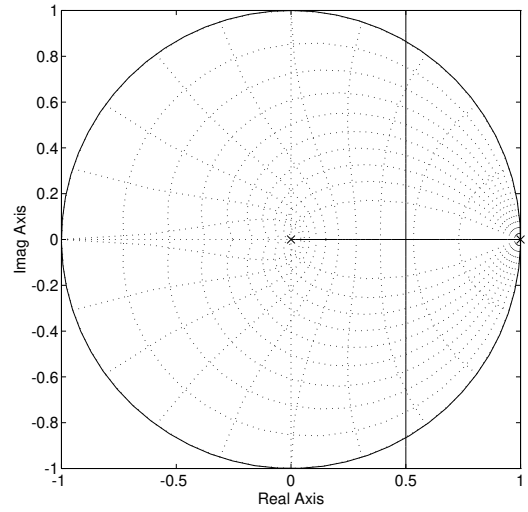


Figure 7.17: Root locus plot for visual servo with actuator position controller plus additional integrator.

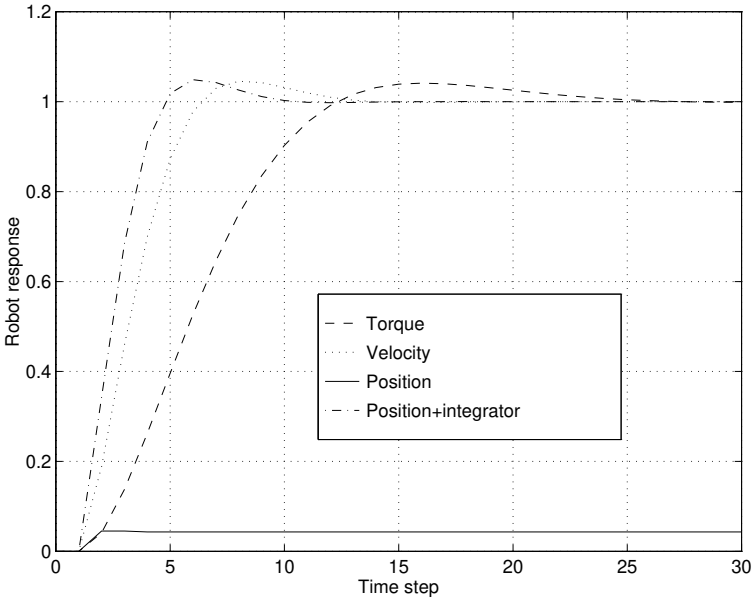


Figure 7.18: Simulation of time responses to target step motion for visual servo systems based on torque, velocity and position-controlled axes. Proportional control with gain selected to achieve a damping factor of 0.7.

7.18. The root locus of this system, Figure 7.17, is also similar to that for velocity mode control. The integrator has turned the position-mode controller into an 'ideal' velocity source, just as used in the experimental system. From a control systems perspective however there are some subtle differences between a velocity-controlled axis and a position-controlled axis plus integrator. In particular the inner position loop will generally be very 'tight', that is high-gain, in order to minimize position error. For an ideal robot this is not a problem, but it will exacerbate resonance problems due to structural and/or drive compliance. It has been shown [213] that for the case of significant structural compliance it is preferable to use a 'soft' velocity loop in order to increase system damping.

7.4.5 Non-linear simulation and model error

The above analysis is based on simple linear models which can mask many 'real-world' effects such as non-linearity and parameter variation. To investigate this, for

each system a pole-placement compensator [24] was designed to place the closed-loop poles at $z = 0.6 \pm 0.4j, 0.4, 0$, and raise the system's Type to 2. Performance was then simulated using a detailed non-linear multi-rate simulation model of the axis dynamics including saturation, Coulombic and static friction.

For axis position and velocity control modes the results were similar to those for the linear simulation case. However the torque-mode controller did not behave as expected — the discrete-time non-linear simulation, using motor model LMOTOR, resulted in very lightly damped oscillatory responses. This was found to be due to a combination of non-linear friction, integral action and the relatively low sample rate. The unmodeled dynamics, particularly stick/slip friction, are complex and have time constants that are shorter than the visual sample interval. It is a 'rule of thumb' that the sample rate should be 4 to 20 times the natural frequency of the modes to be controlled. Armstrong [22] comments on oscillation at low velocity due to the interaction of stick/slip friction with integral control. Another cause of poor performance is that the compensator is linear time-invariant (LTI) based on the assumption of an LTI plant. However linearization of the friction model, (2.68), shows that pole location has a strong dependence on shaft speed, as does plant gain. Improved torque-mode control would require one or more of the following: a higher sample rate, friction measurement and feedforward, adaptive, or robust control.

A more straightforward option is the use of axis velocity control. Velocity feedback is effective in linearizing the axis dynamics and minimizing the effect of parameter variation. Stable compensators were designed and simulated for velocity mode control with Type 2 behaviour. Table 7.2 shows the RMS pixel error for the same simulation run with variations in motor parameter and velocity-loop gain settings. The RMS pixel error is around half that of pole-placement controller, (7.23), based on a multi-rate position-control loop for the same motion amplitude. Interestingly, the system was robust to significant variation in dynamic parameters and became unstable only when the velocity-loop gain was reduced, weakening its ability to mask parameter variation.

7.4.6 Summary

The conclusions from this section can be summarized as follows:

1. Control of actuator torque using only the vision sensor leads to poor performance due to the low sample rate and non-linear axis dynamics. Compensators designed to increase performance and raise system Type are not robust with respect to the aforementioned effects.

2. Closing a velocity loop around the actuator linearizes the axis and considerably reduces the uncertainty in axis parameters.

3. Velocity and position mode control can be used to achieve satisfactory tracking.

Parameter	Value	Change	RMS pixel error
J	41.3×10^{-6}	+25%	6.03
J	24.8×10^{-6}	-25%	5.44
K_τ	66.8×10^{-3}	+25%	5.42
K_τ	40.1×10^{-3}	-25%	6.15
K_{vel}	0.165	-25%	unstable
K_{vel}	0.275	+25%	6.58

Table 7.2: Effect of parameter variation on velocity mode control. Simulations are over 3s with target motion of $\theta_t(t) = 0.2\sin(4.2t)$ radl, and the LMOTOR motor model for joint 6. Compensator designed to place the closed-loop poles at $z = 0.6 \pm 0.4j, 0.4, 0$.

4. Position-mode control requires additional complexity at the axis control level, for no significant performance improvement under visual servoing.

5. Particular implementations of axis position controllers may introduce additional sample rates which will degrade the overall closed-loop performance.

The approaches reported in the literature are interesting when considered in the light of these observations. Most reports have closed the visual position loop around an axis position loop which generally operates at a high sample rate (sample intervals of the order of a few milliseconds). There are fewer reports on the use of a visual position loop around an axis velocity loop. Pool [206] used a hydraulic actuator which is a 'natural' velocity source, albeit with some severe stiction effects. Hashimoto [111] implemented a digital velocity loop on the axes of a Puma robot as did Corke [58, 63]. Weiss [273] proposed visual control of actuator torque, but even with ideal actuator models found that sample intervals as low as 3 ms were required.

Earlier it was suggested that visual fixation should be viewed as a steering problem, continuously changing the robot velocity so as to move toward the goal. Combined with the considerations above this suggests that axis velocity control is appropriate to use in a visual servo system.

7.5 Visual feedforward control

The previous sections have discussed the use of visual feedback control and explored the limits to tracking performance. Feedback control involves manipulation of the closed-loop poles and it has been shown that there are strong constraints on the placement of those poles. Another degree of controller design freedom is afforded by manipulating the zeros of the closed-loop system, and this is achieved by the introduction

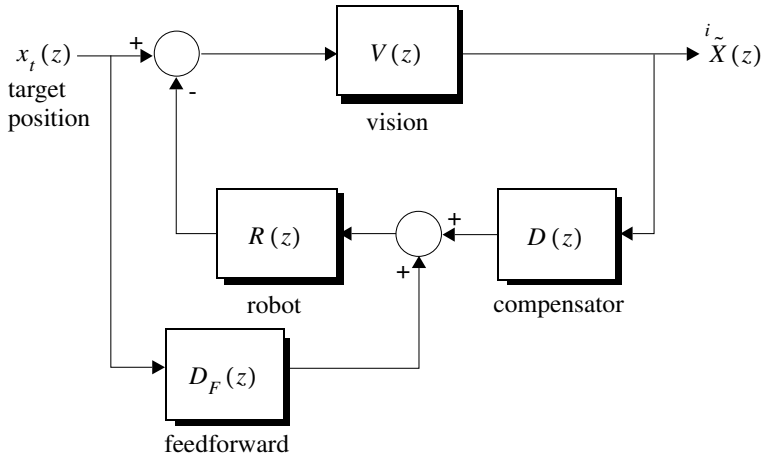


Figure 7.19: Block diagram of visual servo with feedback and feedforward compensation.

of a feedforward term, $D_F(z)$, as shown in Figure 7.19. The closed-loop transfer function becomes

$$\frac{{}^{i}X}{x_t} = \frac{V(z)\,(1 - R(z)D_F(z))}{1 + V(z)R(z)D(z)} \tag{7.35}$$

which has the same denominator as (7.4) but an extra factor in the numerator. Clearly if $D_F(z) \equiv R^{-1}(z)$ the tracking error would be zero, but such a control strategy is not realizable since it requires (possibly future) knowledge of the target position which is not directly measurable. However this information may be estimated as shown in Figure 7.20.

Interestingly the term *visual servoing* has largely replaced the older term *visual feedback*, even though almost all reported visual servo systems are based on feedback control. The remainder of this section introduces a 2-DOF visual fixation controller using camera pan and tilt motion. Following the conclusions above, axis velocity rather than position will be controlled. The velocity demand will comprise estimated target velocity feedforward and image plane error feedback. A block diagram of the controller, using the notation developed in this chapter, is shown in Figure 7.21.

The camera mounting arrangement described in Section 6.3.1 allows camera pan and tilt to be independently actuated by wrist joints 6 and 5 respectively. The 2-DOF fixation problem is thus considered as two independent 1-DOF fixation controllers. For simplicity the following discussion will be for the 1-DOF case but is readily applied to the 2-DOF fixation controller.

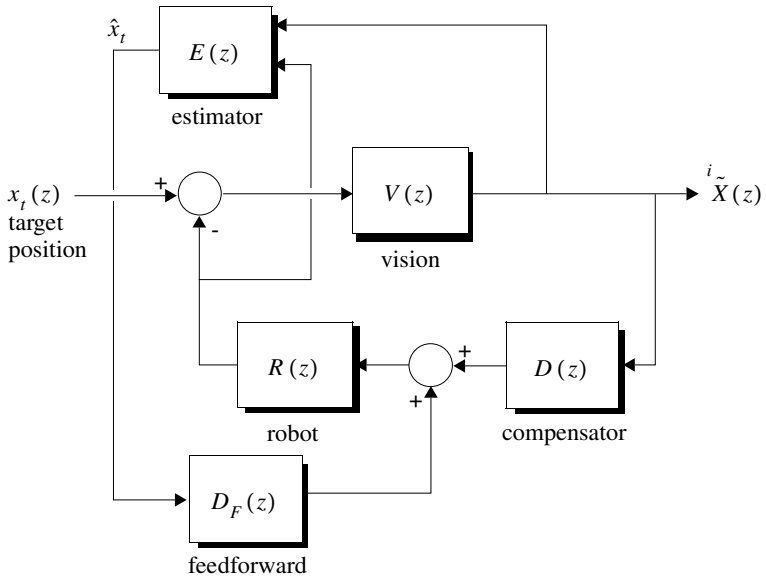


Figure 7.20: Block diagram of visual servo with feedback and estimated feedforward compensation.

7.5.1 High-performance axis velocity control

The experiments described so far have used the artifice of an integrator to implement a velocity-controlled axis. This section investigates two approaches to direct axis velocity control. Such control eliminates two levels of complexity present in the previous system: the integrator and the position loop.

7.5.1.1 Direct access to Unimate velocity loop

Modifications were made to the firmware on the UNIMATE digital servo card to allow the host to directly command the analog velocity loop described in Section 2.3.5. As shown in Figure 7.22 this is a simple variation of the position-control mode, but with the velocity demand being set by the host, not the setpoint interpolator and position loop. From the host computer's perspective this was implemented as a new setpoint command mode, in addition to the existing position and current modes. A previously unused command code was used for this purpose.

This modification allows 'immediate' setting of joint velocity at the video sample rate. However the inherent velocity limit of the velocity loop (see Section 2.3.6) is still a limiting factor and the analog loops were found to have considerable velocity

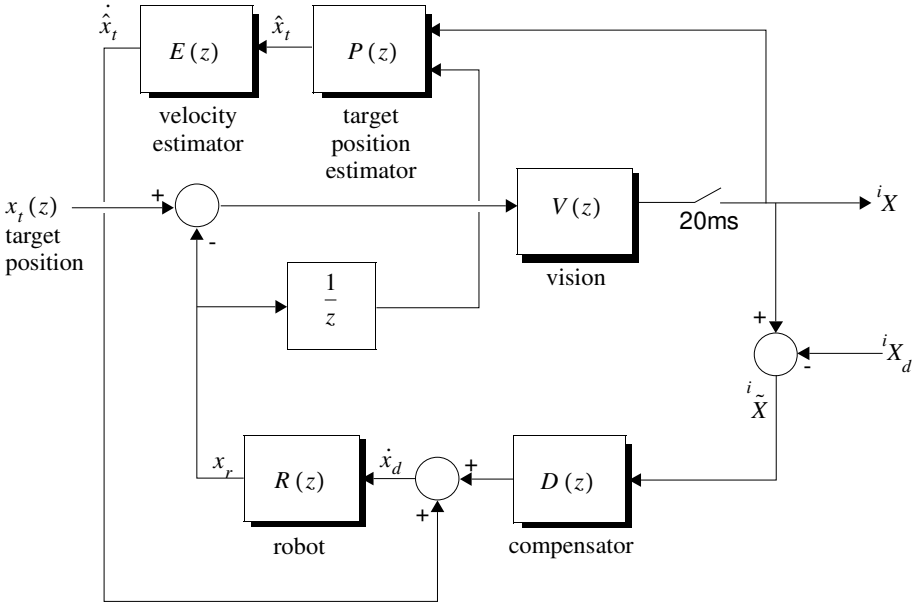


Figure 7.21: Block diagram of velocity feedforward and feedback control structure as implemented. $P(z)$ estimates target bearing from (7.43).

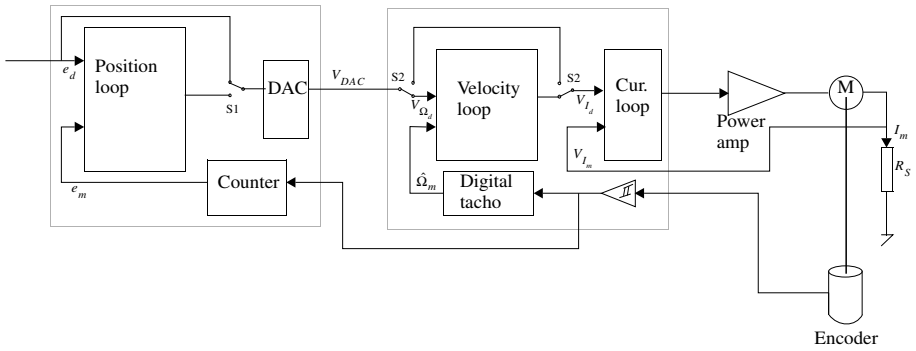


Figure 7.22: Block diagram of Unimation servo in velocity-control mode. Compared with Figure 2.22 the digital position loop has been bypassed by means of switch S1.

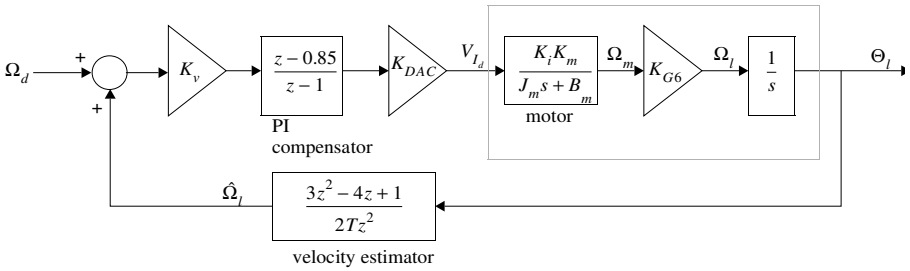


Figure 7.23: Block diagram of digital axis-velocity loop for joint 6. The sample rate is 50 Hz.

offset.

7.5.1.2 Digital axis velocity loop

To overcome the deficiencies in the Unimate analog velocity loop a digital velocity loop has been implemented which commands motor current directly, and operates at the 50 Hz visual sample rate. The overall structure, for joint 6, is shown in Figure 7.23.

A key part of the control structure is the estimation of axis velocity from motor shaft angle measurements. There is considerable literature on this topic, particularly for the case where a discrete position sensor, such as a shaft angle encoder, is used [30, 38, 155]. The two major approaches are:

1. Count the number of encoder pulses occurring in a fixed time interval, typically the sample interval.

2. Measure the elapsed time required for a given number of encoder pulses.

The first approach is easily implemented since most axis controllers provide an encoder counter, for axis position measurement, which can be read periodically. The discrete position sensor results in velocity quantization of

$$\Delta\omega = \frac{\Delta\theta}{T} \tag{7.36}$$

where $\Delta\theta$ is the position quantization level. Velocity resolution can be improved by a higher resolution position sensor, or a longer sampling interval.

Brown et al. [38] compare a number of approaches to velocity estimation. They conclude that at high speed, more than 100 counts per sampling interval, estimators based on backward difference expansions give the best result. At low speed, techniques based on least squares fit are appropriate. The motor shaft velocities corresponding to the 100 count limit are computed and compared with maximum motor

Joint	Counts/rev	Critical vel. (radm/s)	Percent of maximum
1	1000	5.0	4%
2	800	6.3	3.8%
3	1000	5.0	3.9%
4	1000	5.0	1.2%
5	1000	5.0	1.4%
6	500	10.0	2.3%

Table 7.3: Critical velocities for Puma 560 velocity estimators expressed as angular rate and fraction of maximum axis rate. These correspond to 100 encoder pulses per sample interval (in this case 20 ms). Maximum joint velocities are taken from Table 2.21.

speed for the Puma 560 in Table 7.3 — it is clear that the backward difference estimator is appropriate for the majority of the axis velocity range. Belanger [30] compares fixed-gain Kalman filters with first-order finite difference expressions and concludes that the Kalman filter is advantageous at short sample intervals. The Kalman filter is an optimal least squares estimator, and this result agrees with Brown et al. In this application however a Kalman filter will be sub-optimal, since the axis acceleration is not a zero-mean Gaussian signal. A velocity estimator could make use of additional information such as the axis dynamic model and control input. Such estimators, combined with controllers, have been described by Erlic [86] and DeWit [44].

Surprisingly many papers that discuss experimental implementation of model-based control make no mention of the techniques used to estimate motor velocity, yet from (2.84) and (2.86) it is clear that knowledge of motor velocity is required. In this work a second-order, or 3-point, velocity estimator

$$\hat{\dot{\theta}}_i = \frac{3\theta_i - 4\theta_{i-1} + \theta_{i-2}}{2T} \tag{7.37}$$

is used which can be derived from the derivative of a parabola that fits the current and two previous position points. The discrete-time transfer function of this differentiator is

$$\frac{\hat{\Omega}(z)}{\Theta(z)} = \frac{3}{2T}\frac{(z-\frac{1}{3})(z-1)}{z^2} \tag{7.38}$$

and its frequency response is shown in Figure 7.24. Up to 10 Hz the response closely approximates that of an ideal differentiator.

Motor current, set via the Unimate arm interface, is proportional to velocity error with a PI compensator to give a closed-loop DC gain of unity. A detailed SIMULINK model, Figure 7.25, was developed and used to determine suitable values for K_v and

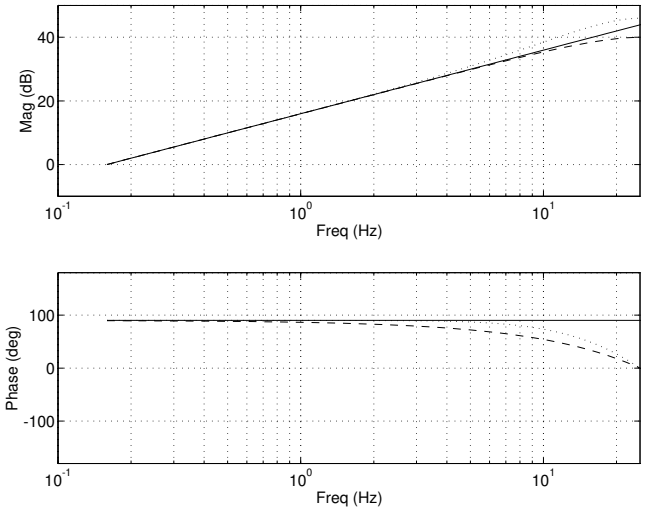


Figure 7.24: Bode plot comparison of differentiators, $\hat{\Omega}(z)/\Theta(z)$, up to the Nyquist frequency of 25 Hz. Ideal differentiator (solid), 1st order difference (dashed), and 2nd order difference (dotted).

the zero in the lag/lead compensator. Experimental step response results for the joint 6 axis are shown in Figure 7.28. This controller is clearly capable of driving the axis faster than the native velocity loop limit of 2.4 radl/s given in Table 2.20.

Determining the linear discrete-time dynamics Ω/Ω_d is difficult since from Figure 7.23 the motor and free integrator must be Z-transformed together when discretizing the system. Instead, the transfer function $\hat{\Omega}/\Omega_d$ will be determined. From (7.29) the discrete-time transfer function of the joint 6 motor and free integrator is

$$\frac{\Theta(z)}{V_{I_d}(z)} = \mathrm{Z}\left\{\frac{1-e^{sT}}{s}\frac{1}{s}\frac{24.4}{s/31.4+1}\right\} \tag{7.39}$$

$$= 0.126\frac{(z+0.81)}{(z-1)(z-0.534)} \tag{7.40}$$

which combined with the velocity estimator (7.38) and PI compensator gives the open-loop transfer function

$$K_v\frac{z-0.85}{z-1}K_{DAC}\frac{\Theta(z)}{V_{I_d}(z)}\frac{\hat{\Omega}(z)}{\Theta(z)} = K\frac{(z-\frac{1}{3})(z-0.85)(z+0.81)}{z^2(z-0.534)(z-1)} \tag{7.41}$$

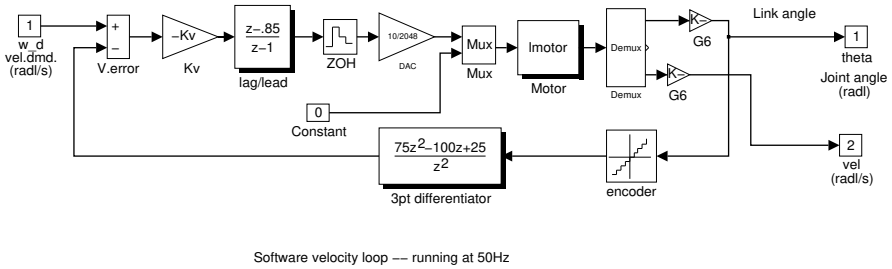


Figure 7.25: SIMULINK model DIGVLOOP: digital axis-velocity loop for joint 6. The sample rate is 50 Hz.

where $K = 6.02 \times 10^{-4} K_v$. The root-locus plot of Figure 7.26 shows the closed-loop poles for the chosen controller parameters. The dominant poles are $z = 0.384$, $-0.121 \pm 0.551\,j$ corresponding to a real pole at 7.6 Hz and a lightly damped pole pair at 15 Hz with damping factor of 0.3. The measured step response shows evidence of this latter mode in the case of the large step demand. The closed-loop bandwidth of nearly 8 Hz is poor when compared with the native analog velocity loop bandwidth of 25 Hz, shown in Figure 2.20, but it is as high as could reasonably be expected given the low sampling rate used. The Bode plot shown in Figure 7.27 indicates that for the standard target motion the phase lag is approximately 18°.

7.5.2 Target state estimation

The end-effector mounted camera senses target position error directly

$$^{i}X = K_{lens}(x_t - x_r) \tag{7.42}$$

which may be rearranged to give a target position estimate

$$\hat{x}_t = \frac{^{i}X}{K_{lens}} + x_r \tag{7.43}$$

in terms of two measurable quantities: ^{i}X the output of the vision system, and x_r the camera angle which can be obtained from the servo system. Both data are required due to the inherent ambiguity with an end-effector mounted camera — apparent motion may be due to target and/or end-effector motion.

From earlier investigations it is known that the vision system introduces a unit delay so the robot position, x_r, should be similarly delayed so as to align the measurements — this is modelled as a $1/z$ block in Figure 7.21. This leads to a delayed target

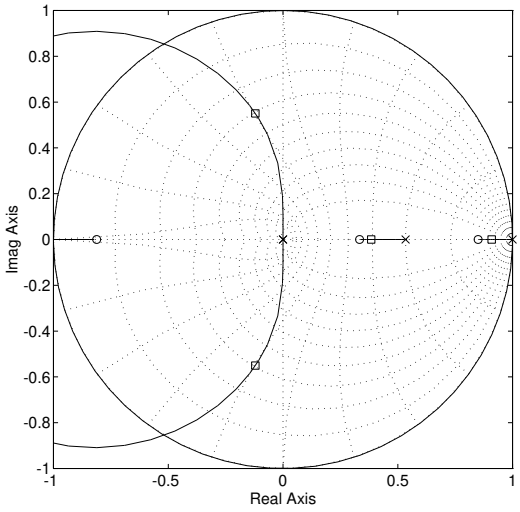
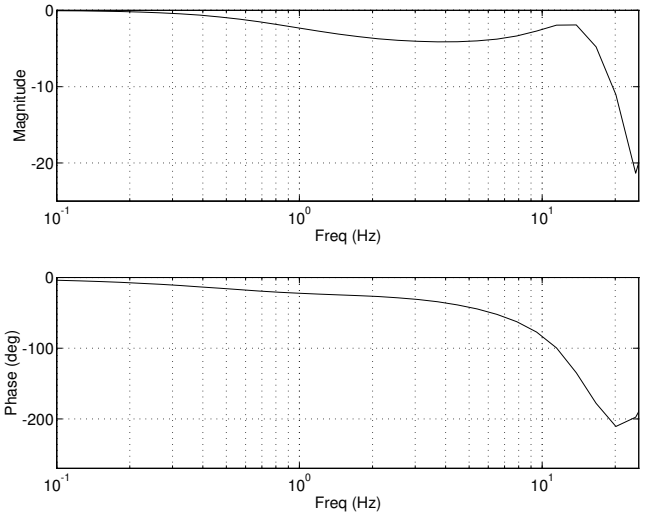


Figure 7.26: Root-locus of closed-loop digital axis-velocity loop. Marked points correspond to closed-loop poles for $K_v = 800$.



.

Figure 7.27: Bode plot of closed-loop digital axis-velocity loop, $\hat{\Omega}(z)/\Omega(z)$.

.

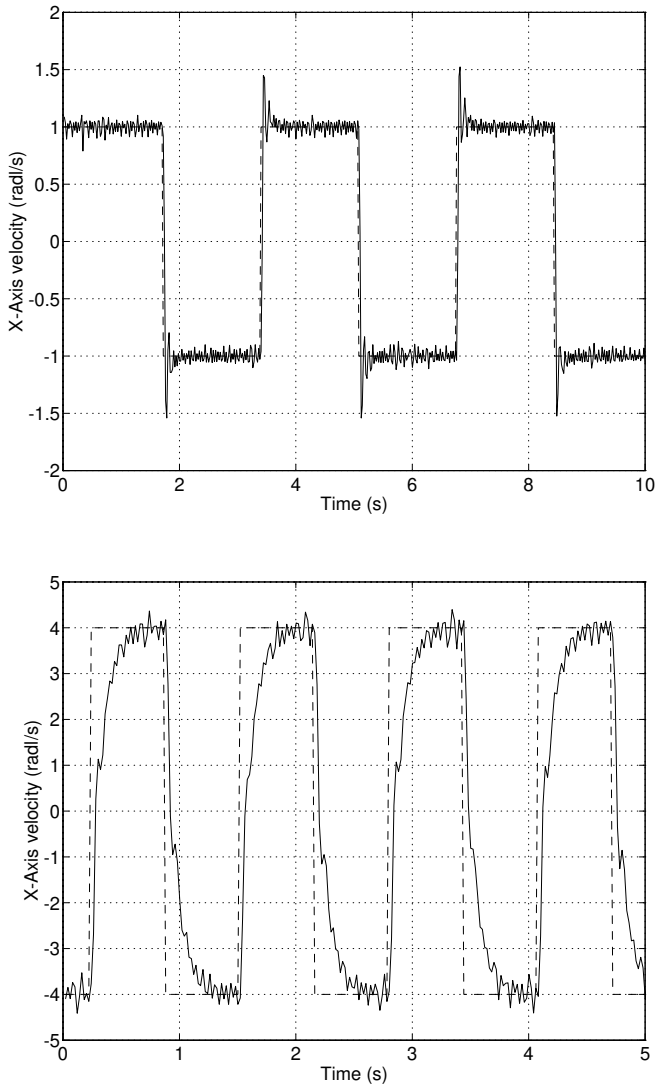


Figure 7.28: Measured response of joint 6 digital axis velocity loop (solid). Top plot is for ± 1 radl/s velocity step demand (dashed), and the bottom plot ± 4 radl/s velocity step demand (dashed). This data was logged by RTVL and the velocity is as computed online using (7.37).

position estimate requiring prediction in order to estimate the current target position and velocity. The estimates must be based on some assumption of target motion, and in this section constant velocity motion will be assumed for estimator design. However tracking performance will be evaluated for the standard sinusoidal test motion (7.10) which seriously challenges that design assumption.

The problem of estimating target velocity from discrete target position measurements is similar to that just discussed in Section 7.5.1.2 for determining the axis velocity from noisy measurements of axis position. However the signal to noise ratio for the vision sensor is significantly poorer than for the encoder. Target position estimates (7.43) will be contaminated by spatial quantization noise in ${}^{i}X_t$. If video fields are used directly there will also be a frame rate pixel jitter due to field pixel offsets, see Section 3.4.1. The remaining sections will examine a number of different approaches to estimating the target state in 1 dimension. Many target state estimation schemes are possible and have been discussed in the literature — the remainder of this section will give a brief summary of only three types and then compare their performance.

