Feedback Control and Sensor Fusion of Vision and Force

Olsson, Tomas

2004

Document Version:
Publisher's PDF, also known as Version of record

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

Document name
LICENTIATE THESIS

Date of issue
October 2004

Document Number
ISRN LUTFD2/TFRT--3235--SE

Author(s)
Tomas Olsson

Supervisor
Rolf Johansson
Anders Robertsson

Sponsoring organisation
EU 5th Framework Growth Project AUTOFETT.

Title and subtitle
Feedback Control and Sensor Fusion of Vision and Force

Abstract

This thesis deals with feedback control using two different sensor types, force sensors and cameras. In many tasks in robotics compliance is required in order to avoid damage to the workpiece. Force and vision are the most useful sensing capabilities for a robot system operating in an unknown or uncalibrated environment. An overview of vision based estimation, control and vision/force control is given.

Two different control algorithms based on a hybrid force/vision structure are presented, using image-based and position-based visual servoing, respectively. The image-based technique is suitable in situations with simple contact geometry in uncalibrated environments, and in situations where the positioning task is naturally specified relative to some visible structure in image space. The position-based technique can handle more complex motions, which require more information about the environment. For a process with linear dynamics in task space, an edge-based visual estimation technique can be used to design an observer with linear error dynamics, which avoids the high computational complexity of the Extended Kalman Filter.

In a dynamic visual tracking system, many different choices of the parameterization of the state space exist. If the actuator- and measurement coordinate systems are rigidly attached, a dual quaternion parameterization can be used to express linear constraints on the estimated motion. This increases robustness when the intrinsic camera parameters are time-varying.

For real-time control applications in general, it is important to minimize the input-output latency, which may otherwise compromise the performance of the control system. In vision-based control systems the latency is dominated by the image processing. A method for multiple cameras, which aims at maximizing the achieved accuracy of the measurements given a fixed computation time, is presented.

Key words
Visual tracking, visual servoing, Kalman filter, resource optimization, sensor based robot control

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
English

Number of pages
121

Recipient's notes

Security classification

Feedback Control and Sensor Fusion of Vision and Force

Tomas Olsson

Department of Automatic Control
Lund Institute of Technology
Lund, October 2004

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

ISSN 0280–5316
ISRN LUTFD2/TFRT--3235--SE

Printed in Sweden,
Lund University, Lund 2004

Contents

Acknowledgments

In the three and a half years that have passed since I first came to the department, many people have helped, supported and inspired me. First of all I would like to thank my supervisor Rolf Johansson, both for giving me the chance to work within the Robotics lab, and for his many invaluable ideas, comments and feedback. My co-supervisor Anders Robertsson is another source of great ideas. His enthusiasm is legendary, as well as his willingness to help other people with whatever problems they have, no matter how busy he is himself. My other colleagues in the Robotlab, Klas Nilsson and Mathias Haage, also deserve a special mention for their work and support.

Many thanks also to all the people at the Department of Automatic Control, who make it such a wonderful and creative environment to work in. Leif and Anders Blomdell provides fantastic computer support. The secretaries Agneta, Britt-Marie and Eva helps out with all kinds of administrative problems. Rolf Braun is acknowledged for his help with the construction of all the strange items that seem to be an inevitable part of robotics experiments.

Thanks also to my fellow PhD students for many rewarding collaborations and coffee room discussions. I have very much enjoyed working together with Johan Bengtsson and Dan Henriksson on different projects. My roommate Staffan Haugwitz deserves credit for his almost unnatural positivity and constant high spirit. The innebandy gang and PiHHP give me a great opportunity to slash people across the legs using a very hard stick every Thursday and Sunday. The poker fanatics are acknowledged for waiting so patiently without giving up on the possibility to steal my money.

The work on the development of the sensor interface for external control was supported by the EU 5th Framework Growth Project *Affordable, flexible system for off-line automated fettling and finishing*, AUTOFETT.

Finally I would like to thank my friends and family for their encouragement and support.

Tomas

1

Introduction

1.1 Motivation

Over the last decades, cameras have emerged as useful sensors for measuring position of objects. In the field of robotics, cameras have reached a widespread usage, in the research community as well as in industry. Many commercial packages for vision processing have been developed, for use in calibration, positioning and inspection, and several other tasks. Additionally, vision sensors will be crucial components in the development of future flexible, mobile and autonomous robot systems. In these applications, there will be a need for robots which are able to work in unstructured, uncalibrated environments, with space-sharing between robots and human users and operators.

In parallel with the developments of new camera sensing techniques for position measurement and calibration, there has been an interest in using camera measurements for closed-loop position control. As early as in the seventies, it was realized that camera feedback could be used to correct the trajectories of a robot, thereby increasing the task accuracy. The term *visual servoing* was introduced for the use of cameras for feedback positioning. In later years, the emergence of cheap, powerful computing power has increased the potential of visual servoing systems dramatically. A review of the research and history of visual servoing from the start until the mid 90s can be found in [Hutchinson *et al.*, 1996].

Two key issues that need to be considered in a successful visual servoing system are *reliability* and *performance*. By reliability we mean the ability of the system to function in any situation that can occur during operation, with respect to changes in the environment such as lighting, (temporary) signal loss due to occlusion of objects, reflections, or rapid unpredicted motions, or internal factors such as singularities. In addition, robustness to uncertainties in the environment or modeling need is an important part of a reliable system. Performance is the ability to achieve tasks when additional timing constraints are present, such as a required cycle-time in a flexible manufacturing system. From a control perspective, performance requirements may be translated into closed-loop bandwidth constraints, in addition to requirements on the servoing capabilities of the system, that is, the ability to follow fast reference trajectories within a given tolerance. Reliability and performance are often conflicting objectives, a reliable system may require extremely large processing power, which may conflict with performance specifications. Conversely, high performance is more easily achieved in well structured, accurately known environments, which have been set up to minimize the risk for disturbances or sensor failures. This conflict makes reliable real-time visual servoing an extremely challenging task.