7.5.2.1 Simple differencing

The simplest velocity estimator possible is a first order difference whose transfer function is

$$\frac{\hat{V}(z)}{Y(z)} = \frac{1}{T}\frac{z-1}{z} \tag{7.44}$$

A second order difference, which takes into account additional data points, is

$$\frac{\hat{V}(z)}{Y(z)} = \frac{1}{2T}\frac{3z^2-4z+1}{z^2} \tag{7.45}$$

and was used for axis velocity control in Section 7.5.1.2.

It is possible to compute visual velocity, or optical flow directly, rather than differentiate the position of a feature with respect to time. It is suggested [37] that the human eye's fixation reflex is driven (to first order) by retinal slip or optical flow. However most machine vision algorithms to compute optical flow, such as that by Horn and Schunck [122] are based principally on a first order difference of image intensity between consecutive frames.

7.5.2.2 $\alpha-\beta$ tracking filters

$\alpha-\beta$ tracking filters were developed in the mid 1950s for radar target tracking, estimating target position and velocity from noisy measurements of range and bearing. In their simplest form these are fixed gain filters based on the assumption that the target acceleration (often referred to as *maneuver*) is zero. The $\alpha-\beta$ filter is

$$\hat{x}_{p_{k+1}} = \hat{x}_k + T\hat{v}_k \tag{7.46}$$

$$\hat{x}_{k+1} = \hat{x}_{p_{k+1}} + \alpha\left[y_{k+1} - \hat{x}_{p_{k+1}}\right] \tag{7.47}$$

$$\hat{v}_{k+1} = \hat{v}_k + \frac{\beta}{T}\left[y_{k+1} - \hat{x}_{p_{k+1}}\right] \tag{7.48}$$

where $\hat{x}_k$ and $\hat{v}_k$ are the position and velocity estimates at sample k respectively, and y_k is the measured position. The above equations may be manipulated into transfer function form

$$\frac{\hat{V}(z)}{Y(z)} = \frac{\beta}{T}\frac{z(z-1)}{z^2 + (\alpha+\beta-2)z + 1 - \alpha} \tag{7.49}$$

from which it is clear that this is simply a first order difference smoothed by a second order system whose poles are manipulated by the parameters α and β. Commonly the filter is treated as having only one free parameter, α, with β computed for critical damping

$$\beta_{CD} = 2 - \alpha - 2\sqrt{1-\alpha} \tag{7.50}$$

or minimal transient error performance (Benedict and Bordner criteria [31])

$$\beta_{BB} = \frac{\alpha^2}{2-\alpha} \tag{7.51}$$

Kalata [141] proposes another parameterization, and introduces the *tracking parameter*, Λ, which is the ratio of position uncertainty due to target maneuverability and sensor measurement, and from which α and β can be determined.

The $\alpha - \beta - \gamma$ filter is a higher-order extension that also estimates acceleration, and should improve the tracking performance for a maneuvering target. Various modifications to the basic filter have been proposed such as time varying gains [170] or maneuver detection and gain scheduling which increases the filter bandwidth as the prediction error increases.

7.5.2.3 Kalman filter

This filter, proposed by Kalman in 1960 [142], is an optimal estimator of system state where the input and measurement noise are zero-mean Gaussian signals with known covariance. A second-order model of the target position can be used whose acceleration input is assumed to be zero-mean Gaussian. For a single DOF the filter states are

$$\underline{X}_k = \begin{bmatrix}\theta_k & \dot{\theta}_k\end{bmatrix}^T \tag{7.52}$$

and the discrete-time state-space target dynamics are

$$\underline{X}_{k+1} = \mathbf{\Phi}\underline{X}_k + \mathbf{\Delta}\underline{\omega}_k \tag{7.53}$$

$$y_k = \mathbf{C}\underline{X}_k + e_k \tag{7.54}$$

where $\underline{\omega}_k$ represents acceleration uncertainty and y_k is the observed target position with measurement noise e_k. For constant-velocity motion the state-transition matrix is

$$\mathbf{\Phi} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{7.55}$$

and the observation matrix

$$\mathbf{C} = [1\ 0] \tag{7.56}$$

where T is the sampling interval. The predictive Kalman filter for one axis [24] is given by

$$\mathbf{K}_k = \mathbf{\Phi}\mathbf{P}_k\mathbf{C}^T(\mathbf{C}\mathbf{P}_k\mathbf{C}^T + \mathbf{R}_2)^{-1} \tag{7.57}$$

$$\underline{\hat{X}}_{k+1} = \mathbf{\Phi}\underline{\hat{X}}_k + \mathbf{K}_k(y_k - \mathbf{C}\underline{\hat{X}}_k) \tag{7.58}$$

$$\mathbf{P}_{k+1} = \mathbf{\Phi}\mathbf{P}_k\mathbf{\Phi}^T + \mathbf{R}_1\mathbf{I}_2 - \mathbf{K}_k\mathbf{C}\mathbf{P}_k\mathbf{\Phi}^T \tag{7.59}$$

where $\mathbf{K}$ is a gain, $\mathbf{P}$ is the error covariance matrix and $\mathbf{I}_2$ is a 2×2 identity matrix. $\mathbf{R}_1 = E\{\underline{\omega}_k\underline{\omega}_k^T\}$ and $\mathbf{R}_2 = E\{e_k^2\}$ are respectively the process and measurement covariance estimates and govern the dynamics of the filter. In this work appropriate values for $\mathbf{R}_1$ and $\mathbf{R}_2$ have been determined through simulation. This filter is predictive, that is $\underline{\hat{X}}_{k+1}$ is the estimated value for the next sample interval based on measurements at time k and assumed target dynamics.

There seems to be considerable mystique associated with the Kalman filter but this second order filter can be expressed simply in transfer function form as

$$\frac{\hat{V}(z)}{Y(z)} = \frac{k_2 z(z-1)}{z^2 + (k_1 - 2)z + (k_2 - k_1 + 1)} \tag{7.60}$$

where k_1 and k_2 are elements of the Kalman gain matrix $\mathbf{K}$. Like the $\alpha - \beta$ filter this is simply a first order difference with a second-order 'smoothing' function. The significant difference with the Kalman filter is that the gains k_1 and k_2 diminish with time according to the equations (7.57) and (7.59). This provides initial fast response then good smoothing once the target dynamics have been estimated. However the gain schedule is completely 'preordained'. That is, the gain trajectory is not dependent upon measured target dynamics or estimation error but is entirely specified by the filter parameters $\mathbf{P}_0$, $\mathbf{R}_1$ and $\mathbf{R}_2$. Berg [32] proposes a modified Kalman filter in which online estimates of covariance are used to set the filter gain. In practice target acceleration does not match the zero-mean Gaussian assumption and is generally correlated from one sample point to the next. Singer et al. [235] show how the dynamic model may be augmented with an extra state to include acceleration and 'whiten' the input to the model. A Wiener filter [235] is a fixed-gain filter whose gain vector is the steady-state gain vector of the Kalman filter.

The Kalman filter equations are relatively complex and time consuming to execute in matrix form. Using the computer algebra package MAPLE the equations were reduced to scalar form, expressed in 'C' as shown in Figure 7.29.

```
/* compute the filter gain */
den = p11 + R2;
k1 = (p11 + T * p12) / den;
k2 = p12 / den;

/* update the state vector */
x1_new = x1 + T * x2 + k1 * (y - x1);
x2_new = x2 + k2 * (y - x1);
x1 = x1_new;
x2 = x2_new;

/* update the covar matrix (symmetric so keep only 3 elements) */
p11_new = R1 + p11 + 2.0 * T * p12 + T * T * p22 -
      k1 * p11 - k1 * p12 * T;
p12_new = p12 + T * p22 - k1 * p12;
p22_new = R1 + p22 - k2 * p12;
p11 = p11_new;
p12 = p12_new;
p22 = p22_new;
```

Figure 7.29: Simplified scalar form of the Kalman filter in 'C' as generated by MAPLE.

7.5.2.4 Estimator initiation

Both the $\alpha-\beta$ and Kalman filter need to be initiated when target tracking commences. For the Kalman filter this involves initializing the error covariance, **P**, generally to a large diagonal value. Both filters require state initialization. Commonly the position is set to the currently observed position, and the initial velocity computed from the current and previous position.

7.5.2.5 Comparison of state estimators

Velocity estimators by definition perform differentiation, which is a noise enhancing operation. To reduce the resultant noise a low-pass filter can be introduced, and it is shown above that the Kalman and $\alpha-\beta$ filter both have this form. The dynamics of the filter provide a tradeoff between estimation noise and response to target acceleration. The $\alpha-\beta$ and Wiener filter are both fixed-gain filters which can be 'tuned' to meet the needs of the application. For the situation, as in this experiment, where the robot is fixated on a target for a long period of time the Kalman filter will converge to fixed gains, and is thus a computationally more expensive implementation of a Wiener or $\alpha-\beta$ filter. Kalata [141] in fact shows how α and β can be related to the steady-state Kalman gains. Singer comments that the Kalman filter is most appropriate when the

"transient period approaches that of the tracking interval".

The Kalman filter could perhaps be viewed as an over-parameterized tracking filter where two covariance matrices must be selected compared to a single parameter for the $\alpha-\beta$ filter. The Kalman filter is an optimal estimator but in this application assumptions about the plant input, zero-mean Gaussian, are violated. Compared to the $\alpha-\beta$ filter it is computationally more expensive in steady-state operation and more difficult to tune.

A number of reports have investigated the target tracking problem and compared different approaches. Hunt [127] is concerned primarily with position extrapolation to cover delay in a vision/robot system and examines 5 different extrapolators spanning linear regression through to augmented Kalman filters. Since none is ideal for all motion she proposes to evaluate all extrapolators and select the one with minimum prediction error in the last time step. Singer and Behnke [235] evaluate a number of tracking filters but in the context of aircraft tracking where target acceleration is minimal. Safadi [218] investigates $\alpha-\beta$ filters designed using the tracking index method for a robot/vision tracking application. He compares tracking performance for sinusoidal test motion, relevant to the turntable tracking experiments used in this work, and finds that the $\alpha-\beta-\gamma$ filter gives the best results. He implements time varying filter gains which converge on optimal gains derived from the tracking index. Visual servo applications using Kalman filters have been reported by Corke [63, 64], $\alpha-\beta$ filters by Allen [7] and Safadi [218], and best-fit extrapolators by Hunt [127].

As already observed the $\alpha-\beta$ and Kalman filters have adjustable parameters which govern the tradeoff between smoothness and response to maneuver. In the experiment the robot will track an object on a turntable so phase lag in the velocity estimate is important and will be the basis of a comparison between approaches. A number of tracking filters have been simulated, see for example Figure 7.30, under identical conditions, where the target follows the standard trajectory (7.10) and the position estimates are corrupted by additive Gaussian measurement noise. It was observed qualitatively that some velocity estimates were much 'rougher' than others, and in an attempt to quantify this a 'roughness' a metric is proposed

$$\rho = 1000\sqrt{\frac{\int_{t_1}^{t_2} \mathrm{H}\,\{\hat{v}(t)\}^2 dt}{t_2 - t_1}} \tag{7.61}$$

where H is a high-pass filter selected to accentuate the 'roughness' in the velocity estimate. In this simulation H is chosen as a 9th order Chebyshev type I digital filter with break frequency of 5 Hz and 3 dB passband ripple. Initial discontinuities in the velocity estimate, see Figure 7.30, induce ringing in the filter so the first 1 s of the filtered velocity estimate is not used for the RMS calculation; that is $t_1 = 1$ s.

The results are summarized in Figure 7.31 and show the fundamental tradeoff between roughness and lag. The first- and second-order difference both exhibit low lag

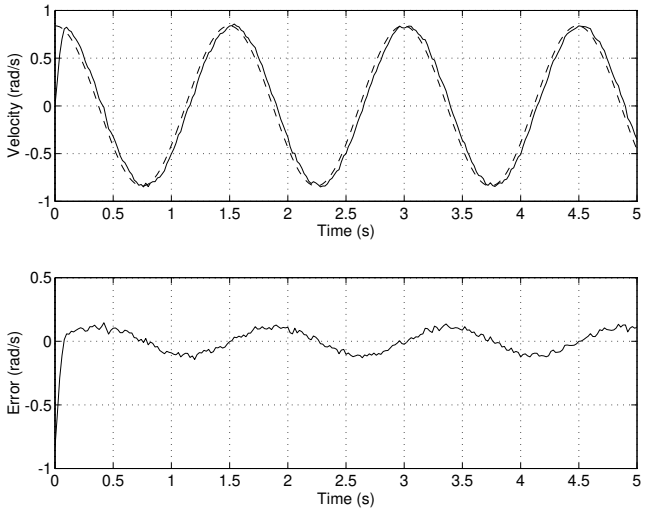


Figure 7.30: Simulated response of $\alpha-\beta$ velocity estimator for $\alpha = 0.65$ and Benedict-Bordner tuning. True velocity (dotted) and filter estimate (solid). Target position is $\theta = 0.2\sin(4.2t)$ rad. Gaussian pixel noise of variance $0.5\,\text{pixel}^2$ has been added to the position 'measurement' data prior to velocity estimation.

and RMS estimation error, but produce the roughest velocity estimates. The $\alpha-\beta$ filters were able to give 'smooth' velocity estimates with low lag and RMS error, see again Figure 7.30. The Benedict-Bordner criterion provide better performance for this target motion than does critically damped tuning. The $\alpha-\beta-\gamma$ filter shows increasing RMS error and roughness as its bandwidth is increased. All the estimators apart from the 2nd order difference and $\alpha-\beta-\gamma$ filters assume constant target velocity. The 2nd order difference and $\alpha-\beta-\gamma$ filters assume constant acceleration and are thus capable of predicting velocity one step ahead. However these filters exhibit the worst roughness and RMS error since the acceleration estimate is based on doubly-differentiated position data.

The arithmetic operation counts of the various filters are compared in Table 7.4. Computation time is not a significant issue, but for the reasons outlined above the variable gain Kalman filter does not appear to confer sufficient advantage to warrant the extra cost.

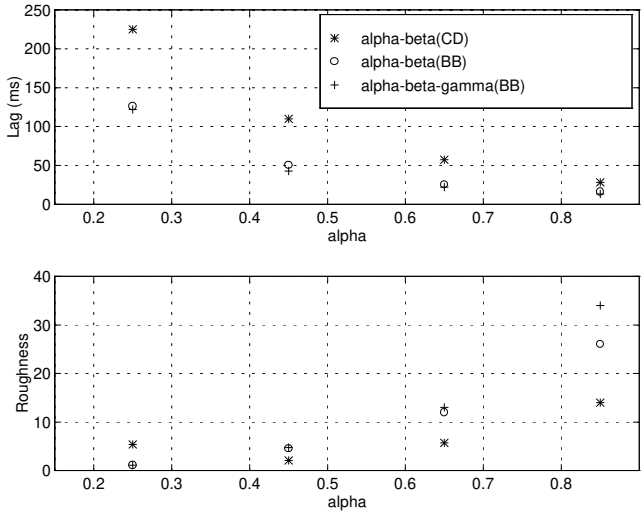


Figure 7.31: Comparison of velocity estimators showing roughness and lag as α varies. Evaluated for sinusoidal target motion at frequency of 4.2 rad/s, lens gain of 640 pixel/rad, and Gaussian quantization noise with standard deviation of 0.5 pixel.

Filter type	*	+
1st order diff	1	1
2nd order diff	3	2
Kalman (matrix form)	68	78
Kalman (simplified)	15	16
$\alpha-\beta$	4	4

Table 7.4: Comparison of operation count for various velocity estimators.

7.5.3 Feedforward control implementation

This section describes the velocity feedforward controller of Figure 7.21 in more detail and with an emphasis on implementation issues. The controller has a variable structure with two modes: gazing or fixating. With no target in view, the system is in gaze mode and maintains a gaze direction by closing position-control loops around each axis based on encoder feedback. In fixation mode the controller attempts to keep the target centroid at the desired image plane coordinate. The velocity demand comprises,

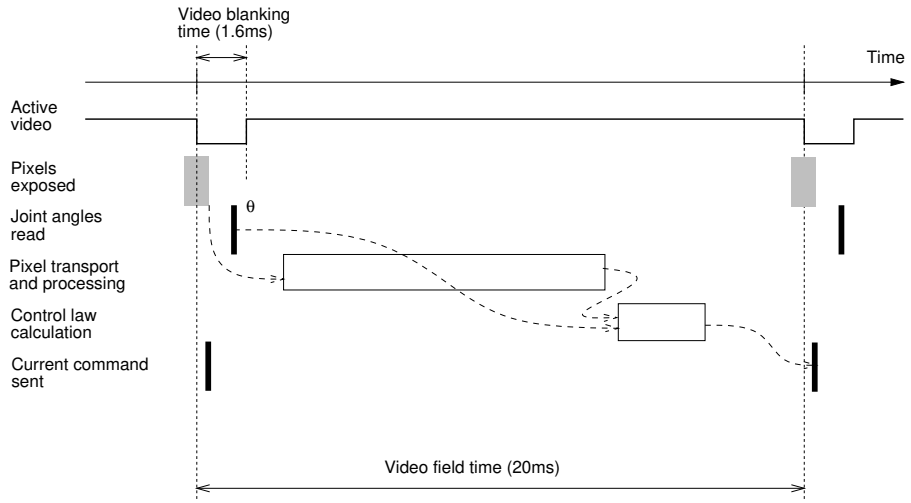


Figure 7.32: Details of system timing for velocity feedforward controller. Note the small time delay between between the center of the pixel exposure interval and the joint angles being read; in practice this is around 1.3 ms.

for each DOF, predicted target velocity feedforward and image plane centroid error feedback

$$\dot{x}_d = \hat{\dot{x}}_t + {}^iK_p\,{}^iX + \begin{cases} {}^iK_i \int {}^iX\,dt & \text{if } |{}^iX| \leq {}^i\Delta \\ 0 & \text{otherwise} \end{cases} \tag{7.62}$$

When the target is within a designated region about the desired image plane centroid integral action is enabled. The integral is reset continuously when the target is outside this region. Both $\alpha-\beta$ and Kalman filters have been used for velocity estimation, $\hat{\dot{x}}_t$, in this work. The transition from gazing to fixation involves initiating the tracking filters. No vision based motion is allowed until the target has been in view for at least two consecutive fields in order to compute a crude target velocity estimate for filter initiation. Intuitively it is unproductive to start moving upon first sighting the target since its velocity is unknown, and the robot may start moving in quite the wrong direction.

The pan and tilt axis axis velocities are controlled by two digital axis-velocity loops as described in Section 7.5.1. There is explicit correction for the wrist axis cross-coupling (2.38). The motor-referenced joint velocities are

$$\dot{\theta}_5 = G_5\omega_{tilt} \tag{7.63}$$

$$\dot{\theta}_6 = G_6\omega_{pan} - G_{56}G_6\omega_{tilt} \tag{7.64}$$

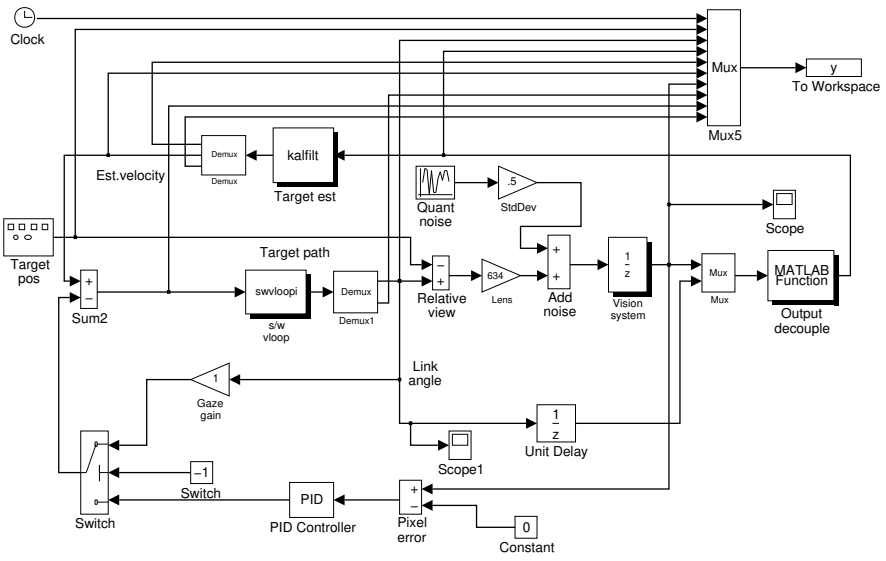


Figure 7.33: SIMULINK model FFVSERVO: feedforward fixation controller.

Figure 7.32 shows details of the timing relationships involved. The actual instant of camera sampling is not signalled by the camera, but has been determined experimentally in Figure 3.14 with respect to the video waveform. The short exposure time is required in order that the camera approximate an ideal sampler of the visual state which is assumed for the control design. Motion blur is not significant since fixation keeps the target fixed with respect to the camera. The robot's joint angles are sampled by a task during the vertical blanking interval and combined with the image plane centroid using (7.43) to estimate the target position. There is some finite time difference between sampling the joint angles and pixel exposure. Analysis of system timestamps indicates that this averages 1.3 ms with a standard deviation of 0.11 ms. It would be possible to use first-order predictors to compute joint angles at the video sampling instant but this has not been attempted.

A single-axis SIMULINK model, shown in Figure 7.33, was created to verify the overall operation and to assist in initial control parameter selection. Figure 7.34 shows the simulated performance when tracking the standard target motion with peak velocity of 1 radl/s. It can be seen that the target velocity estimator is providing most of the velocity demand, but that this feedforward signal is lagging the axis velocity by approximately 70 ms. This is due to lag in the velocity estimator as discussed above, and also the unit delay on position data being passed to the velocity estimator.

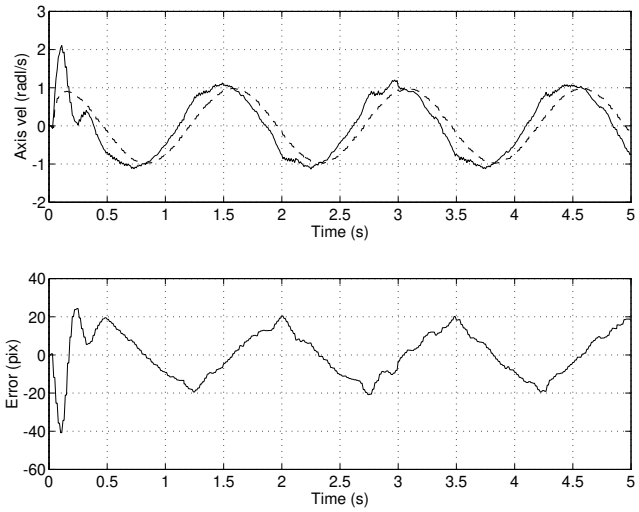


Figure 7.34: Simulation of feedforward fixation controller for standard target motion. Top plot shows total axis velocity demand (solid) and estimated target velocity (dashed). Bottom plot shows the target centroid error. RMS pixel error, excluding the initial transient, is 11 pixels. (${}^iK_p = 0.03$, ${}^iK_i = 0.0042$.)

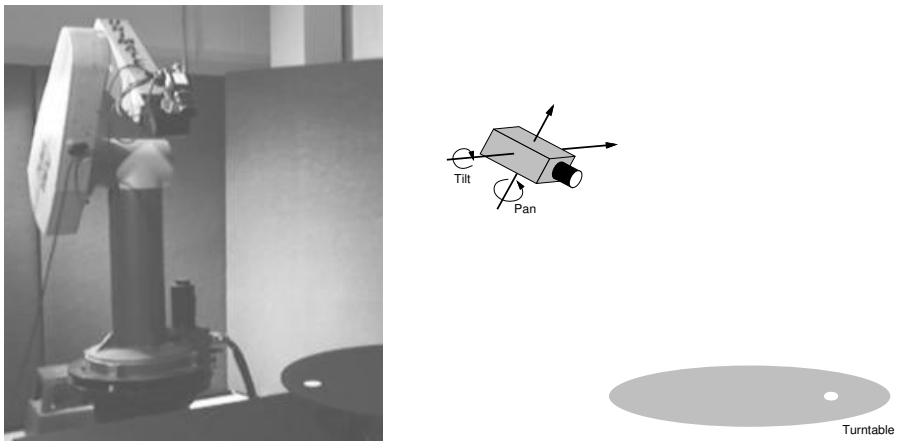


Figure 7.35: Turntable fixation experiment

7.5.4 Experimental results

A number of experiments have been conducted to investigate the performance of the fixation controller. To achieve a particularly challenging motion for the tracking controller a turntable has been built whose rotational velocity can be constant, or a reversing trapezoidal profile. At maximum rotational speed of 40 rpm the former provides targets with the standard motion described in (7.10). The reversing profile results in target motion with a complex Cartesian acceleration profile.

In the first experiment the camera is fixated on a target rotating on a turntable as shown in Figure 7.35. The turntable is rotating at the standard trajectory's angular velocity 4.2 rad/s. Figure 7.36 shows the image plane coordinate error in pixels. It can be seen that the the target is kept within ± 15 pixels of the reference. The earlier, feedback-only strategy results in large following errors and a lag of over 40°. Figure 7.37 shows the measured and feedforward joint velocity for the pan and tilt axes. The joint velocity of up to 2 radl/s is close to the limit of the Unimate velocity loop, 2.3 radl/s. The feedforward velocity signal has the same magnitude as the velocity demand but is slightly lagging, as was the case for the simulation. The other component of the velocity demand is provided by the PI feedback law which is necessary to

- achieve zero position error tracking, since matching target and robot velocity still allows for an arbitrary constant position error;
- overcome lags in the feedforward velocity demand and the velocity loop itself;
- reject the disturbance introduced by a target whose centroid lies off the optical axis which will appear to rotate around the principal point as the camera rolls due to motion of joints 5 and 6 by (6.7);
- accommodate the pose dependent gain relating $\dot{\theta}_5$ to tilt rate in (6.6).

The controller is quite robust to errors in the lens gain estimate used in (7.43). During fixation the centroid error ${}^{i}X$ is small, so minimizing the significance of errors in K_{lens}.

A disturbance may be generated by using the teach pendant to move the first three robot joints. The fixation controller is able to counter this disturbance but the rejection dynamics have not been investigated. The disturbance could also be countered by feeding forward the camera pan/tilt rates due to the motion of those joints. Such a structure would then be similar to the human eye in that the eye muscles accept feedback from the retina, as well as feedforward information from the vestibular system, giving head lateral and rotational acceleration, and head position information from the neck muscles.

Figure 7.38 shows the setup for an experiment where the camera fixates on a ping-pong ball thrown across the system's field of view. Experimental results shown in Figure 7.39 are obtained with the same controller as the previous experiment. The recorded event is very brief, lasting approximately 700 ms. Tracking ceases when the

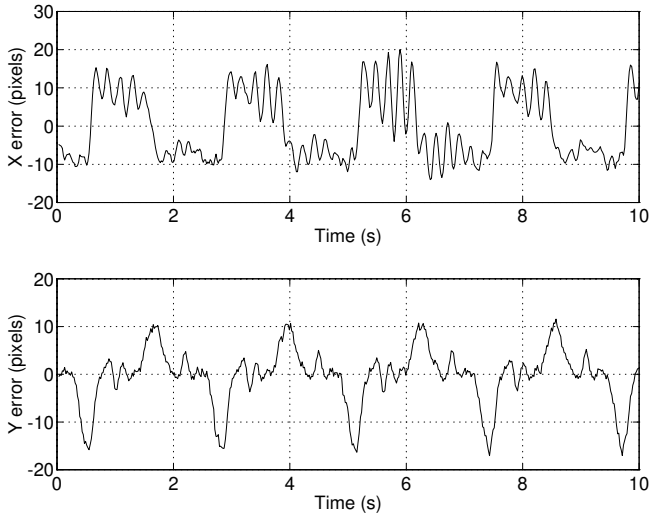


Figure 7.36: Measured tracking performance, centroid error, for target on turntable revolving at 4.2 rad/s. This data was logged by RTVL.

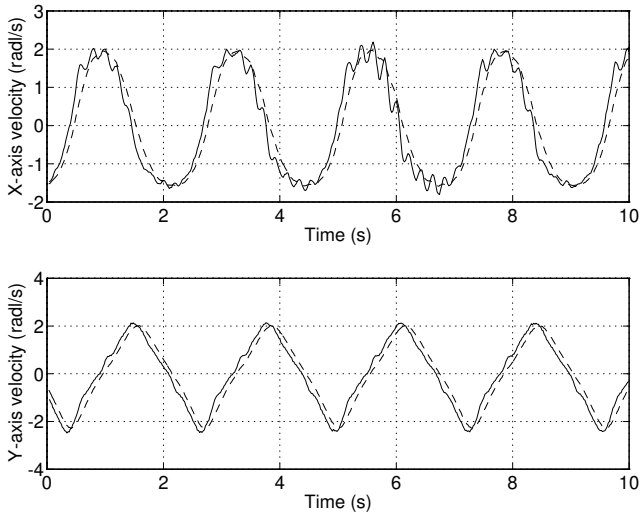


Figure 7.37: Measured tracking velocity for target on turntable revolving at 4.2 rad/s, showing axis velocity (solid) and estimated target velocity (dashed). Note that the Y-axis (tilt) velocity is somewhat triangular — this is due to the $1/S_6$ term in the tilt kinematics given by (6.6). This data was logged by RTVL.

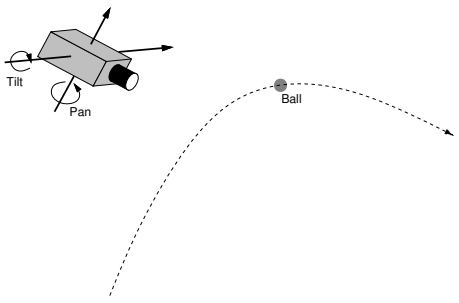


Figure 7.38: Ping pong ball fixation experiment

ball disappears from view as it moves into a poorly lit region of the laboratory. From the centroid error plot it can be seen that the robot achieves an approximately constant centroid error of less than 30 pixels in the vertical, Y, direction in which the ball has constant acceleration. In the horizontal, X, direction the robot is seen to have overshot the target. The measured joint velocities show a peak of over 4 radl/s on the tilt axis, which has a limit due to voltage saturation of 5 radl/s. The apparent size of the ball (area) can be seen to vary with its distance from the camera. The tracking filter in this case is a Kalman filter where the time varying gain is an advantage given the transient nature of the event. The robot does not commence moving until the ball has been in the field of view for a few field times in order for the Kalman filter state estimates to converge.

7.6 Biological parallels

There are some interesting parallels between the control models developed here using control system analysis and synthesis, and those proposed to explain the operation of the human visual fixation reflex. Robinson [214] investigates the control of two human visual behaviours, saccadic motion and smooth pursuit, and how instability is avoided given the presence of significant delay in neural sensing and actuation 'circuits'. The smooth pursuit reflex operates for target motion up to 100 deg/s and experiments reveal that the response is characterized by a delay of 130 ms and a timeconstant of 40 ms. The delay is partitioned as 50 ms for the neural system and 80 ms for peripheral (muscle) delay. Sachadic eye motions are used to direct the fovea to a point of interest and can be considered as 'open-loop' planned moves. The neural delay for saccadic motion is 120 ms compared to 50 ms for fixation motion, and indicates that fixation is a lower level and more reflexive action.

For a constant velocity target the eye achieves a steady-state velocity of 90% of the target velocity, that is, the oculomotor system is of Type 0. There is also physi-

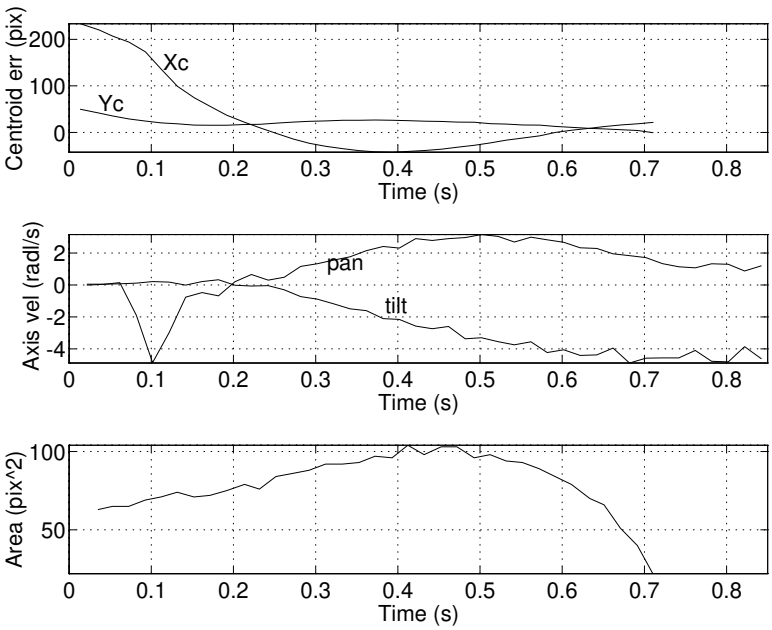


Figure 7.39: Measured tracking performance for flying ping-pong ball. Shown are centroid error, link velocity and observed area of target. This data was logged by RTVL.

ological evidence that the oculomotor control system can be considered a continuous time system. Like the robot visual servo system discussed earlier, the eye controller is driven by retinal position or velocity which measures directly the error between target and eye direction. Since the system is driven by an error signal it is a negative feedback system which admits the possibility of stability problems. However despite the significant delays in the sensor and actuator this feedback system is stable. To explain how the mechanism avoided instability he proposes the existence of an inner positive-feedback loop that cancels the effect of the potentially problematic negative-feedback loop. This structure is shown in block diagram form in Figure 7.40. Ideally the two feedback paths cancel so the resulting dynamics are dictated by the forward path. Such an explanation is somewhat bizarre from a control systems point of view where 'positi ve feedback' is generally considered as something to be avoided. However the proposed model can be interpreted, see Figure 7.41, as being structurally similar to the velocity feedforward control described in the previous section. The effect of the

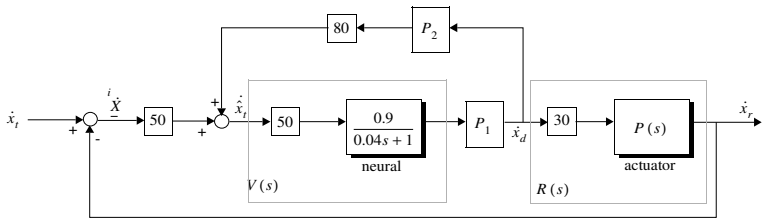


Figure 7.40: Proposed model of oculomotor control system after Robinson [214] with some notational differences to conform with this work. Square boxes containing numbers represent pure time delay in units of milliseconds. P_1 and P_2 are adaptive gains. Actuator dynamics are taken to be $P(s) = 1$.

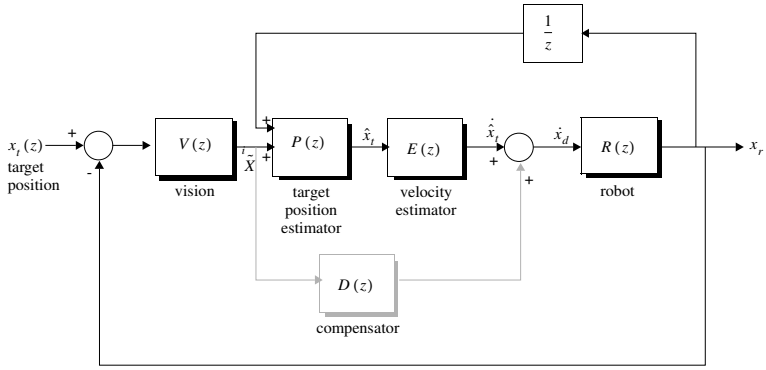


Figure 7.41: Rearrangement of Figure 7.21 to show similarity to Robinson's proposed oculomotor control system. $P(z)$ is defined by (7.43) and is essentially a summation. The path shown dotted represents the less significant feedback path, which if eliminated results in a structure that is identical to Robinson's.

positive feedback is to create an estimate of the target velocity, based on measured retinal velocity and delayed eye velocity command. The feedforward controller of Figure 7.21 is very similar except that target position is estimated from 'retinal' and motor position information and then differentiated to form the principal component of the motor velocity signal.

Eliminating negative feedback, as shown in the model of Figure 7.40, also eliminates its benefits, particularly robustness to parameter variations. In a biological system these variations may be caused by injury, disease or ageing. Robinson proposes that parameter adaption occurs, modelled by the gain terms P_1 and P_2, and provides experimental evidence to support this. Such 'plasticity' in neural circuits is common to much motor learning and involves change over timescales measured in days or

even weeks. Parameter variation again admits instability, if for instance the positive-feedback gain is greater than the negative-feedback gain. Robinson's paper does not discuss the dynamics of the actuator apart from time delay, but it may be that for eye actuation inertia forces are insignificant.

7.7 Summary

The use of machine vision for high-performance motion control is a significant challenge due to the innate characteristics of the vision sensor which include relatively low sample rate, latency and significant quantization. Many of the reports in the literature use low loop gains to achieve stable, but low-speed, closed-loop visual control.

The simple and commonly used proportional feedback controller, while demonstrating an adequate step response, has poor tracking performance. Compensator design, to improve this performance, has followed classical linear principles of increasing system Type and positioning the closed-loop poles by means of PID or pole-placement control. It has been found that the closed-loop poles cannot be made arbitrarily fast and are constrained by a practical requirement for compensator stability. This instability is believed to be due to modelling errors, or plant non-linearities which lead to model errors at saturation.

It is difficult to quantitatively compare the performance of this system with other reports in the literature, due largely to the lack of any common performance metrics. A tracking performance measure was defined in terms of image plane error for a standard sinusoidal target motion. A number of reports on robot heads [231, 270] cite impressive peak velocity and acceleration capabilities, but these are achieved for saccadic not closed-loop visually guided motion. Many reports provide step response data and this gives some measure of closed-loop bandwidth but it has been shown that phase characteristics are critical in tracking applications.

The experimental configuration used in this work, like the majority of reported experimental systems, implements a visual feedback loop around the robot's existing position-control loops. However it has been shown that the redundant levels of control add to system complexity and can reduce closed-loop performance by increasing open-loop latency. Investigation of alternative control structures based on various underlying axis-control methods concluded that axis feedback, at the least velocity feedback, is required to give acceptable performance given the low visual sampling rate and the non-ideality of a real robot axis.

The design constraints inherent in feedback-only control lead to the consideration of feedforward control, which gives additional design degrees of freedom by manipulating system zeros. A 2-DOF variable-structure velocity-feedforward controller is introduced which is capable of high-performance tracking without the stability problems associated with feedback-only control. The feedforward approach effectively transforms the problem from control synthesis to motion estimation.

The next chapter extends the principles developed in this chapter, and applies them to visual servoing for direct end-point control and to 3-DOF translational tracking.

Chapter 8

Further experiments in visual servoing

This chapter extends, and brings together, the principles developed in earlier chapters. Two examples, both involving dynamic visual servoing, will be presented. Firstly 1-DOF visual control of a major robot axis will be investigated with particular emphasis on the endpoint rather than axis motion. Secondly, planar translational camera control will be investigated which requires coordinated motion of the robot's 3 base axes. The simple 2-DOF control of the previous chapter effectively decoupled the dynamics of each axis, with one actuator per camera DOF. In this case there is considerable kinematic and dynamic coupling between the axes involved. This chapter must, by necessity, address a number of issues in robot or motion control such as controller structure artifacts, friction and structural dynamics. Many of these topics have been introduced previously and will not be dealt with in great detail here; rather a 'solution' oriented approach will be taken.

8.1 Visual control of a major axis

Previous sections have described the visual control of the robot's wrist axes, which are almost ideal, in that they have low inertia and friction and are free from structural resonances in the frequency range of interest. This section investigates the application of visual servoing to the control of a major robot axis, in this case the waist axis, and it will be shown that the control problem is non-trivial due to 'real-world' effects such as significant non-linear friction and structural resonances.

In this experiment the camera is mounted on the robot's end-effector, which moves in a horizontal plane over a worktable on which the target object sits at a location that is

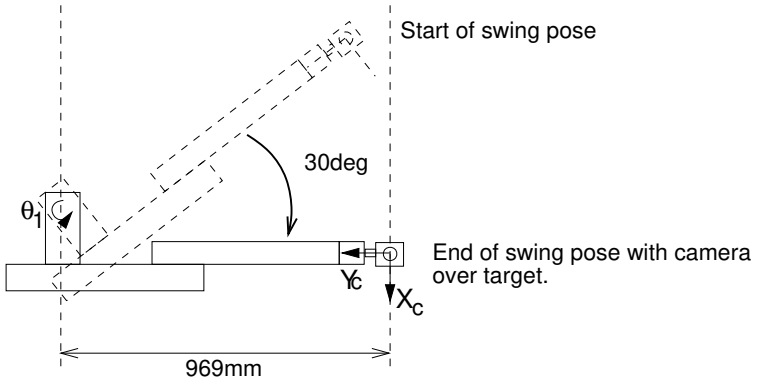


Figure 8.1: Plan view of the experimental setup for major axis control.

only approximately known. The robot's waist joint swings through approximately 30° and is visually servoed to stop with the camera directly over the target. Only 1-DOF is controlled, so the camera's true motion is an arc and the positioning is such that the X-coordinate of the target's image is brought to the desired image plane location. A schematic plan view is shown in Figure 8.1.

This experiment has some similarity with the way in which robots are used to perform part placement in planar assembly applications such as PCB manufacture. That role is typically performed by high-speed SCARA robots which must move quickly from part feeder to the part location, settle, and then release the part. Electronic assembly robots such as the AdeptOne are capable of positioning accuracy of $75\,\mu$m and end-effector speeds of $9\,\mathrm{m/s}$.

8.1.1 The experimental setup

The camera looks downward toward the table. As the camera swings toward the target, the target appears to move in the negative X direction with respect to the camera's coordinate frame. The X component of the target's centroid is taken as a measure of the camera's position relative to the target. The robot arm is well extended so that a large inertia is 'seen' by the joint 1 axis. The arm pose is $[\theta_1,\ 18,\ -123,\ 0,\ 14,\ -90]°$, and using the rigid-body dynamic model from Section 2.2.4 yields an inertia for joint 1 due to link and armature inertias of $3.92\,\mathrm{kg.m^2}$. The camera inertia, computed from known mass and rotation radius, is $0.970\,\mathrm{kg.m^2}$. The total inertia at the link is $4.89\,\mathrm{kg.m^2}$, or in normalized form, $7.5J_{m_1}$. In this pose the camera rotational radius[1] is 969 mm and

[1] The distance between the joint 1 rotational axis and the optical axis of the camera.

Time (s)	$\dot{\theta}_{pk}$ (radl/s)	$\ddot{\theta}_{pk}$ (radl/s^2)
1.0	0.98	2.95
0.8	1.23	4.60
0.6	1.64	8.18
0.5	1.96	11.8
0.4	2.45	18.4

Table 8.1: Peak velocity and acceleration for the test trajectory with decreasing durations.

was determined by a calibration procedure which measured the ratio of target centroid displacement to joint angle rotation.

8.1.2 Trajectory generation

In this experiment the manipulator performs a large motion, and visual information from the target is available only during the latter phase of the move. A trajectory generator is thus required to move the robot to the general vicinity of the target. The trajectory used is a quintic polynomial which has the desirable characteristics of continuous acceleration and computational simplicity. In terms of normalized time $0 \leq \tau \leq 1$ where $\tau = t/T$ and T is the duration of the motion, the joint angle is

$$\theta_d(\tau) = A\tau^5 + B\tau^4 + C\tau^3 + F \tag{8.1}$$

The coefficients are computed from the initial and final joint angles, θ_0 and θ_1 respectively.

$$\begin{aligned} A &= 6(\theta_1 - \theta_0) && (8.2)\\ B &= -15(\theta_1 - \theta_0) && (8.3)\\ C &= 10(\theta_1 - \theta_0) && (8.4)\\ F &= \theta_0 && (8.5) \end{aligned}$$

Velocity is given by

$$\begin{aligned} \frac{d\theta}{dt} &= \frac{d\theta}{d\tau}\frac{d\tau}{dt} && (8.6)\\ &= \frac{1}{T}\left(5A\tau^4 + 4B\tau^3 + 3C\tau^2\right) && (8.7) \end{aligned}$$

and in a similar fashion acceleration is shown to be

$$\frac{d^2\theta}{dt^2} = \frac{1}{T^2}\left(20A\tau^3 + 12B\tau^2 + 6C\tau\right) \tag{8.8}$$

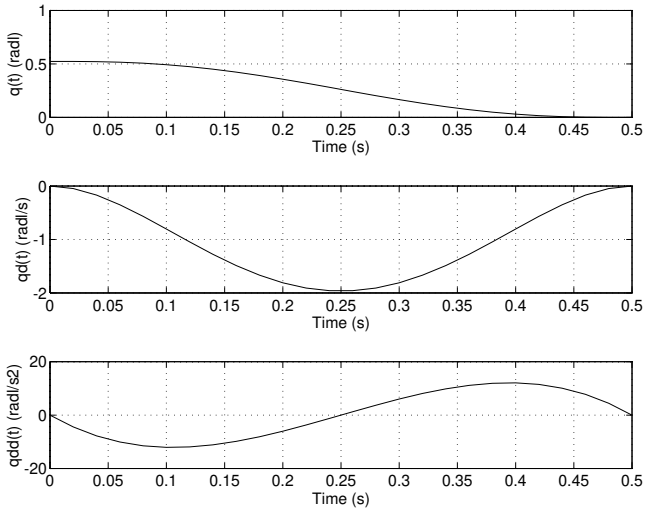


Figure 8.2: Position, velocity and acceleration profile of the quintic polynomial.

The minimum achievable move duration will ultimately be limited by friction and amplifier voltage and current saturation. The peak velocities and accelerations for this trajectory have been computed for a number of time intervals, and these results are summarized in Table 8.1. Based on the manipulator performance limits from Table 2.21 it is clear that the 0.5 s trajectory is at approximately the velocity limit for joint 1, and will be selected as the reference trajectory for this section. Figure 8.2 shows the computed position and velocity as a function of time for a 0.5s swing over 30°.

The average joint velocity is 1 radl/s which in this pose results in an average translational velocity of around 1000 mm/s, the quoted limit of the Puma 560 [255]. The peak camera translational velocity is almost 2000 mm/s, which with an exposure time of 2 ms would result in a maximum motion blur of 4 mm causing a lagging image centroid estimate and some distortion of the target image. These effects decrease as the robot slows down on approach to the target.

8.1.3 Puma 'native' position control

Joint position control is performed by the Unimate position control system described previously in Section 2.3.6. The host issues setpoints computed using (8.1) at 14 ms intervals and these are passed to the digital axis servo. This interpolates between the

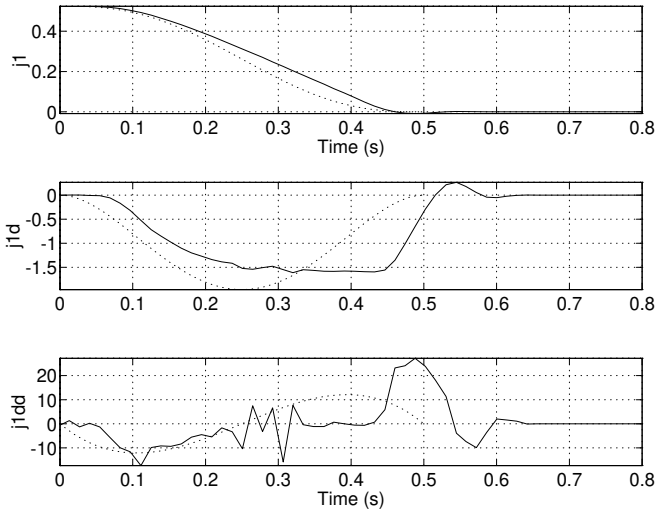


Figure 8.3: Measured axis response for Unimate position control, showing position, velocity and acceleration: measured (solid) and demand (dotted). Data was recorded by the RTVL system at 14 ms intervals, and an off-line procedure using 5-point numerical differentiation was used to obtain the 'measured' velocity and acceleration data. Note the step deceleration at 0.45 s or 70 ms before the first zero crossing of the measured velocity. All quantities are load referenced, angles in radians.

position setpoints emitted by the trajectory generator and closes a position loop around the robot's analog velocity loop at a sample rate of approximately 1 kHz.

The performance of the Unimate servo system is shown in Figure 8.3 which compares the demanded and measured axis position, velocity and acceleration. It can be seen that the measured joint angle lags considerably behind the desired joint angle and that the axis velocity has 'bottomed out' between 0.25 and 0.45 s. The limiting value of -1.6 radl/s is due to velocity loop saturation. Table 2.20 predicts a velocity limit of 1.42 radl/s for joint 1 which is of a similar order to that observed in this experiment[2]. The position loop has very high gain and it can be shown from Section 2.3.6 that the velocity loop demand will be saturated for position error greater than 0.018 radl or 1°. By $t \approx 0.45$ s position error is sufficiently small that the velocity demand rises above the saturation level and the axis begins to decelerate and the acceleration plot shows

[2] Gain variation in the analog electronics, mentioned in Section 2.3.6, or friction variation may account for this discrepancy.

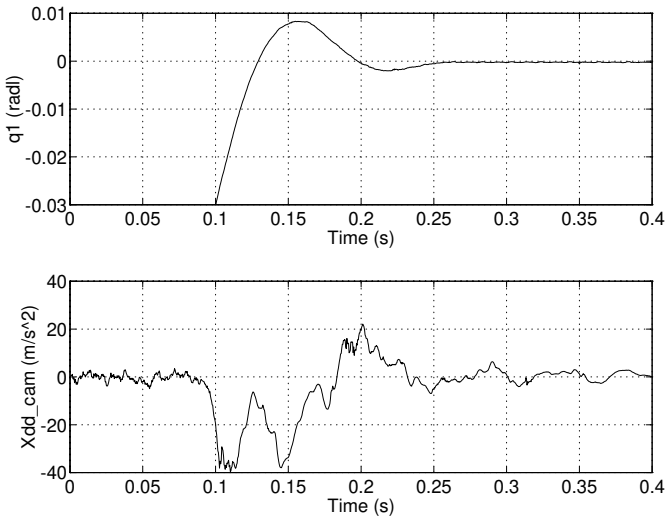


Figure 8.4: Measured joint angle and camera acceleration under Unimate position control. Data was recorded with a 2-channel data-acquisition unit and the time scales of this figure are not aligned with those of Figure 8.3. However the onset of oscillation is around 70 ms before the joint angle peak.

a step occurring at this time. Considerable deceleration can be achieved since back-EMF acts to increase motor current which will then be limited by the current loop, while Coulomb and viscous friction also supply decelerating torque.

Figure 8.4 shows the response of the motor joint angle and the camera acceleration for the 0.5s trajectory recorded using a 2-channel data-acquisition unit. The joint angle has overshot by 8.3×10^{-3} radl resulting in a camera overshoot of 8 mm. From the plot of camera acceleration there is clearly considerable oscillation of the camera, peaking at 4 g. The high deceleration caused by the axis controller, shown in Figure 8.3, has excited a resonance in the manipulator structure at around 20 Hz. The large oscillation between 0.10 and 0.15 s has an amplitude of approximately $15\,\mathrm{m/s^2}$ corresponding to a displacement amplitude of 1 mm. The lower level vibration present after 0.25 s corresponds to a displacement amplitude of around 0.3 mm which is significantly less than 1 pixel. Figure 8.5 compares the displacement of the camera determined by two different means. Over the small angular range involved in motion near the target, the end-effector position can be related directly to link angle, θ_1, by

$$x_m = r\theta_1 \tag{8.9}$$

where r is the radius of camera rotation. The displacement can also be determined from the camera

$$x_c = K_{lens}{}^{i}X \tag{8.10}$$

where the lens gain, in this translational control case, will be a function of target distance as discussed in Section 6.3.2. Lens gain was determined from the image plane distance between the centroids of calibration markers with known spacing, resulting in an estimate $K_{lens} = 0.766\,\text{mm/pixel}$. Figure 8.5 is derived from data logged by the controller itself. Timestamps on target centroid data indicate the time at which the region data was fetched from the APA-512, not when the camera pixels were exposed. However the video-locked timestamping system allows the exposure time to be estimated, and this is used when plotting centroid data in Figure 8.5. There is good correspondence between the two measurements as the robot approaches the target[3]. However at the peak of the overshoot the vision system indicates that the displacement is 1.9 mm greater than that indicated by the axis. Data recorded during the experiment also indicates that joints 4 and 6 rotate by 1.5×10^{-3} and 6×10^{-4} radl during this part of the trajectory, due to inertial forces acting on the camera. In this pose, such rotation would tend to reduce the visually perceived overshoot. From Figure 8.5 the damped natural frequency of the overshoot motion is approximately 7 Hz and this would be largely a function of the Unimate position loop. As discussed in Section 2.3.6 the switchable integral action feature of the position loop[4] results in a poorly-damped low-frequency pole pair. With integral action disabled the overshoot is only 60% of that shown in Figure 8.4 and there is no subsequent undershoot. However the axis stops 37×10^{-3} radm (6 encoders) or 0.5 mm short of the target due to friction effects.

8.1.4 Understanding joint 1 dynamics

The purpose of this section is to develop a simple model of the joint 1 dynamics which includes structural resonance in the transmission and the electro-mechanical dynamics of the motor and current loop. Figure 8.6 shows the measured frequency response function between motor shaft angle and acceleration of the camera, for the robot in the standard pose used for this experiment. There are 3 structural modes in the frequency range measured: the first occurs at 19 Hz which is below the 25 Hz visual Nyquist frequency; the others are above the Nyquist frequency and may result in aliasing.

[3] There is evidence that the joint angle measurement lags approximately 2 ms behind the vision measurement. This may be due to incorrect compensation of timestamped centroid data, or overhead and timing skew in reading encoder data from the Unimate digital servo board. In addition the encoder value returned by the servo is the value at the last clock period. Further refinement of measurement times will not be pursued.

[4] Integral action is configured to switch in when the axis is within 50 encoder counts of the target which is 0.005 radl for this axis.

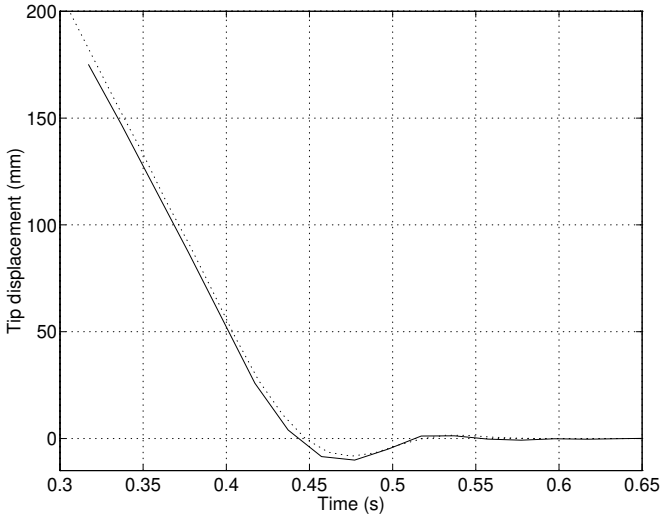


Figure 8.5: Measured tip displacement determined from axis sensing (dotted) and end-effector mounted camera (solid).

The current-loop transfer function was measured, see Figure 8.7, and a model fitted up to 30Hz

$$\frac{\Omega_m}{V_{I_d}} = 6.26 \frac{[0.029,\ 120]}{(33.0)[0.14,\ 165]} \text{radm/Vs} \tag{8.11}$$

This indicates a real pole at 5.3 Hz and a lightly damped complex pole/zero pair due to the mechanical resonance. The anti-resonance in this transfer function corresponds to the 19 Hz resonance seen in the frequency response of Figure 8.6. It is interesting to compare this response with the much simpler one for joint 6 shown in Figure 2.15.

The velocity-loop frequency response function was also measured, and the fitted model is

$$\frac{\Omega_m}{V_{\Omega_d}} = 3.30 \frac{[0.018,\ 117]}{(82.0)[0.31,\ 132]} \text{radm/Vs} \tag{8.12}$$

The structural zeros are unchanged[5] by the action of the velocity loop and the structural poles are more damped. The low-frequency real pole has been pushed out to around 13 Hz. A root-locus showing the effect of velocity feedback on the simple model of of (8.11) is shown in Figure 8.8. The natural frequency of the resonance is reduced marginally and for moderate gains the damping is increased.

[5] Within the accuracy of the estimation procedure.

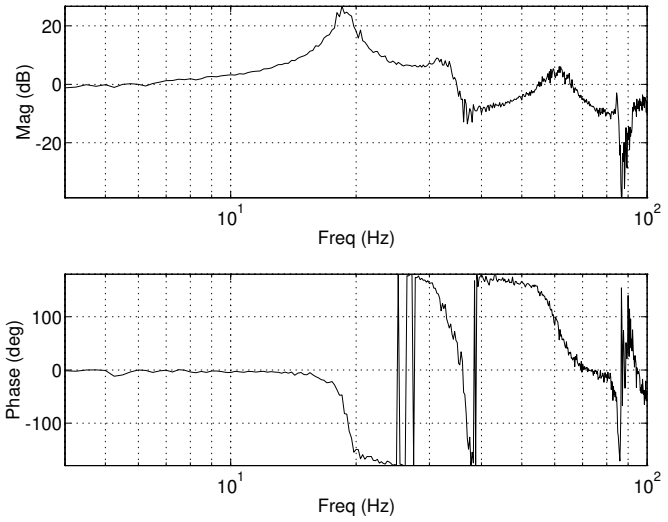


Figure 8.6: Measured frequency response function $1/(j\omega)^2\ddot{X}_{cam}/\Theta_m$ which includes arm structure and transmission dynamics. The first resonance is at 19Hz. Good coherence was obtained above 4Hz.

The main aspects of the joint 1 dynamics are accounted for by the simple two-inertia model shown in Figure 8.9. In this model J_m represents the motor inertia and J_l the motor-referenced inertia of the arm and camera. Elasticity and energy dissipation in the transmission, physically due to torsional compliance in the long vertical drive shaft between the bull gear and link 1, are modelled by the spring K and damper B. Motor friction is represented by B_m, and motor electrical torque is τ_m.

Laplace transformed equations of motion for this system may be written

$$s^2 J_m \Theta_m = \tau_m - B_m s\Theta_m - (Bs+K)(\Theta_m - \Theta_l) \quad (8.13)$$

$$s^2 J_l \Theta_l = (Bs+K)(\Theta_m - \Theta_l) \quad (8.14)$$

These reduce to the transfer functions

$$\frac{\Omega_m}{\tau_m} = \frac{s\Theta_m}{\tau_m} = \frac{J_l s^2 + Bs + K}{\Delta} \quad (8.15)$$

$$\frac{\Omega_l}{\tau_m} = \frac{s\Theta_l}{\tau_m} = \frac{Bs+K}{\Delta} \quad (8.16)$$

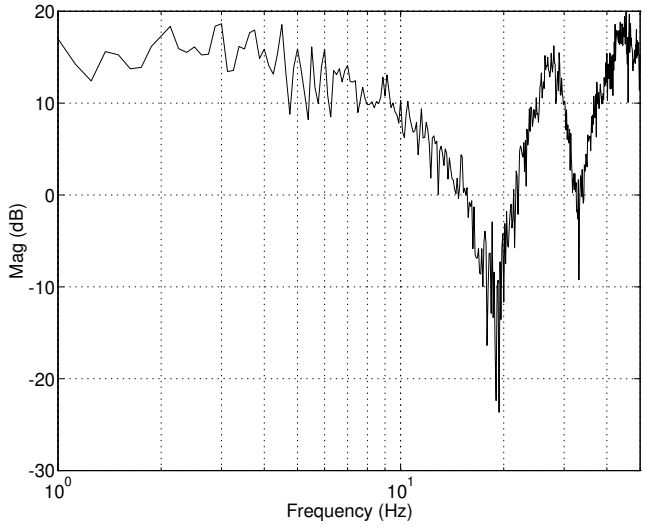


Figure 8.7: Measured frequency response function magnitude for joint 1 motor and current loop $\Omega_m(s)/V_{I_d}(s)$.

where

$$\Delta = J_m J_l s^3 + [(J_m + J_l)B + B_m J_l]s^2 + [(J_m + J_l)K + BB_m]s + KB_m \tag{8.17}$$

The transfer function from motor position to load position, measured in Figure 8.6, is given by this simple model as

$$\frac{\Theta_l}{\Theta_m} = \frac{Bs + K}{J_l s^2 + Bs + K} \tag{8.18}$$

$$= \frac{(\sigma_0)}{[\zeta_0, \omega_0]} \tag{8.19}$$

in the shorthand notation employed previously. The numerator of the transfer function (8.15) is the same as the denominator of (8.18) and this was observed in the measured transfer functions shown in Figures 8.6 and 8.7. In (8.19),

$$\sigma_0 = \frac{K}{B} \tag{8.20}$$

$$\omega_0 = \sqrt{\frac{K}{J_l}} \tag{8.21}$$

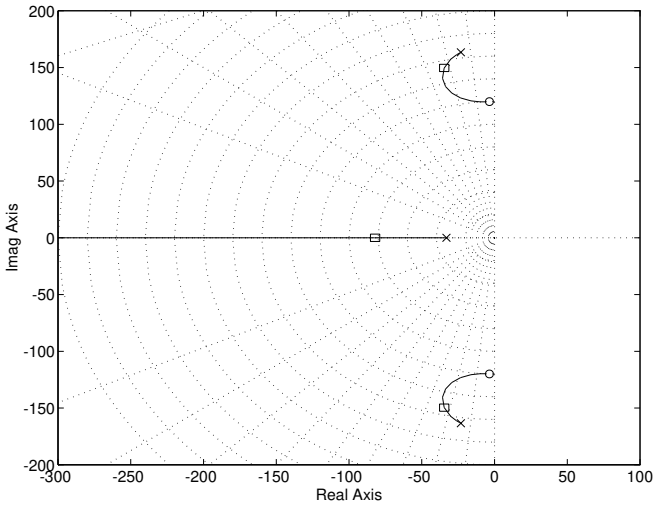


Figure 8.8: Root-locus for motor and current loop with single-mode structural model. The marked closed-loop poles correspond to a loop gain of 11.5 which places the real current loop pole at -82 rad/s as for the velocity-loop model (8.12).

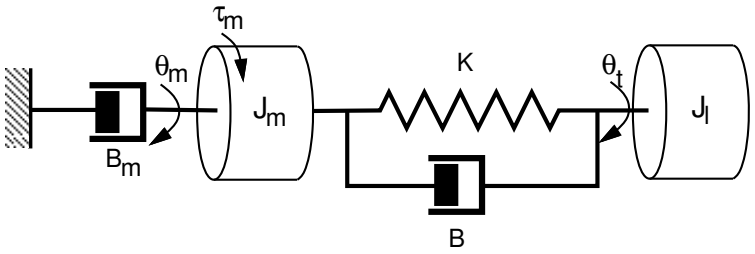


Figure 8.9: Schematic of simple two-inertia model.

$$\zeta_0 = \frac{B}{2\sqrt{KJ_l}} \tag{8.22}$$

Inspection of Figure 8.6 indicates that $\omega_0/2\pi = 19$ Hz ($\omega_0 = 120$ rad/s) and that σ_0 is at some considerably higher frequency. From (8.21) it is clear that the resonant frequency decreases as the 'outboard' inertia, J_l, increases. The selected robot pose has almost maximum possible inertia about the joint 1 axis thus bringing the resonance

as low as possible.

Considering now the motor current-to-velocity transfer function measured in Figure 8.7, the model version is, from (8.15) and (8.17),

$$\frac{\Omega_m}{\tau_m} = \frac{1}{B_m} \frac{[\zeta_0, \omega_0]}{(\sigma)[\zeta, \omega_n]} \tag{8.23}$$

The experimental model (8.11) indicates that

$$\begin{aligned} \omega_0 &= 120 && (8.24)\\ \zeta_0 &= 0.029 && (8.25)\\ \sigma &= 33 && (8.26)\\ \omega &= 165 && (8.27)\\ \zeta &= 0.14 && (8.28) \end{aligned}$$

Using the previously obtained estimate of $J_m + J_l = 7.5 J_m$, and the parameter value $J_m = 200 \times 10^{-6}\,\mathrm{kg.m^2}$, it can be deduced that

$$\begin{aligned} J_l &= 1.3 \times 10^{-3}\,\mathrm{kg.m^2} && (8.29)\\ K = J_l \omega_0^2 &= 18.7\,\mathrm{N.m.s/rad} && (8.30)\\ B = 2\zeta_0 \sqrt{K J_l} &= 9.05 \times 10^{-3}\,\mathrm{N.m.s/rad} && (8.31)\\ B_m &= 0.032\,\mathrm{N.m.s/rad} && (8.32) \end{aligned}$$

Hence $\sigma_0 = K/B = 2069\,\mathrm{rad/s}$ (329 Hz), too high to be revealed in the measurement of Figure 8.6.

If there was no compliance in the transmission from motor to load, (8.15) would reduce to

$$\begin{aligned} \frac{\Omega_m}{\tau_m} &= \frac{1}{(J_m + J_l)s + B_m} && (8.33)\\ &= \frac{1/B_m}{(\sigma_1)} && (8.34) \end{aligned}$$

where $\sigma_1 = B_m/(Jm + J_l)$. The real pole of the compliant system at $s = -\sigma$ is different from $-\sigma_1$. Numerical investigation shows that for for high stiffness and low damping σ will be very close to but slightly more negative than σ_1.

8.1.5 Single-axis computed torque control

As a prelude to trialing visual servo control of the waist axis, a single-axis computed-torque velocity and position controller was implemented. Position control is necessary

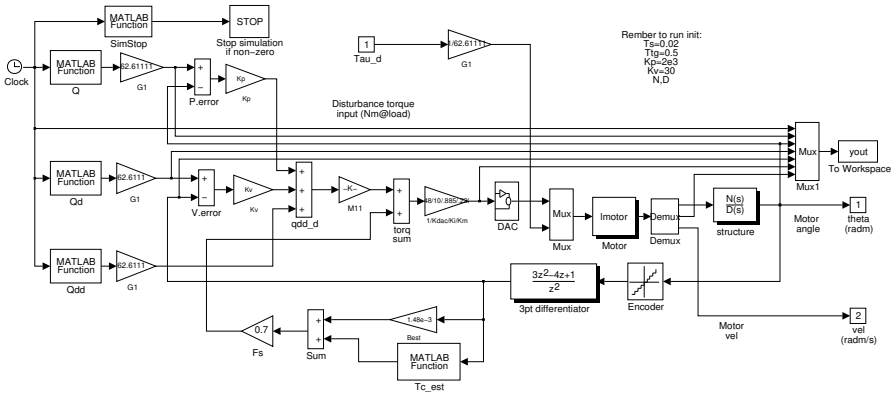


Figure 8.10: SIMULINK model CTORQUEJ1: the joint 1 computed torque controller.

for that phase of motion where the target is not visible and the polynomial trajectory generator is providing axis setpoints. Velocity control will be used when the target is visible. Such an axis controller can run at an arbitrary sample rate, eliminating the multi-rate problems discussed earlier. Also, by bypassing the Unimate velocity loop, it is possible to achieve higher speed joint motion.

A SIMULINK block diagram of the controller is shown in Figure 8.10. It is a straightforward implementation of (2.84) where $\mathbf{G}$ and $\mathbf{C}$ are zero and $\mathbf{M}$ is a scalar constant, resulting in

$$\tau_d = \mathbf{M}\left\{\ddot{\theta}_d + K_p(\theta_d - \theta) + K_v(\dot{\theta}_d - \hat{\dot{\theta}})\right\} + \mathbf{F}(\hat{\dot{\theta}}) \tag{8.35}$$

where $\theta = \theta_1$ in this case. Friction feedforward, $\mathbf{F}(\hat{\dot{\theta}})$ is based on the measured axis friction data shown in Table 2.12. A 3-point derivative (7.37) is used to estimate velocity for friction compensation and velocity feedback.

Based on experience with the joint 5 and 6 velocity loops it seemed reasonable to use the 20 ms visual sample interval to close the axis velocity and position loops. However it quickly became clear when setting the control gains that it was not possible to meet the objectives of tight trajectory following and stability. The fundamental difference between this axis and joints 5 and 6 is the resonance previously discussed. When the resonance (8.18) was introduced into the SIMULINK model it exhibited similar behaviour to the real robot. Paul [199] comments briefly on this issue and suggests that the sampling rate be at least 15 times the frequency of the highest structural resonance. From Figure 8.7 the highest frequency resonant peak in the range measured is approximately 45 Hz, which would indicate a desired sampling rate of 675 Hz. SIMULINK simulation of the controller and resonance indicated that good

performance could be achieved at a sample rate of 250 Hz, with a noticeable improvement at 500 Hz. The discontinuous nature of processes such as stick/slip and Coulomb friction could be considered as having harmonics up to very high frequencies and is further support for the requirement of a high sample rate.

In practice the sample rate is limited by the processor used for control. The single-axis computed-torque function (8.35) executes in less than 400μs including axis velocity estimation and servo communications overhead, but many other processes, particularly those concerned with vision system servicing, need to run in a timely manner. Another consideration in sampling is to avoid *beating* problems when the 50 Hz vision based position-loop is closed around the axis-velocity controller. For these reasons a sample rate of 2 ms was selected, generated from a hardware counter driven by the system's pixel clock. The computed-torque controller thus runs at exactly 10 times the visual sampling rate.

Friction compensation would ideally make the arm appear to be frictionless if the position and velocity feedback were disabled. In practice, by pushing the arm, it can be seen that this ideal is approximated but there are some problems. There is a tendency for over-compensation so that, once set moving, the arm accelerates slightly and glides away. This clearly indicates a mismatch in friction parameters and may be due to frictional dependence on pose, load or temperature. The effect of stiction is also very noticeable since friction compensation provides no torque at zero velocity. In this implementation an overall friction compensation scaling term is introduced and in practice is set to 80%. When pushing on the arm it is necessary to 'break' stiction, that is, provide sufficient torque to cause the joint to move, at which point friction compensation comes into effect.

Friction compensation of this form noticeably improves performance during high-speed motion. However initial experiments at low speed resulted in extremely poor motion quality with pronounced stick/slip and oscillatory behaviour. At low speed, $|\dot{\theta}_d| < \dot{\theta}_{min}$, friction compensation is based on the demanded rather than estimated joint velocity as discussed in Section 2.5.2. In practice the value of $\dot{\theta}_{min}$ was found to be critical to low-speed performance and that an appropriate setting is given by

$$\dot{\theta}_{min} > 2\Delta\omega \tag{8.36}$$

where $\Delta\omega$ is the estimated velocity 'quanta' given by (7.36) and is a function of sampling interval.

Figure 8.11 shows the measured response of the axis controller to a step velocity demand. The velocity estimate, as used in the online controller, is extremely noisy, which imposes an upper limit on the velocity feedback gain, K_v. A single-pole digital filter

$$\frac{\hat{\Omega}'}{\hat{\Omega}} = \frac{(1-\lambda)z}{z-\lambda} \tag{8.37}$$

is used to create a smoothed velocity estimate, $\hat{\dot{\theta}}'$. In practice a value of $\lambda = 0.85$ is

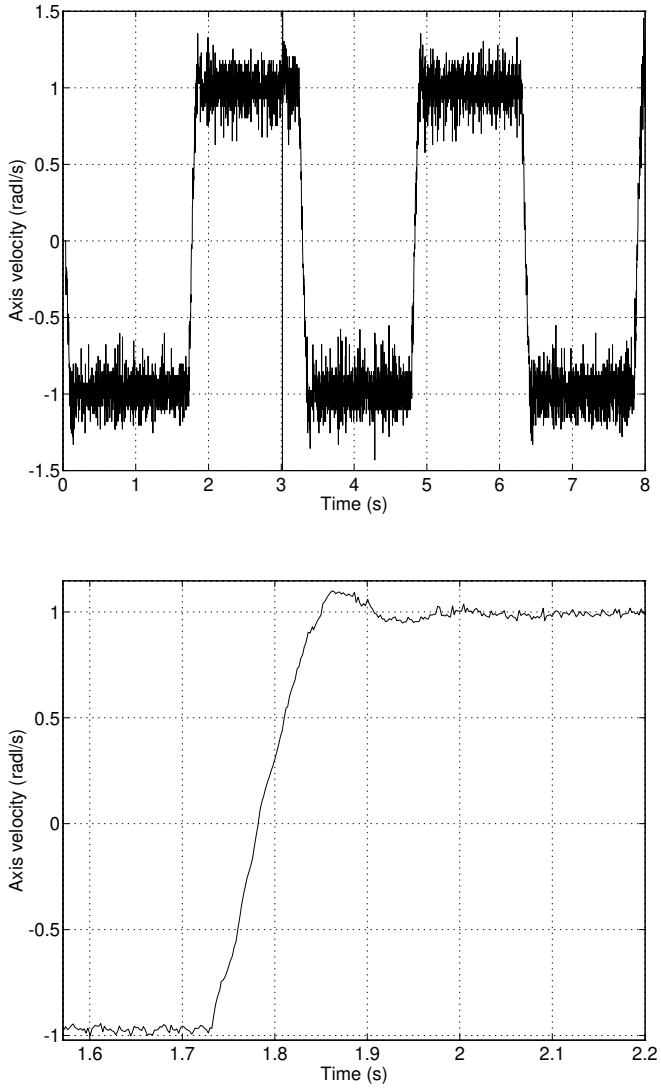


Figure 8.11: Measured axis velocity step response of single-axis computed-torque control. Demand is ± 1 radl/s. Joint angle was recorded by the RTVL system at 2 ms intervals, and velocity was estimated using the same 3-point numerical differentiation as performed in the online controller. The bottom plot is a more detailed view of the velocity estimate after it has been 'cleaned up' off-line by the same single-pole digital filter used online.

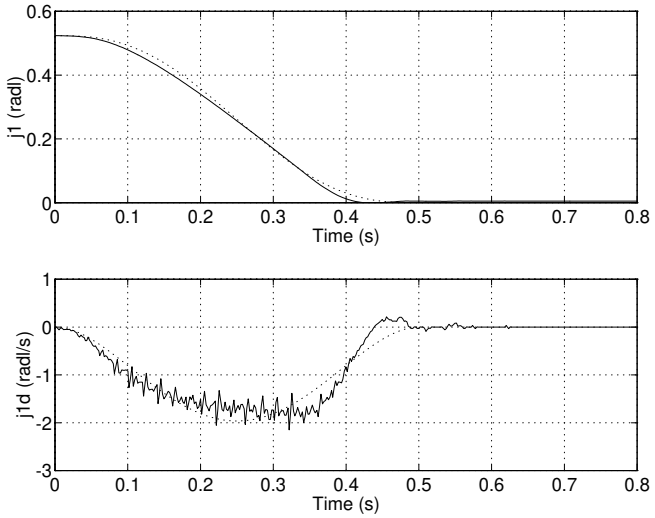


Figure 8.12: Measured axis response of single-axis computed-torque control showing axis position and velocity: measured (solid) and demand (dotted). Joint angle was recorded by the RTVL system at 2 ms intervals, and an off-line procedure using 5-point numerical differentiation was used to obtain the 'measured' velocity data. The axis has overshot and stopped 0.005 radl short of the target.

used, giving a filter bandwidth of 13 Hz. The bottom plot in Figure 8.11 is a filtered version from which it can be seen that there is very slight overshoot and a rise time of around 0.1 s leading to a bandwidth estimate of approximately 5 Hz.

Figure 8.12 shows the performance of the computed-torque controller for the standard 0.5 s swing trajectory. The computed torque controller (8.35) and the trajectory generator (8.1), (8.7) and (8.8) are executed every 2 ms. The control gains have been set empirically so as to achieve good trajectory following with minimum overshoot. There is again evidence of velocity saturation but at a higher speed than for the Unimate's native velocity controller. The final value of the joint angle is less than the demand since the axis has overshot and stopped due to stiction. For over-damped motion, stiction would stop the robot short of its target. Increased proportional gain, K_p, would reduce this error but is limited in practice by stability criteria. Increasing both K_p and K_v leads to rough motion due to noise on the velocity estimate. Integral action was experimented with, but it was difficult to achieve both stability and accurate target settling.

Compared to the native Unimate position and velocity loops this controller has

Characteristic	Unimate	Computed-torque
sample rate	position loop at 1 ms interval	position loop at 2 ms interval
velocity estimation	analog synthesis from encoder signals with considerable low frequency gain boost due to low-pass filter	digital estimation from encoder signals at 2 ms interval
integral action	analog implementation	digital implementation at 2 ms interval

Table 8.2: Comparison of implementational differences between the native Unimate controller and computed-torque controller implemented.

inferior low-speed performance, particularly in terms of achieving the target position. The significant implementational differences between the two controllers are summarized in Table 8.2. A number of papers [162, 247] have examined the performance of Puma computed-torque controllers but the performance metric used is always high-speed trajectory following error. Apart from noting this issue it will not be pursued further since the experiment is concerned primarily with high-speed motion.

The few narrow spikes on the velocity estimate are due to timing jitter, that is, non-uniform time steps in sampling the axis position. They are present even during motion with no velocity feedback. Statistics show the standard deviation in sample timing is generally 0.1 ms or 5% of the sample interval. In some instances the timing jitter is more severe due to other activity under the real-time operating system such as interrupt level processing. First attempts at control using this high sample rate yielded velocity plots with a marked oscillation at approximately 75 Hz. Investigation of this phenomenon showed it to be due to interaction between the 2 ms computed-torque process requesting the axis position and the 984 μs process in the Unimate axis servo board which updates the internal 16-bit software encoder register from the hardware encoder counter. This was overcome by modifying the Unimate axis firmware so that when in current control mode it will return the instantaneous encoder value, not that obtained at the last servo clock tick.

8.1.6 Vision based control

This section introduces a hybrid visual control strategy capable of stopping the robot directly over a randomly-placed target. The first phase of the motion is under the control of the trajectory generator as previously. Once the target comes into view the centroid data from the end-mounted camera is used to bring the robot to a stop over the target. The trajectory generator and computed torque control law are executed

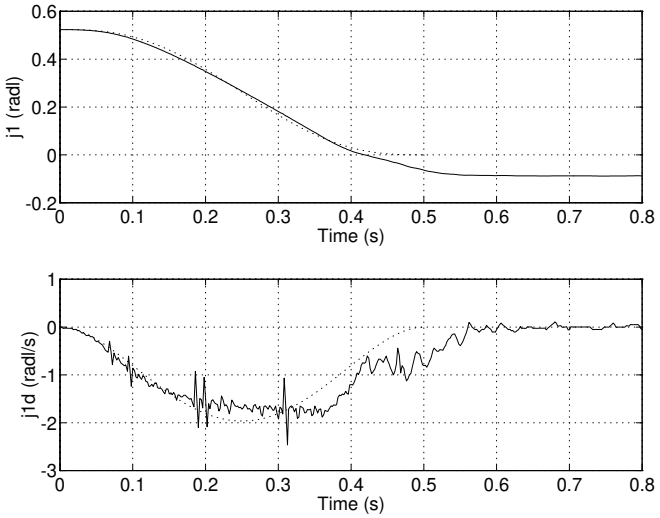


Figure 8.13: Measured axis response under hybrid visual control showing position and velocity: measured (solid) and trajectory-generator demand (dotted). Note that the axis position has settled at a negative value in order to move the manipulator over the target. The system switched to visual servoing at approximately $t = 0.4$ s.

at a 2 ms interval as in the previous section, until the target comes into view. Under visual control the centroid and centroid velocity are estimated every 20 ms and used to compute the axis velocity demand

$$\dot{\theta}_d = {}^iK_p({}^iX_d - {}^iX) + {}^iK_v{}^i\hat{\dot{X}} \tag{8.38}$$

using a simple PD control law. The computed torque control is still executed at a 2 ms interval but the axis position error and trajectory acceleration feedforward terms are dropped giving

$$\tau_d = \mathbf{M}\left\{K_v(\dot{\theta}_d - \hat{\dot{\theta}})\right\} + \mathbf{F}(\hat{\dot{\theta}}) \tag{8.39}$$

where K_v has the same value as previously. The gains for the visual servo loop, ${}^iK_p$ and ${}^iK_v$, were set empirically and it was possible to achieve good steady-state error performance with acceptable overshoot. Relatively high gains were possible due to the effectiveness of the computed-torque velocity loop. Once again the high level of friction was a problem at low speed, and attempts to achieve critical damping led to the robot stopping short of the target. The addition of an integral term to (8.38) is

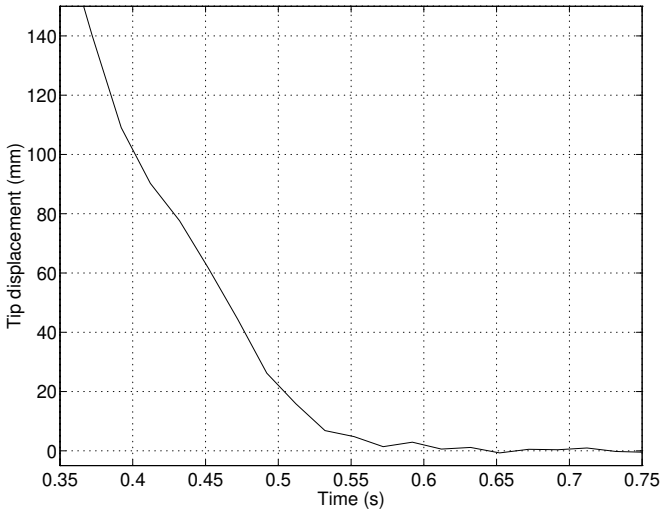


Figure 8.14: Measured tip displacement determined from end-effector mounted camera for hybrid visual control strategy.

helpful in the longer term in bringing the robot to the destination, but high levels of integral gain again lead to oscillatory behaviour.

Figure 8.13 shows a typical trajectory for a target displaced approximately 20 mm from the destination of the trajectory generator. The effect of this displacement can be seen in the final value of the joint angle which is -0.0884 radl rather than the trajectory generator demand of 0 radl. The target comes into view at 0.32 s and the transition in control strategies occurred 4 field times later at approximately 0.40 s. The transition is smooth, helped by the axis being velocity-limited at that point. Figure 8.14 shows the end-point error sensed by the camera scaled into length units. The end-point has followed an almost critically damped trajectory and by 0.60 s the end-point has settled to within 1 pixel or 0.78 mm of the target whose position was unknown.

8.1.7 Discussion

This section has provided a preliminary investigation into some of the issues involved in the control, using vision and axis feedback, of a realistic robot axis. The axis chosen has a number of serious non-idealities that include resonance and high levels of stiction, viscous and Coulomb friction. A high-bandwidth velocity loop was implemented using a single-axis computed-torque controller with velocity estimated from

measured axis position. Although the wrist axes could be velocity-controlled at 50 Hz, this axis has a number of resonances up to 45 Hz which seriously limit the achievable closed-loop performance. The dynamics of the stick-slip and Coulomb friction effect also have very short time constants requiring a high sample rate. The digital velocity loop was run at 500 Hz and provided high-quality velocity control for the overlying trajectory generator or visual feedback loop.

A challenging trajectory was chosen at the performance limit of the robot. The Unimate axis controller, with its lower velocity capability, was unable to follow the trajectory and accumulated increasing position error. As the axis approached the destination it decelerated very rapidly, thereby exciting structural resonances in the arm. These resonances could be detected by an accelerometer mounted on the camera and also by their effect on the motor's transfer function. This latter effect caused difficulty with a 50 Hz digital velocity loop and the sample rate had to be raised to 500 Hz. The computed-torque controller was able to achieve a more rapid response with similar overshoot, but was more prone to stiction and currently has poor low-speed motion capability. The hybrid controller, using visual feedback for final positioning, was able to reliably position the robot to within 1 pixel of the centroid of a randomly placed target.

Interestingly the visual loop, despite its low sample rate, was seemingly unaffected by the structural resonance. This is partly due to the low displacement amplitude of the oscillation. In the example of Section 8.1.3 the peak amplitude is only 1 mm or approximately 1.3 pixels despite the significant axis acceleration step. Close inspection of the visually sensed tip displacement in Figure 8.5 shows no evidence of this oscillation, which will be masked by the motion blur effect discussed in Section 3.5.6.

Visual servoing is likely to be most useful in controlling manipulators that are either very rigid or, more interestingly, less stiff (and hence cheaper) with vibrations occurring well below the Nyquist frequency and thus amenable to visual closed-loop endpoint control. The latter is quite feasible, with the bandwidth achievable limited primarily by the sample rate of the visual sensor. Cannon [42] observes that control of the manipulator's end-point using a non-colocated sensor complicates control design, but can result in superior performance. Unfortunately the Puma robot available for this work is more difficult to control, since significant resonances exist on either side of the Nyquist frequency, even though the robot pose was selected in order to minimize those frequencies.

8.2 High-performance 3D translational visual servoing

This section describes an experiment in high-performance translational visual servoing. Three robot DOF are controlled by image features so as to keep the camera at a constant height vertically above a target rotating on a turntable. A plan view is shown in Figure 8.15. The controller will use visual feedforward of target velocity

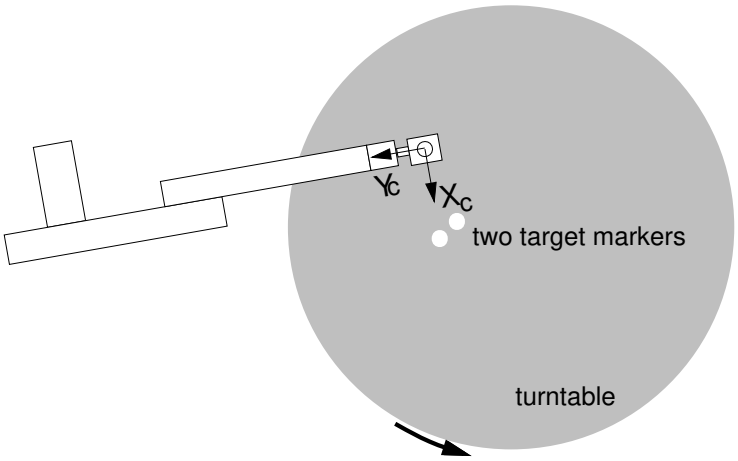


Figure 8.15: Plan view of the experimental setup for translational visual servoing.

and computed-torque axis velocity-control loops since axis interaction torques are expected to be significant at the anticipated joint speeds and accelerations.

8.2.1 Visual control strategy

In order to control 3 robot DOF we require 3 'pieces' of information from the image. As in previous examples the centroid gives an X and Y target displacement with respect to the camera and the extra information required is target distance. From (3.66), for the X-axis, we can write

$$ {}^{c}\hat{x}_t = \frac{{}^{c}z - f}{\alpha_x f} \left({}^{i}X - X_0 \right) \tag{8.40} $$

which gives the estimated target position relative to the camera in terms of the image plane coordinate and target distance. A similar expression can be written for the Y-axis. The camera coordinate frame can be determined from measured joint angles and forward kinematics

$$ {}^{0}\mathbf{T}_c = \mathrm{K}\left(\underline{\theta}\right) {}^{t_6}\mathbf{T}_c \tag{8.41} $$

which provides the camera pose in world coordinates allowing the target position in world coordinates to be estimated

$$ {}^{0}\underline{\hat{x}}_t = {}^{0}\mathbf{T}_c \, {}^{c}\underline{x}_t \tag{8.42} $$

where ${}^{c}\underline{x}_t = [{}^{c}x_t, {}^{c}y_t, {}^{c}z_t]$.

The remaining problem is estimation of target distance, ${}^{c}z_t$, which is required to control the robot's Z-axis Cartesian motion and is also required in (8.40). Using a monocular view the approaches to distance estimation are rather limited. *Stadimetry* is a simple technique based on the apparent size of the object but is sensitive to changes in lighting level and threshold as discussed in Section 4.1.3.4. A more robust approach is based on the distance between the centroids of two features since centroid was previously shown to be robust with respect to lighting and threshold. Consider a scene with two circular markers where the X and Y image plane components of the centroid difference are

$$ {}^{i}\Delta_X = {}^{i}X_1 - {}^{i}X_2, \quad {}^{i}\Delta_Y = {}^{i}Y_1 - {}^{i}Y_2 \tag{8.43} $$

The X and Y axes have different pixel scale factors, see Table 3.8, so these displacements must be scaled to length units

$$ \Delta_x = \frac{{}^{c}z_t\,{}^{i}\Delta_X}{\alpha_x f}, \quad \Delta_y = \frac{{}^{c}z_t\,{}^{i}\Delta_Y}{\alpha_y f} \tag{8.44} $$

The distance between the centers of the markers

$$ \Delta = \sqrt{\Delta_x^2 + \Delta_y^2} \tag{8.45} $$

is known, allowing an expression for range to be written

$$ {}^{c}\hat{z}_t = \frac{f\Delta}{\sqrt{\left(\frac{{}^{i}\Delta_X^2}{\alpha_x}\right)^2 + \left(\frac{{}^{i}\Delta_Y^2}{\alpha_x}\right)^2}} \tag{8.46} $$

The centroid used for fixation purposes is the mean centroid of the two circle features. The target position estimates in the world frame are given by (8.42) and the X and Y components are input to tracking filters in order to estimate the target's world-frame Cartesian velocity, ${}^{0}\hat{\dot{\underline{x}}}_t$. This provides the velocity feedforward component of the robot's motion. The feedback Cartesian velocity component is derived from image-plane centroid error and estimated target range error

$$ {}^{c}\dot{x}_e = {}^{i}K_p({}^{i}X_d - {}^{i}X) + {}^{i}K_v\,{}^{i}\hat{\dot{X}} \tag{8.47} $$

$$ {}^{c}\dot{y}_e = {}^{i}K_p({}^{i}Y_d - {}^{i}Y) + {}^{i}K_v\,{}^{i}\hat{\dot{Y}} \tag{8.48} $$

$$ {}^{c}\dot{z}_e = K_z({}^{c}\hat{z}_t - {}^{c}z_d) \tag{8.49} $$

As shown in Figure 8.16 the two velocity components are summed in the wrist reference frame

$$ {}^{t6}\dot{\underline{x}}_d = \underbrace{{}^{t6}\mathbf{J}_0\,{}^{0}\hat{\dot{\underline{x}}}_t}_{\text{feedforward}} + \underbrace{{}^{t6}\mathbf{J}_c\,{}^{c}\dot{\underline{x}}_e}_{\text{feedback}} \tag{8.50} $$

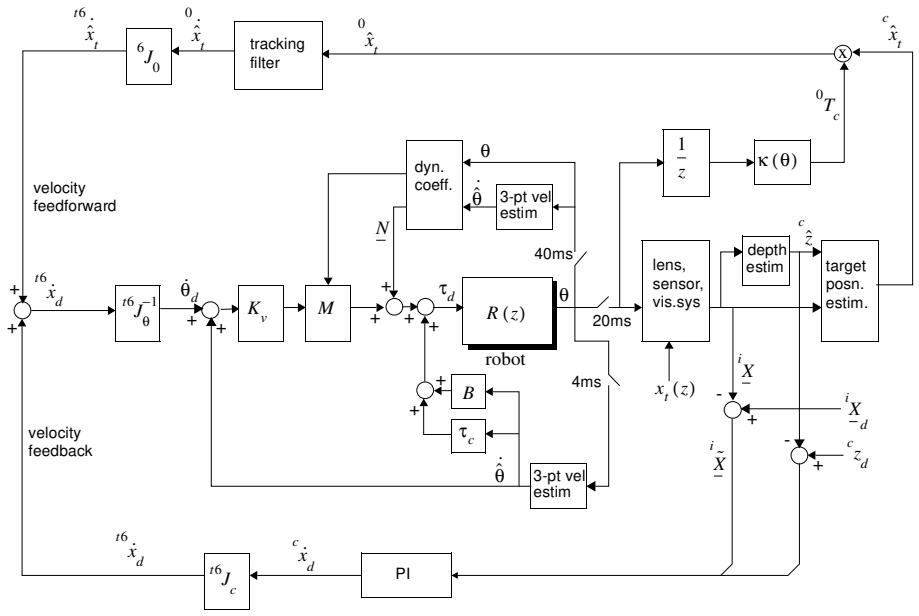


Figure 8.16: Block diagram of translational control structure.

The camera-mount Jacobian, ${}^{t6}\mathbf{J}_c$, is a constant and is determined (from ${}^{t6}\mathbf{T}_c$ given earlier in (4.73)) using (2.10) to be

$$ {}^{t6}\mathbf{J}_c = {}^{c}\mathbf{J}_{t6}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \tag{8.51} $$

and involves only transpositions and sign changing. The other Jacobian, ${}^{t6}\mathbf{J}_0$, is configuration dependent and determined online using (2.10) and the current estimate of ${}^{0}\mathbf{T}_{t6}$.

In the final step the total Cartesian velocity demand is resolved to joint velocity by the manipulator Jacobian

$$ \dot{\underline{\theta}}_d = {}^{t6}\mathbf{J}_\theta(\underline{\theta})^{-1\,t6}\dot{\underline{x}}_d \tag{8.52} $$

to provide the axis velocity demand. The inverse manipulator Jacobian is computed using the method of Paul and Zhang [202]. All Jacobians in the above equations are 3×3.

8.2.2 Axis velocity control

High-performance axis velocity control will be required for the robot's base axes, and from experience in the previous section a high sample rate will be required. Computed-torque control will be used to compensate for rigid-body dynamic effects which are expected to be significant in this experiment. For online dynamics computation it is desirable to divide the computation into two components, one executed at the axis servo rate and another, generally coefficient computation, executed at a lower rate. There are two motivations for this. The first, which is widely mentioned [149, 201, 228], is that this reduces the amount of computation that need be done at the high sample rate and thus significantly lowers the burden on the processor. Secondly, Sharkey et al. [228] suggest that it may be counter-productive to compute joint torques at a high-rate based on a dynamic model that does not take into account higher-order manipulator dynamic effects. The RNE procedure described earlier in Section 2.2.2 computes torque as a function of $\underline{\theta}$, $\dot{\underline{\theta}}$ and $\ddot{\underline{\theta}}$ and suffers the objections just mentioned when computed at the high sample rate.

In reports of controllers using this dual-rate approach [149, 228] the coefficient matrices, **M**, **C**, and **G** are computed at a low rate while the torque equation (2.11) is executed at the higher sample rate. However this raises the question of how to calculate these coefficient matrices. Symbolic algebra approaches could be used but the run-time computational cost may be significant since the efficient factorization of the RNE procedure cannot be exploited. Armstrong et al. [20] provide closed-form expressions for the coefficient matrices derived by symbolic analysis and significance-based simplification; however the kinematic conventions are different to those used in this work. Instead, the approach taken was to begin with the symbolic sum-of-product torque expressions developed earlier and apply the culling procedure of Section 2.6.3 to automatically eliminate terms below 1% significance. The expressions were then symbolically partitioned

$$\underline{\tau} = \mathbf{M}(\underline{\theta})\ddot{\underline{\theta}} + \mathbf{N}(\underline{\theta}, \dot{\underline{\theta}}) \tag{8.53}$$

where **M** is the 3×3 manipulator inertia matrix and **N** is a 3-vector comprising lumped gravity, centripetal and Coriolis torques. Given coefficients **M** and **N** equation (8.53) can be evaluated with only 9 multiplications and 9 additions resulting in a very low computational burden of only $81\,\mu$s. The coefficients **M** and **N**, updated at 25 Hz, are computed by a MAPLE generated 'C' function that takes 1.5 ms to execute. The low rate of coefficient update is not a problem in this situation since the pose cannot change significantly over the interval, and axis velocity can be assumed piecewise constant since the demand is only updated at the video sample rate of 50 Hz.

Evaluating the coefficients from the expanded torque equation is computationally less efficient than the RNE algorithm. However this approach allows the computation to be partitioned into a 'lightweight' component for execution at a high servo rate and a moderately more expensive component to execute at a lower sample rate. The

computational burden can be expressed as the fraction

$$\frac{\text{execution time}}{\text{period}}$$

Computing the 3-axis RNE (symbolically simplified) at 4 ms (using timing data from Table 2.25) results in a burden of

$$\frac{780}{4000} = 20\%$$

while the dual-rate approach has two, relatively small, components

$$\frac{81}{4000} + \frac{1500}{40000} = 6\%$$

totalling less than one third the burden of the straightforward RNE. The dual-rate approach also satisfies the problem of unwanted coupling of unmodelled manipulator dynamics.

The velocity controller, evaluated at the 4 ms period, uses simple proportional gain

$$\underline{\tau}_d = \mathbf{M}\mathbf{K}_v(\dot{\underline{\theta}}_d - \hat{\underline{\theta}}) + \mathbf{N} + \mathbf{F}(\hat{\underline{\theta}}) \tag{8.54}$$

and friction compensation as described in the previous section. **M** and **N** are updated at a 40 ms interval.

8.2.3 Implementation details

To implement this control strategy under VxWorks and RTVL a number of concurrent communicating tasks, shown in Figure 8.17, are used:

`field` is a high-priority task that samples the joint angles into the shared variable `j6_vis` at the beginning of vertical blanking. It then activates the `centroid`, `viskine` and `dynpar` tasks.

`viskine` is a 20 ms periodic task that computes the camera coordinate frame ${}^{o}\mathbf{T}_{c}$ and the inverse manipulator Jacobian ${}^{t_6}\mathbf{J}_{\theta}$ from current joint angles `j6_vis`.

`centroid` is a 20 ms periodic task that processes region data, estimates target range by (8.46), Cartesian velocity by (8.47) - (8.50), which are then resolved to joint rate demands and written to `thd_d`.

`torque` is a 4 ms periodic task that performs the computed-torque axis velocity loop calculations for joints 1 to 3.

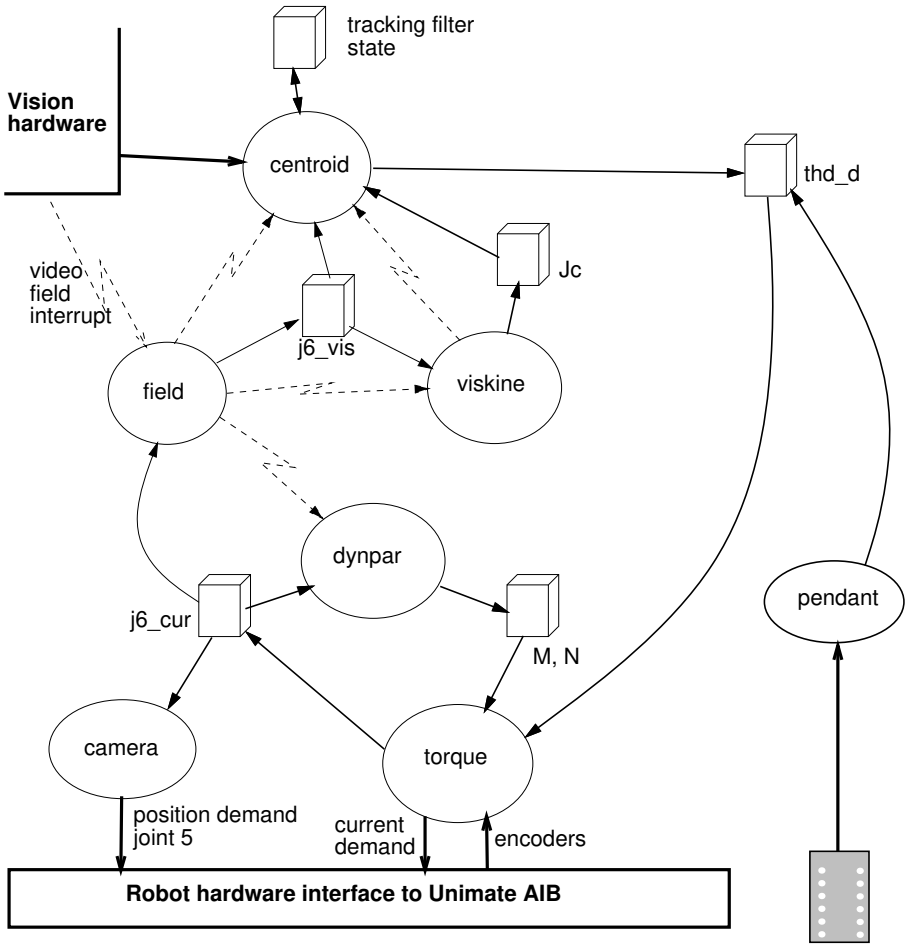


Figure 8.17: Task structure for translation control. Arrows represent data flow, lighting bolts are events, ellipses are tasks and '3D' boxes are datastructures.

`dynpar` is a 40 ms periodic task that updates the rigid-body coefficients **M** and **N** based on current joint angles and velocity estimated over the last three time steps using a 3-point derivative.

`camera` is a 7 ms periodic task that is responsible for keeping the camera's optical axis normal to the motion plane. Motion of the lower three joints changes the orientation of the camera which this task counters by appropriate wrist motion.

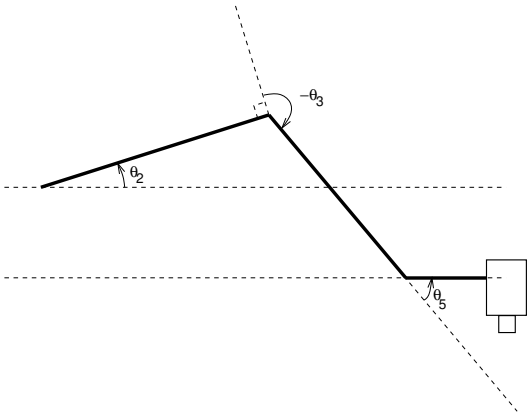


Figure 8.18: Camera orientation geometry. The camera can be maintained normal to the working surface if $\theta_5 = -(\theta_2 + \theta_3 + \pi)$.

Function	Period	Execution time	
	μs	μs	% of CPU
`torque`	4000	1844	46
`viskine`	20,000	3800	19
`dynpar`	40,000	1500	4
`centroid`	20,000	1300	7
Total			76

Table 8.3: Summary of task execution times. These are average elapsed times for execution of code segments and no allowance has been made for the effects of task preemption during execution.

From the simple geometry depicted in Figure 8.18 it is clear that compensation can be achieved by motion of a single wrist axis

$$\theta_5 = -(\theta_2 + \theta_3 + \pi) \tag{8.55}$$

`pendant` is a low-priority continuous task that communicates with the robot teach pendant using the ARCL pendant communication primitives. Several modes of operation can be selected by the teach pendant and include visual fixation control, manual joint velocity control and manual Cartesian velocity control.

Execution times of the various modules are summarized in Table 8.3 in absolute terms and as a fraction of total computing burden. The torque task consumes

the largest fraction, but a good deal of that time is taken up with communications overhead: 680μs to read the three encoders, and 510μs to transmit the motor current demands. Velocity estimation, computed torque computation, friction compensation and current clipping take only 220μs/axis. These computations account for 76% of the computational resource, and additional processing is required for data logging, user interface, graphics and networking. Clearly the processor, in this case a single 33 MHz 68030, is close to its limit.

8.2.4 Results and discussion

The tracking performance of the controller for the standard target motion is shown in Figure 8.19. The error magnitude in the Y direction is similar to that displayed by the 2-DOF pan/tilt controller of Section 7.5. The centroid error is significantly greater in the X direction than the Y direction and this is believed to be due to the poor dynamic performance of the waist axis. Performance of that axis is limited by significant friction and the relatively low velocity gain needed to ensure stability. Figure 8.20 shows the results of the same experiment but with centripetal and Coriolis feedforward disabled. Clearly the Y-axis tracking performance is substantially degraded with only partial compensation of manipulator rigid-body dynamic effects. Disabling target velocity feedforward results in very poor performance, and it is not possible to keep the target in the camera's field of view even at 20% of the velocity demonstrated here. The manipulator joint rates shown in Figure 8.21 peak at approximately half the fundamental maximum joint rates established in Table 2.21. At the peak velocity there is some tendency toward oscillation, particularly for joints 1 and 2.

The logged joint angle trajectory data was transformed off-line by the manipulator forward kinematics to determine the camera's Cartesian trajectory which is shown in Figure 8.22. The camera's height, ${}^{c}z$, is not constant and has a peak-to-peak amplitude of approximately 40 mm. This performance is due to the relatively low gain proportional control strategy (8.49) used for this DOF. The Cartesian camera velocity is shown in Figure 8.23 along with the online estimated target velocity. The velocity feedforward signal is a good estimate of the actual target velocity. The peak tip speed is 350 mm/s and this is approximately one third of the manufacturer's specified maximum. The robot appeared to be 'working hard' and with considerable gear noise.

The Cartesian path of the camera, shown in Figure 8.24, is only approximately a circle and two factors contribute to this distortion. Firstly, non-ideal tracking performance causes the camera to deviate from the path of the target. Secondly, the camera is not maintained in an orientation normal to the XY plane. The contribution of these two effects can be understood by looking at the online target position estimates. These are computed from target centroid and joint angle data and make no assumptions about tracking performance. Figure 8.25 shows this estimated target path which is clearly distorted like the path actually followed by the camera. Similar plots,

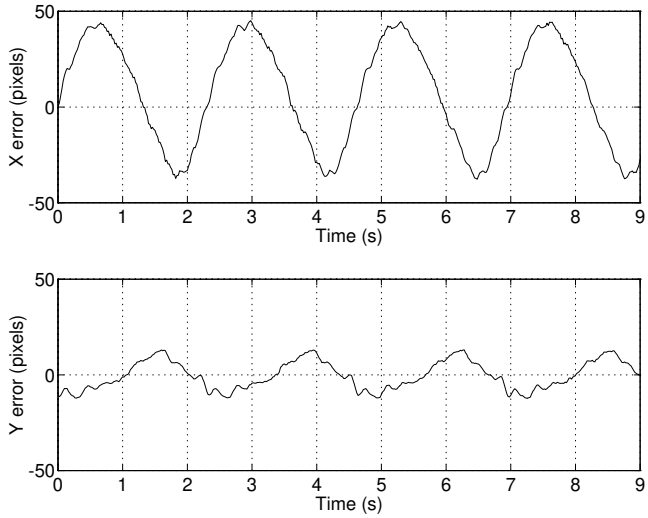


Figure 8.19: Measured centroid error for translational visual servo control with estimated target velocity feedforward. RMS pixel error is 28 and 7.3 pixels for the X and Y directions respectively.

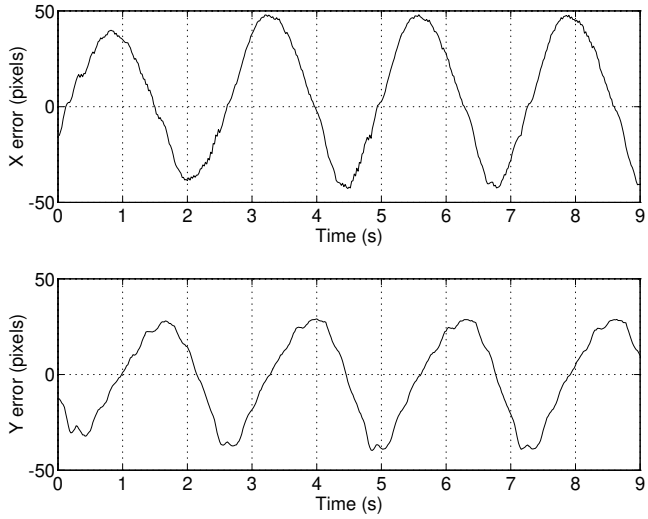


Figure 8.20: Measured centroid error for translational visual servo control with estimated target velocity feedforward, but without centripetal and Coriolis feedforward. RMS pixel error is 30 and 22 pixels for the X and Y directions respectively.

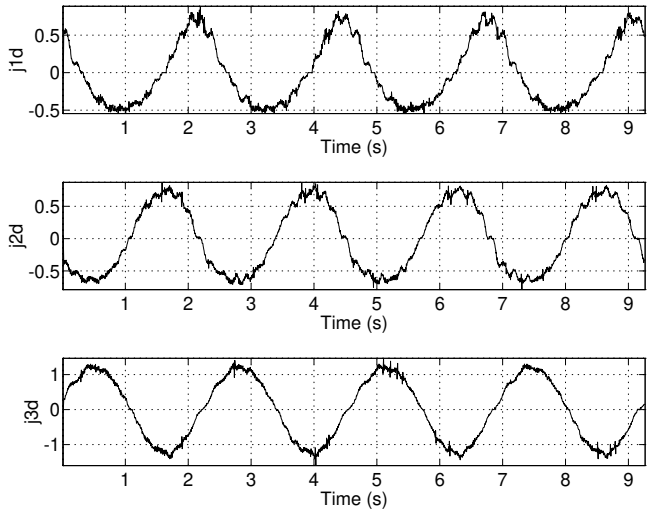


Figure 8.21: Measured joint rates (radl/s) for translational visual servo control with estimated target velocity feedforward. Velocity is estimated off-line using a 5-point numerical derivative of joint angles measured by RTVL.

but for significantly lower camera speed, are almost ideal circles. At the joint speeds used in this experiment the wrist axes, under control of the Unimate position loops, are unable to accurately track the wrist orientation demand (8.55). From the recorded data it appears that the actual θ_5 lags the demand by approximately 21 ms resulting in peak-to-peak orientation errors of 0.043 radl and this is verified using the SIMULINK model of Figure 2.26. In addition, dynamic forces acting on the camera result in small rotations of joints 4 and 6. These seemingly small orientation errors, coupled with the nominal 500 mm target distance, result in considerable translational errors.

It is interesting to contrast the control structure just described with some other structures described in the literature. Operational space control, proposed by Khatib [149], would appear to be an ideal candidate for such an application since a camera is an operational (or task) space sensor. In fact an example of vision based control using the operational space formulation has been recently described by Woodfill et al. [284]. Operational space control transforms the robot and sensor data into the operational space where degrees of freedom can be selected as position or force controlled and desired manipulator end-point forces are computed. Finally these forces are transformed to joint space and actuate the manipulator. As expressed in Khatib [149] the

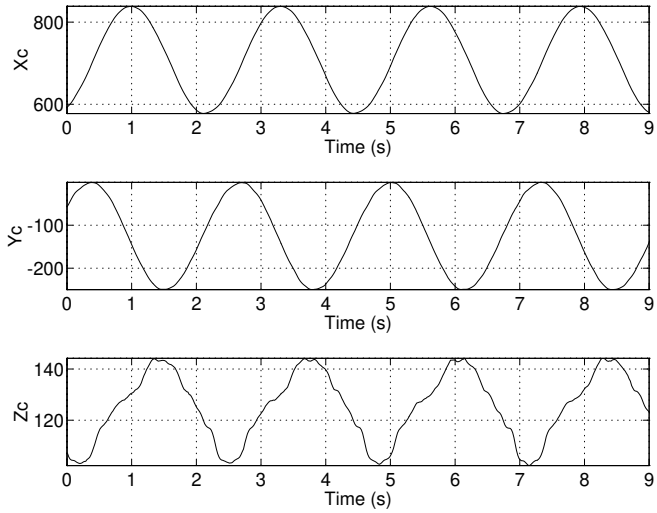


Figure 8.22: Measured camera Cartesian position (mm) for translational visual servo control with estimated target velocity feedforward. Cartesian path data is estimated off-line from joint angles (measured by RTVL) using forward manipulator kinematics.

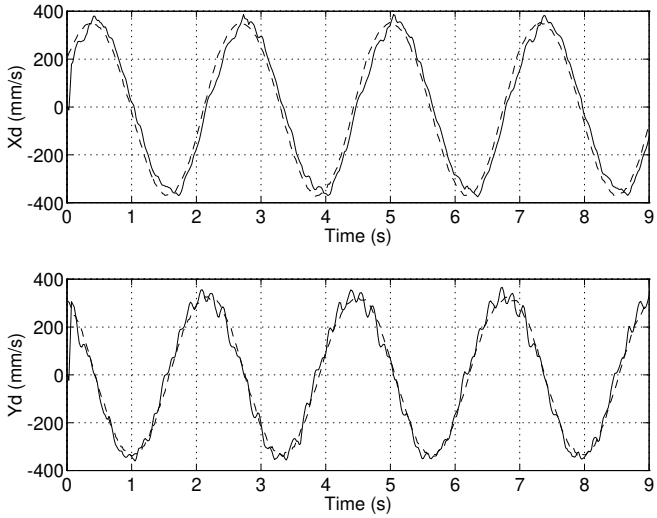


Figure 8.23: Measured camera Cartesian velocity (solid) with estimated target velocity feedforward (dashed). Camera velocity is estimated off-line using a 5-point numerical derivative of the Cartesian position data of Figure 8.22.

manipulator torque loop is closed at a rate of 200 Hz which is likely to be dictated by computational limits. If a machine vision sensor is used then the torque loop would be closed at a maximum of 60 Hz (assuming RS170 video) which has been shown to be too low to achieve high-performance from the non-ideal Puma axes. The structure proposed in this section is hierarchical and more appropriate for the case of a low sample rate sensor such as a machine vision system.

8.3 Conclusion

This chapter has extended and brought together many of the principles established in earlier chapters by means of two examples: 1-DOF visual control of a major robot axis, and 3-DOF translational camera control. The simple 2-DOF control of the previous chapter effectively decoupled the dynamics of each axis, with one actuator per camera DOF. In the 3-DOF translational case there is considerable kinematic and dynamic coupling between the axes involved. To achieve high-performance tracking a multi-rate hierarchical control system was developed. Target velocity feedforward was found to be essential for the accurate tracking at high speed shown in Section 8.2. Feedforward of manipulator rigid-body dynamics, that is computed-torque control, has also been shown to increase the tracking accuracy. The end-point control experiment of Section 8.1 did not use visual feedforward, but in that case the target was stationary and the manipulator was decelerating.

The controller computational hardware, which has served well over a period of many years, is now at the limit of its capability. Working in such a situation is increasingly difficult and unproductive and precludes the investigation of more sophisticated control structures. The act of enabling data logging for the experiment of Section 8.2 now causes a visible degradation in performance due to increased latencies and violations of designed timing relationships. The experimental facility has however demonstrated that sophisticated control techniques incorporating machine vision, rigid-body dynamics and friction compensation can be achieved using only a single, modest, processor.

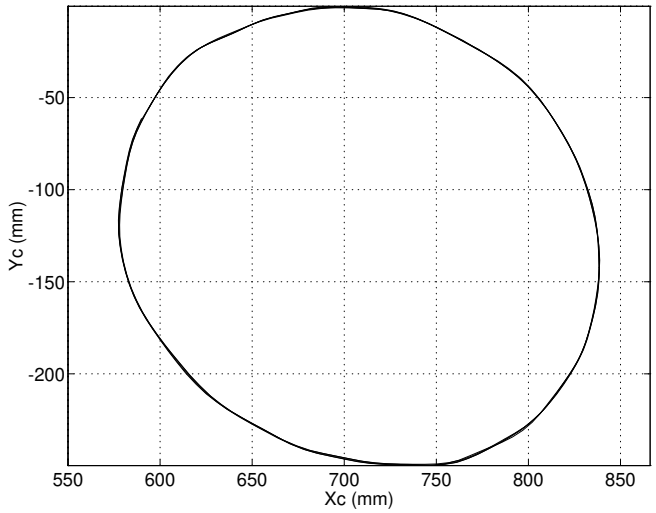


Figure 8.24: Measured camera Cartesian path in the XY plane. Cartesian path data as per Figure 8.22.

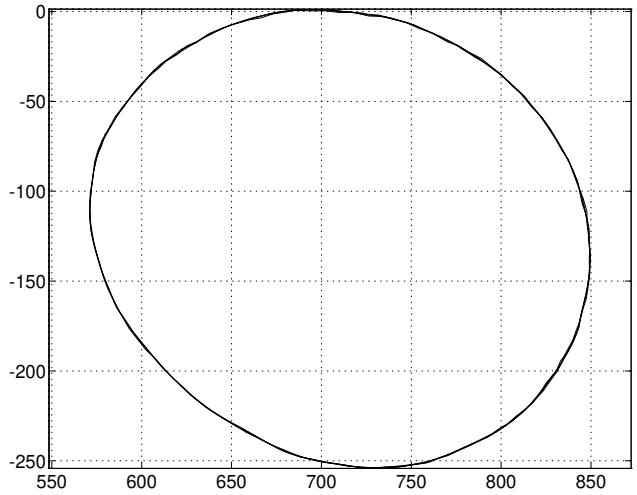


Figure 8.25: Measured camera Cartesian path estimate from online tracking filter recorded by RTVL.

Chapter 9

Discussion and future directions

9.1 Discussion

This book has presented for the first time a detailed investigation of the many facets of a robotic visual servoing system with particular emphasis on dynamic characteristics and high-performance motion control. The use of machine vision for high-performance motion control is a significant challenge due to the innate characteristics of the vision sensor which include relatively low sample rate, latency and coarse quantization.

A distinction has been established between visual servo *kinematics* and visual servo *dynamics*. The former is well addressed in the literature and is concerned with how the manipulator should move in response to perceived visual features. The latter is concerned with dynamic effects due to the manipulator and machine vision sensor which must be explicitly addressed in order to achieve high-performance control. This problem is generic to all visually controlled machines no matter what approach is taken to feature extraction or solving the visual kinematic problem.

Weiss proposed a robot control scheme that entirely does away with axis sensors — dynamics and kinematics are controlled adaptively based on visual feature data. This concept has a certain appeal but in practice is overly complex to implement and appears to lack robustness. The concepts have only ever been demonstrated in simulation for up to 3-DOF and then with simplistic models of axis dynamics which ignore 'real world' effects such as Coulomb friction and stiction. However the usefulness of such a control approach is open to question. It is likely that robots will always have axis position and/or velocity sensors since not all motion will be, or can be, visually guided. Ignoring these sensors adds greatly to the control complexity and has been shown to lead to inferior control performance. Structural resonances and the time constants of stick-slip friction, ignored by Weiss, dictate a sample interval of less than

4 ms, which is not currently achievable with off-the-shelf video or image processing components. If constrained by the vision system to low sample rates, high-bandwidth axis level feedback is required for high-performance and robust control. Axis position and velocity information are also essential for model-based dynamic control and Jacobian computation.

In order to overcome low visual sample rates some researchers have proposed the use of trajectory generators that continuously modify their goal based on visual input. In this book the task has instead been considered as a 'steering problem', controlling the robot's velocity so as to guide it toward the goal. Conceptually this leads to a visual position loop closed around axis velocity loops. This structure is analogous to the common robot control structure, except that position is sensed directly in task space rather than joint space.

The term 'high performance' has been widely used in this book and was defined at the outset as robot motion which approaches or exceeds the performance limits stated by the manufacturer. The limits for the robot used were established in Chapter 2. There is also an implied fidelity criterion which was defined later in terms of pixel error for tracking applications.

The performance achieved is a consequence of the detailed understanding of the dynamics of the system to be controlled (the robot) and the sensor (the camera and vision system). Despite the long history of research in these areas individually, and combined in visual servoing, it is apparent that much of the data required for modelling is incomplete and spread through a very diverse literature. This book has attempted to draw together this disparate information and present it in a systematic and consistent manner.

A number of different control structures have been demonstrated in experiments and simulation. Feedback-only controllers were shown to be capable of high performance but were found to be rather 'brittle' with respect to actuator saturation. Their design was also found to be seriously constrained in order to achieve compensator stability which has been shown to be important. Feedforward increases the design degrees of freedom and a control strategy based on estimated target velocity feedforward was introduced. With no *a priori* knowledge of target motion the controller demonstrates sufficient performance to fixate on an object rotating on a turntable or a ping-pong ball thrown in front of the robot.

The key conclusions from this work are that in order to achieve high-performance visual servoing it is necessary to minimize open-loop latency, have an accurate dynamic model of the system and to employ a feedforward type control strategy. Prediction can be used to overcome latency but at the expense of reduced high frequency disturbance rejection. Open-loop latency is reduced by choice of a suitable control architecture. An accurate dynamic model is required for control synthesis. Feedback-only controllers have a loop gain limit due to the significant delay in pixel transport and processing. Simple feedback controllers have significant phase lag characteristics

which lead to poor tracking. More sophisticated feedback controllers can overcome this but the solution space becomes very constrained and the controllers are not robust with respect to plant parameter variation. Feedforward control results in a robust controller with excellent tracking capability.

9.2 Visual servoing: some questions (and answers)

This section poses a number of pertinent questions that were posed at the outset of the original PhD research program. They are worth repeating here, along with answers in terms of material covered in this book, since they provide a succinct encapsulation of the major conclusions of this research.

1. *Can machine vision be used to control robot manipulators for dynamically challenging tasks, by providing end-point relative sensing?* Machine vision and end-point sensing can be used for dynamically challenging tasks if the general principles summarized above are observed. The limits to visual servo dynamic performance have been investigated and suitable control structures have been explored by analysis, simulation and experiment. Commonly used performance measures such as step response have been shown to be inadequate for the general target tracking problem. Alternative measures, based on image plane error, have been proposed to enable more meaningful comparison of results.

2. *What are the limiting factors in image acquisition and processing?* These issues were addressed principally in Chapter 3. Image feature extraction, by specialized hardware or even software, is now easily capable of 50 or 60 Hz operation, and the fundamental limit to visual servo performance is now the sensor frame rate. It has been shown that the requirements for ideal visual sampling and reducing motion blur require short exposure times, and consequently considerable scene illumination, which may be a problem in some applications due to the heat generated. At higher frame rates, and in order to maintain the ideal sampler approximation, even shorter exposure times would be required, necessitating increased illumination or more sensitive sensors. A camera-mounted pulsed-LED lighting system has been demonstrated that provides light when and where required.

   More robust scene interpretation is definitely required if visually servoed systems are to move out of environments lined with black velvet. Optical flow based approaches show promise, and have been demonstrated at 60 Hz with specialized processing hardware.

   Maintaining adequate focus is often considered important for visual servoing and is difficult to achieve given variation in relative target distance. The lens system may be configured for large depth of field but this further exacerbates the

lighting problems just mentioned. The upper bound on image clarity has been shown to be determined by effects such as lens aberration and to a lesser extent pixel re-sampling. It has also been shown that binary image processing in the presence of edge gradients results in feature width estimates that are functions of threshold and overall illumination, but that centroid estimates are unbiased. For image processing approaches based on edges or texture, for instance optical flow or interest operators, image clarity would be important.

3. *How can the information provided by machine vision be used to determine the pose of objects with respect to the robot?* This is what has been termed in this book the *kinematics of visual control* and a number of techniques were reviewed in Chapter 4. This problem is solved for all practical purposes, and the computational costs of the various proposed approaches, image based and position based, are comparable and readily achieved. Control of a 6-DOF robot was demonstrated by Ganapathy [98] a decade ago using a closed-form solution that executed in only 4 ms on a 68000 processor. An iterative approach has even been patented [289].

   The image-based technique has been demonstrated to work well, but the advantages seem illusory, and the problem of image feature Jacobian update in the general case remains, although adaptation and general learning schemes have been demonstrated. The 3-D scene interpretation problem can be eased significantly by using 3-D sensors. Sensors based on structured lighting are now compact and fast enough to use for visual servoing.

4. *Can robot tasks be specified in terms of what is seen?* For a large number of robotic tasks the answer is clearly 'yes' but this book has provided only partial answers to the question. A number of papers reviewed in Chapter 4 have discussed how tasks can be described in terms of image features and desired trajectories of those features but most of the systems described are single purpose. There appears to have been little work on languages for general specification of visually servoed tasks. In a semi-automated situation such as time delay teleoperation an operator may be able to select image features and indicate their desired configuration and have the execution performed under remote closed-loop visual control. There are some reports on this topic [198].

5. *What effects do robot electro-mechanical and machine vision dynamics have on closed-loop dynamic performance?* The vision system was modelled in detail in Section 6.4 where it was shown that the dominant characteristics were delay due to pixel transport and gain due to lens perspective. The effective delay was shown to depend on the camera exposure period, and for some visual servoing configurations the gain was shown to depend on target distance. Both characteristics can have significant effect on closed-loop system stability.

In Section 6.4 the electro-mechanical dynamics could be represented by a unit delay due to the action of a position control loop. When the axes are velocity controlled the dynamics are first-order and the effective viscous damping is strongly dependent on motion amplitude due to the effect of Coulomb friction. For visual servo control of a major axis, as described in Chapter 8, additional dynamic effects such as structural resonance and significant stiction are encountered.

Structural dynamics were discussed in Section 8.1.4 and are important when the manipulator has significant structural or transmission compliance, and high endpoint acceleration is required. Visual servoing, by providing direct endpoint position feedback, can be used to control endpoint oscillation, Section 8.1. This may one day be a viable alternative to the common approach of increasing link stiffness which then necessitates higher torque actuators.

6. *What control architecture is best suited for such an application?* Many aspects of the control architecture employed in this work are very well suited to this task since it provides the prerequisites identified in Section 6.1, in particular:

   - a high-frame-rate low-latency vision system which reduces latency to the minimum possible by overlapping pixel transport with region extraction;
   - a high bandwidth communications path between the vision system and robot controller achieved by a shared backplane.

   The control architecture has evolved as shortcomings were identified, and an obvious weakness in the early work was added latency due to servo setpoint double handling and the multi-rate control structure. In later work this was eliminated by implementing custom axis velocity loops which were also capable of higher speed. The greatest bottleneck in the architecture described in this book is now the communications link to the Unimate servos and the digital servo boards themselves, particularly when implementing high sample rate axis control. A more appropriate architecture may involve direct motor control by a dedicated motor control card, perhaps another VMEbus processor, and dispensing with the Unimate digital servo board. A different robot may also eliminate some of these difficulties.

   The control problem has considerable inherent parallelism: axis control, visual feature processing, and application program execution. It seems likely that the control task can be partitioned into a number of cooperating parallel processes each of which have relatively modest computational and communications requirements. These processes could also be geographically distributed into the motors and cameras, ultimately leading to an implementation comprising a network of communicating 'smart' entities. This offers little advantage for a small

machine such as a Puma, but has significant benefits for control of large machines in applications such as, for example, mining.

9.3 Future work

There are many interesting topics related to visual servoing that remain to be investigated. These include:

1. The Puma robot has limited the control performance in this work, primarily because of friction. It would be interesting to investigate the performance that could be achieved using a direct drive robot or some of the robotic head and eye devices that have been recently reported [157].

2. This book has alluded to the limiting nature of cameras that conform to common video standards. The image feature extraction hardware used in this work [25] is capable of processing over 20 Mpixels/s. In conjunction with lower resolution images, say 256×256 pixels, this would allow at least 300 frames/s giving a visual Nyquist frequency of 150 Hz. Cameras with such capability do exist, for instance from Dalsa in Canada, but they are extremely bulky and weigh almost 1 kg. However the market for 'digital cameras' is expanding and the nascent standards such as that proposed by AIA will expand the range of cameras and video formats. A more difficult path would be to develop a camera from a CCD sensor chip.

   Such high visual sample rates would allow for visual control of structural modes and would then provide a viable approach to high-quality endpoint control. Such rates may also allow axis-level velocity feedback to be dispensed with, but such feedback is relatively inexpensive in terms of hardware requirements and has been shown to provide significant advantage.

3. The issue of languages and operator interfaces for visual description of tasks has not received much attention in the literature and would seem to offer considerable scope for investigation. Potentially an operator could describe a task by indicating visual features on a screen. The robot system would then bring a tool to the indicated location and perform the task.

4. Clearly more robust scene interpretation is required if the systems are to move out of environments lined with black velvet. There has been some recent work based on greyscale feature tracking and optical flow at sample rates in the range 25 to 60 Hz. In many cases the latency, due to pipelining, is several sample intervals which is poor from a control point of view. Research into low-cost dedicated architectures with high sample rate and low latency is needed to address this issue.

Bibliography

[1] P. Adsit. *Real-Time Intelligent Control of a Vision-Servoed Fruit-Picking Robot*. PhD thesis, University of Florida, 1989.

[2] J. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images — a review. *Proc. IEEE*, 76(8):917–935, August 1988.

[3] G. Agin. Calibration and use of a light stripe range sensor mounted on the hand of a robot. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 680–685, 1985.

[4] R. Ahluwalia and L. Fogwell. A modular approach to visual servoing. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 943–950, 1986.

[5] J. S. Albus. *Brains, behavior and robotics*. Byte Books, 1981.

[6] B. Alexander and K. Ng. Sub-pixel accuracy image feature location via the centroid. In A. Maeder and B. Jenkins, editors, *Proc. Digital Image Computing: Techniques and Applications*, pages 228–236, Melbourne, December 1991. Australian Pattern Recognition Soc.

[7] P. K. Allen, A. Timcenko, B. Yoshimi, and P. Michelman. Real-time visual servoing. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1850–1856, 1992.

[8] P. K. Allen, B. Yoshimi, and A. Timcenko. Real-time visual servoing. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 851–856, 1991.

[9] J. Aloimonos, I. Weiss, and A. Badyopadhyay. Active vision. *Int. J. Computer Vision*, 1:333–356, January 1988.

[10] C. H. An, C. G. Atkeson, J. D. Griffiths, and J. M. Hollerbach. Experimental evaluation of feedforward and computed torque control. *IEEE Trans. Robot. Autom.*, 5(3):368–373, June 1989.

[11] C. H. An, C. G. Atkeson, and J. M. Hollerbach. Experimental determination of the effect of feedforward control on trajectory tracking errors. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 55–60, 1986.

[12] G. B. Andeen, editor. *Robot Design Handbook*. McGraw-Hill, 1988.

[13] N. Andersen, O. Ravn, and A. Sørensen. Real-time vision based control of servomechanical systems. In *Proc. 2nd International Symposium on Experimental Robotics*, Toulouse, France, June 1991.

[14] C. H. Anderson, P. J. Burt, and G. S. van der Wal. Change detection and tracking using pyramid transform techniques. In *Proceeding of SPIE*, volume 579, pages 72–78, Cambridge, Mass., September 1985. SPIE.

[15] H. L. Anderson, R. Bajcsy, and M. Mintz. Adaptive image segmentation. Technical Report MS-CIS-88-26, GRASP Lab, University of Pennsylvania, April 1988.

[16] R. L. Andersson. Real-time gray-scale video processing using a moment-generating chip. *IEEE Trans. Robot. Autom.*, RA-1(2):79–85, June 1985.

[17] R. Andersson. *Real Time Expert System to Control a Robot Ping-Pong Player*. PhD thesis, University of Pennsylvania, June 1987.

[18] R. Andersson. Computer architectures for robot control: a comparison and a new processor delivering 20 real Mflops. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1162–1167, May 1989.

[19] R. Andersson. A low-latency 60Hz stereo vision system for real-time visual control. *Proc. 5th Int.Symp. on Intelligent Control*, pages 165–170, 1990.

[20] B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the Puma 560 arm. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 1, pages 510–18, Washington, USA, 1986.

[21] B. Armstrong. Friction: Experimental determination, modeling and compensation. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1422–1427, 1988.

[22] B. Armstrong. *Dynamics for Robot Control: Friction Modelling and Ensuring Excitation During Parameter Identification*. PhD thesis, Stanford University, 1988.

[23] W. Armstrong. Recursive solution to the equations of motion of an n-link manipulator. In *Proc. 5th World Congress on Theory of Machines and Mechanisms*, pages 1343–1346, Montreal, July 1979.

[24] K. J. Åström and B. Wittenmark. *Computer Controlled Systems: Theory and Design*. Prentice Hall, 1984.

[25] Atlantek Microsystems, Technology Park, Adelaide. *APA-512+ Area Parameter Accelerator Hardware Manual*, April 1993.

[26] R. Bajcsy. Active perception. *Proc. IEEE*, 76(8):996–1005, August 1988.

[27] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, 1982.

[28] B. Batchelor, D. Hill, and D. Hodgson, editors. *Automated visual inspection*. IFS, 1985.

[29] A. Bejczy. Robot arm dynamics and control. Technical Report NASA-CR-136935, NASA JPL, February 1974.

[30] P. Bélanger. Estimation of angular velocity and acceleration form shaft encoder measurements. Technical Report TR-CIM-91-1, Mc Gill University, 1991.

[31] T. R. Benedict and G. W. Bordner. Synthesis of an optimal set of radar track-while-scan smoothing equations. *IRE Transactions on Automatic Control*, pages 27–32, 1962.

[32] R. Berg. Estimation and prediction for maneuvering target trajectories. *IEEE Trans. Automatic Control*, AC-28(3):294–304, March 1983.

[33] P. Besl. Active, optical range imaging sensors. *Machine Vision and Applications*, 1:127–152, 1988.

[34] R. C. Bolles. Verification vision for programmable assembly. In *Proc 5th International Joint Conference on Artificial Intelligence*, pages 569–575, Cambridge, MA, 1977.

[35] R. Bolles and R. Cain. Recognizing and locating partially visible objects: the local-feature-focus method. *Int. J. Robot. Res.*, 1(3):57–82, 1982.

[36] M. Bowman and A. Forrest. Visual detection of differential movement: Applications to robotics. *Robotica*, 6:7–12, 1988.

[37] C. Brown. Gaze controls with interactions and delays. *IEEE Trans. Syst. Man Cybern.*, 20(1):518–527, 1990.

[38] R. Brown, S. Schneider, and M. Mulligan. Analysis of algorithms for velocity estimation from discrete position versus time data. *IEEE Trans. Industrial Electronics*, 39(1):11–19, February 1992.

[39] R. Bukowski, L. Haynes, Z. Geng, N. Coleman, A. Santucci, K. Lam, A. Paz, R. May, and M. DeVito. Robot hand-eye coordination rapid prototyping environment. In *Proc. ISIR*, pages 16.15–16.28, October 1991.

[40] J. Burdick. An algorithm for generation of efficient manipulation dynamic equations. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 212–218, 1986.

[41] G. Buttazzo, B. Allotta, and F. Fanizza. Mousebuster: a robot system for catching fast moving objects by vision. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 932–937, 1993.

[42] R. H. Cannon and E. Schmitz. Initial experiments on the end-point control of a flexible one-link robot. *Int. J. Robot. Res.*, 3(3):62–75, Fall 1984.

[43] C. Canudas De Wit. Experimental results on adaptive friction compensation in robot manipulators: Low velocities. In V. Hayward and O. Khatib, editors, *Experimental Robotics 1*, pages 196–214. Springer Verlag, 1989.

[44] C. Canudas De Wit and N. Fixot. Robot control via robust estimated state feedback. *IEEE Trans. Autom. Control*, 36(12):1497–1501, December 1991.

[45] D. Carmer and L. Peterson. Laser radar in robotics. *Proc. IEEE*, 84(2):299–320, February 1996.

[46] G. Cesareo, F. Nicolo, and S. Nicosia. DYMIR: a code for generating dynamic model of robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 115–120, 1984.

[47] B. Char et al. *Maple V language reference manual*. Springer-Verlag, 1991.

[48] F. Chaumette, P. Rives, and B. Espiau. Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2248–2253, 1991.

[49] W. F. Clocksin, J. S. E. Bromley, P. G. Davey, A. R. Vidler, and C. G. Morgan. An implementation of model-based visual feedback for robot arc welding of thin sheet steel. *Int. J. Robot. Res.*, 4(1):13–26, Spring 1985.

[50] Computer Group Microcomputer Division, Motorola Inc., 2900 South Diablo Way, Tempe, AZ 85282. *MVME147 MPU VMEmodule: MVME712/MVME712M Transition Module: User's Manual*, April 1988.

[51] D. Coombs and C. Brown. Cooperative gaze holding in binocular vision. *IEEE Control Systems Magazine*, 11(4):24–33, June 1991.

[52] P. I. Corke. *High-Performance Visual Closed-Loop Robot Control*. PhD thesis, University of Melbourne, Dept. Mechanical and Manufacturing Engineering, July 1994.

[53] P. I. Corke. An automated symbolic and numeric procedure for manipulator rigid-body dynamic significance analysis and simplification. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1018–1023, 1996.

[54] P. I. Corke and H. L. Anderson. Fast image segmentation. In *International Conference on Automation, Robotics and Computer Vision*, pages 988–992, Singapore, September 1990.

[55] P. Corke. ARCL application programmer's reference manual. MTM-180, CSIRO Division of Manufacturing Technology, February 1990.

[56] P. Corke. The Unimation Puma servo system exposed. MTM-226, CSIRO Division of Manufacturing Technology, 1991.

[57] P. Corke. An experimental facility for robotic visual servoing. In *Proc. IEEE Region 10 Int. Conf.*, pages 252–256, Melbourne, 1992.

[58] P. Corke. Experiments in high-performance robotic visual servoing. In *Proc. International Symposium on Experimental Robotics*, pages 194–200, Kyoto, October 1993.

[59] P. Corke. Visual control of robot manipulators — a review. In K. Hashimoto, editor, *Visual Servoing*, volume 7 of *Robotics and Automated Systems*, pages 1–31. World Scientific, 1993.

[60] P. Corke and B. Armstrong-Hélouvry. A meta-study of PUMA 560 dynamics: A critical appraisal of literature data. *Robotica*, 13(3):253–258, 1995.

[61] P. Corke and B. Armstrong-Hélouvry. A search for consensus among model parameters reported for the PUMA 560 robot. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1608–1613, San Diego, May 1994.

[62] P. Corke and M. Good. Dynamic effects in high-performance visual servoing. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1838–1843, Nice, May 1992.

[63] P. Corke and M. Good. Controller design for high-performance visual servoing. In *Proc. IFAC 12th World Congress*, pages 9–395 to 9–398, Sydney, 1993.

[64] P. Corke and R. Kirkham. The ARCL robot programming system. In *Proc.Int.Conf. of Australian Robot Association*, pages 484–493, Brisbane, July 1993. Australian Robot Association, Mechanical Engineering Publications (London).

[65] P. Corke and R. Paul. Video-rate visual servoing for robots. In V. Hayward and O. Khatib, editors, *Experimental Robotics 1*, volume 139 of *Lecture Notes in Control and Information Sciences*, pages 429–451. Springer-Verlag, 1989.

[66] P. Corke and R. Paul. Video-rate visual servoing for robots. Technical Report MS-CIS-89-18, GRASP Lab, University of Pennsylvania, February 1989.

[67] P. Corke, M. Roberts, R. Kirkham, and M. Good. Force-controlled robotic deburring. In *Proc. 19th ISIR*, pages 817–828, Sydney, November 1988.

[68] P. Y. Coulon and M. Nougaret. Use of a TV camera system in closed-loop position control of mechnisms. In A. Pugh, editor, *International Trends in Manufacturing Technology ROBOT VISION*, pages 117–127. IFS Publications, 1983.

[69] J. J. Craig. *Introduction to Robotics*. Addison Wesley, second edition, 1989.

[70] R. Cunningham. Segmenting binary images. *Robotics Age*, pages 4–19, July 1981.

[71] M. L. Cyros. Datacube at the space shuttle's launch pad. *Datacube World Review*, 2(5):1–3, September 1988. Datacube Inc., 4 Dearborn Road, Peabody, MA.

[72] Data Translation, 100 Locke Drive, Marlborough, MA 01752-1192. *DT778 Series - User Manual*, 1984.

[73] Datacube Inc., 4 Dearborn Road, Peabody, MA. *DIGIMAX digitizer and display module*, 1988.

[74] Datacube Inc. *Installation and Software*, 1988.

[75] Datacube Inc., 4 Dearborn Road, Peabody, MA. *MAXbus specification*, SP00-5 edition, September 1988.

[76] E. Dickmanns and V. Graefe. Applications of dynamic monocular machine vision. *Machine Vision and Applications*, 1:241–261, 1988.

[77] E. Dickmanns and V. Graefe. Dynamic monocular machine vision. *Machine Vision and Applications*, 1:223–240, 1988.

[78] E. Dickmanns and F.-R. Schell. Autonomous landing of airplanes by dynamic machine vision. In *Proc. IEEE Workshop on Applications of Computer Vision*, pages 172–179. IEEE Comput. Soc. Press, November 1992.

[79] J. Dietrich, G. Hirzinger, B. Gombert, and J. Schott. On a unified concept for a new generation of light-weight robots. In V. Hayward and O. Khatib, editors, *Experimental Robotics 1*, volume 139 of *Lecture Notes in Control and Information Sciences*, pages 287–295. Springer-Verlag, 1989.

[80] R. Duda and P. Hart. *Pattern classification and scene analysis*. Wiley, 1973.

[81] K. A. Dzialo and R. J. Schalkoff. Control implications in tracking moving objects using time-varying perspective-projective imagery. *IEEE Trans. Ind. Electron.*, IE-33(3):247–253, August 1986.

[82] EG&G Reticon, 345 Potrero Ave., Sunyvale, CA. *Depth of Field Characteristics using Reticon Image Sensing Arrays and Cameras*. Application Note 127.

[83] EG&G Reticon, 345 Potrero Ave., Sunyvale, CA. *Solid State Camera Products*, 1991/2.

[84] J. Engelberger. *Robotics in Practice*. Kogan Page, 1980.

[85] S. D. Eppinger and W. P. Seering. Introduction to dynamic models for robotic force control. *IEEE Control Systems Magazine*, 7(2):48–52, 1987.

[86] M. Erlic and W.-S. Lu. Manipulator control with an exponentially stable velocity observer. In *Proc. ACC*, pages 1241–1242, 1992.

[87] W. Faig. Calibration of close-range photogrammetric systems: Mathematical formulation. *Photogrammetric Engineering and Remote Sensing*, 41(12):1479–1486, December 1975.

[88] H. Fässler, H. Beyer, and J. Wen. A robot pong pong player: optimized mechanics, high performance 3D vision, and intelligent sensor control. *Robotersysteme*, 6:161–170, 1990.

[89] R. Featherstone. *Robot Dynamics Algorithms*. Kluwer Academic Publishers, 1987.

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

[91] J. Feddema. *Real Time Visual Feedback Control for Hand-Eye Coordinated Robotic Systems*. PhD thesis, Purdue University, 1989.

[92] J. Feddema and O. Mitchell. Vision-guided servoing with feature-based trajectory generation. *IEEE Trans. Robot. Autom.*, 5(5):691–700, October 1989.

[93] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[94] J. P. Foith, C. Eisenbarth, E. Enderle, H. Geisselmann, H. Ringschauser, and G. Zimmermann. Real-time processing of binary images for industrial applications. In L. Bolc and Z. Kulpa, editors, *Digital Image Processing Systems*, pages 61–168. Springer-Verlag, Germany, 1981.

[95] G. Franklin and J. Powell. *Digital Control of dynamic systems*. Addison-Wesley, 1980.

[96] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics. Control, Sensing, Vision and Intelligence*. McGraw-Hill, 1987.

[97] S. Ganapathy. Camera location determination problem. Technical Memorandum 11358-841102-20-TM, AT&T Bell Laboratories, November 1984.

[98] S. Ganapathy. Real-time motion tracking using a single camera. Technical Memorandum 11358-841105-21-TM, AT&T Bell Laboratories, November 1984.

[99] C. Geschke. A robot task using visual tracking. *Robotics Today*, pages 39–43, Winter 1981.

[100] C. C. Geschke. A system for programming and controlling sensor-based robot manipulators. *IEEE Trans. Pattern Anal. Mach. Intell.*, PAMI-5(1):1–7, January 1983.

[101] A. Gilbert, M. Giles, G. Flachs, R. Rogers, and H. Yee. A real-time video tracking system. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2(1):47–56, January 1980.

[102] R. M. Goor. A new approach to minimum time robot control. Technical Report GMR-4869, General Motors Research Laboratories, November 1984.

[103] G. Hager, S. Hutchinson, and P. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, October 1996.

[104] G. Hager, W.-C. Chang, and A. Morse. Robot feedback control based on stereo vision: Towards calibration-free hand-eye coordination. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2850–2856, 1994.

[105] E. Hall. *Computer Image Processing and Recognition*. Academic Press, 1979.

[106] R. M. Haralick and L. G. Shapiro. Survey: Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, 29:100–132, 1985.

[107] R. Haralick and L. Shapiro. *Computer and Robot Vision*. Addison-Wesley, 1993.

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

[109] R. S. Hartenberg and J. Denavit. A kinematic notation for lower pair mechanisms based on matrices. *Journal of Applied Mechanics*, 77(2):215–221, June 1955.

[110] H. Hashimoto, T. Kubota, W.-C. Lo, and F. Harashima. A control scheme of visual servo control of robotic manipulators using artificial neural network. In *Proc. IEEE Int.Conf. Control and Applications*, pages TA–3–6, Jerusalem, 1989.

[111] K. Hashimoto, T. Kimoto, T. Ebine, and H. Kimura. Manipulator control with image-based visual servo. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2267–2272, 1991.

[112] K. Hashimoto and H. Kimura. A new parallel algorithm for inverse dynamics. *Int. J. Robot. Res.*, 8(1):63–76, February 1989.

[113] K. Hashimoto, K. Ohashi, and H. Kimura. An implementation of a parallel algorithm for real-time model-based control on a network of microprocessors. *Int. J. Robot. Res.*, 9(6):37–47, December 1990.

[114] M. Hatamian. A real-time two-dimensional moment generating algorithm and its single chip implementation. *IEEE Trans. Acoust. Speech Signal Process.*, 34(3):546–553, June 1986.

[115] V. Hayward and R. P. Paul. Robot manipulator control under UNIX — RCCL: a Robot Control C Library. *Int. J. Robot. Res.*, 5(4):94–111, 1986.

[116] J. Hill and W. T. Park. Real time control of a robot with a mobile camera. In *Proc. 9th ISIR*, pages 233–246, Washington, DC, March 1979.

[117] R. E. Hill. A new algorithm for modeling friction in dynamic mechanical systems. *TDA Progress Report 42-95*, pages 51–57, July 1988.

[118] C.-S. Ho. Precision of digital vision systems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 5(6):593–601, November 1983.

[119] J. M. Hollerbach. Dynamics. In M. Brady, J. M. Hollerbach, T. L. Johnson, T. Lozano-Perez, and M. T. Mason, editors, *Robot Motion - Planning and Control*, pages 51–71. MIT, 1982.

[120] J. Hollerbach. A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Trans. Syst. Man Cybern.*, SMC-10(11):730–736, November 1980.

[121] R. Hopwood. Design considerations for a solid-state image sensing system. In *Minicomputers and Microprocessors in Optical Systems*, volume 230, pages 178–188. SPIE, 1980.

[122] B. K. P. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[123] K. Hosoda and M. Asada. Versatile visual servoing without knowledge of true Jacobian. In *Proc. IROS*, September 1994.

[124] N. Houshangi. Control of a robotic manipulator to grasp a moving target using vision. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 604–609, 1990.

[125] M. K. Hu. Visual pattern recognition by moment invariants. *IRE Trans. Info. Theory*, 8:179–187, February 1962.

[126] E. Hudson. Ultra-fine pitch SMD automated assembly. *Industrial automation journal*, pages 17–22, January 1992. Singapore Industrial Automation Association.

[127] A. E. Hunt. Vision-based predictive robotic tracking of a moving target. Master's thesis, Dept. Electrical Engineering, Carnegie-Mellon University, Pittsburgh, PA, USA., January 1982.

[128] H. Inoue, T. Tachikawa, and M. Inaba. Robot vision server. In *Proc. 20th ISIR*, pages 195–202, 1989.

[129] H. Inoue, T. Tachikawa, and M. Inaba. Robot vision system with a correlation chip for real-time tracking, optical flow, and depth map generation. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1621–1626, 1992.

[130] A. Izaguirre, M. Hashimoto, R. Paul, and V. Hayward. A new computational structure for real time dynamics. Technical Report MS-CIS-87-107, University of Pennsylvania, December 1987.

[131] A. Izaguirre and R. Paul. Automatic generation of the dynamic equations of the robot manipulators using a LISP program. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 220–226, 1986.

[132] A. Izaguirre, P. Pu, and J. Summers. A new development in camera calibration: Calibrating a pair of mobile cameras. *Int. J. Robot. Res.*, 6(3):104–116, 1987.

[133] M. Jägersand, O. Fuentes, and R. Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *Proc. IEEE Int. Conf. Robotics and Automation*, page to appear, 1996.

[134] W. Jang and Z. Bien. Feature-based visual servoing of an eye-in-hand robot with improved tracking performance. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2254–2260, 1991.

[135] W. Jang, K. Kim, M. Chung, and Z. Bien. Concepts of augmented image space and transformed feature space for efficient visual servoing of an "eye-in-hand robot". *Robotica*, 9:203–212, 1991.

[136] R. A. Jarvis. A perspective on range finding techniques for computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):122–139, March 1983.

[137] M. Kabuka, J. Desoto, and J. Miranda. Robot vision tracking system. *IEEE Trans. Ind. Electron.*, 35(1):40–51, February 1988.

[138] M. Kabuka and R. Escoto. Real-time implementation of the Newton-Euler equations of motion on the NEC μPD77230 DSP. *Micro Magazine*, 9(1):66–76, February 1990.

[139] M. Kabuka, E. McVey, and P. Shironoshita. An adaptive approach to video tracking. *IEEE Trans. Robot. Autom.*, 4(2):228–236, April 1988.

[140] M. Kahn. The near-minimum time control of open-loop articulated kinematic linkages. Technical Report AIM-106, Stanford University, 1969.

[141] P. R. Kalata. The tracking index: a generalized parameter for $\alpha-\beta$ and $\alpha-\beta-\gamma$ target trackers. *IEEE Trans. Aerosp. Electron. Syst.*, AES-20(2):174–182, March 1984.

[142] R. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, pages 35–45, March 1960.

[143] T. Kane and D. Levinson. The use of Kane's dynamical equations in robotics. *Int. J. Robot. Res.*, 2(3):3–21, Fall 1983.

[144] H. Kasahara and S. Narita. Parallel processing of robot-arm control computation on a multimicroprocessor system. *IEEE Trans. Robot. Autom.*, 1(2):104–113, June 1985.

[145] H. Kazerooni and S. Kim. A new architecture for direct drive robots. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 442–445, 1988.

[146] T. Kenjo and S. Nagamori. *Permanent-Magnet and Brushless DC motors*. Clarendon Press, Oxford, 1985.

[147] E. Kent, M. Shneier, and R. Lumia. PIPE - Pipelined Image Processing Engine. *J. Parallel and Distributed Computing*, 2:50–7, December 1991.

[148] W. Khalil, M. Gautier, and J. F. Kleinfinger. Automatic generation of identification models of robots. *Proc. IEEE Int. Conf. Robotics and Automation*, 1(1):2–6, 1986.

[149] O. Khatib. A unified approach for motion and force control of robot manipulators: the operational space formulation. *IEEE Trans. Robot. Autom.*, 3(1):43–53, February 1987.

[150] P. K. Khosla and S. Ramos. A comparative analysis of the hardware requirements for the Lagrange-Euler and Newton-Euler dynamic formulations. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 291–296, 1988.

[151] P. Khosla. *Real-Time Control and Identification of Direct-Drive Manipulators*. PhD thesis, Carnegie-Mellon University, 1986.

[152] P. Khosla and T. Kanade. Experimental evaluation of the feedforward compensation and computed-torque control schemes. *Proc. American Control Conference*, pages 790–798, 1986.

[153] P. Khosla and T. Kanade. Real-time implementation and evaluation of computed-torque scheme. *IEEE Trans. Robot. Autom.*, 5(2):245–253, April 1989.

[154] P. Khosla, N. Papanikolopoulos, and B. Nelson. Dynamic sensor placement using controlled active vision. In *Proc. IFAC 12th World Congress*, pages 9.419–9.422, Sydney, 1993.

[155] R. D. Klafter, T. A. Chmielewski, and M. Negin. *Robotic Engineering - an Integrated Approach*. Prentice Hall, 1989.

[156] R. Kohler. A segmentation system based on thresholding. *Computer Graphics and Image Processing*, pages 319–338, 1981.

[157] E. Krotkov, C. Brown, and J. Crowley, editors. *Active Computer Vision*. IEEE, Nice, May 1992. Notes of IEEE Conf. Robotics and Automation workshop M5.

[158] M. Kuperstein. Generalized neural model for adaptive sensory-motor control of single postures. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 140–143, 1988.

[159] R. H. Lathrop. Parallelism in manipulator dynamics. *Int. J. Robot. Res.*, 4(2):80–102, Summer 1985.

[160] R. Lathrop. Constrained (closed-loop) robot simulation by local constraint propogation. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 689–694, 1986.

[161] M. B. Leahy, K. P. Valavanis, and G. N. Saridis. Evaluation of dynamic models for PUMA robot control. *IEEE Trans. Robot. Autom.*, 5(2):242–245, April 1989.

[162] M. Leahy. Experimental analysis of robot control: A performance standard for the Puma560. In *4th Int. Symp. Intelligent Control*, pages 257–264. IEEE, 1989.

[163] M. Leahy, V. Milholen, and R. Shipman. Robotic aircraft refueling: a concept demonstration. In *Proc. National Aerospace and Electronics Conf.*, pages 1145–50, May 1990.

[164] M. Leahy, L. Nugent, K. Valavanis, and G. Saridis. Efficient dynamics for PUMA-600. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 519–524, 1986.

[165] M. Leahy and G. Saridis. Compensation of industrial manipulator dynamics. *Int. J. Robot. Res.*, 8(4):73–84, 1989.

[166] C. S. G. Lee. Robot arm kinematics, dynamics and control. *IEEE Computer*, 15(12):62–80, December 1982.

[167] C. S. G. Lee, B. Lee, and R. Nigham. Development of the generalized D'Alembert equations of motion for mechanical manipulators. In *Proc. 22nd CDC*, pages 1205–1210, San Antonio, Texas, 1983.

[168] C. S. G. Lee, T. N. Mudge, and J. L. Turney. Hierarchical control structure using special purpose processors for the control of robot arms. In *Proc. Pattern Recognition, Image Processing Conf.*, pages 634–640, 1982.

[169] R. Lenz and R. Tsai. Techniques for calibration of the scale factor and image center for high accuracy 3-D machine vision metrology. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):713–720, September 1988.

[170] C. Lin and P. Kalata. The tracking index for continuous target trackers. In *Proc. ACC*, pages 837–841, 1992.

[171] Z. Lin, V. Zeman, and R. V. Patel. On-line robot trajectory planning for catching a moving object. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1726–1731, 1989.

[172] Y. L. C. Ling, P. Sadayappan, K. W. Olson, and D. E. Orin. A VLSI robotics vector processor for real-time control. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 303–308, 1988.

[173] M. Liu. Puma 560 robot arm analogue servo system parameter identification. Technical Report ASR-91-1, Dept. Mechanical and Manufacturing Engineering, University of Melbourne, February 1991.

[174] L. Ljung. *System Identification Toolbox*. The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760, 1988.

[175] J. Lloyd. Implementation of a robot control development environment. Master's thesis, Mc Gill University, December 1985.

[176] J. Y. S. Luh and C. S. Lin. Scheduling of parallel computation for a computer-controlled mechanical manipulator. *IEEE Trans. Syst. Man Cybern.*, 12(2):214–234, March 1982.

[177] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul. On-line computational scheme for mechanical manipulators. *ASME Journal of Dynamic Systems, Measurement and Control*, 102:69–76, 1980.

[178] J. Maciejowski. *Multivariable Feedback design*. Addison Wesley, 1989.

[179] A. G. Makhlin. Stability and sensitivity of servo vision systems. *Proc 5th Int Conf on Robot Vision and Sensory Controls - RoViSeC 5*, pages 79–89, October 1985.

[180] H. Martins, J. Birk, and R. Kelley. Camera models based on data from two calibration planes. *Computer Graphics and Image Processing*, 17:173–180, 1980.

[181] The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760. *Matlab User's Guide*, January 1990.

[182] The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760. *Simulink User's Guide*, 1993.

[183] B. Mel. *Connectionist Robot Motion Planning*. Academic Press, 1990.

[184] A. Melidy and A. A. Goldenberg. Operation of PUMA 560 without VAL. *Proc. Robots 9*, pages 18.61–18.79, June 1985.

[185] W. Miller. Sensor-based control of robotic manipulators using a general learning algorithm. *IEEE Trans. Robot. Autom.*, 3(2):157–165, April 1987.

[186] F. Miyazaki and S. Aritmoto. Sensory feedback for robot manipulators. *J. Robot. Syst.*, 2(1):53–71, 1985.

[187] J. Mochizuki, M. Takahashi, and S. Hata. Unpositioned workpieces handling robot with visual and force sensors. *IEEE Trans. Ind. Electron.*, 34(1):1–4, February 1987.

[188] Motorola Inc. *VMEbus Specification Manual*, June 1985.

[189] J. J. Murray. *Computational Robot Dynamics*. PhD thesis, Carnegie-Mellon University, 1984.

[190] P. V. Nagy. The PUMA 560 industrial robot: Inside-out. In *Robots 12*, pages 4.67–4.79, Detroit, June 1988. SME.

[191] S. Negahdaripour and J. Fox. Undersea optical stationkeeping: Improved methods. *J. Robot. Syst.*, 8(3):319–338, 1991.

[192] C. Neuman and J. Murray. Customized computational robot dynamics. *J. Robot. Syst.*, 4(4):503–526, 1987.

[193] R. Nevatia. Depth measurement by motion stereo. *Computer Graphics and Image Processing*, 5:203–214, 1976.

[194] R. Nigham and C. S. G. Lee. A multiprocessor-based controller for the control of mechanical manipulators. *IEEE Trans. Robot. Autom.*, 1(4):173–182, December 1985.

[195] D. Orin, R. McGhee, M. Vukobratovic, and G. Hartoch. Kinematics and kinetic analysis of open-chain linkages utilizing Newton-Euler methods. *Mathematical Biosciences. An International Journal*, 43(1/2):107–130, February 1979.

[196] N. Papanikolopoulos, P. Khosla, and T. Kanade. Adaptive robot visual tracking. In *Proc. American Control Conference*, pages 962–967, 1991.

[197] N. Papanikolopoulos, P. Khosla, and T. Kanade. Vision and control techniques for robotic visual tracking. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 857–864, 1991.

[198] N. Papanikolopoulos and P. Khosla. Shared and traded telerobotic visual control. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 878–885, 1992.

[199] R. P. Paul. *Robot Manipulators: Mathematics, Programming, and Control*. MIT Press, Cambridge, Massachusetts, 1981.

[200] R. P. Paul, B. Shimano, and G. E. Mayer. Kinematic control equations for simple manipulators. *IEEE Trans. Syst. Man Cybern.*, 11(6):449–455, June 1981.

[201] R. P. Paul and H. Zhang. Design of a robot force/motion server. In *Proc. IEEE Int. Conf. Robotics and Automation*, volume 3, pages 1878–83, Washington , USA, 1986.

[202] R. P. Paul and H. Zhang. Computationally efficient kinematics for manipulators with spherical wrists. *Int. J. Robot. Res.*, 5(2):32–44, 1986.

[203] R. Paul. Modelling, trajectory calculation and servoing of a computer controlled arm. Technical Report AIM-177, Stanford University, Artificial Intelligence Laboratory, 1972.

[204] R. Paul, M. Rong, and H. Zhang. Dynamics of Puma manipulator. In *American Control Conference*, pages 491–96, June 1983.

[205] M. Penna. Camera calibration: a quick and easy way to determine scale factor. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(12):1240–1245, December 1991.

[206] T. Pool. *Motion Control of a Citrus-Picking Robot*. PhD thesis, University of Florida, 1989.

[207] R. E. Prajoux. Visual tracking. In D. Nitzan et al., editors, *Machine intelligence research applied to industrial automation*, pages 17–37. SRI International, August 1979.

[208] K. Przytula and J. Nash. A special purpose coprocessor for robotics and signal processing. In D. Lyons, G. Pocock, and J. Korein, editors, *Workshop on Special Computer Architectures for Robotics*, pages 74–82. IEEE, Philadelphia, April 1988. In conjunction with IEEE Conf. Robotics and Automation.

[209] M. H. Raibert and B. K. P. Horn. Manipulator control using the configuration space method. *The Industrial Robot*, pages 69–73, June 1978.

[210] O. Ravn, N. Andersen, and A. Sørensen. Auto-calibration in automation systems using vision. In *Proc. Third International Symposium on Experimental Robotics*, pages 201–210, Kyoto, October 1993.

[211] P. Rives, F. Chaumette, and B. Espiau. Positioning of a robot with respect to an object, tracking it and estimating its velocity by visual servoing. In V. Hayward and O. Khatib, editors, *Experimental Robotics 1*, volume 139 of *Lecture Notes in Control and Information Sciences*, pages 412–428. Springer-Verlag, 1989.

[212] A. Rizzi and D. Koditschek. Preliminary experiments in spatial robot juggling. In *Proc. 2nd International Symposium on Experimental Robotics*, Toulouse, France, June 1991.

[213] M. Roberts. Control of resonant robotic systems. Master's thesis, University of Newcastle, Australia, March 1991.

[214] D. Robinson. Why visuomotor systems don't like negative feedback and how they avoid it. In M. Arbib and A. Hanson, editors, *Vision, Brain and Cooperative Behaviour*, chapter 1. MIT Press, 1987.

[215] C. Rosen et al. Machine intelligence research applied to industrial automation. Sixth report. Technical report, SRI International, 1976.

[216] C. Rosen et al. Machine intelligence research applied to industrial automation. Eighth report. Technical report, SRI International, 1978.

[217] A. Rosenfeld and A. C. Kak. *Digital Picture Processing*. Academic Press, 1982.

[218] R. B. Safadi. An adaptive tracking algorithm for robotics and computer vision application. Technical Report MS-CIS-88-05, University of Pennsylvania, January 1988.

[219] P. K. Sahoo, S. Soltani, and A. K. C. Wong. A survey of thresholding techniques. *Computer Vision, Graphics, and Image Processing*, 41:233–260, 1988.

[220] T. Sakaguchi, M. Fujita, H. Watanabe, and F. Miyazaki. Motion planning and control for a robot performer. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 925–931, 1993.

[221] B. Saleh and M. Teich. *Fundamentals of Photonics*. Wiley, 1991.

[222] C. Samson, B. Espiau, and M. L. Borgne. *Robot Control: the Task Function Approach*. Oxford University Press, 1990.

[223] A. C. Sanderson and L. E. Weiss. Image-based visual servo control using relational graph error signals. *Proc. IEEE*, pages 1074–1077, 1980.

[224] A. C. Sanderson and L. E. Weiss. Adaptive visual servo control of robots. In A. Pugh, editor, *Robot Vision*, pages 107–116. IFS, 1983.

[225] A. C. Sanderson and L. E. Weiss. Dynamic sensor-based control of robots with visual feedback. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 102–109, 1986.

[226] A. C. Sanderson, L. E. Weiss, and C. P. Neuman. Dynamic visual servo control of robots: an adaptive image-based approach. *Proc. IEEE*, pages 662–667, 1985.

[227] S. Sawano, J. Ikeda, N. Utsumi, H. Kiba, Y. Ohtani, and A. Kikuchi. A sealing robot system with visual seam tracking. In *Proc. Int. Conf. on Advanced Robotics*, pages 351–8, Tokyo, September 1983. Japan Ind. Robot Assoc., Tokyo, Japan.

[228] P. Sharkey, R. Daniel, and P. Elosegui. Transputer based real time robot control. In *Proc. 29th CDC*, volume 2, page 1161, Honolulu, December 1990.

[229] P. Sharkey and D. Murray. Coping with delays for real-time gaze control. In *Sensor Fusion VI*, volume 2059, pages 292–304. SPIE, 1993.

[230] P. Sharkey, D. Murray, S. Vandevelde, I. Reid, and P. McLauchlan. A modular head/eye platform for real-time reactive vision. *Mechatronics*, 3(4):517–535, 1993.

[231] P. Sharkey, I. Reid, P. McLauchlan, and D. Murray. Real-time control of a reactive stereo head/eye platform. *Proc. 29th CDC*, pages CO.1.2.1–CO.1.2.5, 1992.

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

[233] Y. Shiu and S. Ahmad. Finding the mounting position of a sensor by solving a homogeneous transform equation of the form Ax=xB. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1666–1671, 1987.

[234] W. M. Silver. On the equivalance of Lagrangian and Newton-Euler dynamics for manipulators. *Int. J. Robot. Res.*, 1(2):60–70, Summer 1982.

[235] R. A. Singer and K. W. Behnke. Real-time tracking filter evaluation and selection for tactical applications. *IEEE Trans. Aerosp. Electron. Syst.*, AES-7(1):100–110, January 1971.

[236] S. Skaar, W. Brockman, and R. Hanson. Camera-space manipulation. *Int. J. Robot. Res.*, 6(4):20–32, 1987.

[237] G. Skofteland and G. Hirzinger. Computing position and orientation of a freeflying polyhedron from 3D data. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 150–155, 1991.

[238] O. Smith. Closer control of loops with dead time. *Chem. Eng. Prog. Trans*, 53(5):217–219, 1957.

[239] I. Sobel. On calibrating computer controlled cameras for perceiving 3-D scenes. *Artificial Intelligence. An International Journal*, 5:185–198, 1974.

[240] M. Srinivasan, M. Lehrer, S. Zhang, and G. Horridge. How honeybees measure their distance from objects of unknown size. *J. Comp. Physiol. A*, 165:605–613, 1989.

[241] G. Stange, M. Srinivasan, and J. Dalczynski. Rangefinder based on intensity gradient measurement. *Applied Optics*, 30(13):1695–1700, May 1991.

[242] T. M. Strat. Recovering the camera parameters from a transformation matrix. In *Proc. Image Understanding Workshop*, 1984.

[243] I. E. Sutherland. Three-dimensional data input by tablet. *Proc. IEEE*, 62(4):453–461, April 1974.

[244] K. Tani, M. Abe, K. Tanie, and T. Ohno. High precision manipulator with visual sense. In *Proc. ISIR*, pages 561–568, 1977.

[245] T. Tarn, A. K. Bejczy, X. Yun, and Z. Li. Effect of motor dynamics on nonlinear feedback robot arm control. *IEEE Trans. Robot. Autom.*, 7(1):114–122, February 1991.

[246] T. J. Tarn, A. K. Bejczy, S. Han, and X. Yun. Inertia parameters of Puma 560 robot arm. Technical Report SSM-RL-85-01, Washington University, St. Louis, MO., September 1985.

[247] T. Tarn, A. Bejczy, G. Marth, and A. Ramadorai. Performance comparison of four manipulator servo schemes. *IEEE Control Systems Magazine*, 13(1):22–29, February 1993.

[248] A. R. Tate. Closed loop force control for a robotic grinding system. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachsetts, 1986.

[249] F. Tendick, J. Voichick, G. Tharp, and L. Stark. A supervisory telerobotic control system using model-based vision feedback. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2280–2285, 1991.

[250] W. Thomas, Jr., editor. *SPSE Handbook of Photographic Science and Engineering*. John Wiley and Sons, 1973.

[251] M. Tomizuka. Zero phase error tracking algorithm for digital control. *Journal of Dynamic Systems, Measurement and Control*, 109:65–68, March 1987.

[252] R. Tsai. A versatile camera calibration technique for high accuracy 3-D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Trans. Robot. Autom.*, 3(4):323–344, August 1987.

[253] R. Tsai and R. Lenz. A new technique for fully autonomous and efficient 3D robotics hand/eye calibration. *IEEE Trans. Robot. Autom.*, 5(3):345–358, June 1989.

[254] J. Uicker. *On the Dynamic Analysis of Spatial Linkages Using 4 by 4 Matrices*. PhD thesis, Dept. Mechanical Engineering and Astronautical Sciences, NorthWestern University, 1965.

[255] Unimation Inc., Danbury, CT. *Unimate PUMA 500/600 Robot, Volume 1: Technical Manual*, April 1980.

[256] J. Urban, G. Motyl, and J. Gallice. Real-time visual servoing using controlled illumination. *Int. J. Robot. Res.*, 13(1):93–100, February 1994.

[257] K. Valavanis, M. Leahy, and G. Saridis. Real-time evaluation of robotic control methods. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 644–649, 1985.

[258] S. Venkatesan and C. Archibald. Realtime tracking in five degrees of freedom using two wrist-mounted laser range finders. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2004–2010, 1990.

[259] D. Vernon and M. Tistarelli. Using camera motion to estimate range for robotic parts manipulation. *IEEE Trans. Robot. Autom.*, 6(5):509–521, October 1990.

[260] A. Verri and T. Poggio. Motion field and optical flow: Qualitative properties. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(5):490–498, May 1989.

[261] Vision Systems Limited, Technology Park, Adelaide. *APA-512MX Area Parameter Accelerator User Manual*, October 1987.

[262] R. Vistnes. Breaking away from VAL. Technical report, Unimation Inc., 1981.

[263] M. Vuskovic, T. Liang, and K. Anantha. Decoupled parallel recursive Newton-Euler algorithm for inverse dynamics. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 832–855, 1990.

[264] P. Vuylsteke, P. Defraeye, A. Oosterlinck, and H. V. den Berghe. Video rate recognition of plane objects. *Sensor Review*, pages 132–135, July 1981.

[265] M. W. Walker and D. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *ASME Journal of Dynamic Systems, Measurement and Control*, 104:205–211, 1982.

[266] R. Walpole and R. Myers. *Probability and Statistics for Engineers and Scientists*. Collier Macmillan, 1978.

[267] C. Wampler. *Computer Methods in Manipulator Kinematics, Dynamics, and Control: a Comparative Study.* PhD thesis, Stanford University, 1985.

[268] J. Wang and G. Beni. Connectivity analysis of multi-dimensional multi-valued images. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1731–1736, 1987.

[269] J. Wang and W. J. Wilson. Three-D relative position and orientation estimation using Kalman filter for robot control. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2638–2645, 1992.

[270] A. Wavering, J. Fiala, K. Roberts, and R. Lumia. Triclops: A high-performance trinocular active vision system. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 410–417, 1993.

[271] T. Webber and R. Hollis. A vision based correlator to actively damp vibrations of a coarse-fine manipulator. RC 14147 (63381), IBM T.J. Watson Research Center, October 1988.

[272] A. Weir, P. Dunn, P. Corke, and R. Burford. High speed vision processing. In *Workshop on Imaging Software and Hardware*. University of Sydney, February 1985.

[273] L. Weiss. *Dynamic Visual Servo Control of Robots: an Adaptive Image-Based Approach*. PhD thesis, Carnegie-Mellon University, 1984.

[274] 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 *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2376–2384, 1991.

[275] J. S. Weszka. A survey of threshold selection techniques. *Computer Graphics and Image Processing*, 7:259–265, 1978.

[276] D. E. Whitney. Historical perspective and state of the art in robot force control. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 262–8, 1985.

[277] D. Whitney and D. M. Gorinevskii. The mathematics of coordinated control of prosthetic arms and manipulators. *ASME Journal of Dynamic Systems, Measurement and Control*, 20(4):303–309, 1972.

[278] J. Wilf and R. Cunningham. Computing region moments from boundary representations. JPL 79-45, NASA JPL, November 1979.

[279] E. Williams and R. Hall. *Luminescence and the light emitting diode*. Pergamon, 1978.

[280] W. Wilson. Visual servo control of robots using Kalman filter estimates of relative pose. In *Proc. IFAC 12th World Congress*, pages 9–399 to 9–404, Sydney, 1993.

[281] Wind River Systems, Inc., 1351 Ocean Avenue, Emeryville CA 94608. *VxWorks 4.00 Volume 1: User Manual*, 1988.

[282] P. Wolf. *Elements of Photogrammetry*. McGraw-Hill, 1974.

[283] K. W. Wong. Mathematic formulation and digital analysis in close-range photogrammetry. *Photogrammetric Engineering and Remote Sensing*, 41(11):1355–1373, November 1975.

[284] J. Woodfill, R. Zabih, and O. Khatib. Real-time motion vision for robot control in unstructured environments. In L. Demsetz and P. Klarer, editors, *Robotics for challenging environments*. ASCE, New York, 1994.

[285] Y. Yakimovsky and R. Cunningham. A system for extracting three-dimensional measurements from a stereo pair of TV cameras. *Computer Graphics and Image Processing*, 7:195–210, 1978.

[286] M. Y. H. Yii, D. G. Holmes, and W. A. Brown. Real-time control of a robot manipulator using low-cost parallel processors. In *IEEE Workshop on Motion Control*, 1989.

[287] K. Youcef-Toumi and H. Asada. The design and control of manipulators with decoupled and configuration-invariant inertia tensors. In *Proc. ACC*, pages 811–817, Seattle, 1986.

[288] J.-C. Yuan. A general photogrammetric method for determining object position and orientation. *IEEE Trans. Robot. Autom.*, 5(2):129–142, April 1989.

[289] J.-C. Yuan, F. Keung, and R. MacDonald. Telerobotic tracker. Patent EP 0 323 681 A1, European Patent Office, Filed 1988.

[290] D. B. Zhang, L. V. Gool, and A. Oosterlinck. Stochastic predictive control of robot tracking systems with dynamic visual feedback. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 610–615, 1990.

[291] N. Zuech and R. Miller. *Machine Vision*. Fairmont Press, 1987.

Appendix A

Glossary

ADC	analog to digital converter
affine	An *affine* transformation is similar to a conformal transformation except that scaling can be different for each axis — shape is not preserved, but parallel lines remain parallel.
AGC	automatic gain control
AIA	Automated Imaging Association.
ALU	Arithmetic logic unit, one component of a CPU
APA	Area Parameter Accelerator, an image feature extraction board manufactured by Atlantek Microsystems, Adelaide, Australia. See Appendix C.
ARC	Australian Research Council, a national agency that funds research projects
ARCL	Advanced Robot Control Library, an RCCL like package for on-line control and offline simulation of robot programs [64].
ARX	autoregressive with exogenous inputs
ARMAX	autoregressive moving average with exogenous inputs
CCD	Charge coupled device
CCIR	International Radio Consultative Committee, a standards body of the UN
CID	Charge injection device
CIE	Commission Internationale de l'Eclairage
CMAC	Cerebellar model arithmetic computer [5]
conformal	A *conformal* transformation is one which preserves shape — translation, rotation and scaling are all conformal.
COG	Center of gravity
CPU	Central processing unit

CRT	Cathode ray tube.
CSIRO	Commonwealth Scientific and Industrial Research Organization, the Australian national research organization.
CTF	Contrast transfer function. Similar to MTF but measured with square wave rather than sine wave excitation.
DAC	digital to analog converter
DH	Denavit-Hartenberg
DIGIMAX	A video digitizer board manufactured by Datacube Inc., Danvers MA, USA.
DOF	Degrees of freedom
EMF	electro-motive force, measured in Volts
FFT	Fast Fourier transform, an efficient algorithm for computing a discrete Fourier transform
fovea	the high resolution region of the eye's retina.
FPGA	field-programmable gate array
IBVS	Image based visual servoing
LED	light emitting diode
LQG	linear quadratic Gaussian
LTI	linear time invariant
MAPLE	a symbolic algebra package from University of Waterloo [47].
MATLAB	An interactive package for numerical analysis, matrix manipulation and data plotting from 'The MathWorks' [181].
MAXBUS	a digital video interconnect standard from Datacube Inc.
MAXWARE	'C' language libraries from Datacube Inc. for the control of their image processing modules
MDH	Modified Denavit-Hartenberg
MIMO	multiple-input multiple-output
MMF	magneto-motive force, measured in Ampere turns. Analogous to voltage in a magnetic circuit.
MRAC	model reference adaptive control
MTF	Modulation transfer function
NE	Newton-Euler
NFS	Network File System, a protocol that allows computers to access disks on remote computers via a network.
NMOS	N-type metal oxide semiconductor
PBVS	Position based visual servoing
PD	proportional and derivative
PI	proportional and integral
PID	proportional integral and derivative
principal point	Principal point, the point where the camera's optical axis intersects the image plane.

Puma	A type of robot originally manufactured by Unimation Inc, and subsequently licenced to Kawasaki. Probably the most common laboratory robot in the world.
RCCL	Robot Control C Library, a software package developed at Purdue and McGill Universities for robot control.
RMS	Root mean square.
RNE	Recursive Newton-Euler
RPC	a mechanism by which a computer can execute a procedure on a remote computer. The arguments are passed and the result returned via a network.
RS170	Recommended standard 170, the video format used in the USA and Japan.
RTVL	Real-time vision library. A software package to facilitate experimentation in real-time vision, see Appendix D.
saccade	a rapid movement of the eye as it jumps from fixation on one point to another.
SHAPE	a 3D shape measurement system developed at Monash University, Melbourne.
SIMULINK	A block-diagram editing and non-linear simulation add on for MATLAB.
SISO	single-input single-output.
SNR	Signal to noise ratio, generally expressed in dB. SNR = $\overline{x^2}/\sigma^2$
Unimate	Generic name for robots and controllers manufactured by Unimation Inc.
VAL	the programming language provided with Unimate robots.
VLSI	Very large scale integrated (circuit)
VxWorks	a real-time multi-tasking operating system from WindRiver Systems [281].
ZOH	zero-order hold.

Appendix B

This book on the Web

Details on source material from this book can be obtained via the book's home page at `http://www.cat.csiro.au/dmt/programs/autom/pic/book.htm`. Material available includes:

- Cited technical papers by the author
- Robotics Toolbox for MATLAB
- MAPLE code for symbolic manipulation of robot equations of motion
- SIMULINK models for robot and visual servo systems
- Links to other visual servo resources available on the World Wide Web
- Ordering details for the accompanying video tape.
- Visual servoing bibliography
- Errata

Appendix C

APA-512

The APA-512 [261] is a VMEbus boardset designed to accelerate the computation of area parameters of objects in a scene. It was conceived and prototyped by the CSIRO Division of Manufacturing Technology, Melbourne, Australia, in 1982-4, [272] and is now manufactured by Atlantek Microsystems Ltd. of Adelaide, Australia. The APA performs very effective data reduction, reducing a 10 Mpixel/s stream of grey-scale video data input via MAXBUS, to a stream of feature vectors representing objects in the scene, available via onboard shared memory.

The APA, see Figure C.1, accepts video input from a MAXBUS connector, binarizes it, and passes it via a framebuffer to the connectivity logic. The frame buffer allows images to be loaded via the VMEbus, and also acts as a pixel buffer to match the processing rate to the incoming pixel rate. Pixels arrive at 10 Mpixel/s during

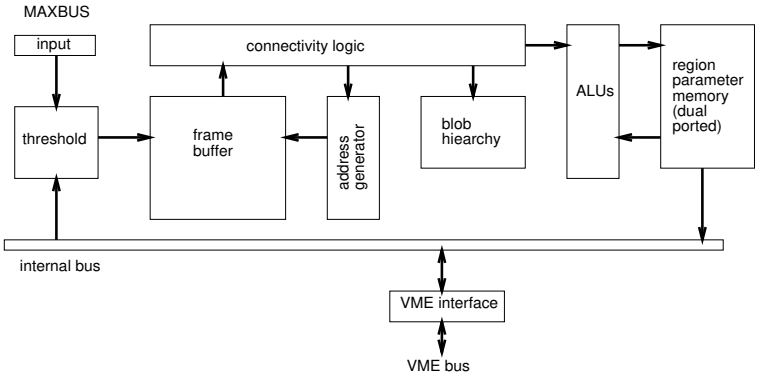


Figure C.1: APA-512 block diagram.

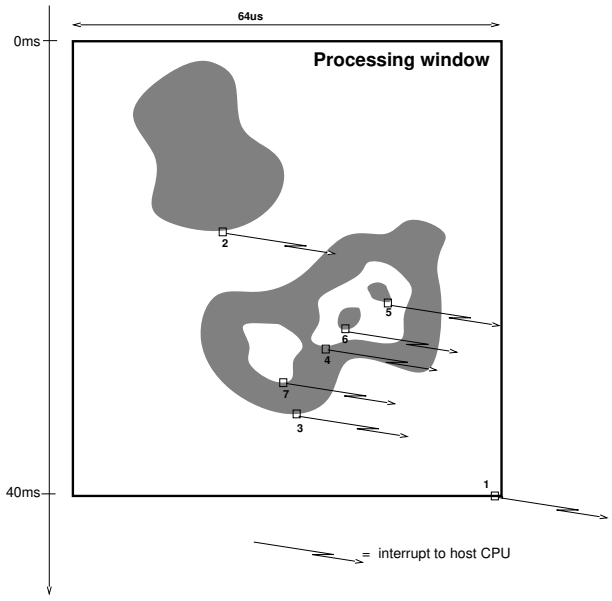


Figure C.2: APA region data timing, showing region completion interrupts for a non-interlaced frame. The host is notified of completed regions one raster scan line after the last pixel of the region.

each video line but the peak processing rate is over 20 Mpixel/s. The connectivity logic examines each pixel and its neighbours and commands a bank of ALUs to update the primitive region features which are kept in shared memory. The ALUs are implemented by custom gate arrays. Each region is assigned a unique integer label in the range 0 to 254, and which also serves as the address of the region's primitive feature data in the shared memory. The connectivity analysis is single-pass, of the type described by Haralick as *simple-linkage region growing* [106]. For each region the following parameters are computed by the APA:

- Σi, number of pixels (zeroth moment);
- Σx, Σy (first moments);
- Σx^2, Σy^2, Σxy (second moments);
- minimum and maximum x and y values for the region;
- perimeter length;
- a perimeter point;
- region color (0 or 1);
- window edge contact.

Figure C.2 shows when region completion interrupts are generated with respect to

the image which is input in raster scan fashion. One line time after the last pixel of the region its completion can be detected and the host computer notified by interrupt or pollable status flag and that region's label is placed in a queue. The host would read a label from the queue and then read the region's primitive feature data from the APA's shared memory. Information about regions is thus available well before the end of the field or frame containing the region.

From the fundamental parameters, a number of commonly used image features discussed in Section 4.1 such as:

- area
- centroid location
- circularity
- major and minor equivalent ellipse axis lengths
- object orientation (angle between major axis and horizontal)

may be computed by the host processor. These represent a substantial subset of the so-called 'SRI parameters' defined by Gleason and Agin at SRI in the late 70's.

The perimeter point is the coordinate of one pixel on the region's perimeter, and is used for those subsequent operations that require traversal of the perimeter. The edge contact flag, when set, indicates that the region touches the edge of the processing window and may be partially out of the image, in this case the parameters would not represent the complete object.

Perimeter is computed by a scheme that examines a 3x3 window around each perimeter point as shown in Figure C.3. The lookup table produces an appropriate perimeter length contribution depending upon the slope of the perimeter at that point. Experiments reveal a worst case perimeter error of 2% with this scheme.

The APA-512 computes these parameters for each of up to 255 *current* regions within the scene. Processing of the data is done in raster scan fashion, and as the end of a region is detected the region label is placed in a queue and the host is notified by an interrupt or a pollable status flag. The host may read the region parameters and then return the region label to the APA for reuse later in the frame, thus allowing processing of more than 255 objects within one frame. This feature is essential for processing non-trivial scenes which can contain several hundred regions of which only a few are of interest. Maximum processing time is one video frame time.

An additional feature of the APA is its ability to return region hierarchy information as shown in Figure C.4. When a region is complete the APA may be polled to recover the labels of already completed regions which were topologically contained within that region. This makes it possible to count the number of holes within an object, and compute the area of enclosed holes or internal perimeter.

The APA was designed to process non-interlaced video, but can be coerced into working with interlaced video thus eliminating the deinterlacing process. The APA processes the interlaced video as one large frame, see Figure C.5. The active processing window is set to the upper position before the even field commences. This

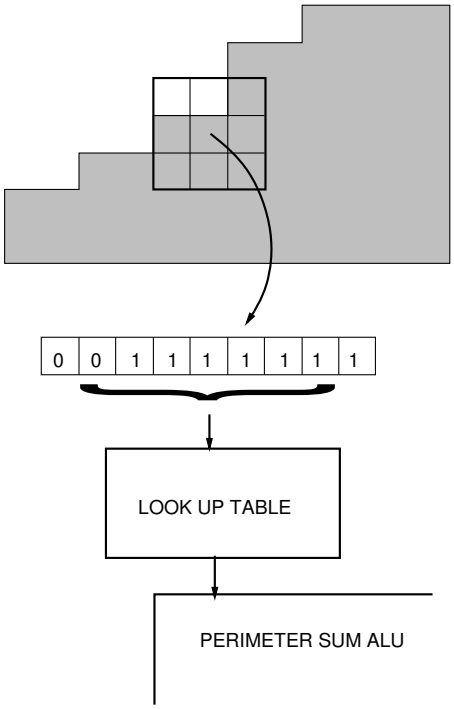


Figure C.3: Perimeter contribution lookup scheme.

window stops processing before the blanking interval during which time the APA will continue loading the video signal which will comprise the equalization and serration pulses. Prior to the odd field commencing, the processing window is set to the lower position. The biggest drawback with this approach is that the bottom processing window comprises only $511 - 313 = 198$ lines rather than the 287 active lines of a video field. This is due to the APA unnecessarily loading lines during the vertical blanking interval and also the CCIR video format having 574 active lines. Normally when working with 512×512 images the lower 31 lines of each field are lost.

The APA-512 is controlled by the host computer using a 'C' language library that is compatible with Datacube's MAXWARE 3.1 [74]. To achieve maximum performance with a Sun workstation host a Unix device driver was written to efficiently move data from the APA to the application program's memory space. For operation under VxWorks the interface library was simply cross-compiled and the device driver replaced by a simple structure in which APA interrupt handlers raise semaphores to unblock tasks which service the APA.

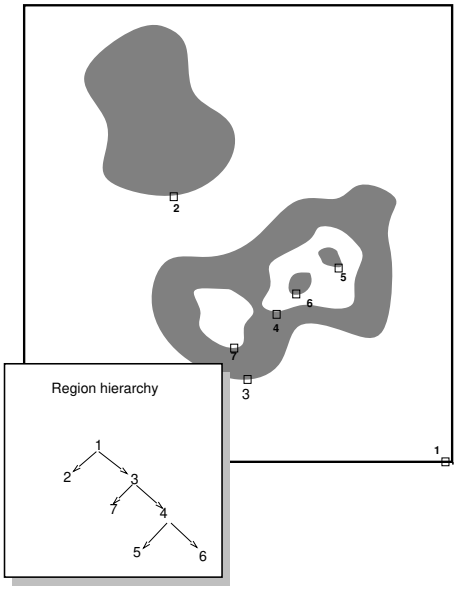


Figure C.4: Hierarchy of binary regions.

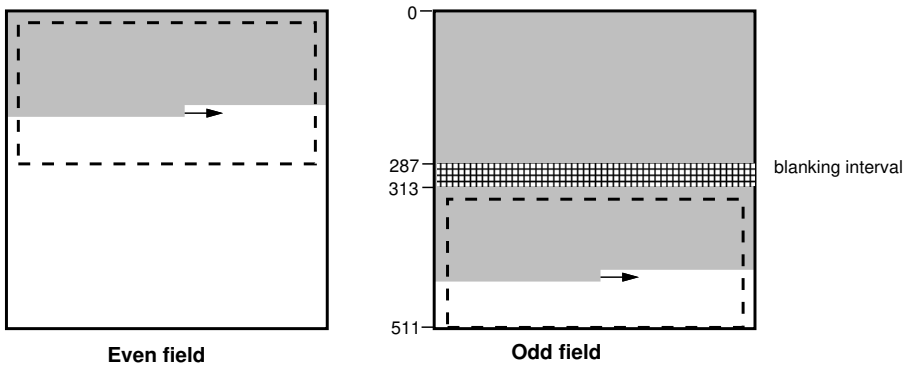


Figure C.5: Field mode operation. Stippled region shows the progress of the raster scan. Dashed box is the active APA processing window.

Appendix D

RTVL: a software system for robot visual servoing

This appendix discusses a software system developed within the Division's Melbourne Laboratory to facilitate research into video-rate robotic visual servoing. Early experimental work in visual servoing showed that quite simple applications rapidly became bloated with detailed code dealing with the requirements of vision and robot control, graphical display, diagnostics, data logging and so on [57]. Considerable work has gone into the design of the software system known as RTVL for real-time vision library. RTVL provides extensive functions to the user's application program encompassing visual-feature extraction, data-logging, remote variable setting, and graphical display.

RTVL provides visual-servo specific extensions to VxWorks and is loaded into memory at system boot time to provides all the infrastructure required for visual servoing. Visual servo applications programs are loaded subsequently and access RTVL via a well-defined function call interface. Internally it comprises a number of concurrent tasks and shared data structures. Each RTVL module has its own initialization and cleanup procedure as well as one or more procedures, accessible from the VxWorks shell, to show operating statistics or enable debugging output. The whole system is highly parameterized and the parameters are parsed from a text file at startup. All parameters are global variables, and thus may be inspected or altered from the VxWorks shell allowing many operating characteristics to be changed online. A schematic of the system, showing both hardware and software components, is given in Figure D.1.

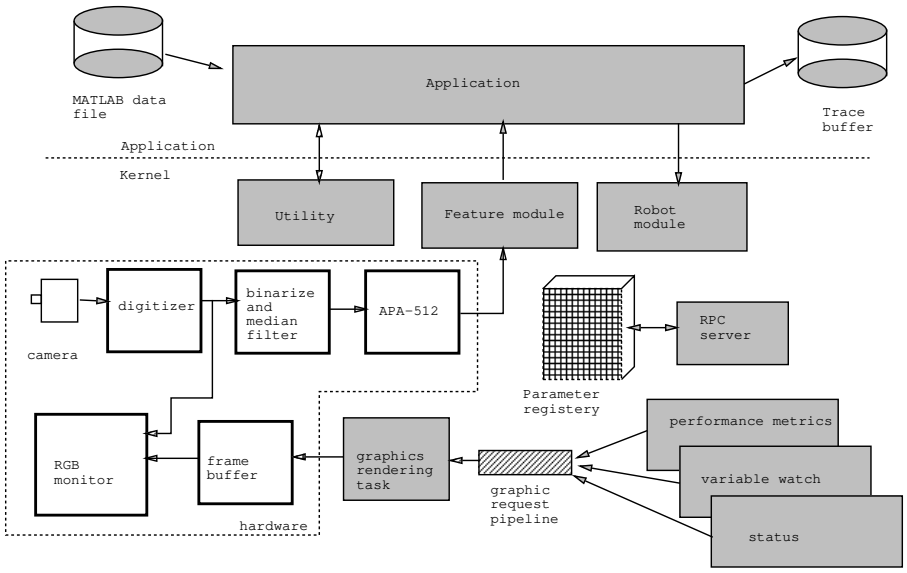


Figure D.1: Schematic of the experimental system

D.1 Image processing control

The image processing pipeline shown in Figure D.1 requires software intervention during each video blanking time to control video datapaths and set the APA processing window to allow field rate processing as discussed in Appendix C. The `field` task is activated at the start of video blanking and invokes functions that have been previously registered for callback. A number of permanent callbacks are used by RTVL itself and others may be registered by user application programs. The blanking interval is relatively narrow, lasting only 1.6 ms, so this task runs at high priority in order to accomplish all its tasks within the blanking interval. Datacube's MAXWARE software libraries are used, with a custom written low level module to allow operation under VxWorks.

D.2 Image features

The feature extraction process is simplistic and reports the first (in raster order) n regions which meet the application program's acceptance criteria. These are expressed in terms of a boolean screening function applied to all extracted regions in the scene. Typically screening is on the basis of object color (black or white), upper and lower

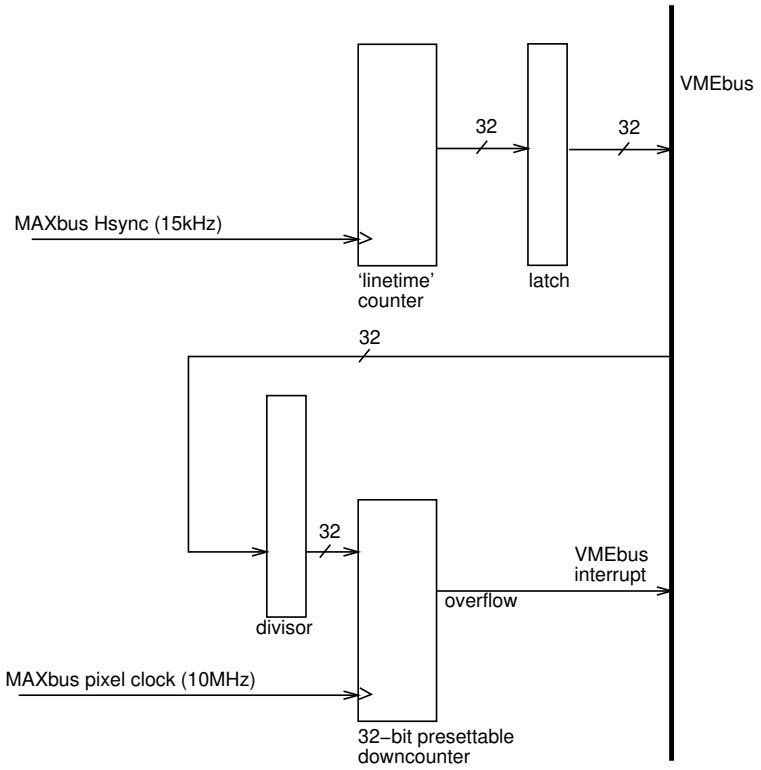


Figure D.2: Block diagram of video-locked timing hardware.

bounds on area, and perhaps circularity (4.11). Functions exist to compute useful measures such as centroid, circularity and central moments from the returned region datastructures. An application obtains region data structures by means of the `regGet()` function.

D.3 Time stamps and synchronized interrupts

Timestamping events is essential in understanding the temporal behavior of a complex multi-tasking sensor-based system. It is also desirable that the mechanism has low overhead so as not to impact on system performance. To meet this need a novel timing board has been developed that provides, via one 32-bit register, a count of video lines since midnight, see Figure D.2. Each video line lasts 64μs and this provides adequate timing resolution, but more importantly since the count is derived from

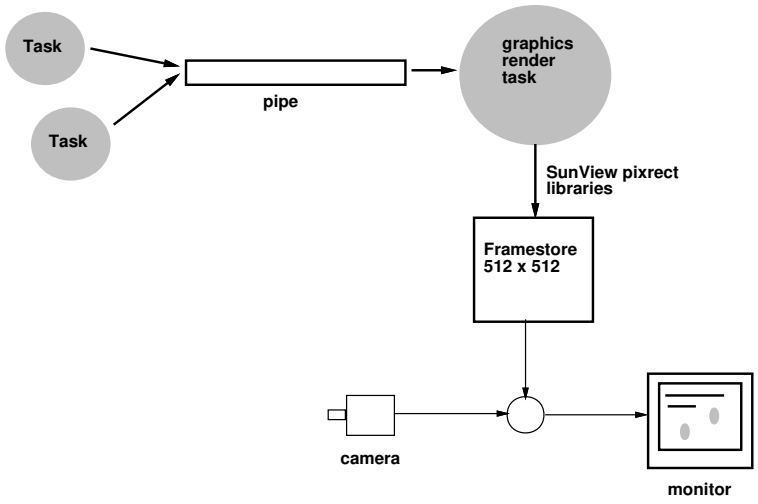


Figure D.3: Graphics rendering subsystem.

MAXBUS horizontal synchronization signals it gives a time value which can be directly related to the video waveform. The time value can be readily converted into frame number, field number, field type (odd or even) and line number within the field or frame, as well as into conventional units such as time of day in hours, minutes and seconds. A comprehensive group of macros and functions is provided to perform these conversions. For debugging and performance analysis this allows the timing of events with respect to the video waveform to be precisely determined.

In addition, for custom manipulator axis control strategies, a source of periodic interrupts with a period in the range 1 to 5 ms is required. To avoid beating problems with the lower rate visual control loops it is desirable that the servo loops operate at a sub-multiple of the vision field time. A programmable down counter on the timing board divides the MAXBUS pixel clock by a 32 bit divisor to create suitable interrupts. Divisor values are computed by

$$n = 9625000 * T$$

where T is the desired period in units of seconds. The 2 and 4 ms period servo loops used in Chapter 8 are clocked in this manner.

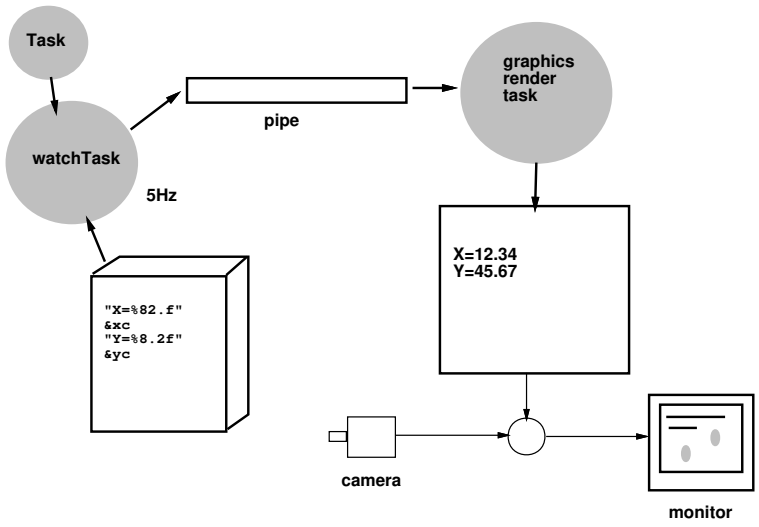


Figure D.4: Variable watch facility.

D.4 Real-time graphics

The system uses a framebuffer to hold graphical data which is overlaid on the live image from the camera. Almost all information about the system can be learned from this display. Since many tasks need to display graphical information quickly and cannot afford to wait all requests are queued, see Figure D.3. The graphical requests queue is serviced by a low-priority task which renders the graphical entities into the framebuffer using the SunView pixrect libraries which has been 'fooled' into treating the ROISTORE as a memory resident pixrect. Functions are provided which allow an application to write strings to the graphic display and update a variety of tracking cursors.

D.5 Variable watch

The 'watch' package allows a number of specified variables to be monitored and their value displayed on the real-time graphics display as shown in Figure D.4. The variable values are updated on the screen at 5 Hz, and may be used to monitor various internal program variables, as shown in Figure 6.8.

D.6 Parameters

Most parts of the RTVL kernel are controlled by parameters, rather than compiled-in constants. Parameters can be integer, float or string variables, or be attached to a function which is invoked when the parameter is changed. Whilst parameters can be set manually via the VxWorks shell, the preferred mechanism is via the interactive control facility described next. Parameters are initialized from a parameter file read when the kernel starts up. As each parameter is initialized from the file, it is registered with the parameter manager, which is a remote procedure call (RPC) server task which can modify or return the value of any parameter, see Figure D.5. User applications can also use the parameter facilities, by explicitly registering the type and location of application program parameters.

D.7 Interactive control facility

Various operating parameters of the RTVL kernel such as threshold may be changed by simply moving a slider on the control panel. A popup window lists all parameters registered under RTVL, and double-clicking brings up a slider, or value entry box, for that parameter. The XView program is an RPC client to the parameter server task running under VxWorks to examine and modify the parameter.

Convenient control of applications is facilitated by a mechanism that allows program variables to be registered with a remote procedure call (RPC) server. The client is an interactive control tool running under OpenWindows on an attached workstation computer. A list of variables registered under the real-time system can be popped up, and for user selected variables a value slider is created which allows that variable to be adjusted. Variables can be boolean, integer, floating point scalar or vector.

The named parameters can be conveniently examined and modified by the interactive control tool, an XView application running under OpenWindows.

A remote cursor facility has also been implemented, whereby a cursor on the RTVL display is slaved to the mouse on the OpenWindows workstation. An application program can wait for a 'pick' event

```
superGetPick(&coord);
```

Unsolicited picks are grabbed by the region processing subsystem which will display the feature vector for the region under the cursor.

D.8 Data logging and debugging

The VxWorks shell allows any global program variable to be examined or modified interactively. However it has been found convenient to build facilities to 'log' variables

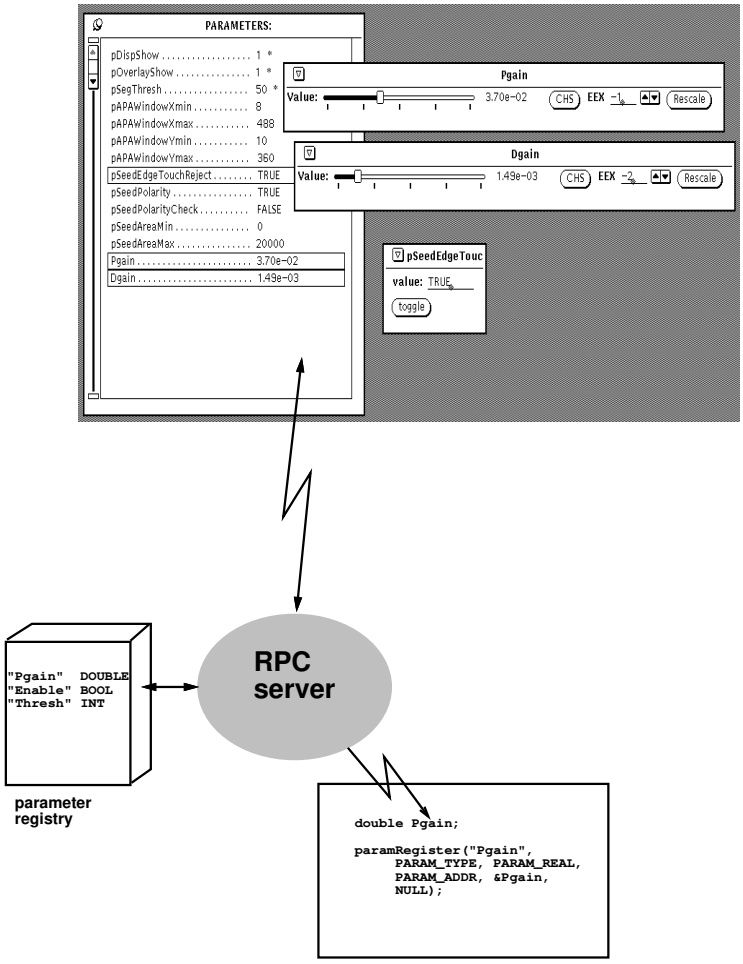


Figure D.5: On-line parameter manipulation.

of interest onto the main color display, and also to record time histories of variables for data logging purposes.

The 'watch' facility continuously displays the values of a nominated list of variables in the graphics plane which is superimposed on the live camera image, see Figure 6.8.

The RTVL kernel provides built-in facilities to log multiple variables during an experiment. This is essential in trying to debug an application, or monitor the closed-loop control system performance. Traced variables are timestamped and placed into

a large circular buffer using a low-overhead macro package. The trace buffer can be dumped to disk, and off-line tools can be used to extract the time histories of the variables of interest. These time records can be analyzed by a number of tools such as MATLAB [181] for analysis and plotting.

D.9 Robot control

Robot control functions are layered on top of the ARCL package introduced in Section 6.2.3. RTVL can use ARCL facilities to implement a 6-DOF Cartesian velocity controller. This approach is useful for quick application development, but the problems discussed in the body of the book such as increased latency from data double handling and the limitations of purely feedback control will apply. For higher performance it is necessary for the programmer to implement custom axis control loops.

D.10 Application program facilities

MATLAB is used as a tool for both controller design and data analysis. The RTVL kernel includes library routines for reading MATLAB's binary MAT-files, as well as for performing matrix arithmetic. This allows straightforward implementation of state-feedback controllers and state-estimators designed using MATLAB. On the current system, a 4-state estimator and state-feedback control can be executed in around $500\mu s$.

D.11 An example — planar positioning

Servoing in a plane orthogonal to the camera view axis has been demonstrated by a number of authors with varying levels of performance. The most appropriate features to use are the object centroid (x_c, y_c), computed simply from the 1st order moments;

$$x_c = \frac{m_{10}}{m_{00}}, y_c = \frac{m_{01}}{m_{00}}$$

The essence of the application program is:

```
1  planar()
2  {
3      Region  r;
4      double  xc, yc, xgain, ygain, cartRates[6];
5      int     nfeat;
6
7      paramRegister("xgain", &xgain, PARAM_TYPE, PARAM_REAL, 0);
8      paramRegister("ygain", &ygain, PARAM_TYPE, PARAM_REAL, 0);
```

```
 9      watchStart("X=%f", &x, "Y=%f", &y, NULL);
10      regFilterFunc(filterfn);    /* specify region screening funcn */
11      robotOpen();
12      for (;;) {      /* loop forever */
13          nfeat = regGet(&r, 1, filterfn); /* get a feature */
14
15          xc = XC(r);     /* find the centroid */
16          yc = YC(r);
17
18          /* compute the desired cartesian velocity */
19          cartRates[X] = xc * xgain;
20          cartRates[Y] = yc * ygain;
21
22          /* set the robot velocity */
23          robotSetRates(cartRates);
24      }
24  }
```

Lines 7-8 register the two control gains with the parameter manager to allow setting by the remote interactive control tool. Line 9 causes the centroid coordinates in pixels to be monitored on the real-time display. Line 12 requests a single feature that is accepted by the low-level region screening function `filterfn` (not shown here). Next the centroid is computed from simple moment features within the `Region` datatype, the control gain is computed, and then the X and Y Cartesian velocities of the robot are set.

D.12 Conclusion

A powerful experimental facility for research into robotic visual closed-loop control has been described. Pipelined image processing hardware and a high-speed region-analysis boardset are used to extract visual features at 50Hz. The RTVL kernel takes advantage of the real-time multi-tasking environment to provide facilities for real-time graphics, status display, diagnostic tracing, interactive control via a remote OpenWindows control tool, and MATLAB data import. An important design aim was to decouple the actual application from the considerable infrastructure for visual-feature extraction and robot control.

Appendix E

LED strobe

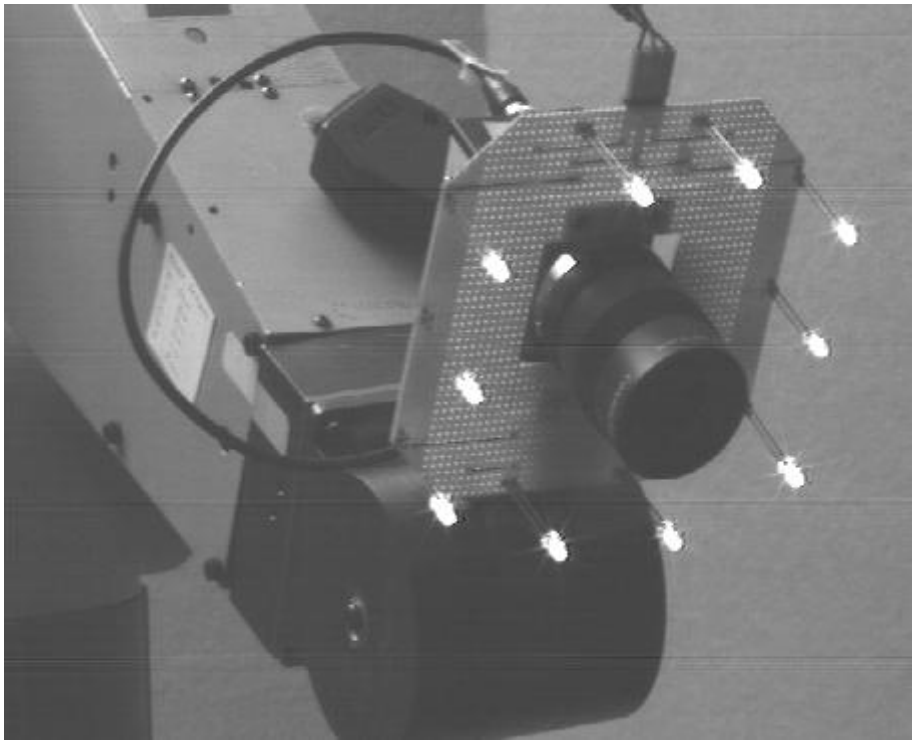


Figure E.1: Photograph of camera mounted solid-state strobe.

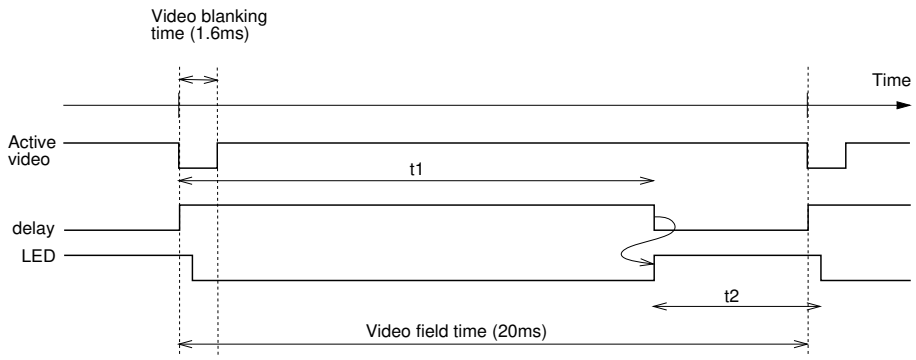


Figure E.2: Derivation of LED timing from vertical synchronization pulses.

The high-power LED illumination system built in the course of this work is shown in Figure E.1. Ten LEDs are arranged in a circle around the lens providing coaxial illumination of the robot's area of interest. Figure E.2 shows how the LED timing pulses are derived from vertical synchronization pulses output by the system's DIGIMAX video digitizer. The first delay, t_1, positions the pulse with respect to the vertical synchronization pulse and is used to align the LED pulse with the shutter opening. The second delay, t_2, governs the length of the LED pulse. LED timing can be readily set by pointing the camera at the LED while observing the camera output on a monitor. The LED appears lit only when the LED pulse overlaps the camera exposure interval. For a short duration pulse, the bounds of the exposure interval can be explored. Such an approach was used to determine the exposure intervals shown in Figure 3.14.

The peak pulse current was established experimentally using a circuit which allowed the current waveform to be monitored while adjusting the magnitude of the applied voltage pulse. As the peak current amplitude is increased the observed current waveform ceases to be square and rises during the pulse. This is indicative of the onset of thermal breakdown within the junction, and if sustained was found to lead to permanent destruction of the LED. The relationship between permissible current pulse amplitude and duty cycle is likely to be nonlinear and dependent upon thermal properties of the LED junction and encapsulation. The final LED control unit provides an adjustable constant current drive to avoid destructive current increase due to thermal breakdown.

In practice it was found that the relationship between current and brightness was non-linear as shown in Figure E.5. Further investigation with a high-speed photodiode sensor, see Figure E.3, shows that the light output of the LED is a narrow initial spike with a slower exponential falloff. Figure E.4 shows more detail of the initial spike. The light output increases with current which has a limited rise time due to inductance

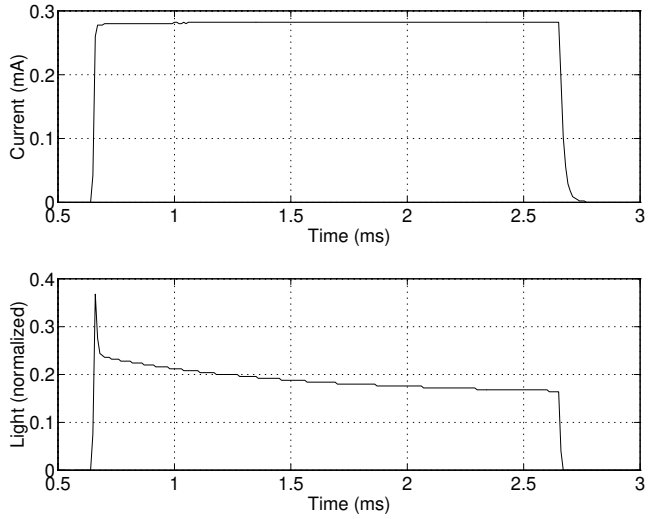


Figure E.3: LED light output as a function of time. Note the very narrow initial peak.

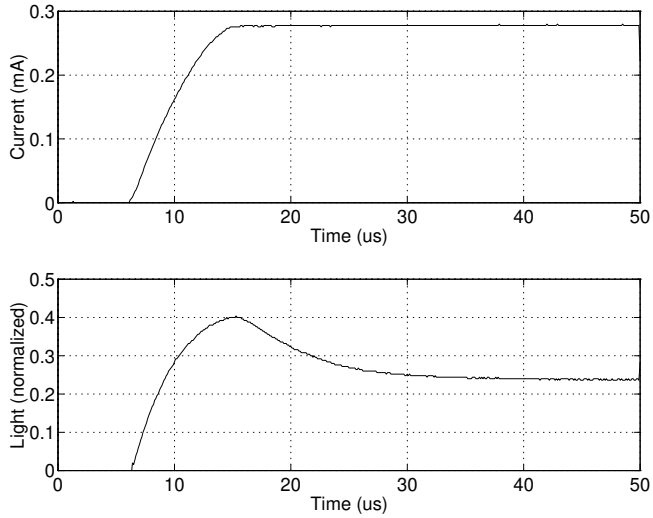


Figure E.4: LED light output during initial peak. Current (and light output) rise time is limited by circuit inductance.

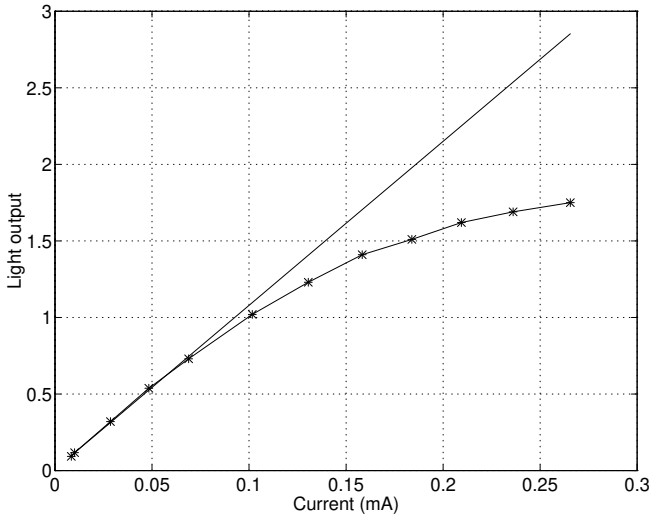


Figure E.5: Measured LED intensity as a function of current. Intensity is the average intensity over many light pulses as measured with an averaging light-meter.

in the long connecting cable. After the peak, light output falls off exponentially with a time constant of approximately 10μs. The two time constants involved in light falloff are suspected to be due to heating effects: the fast mode from heating of the small junction region, and the slow mode from heating the entire package. At high temperature an LED has a reduced quantum efficiency and its spectral characteristics may alter [279].

This light source has been used usefully in many experiments. The most significant difficulties are that the light output was not as great as hoped for, due to the unexpected thermal effects described above, and the rather uneven nature of the illumination.

Index