In order to increase the reliability and performance, it is crucial to exploit all available information. Dynamic and geometric models can be used to increase reliability, by increasing the predictive capabilities of the visual processing and thereby decreasing the risk for signal loss, as well as for improving performance. By suitable parameterizations of the state space, we can avoid singularities and incorporate geometric constraints on the states, thereby making the vision processing more robust. However, perhaps the most important improvements in reliability and flexibility can be achieved by using multiple sensors. Stereo- and multi-camera systems will improve tracking reliability and accuracy considerably, compared to single camera systems, due to both sensor redundancy and geometric factors. Similarly, other sensors such as range- and force sensors, may complement the vision system. In this thesis, two methods for combined force/vision control are presented and analyzed.

1.2 Outline and Related Publications

This section contains a brief outline of each chapter in the rest of the thesis, with references to related publications.

Chapter 2: Background

This chapter gives a short introduction to the subjects of vision based control, estimation, and combined force/vision control.

Chapter 3: Interface for External Control

This chapter presents the interface developed for external control, designed by extending an ABB S4CPlus industrial robot control system. Results from experiments, where the designed interface was used for force controlled grinding and deburring, are presented.

Publications

Blomdell, A, ; Bolmsjö, G. ; Brogårdh, T ; Cederberg, P. ; Isaksson, M. ; Johansson, R. ; Haage, M ; Nilsson, K ; Olsson, M. ; Olsson, T. ; Robertsson, A. ; Wang, J.J (2004): "Extending an industrial robot controller with a fast open sensor interface - implementation and applications" In *Robotics and Automation Magazine, to appear*.

Johansson, R. ; Robertsson, A. ; Nilsson, K. ; Brogårdh, T. ; Cederberg, P. ; Olsson, M ; Olsson, T. ; Bolmsjö, G. (2004): "Sensor Integration in Task-level Programming and Industrial Robotic Task Execution Control" In *Industrial Robot: An International Journal, 31:3, pp. 284-296*.

The interface described in this chapter is the result of the efforts of several people. Anders Robertsson and Klas Nilsson were responsible for the overall design of the interface, supported by Mats Isaksson and many others. Anders Blomdell, among many other contributions, developed the software for the Linux PowerPC platform running the external controllers. Mathias Haage and Klas Nilsson designed and implemented the force control extensions to the robot programming language RAPID. J.J. Wang provided most of the code for the kinematics library, as well as other software. Tomas Olsson and Anders Robertsson developed and tested force- and impedance controllers, and performed most of the experiments together with Klas Nilsson.

Chapter 4: Image-Based Hybrid Vision/Force Control

In this chapter a methods for image-based hybrid vision/force control is presented. An image-based visual servoing technique is used together with force feedback to perform experiments with drawing on a planar surface, while the position of the surface is simultaneously estimated using the available sensor data.

Publications

Olsson, T. and Bengtsson, J. and Johansson, R. and Malm, H. (2002): "Force Control and Visual Servoing Using Planar Surface Identification" In *Proceedings of the IEEE International Conference on Robotics and Automation*. Washington D.C., USA.

Chapter 5: Position-Based Hybrid Vision/Force Control

In this chapter, methods and experiments using a position-based hybrid force/vision control algorithm are presented. An observer using point-to-contour error meaurements is used to estimate the states of a linear system, with a computational complexity which is considerably smaller than for an EKF. Impedance control with inner motion control is used to obtain compliant behavior in the force controlled directions.

Publications

Olsson, T. and and Johansson, R. and Robertsson, A. (2004): "Flexible Force-Vision Control for Surface Following using Multiple Cameras" In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. Sendai, Japan.

Chapter 6: Rigid-Body Tracking using Dual Quaternions

In this chapter we present methods for real-time rigid body tracking with simultaneous calibration and tracking of intrinsic parameters, based on a dual quaternion parameterization of the object pose.

Publications

Olsson, T. and Bengtsson, J. and Robertsson, A. and Johansson, R. (2003): "Visual Position Tracking using Dual Quaternions with Hand-Eye Motion Constraints" In *Proceedings of the IEEE International Conference on Robotics and Automation*. Taipei, Taiwan.

Chapter 7: Multi-Camera Feedback Control with Time Constraints

In this chapter we present methods for multi-camera real-time rigid body tracking with time constraints. A simple algorithm is presented, which attempts to select active cameras and feature searches to minimize the effective measurement covariance. Simulations are used to validate the approach.

Publications

Henriksson, D. and Olsson, T. (2004): "Maximizing the Use of Computational Resources in Multi-Camera Feedback Control" In *Proceedings of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS04)*. Toronto, Canada.

2

Background

2.1 Introduction

The use of external sensing in robotics has been an active research topic for many years. Sensors in general, and vision sensors in particular, are used in the research community as well as in industry, for tasks such as calibration, positioning and inspection.

Although current industrial robot systems have very limited support for external sensor feedback, much research on vision feedback has been presented within the research community. In this chapter we present some background on the use of cameras and other external sensors for feedback control, with special focus on visual tracking and feedback, and combined vision/force control.

2.2 Visual Sensing and Vision Based Control

A general dynamical model for visual servoing is given by

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{v}) \tag{2.1}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) + \mathbf{e} \tag{2.2}$$

where $\mathbf{f}$ is a function of the state $\mathbf{x}$, the control input $\mathbf{u}$, and the disturbance $\mathbf{v}$, and the image-space output $\mathbf{y}$ is disturbed by additive noise

input $\mathbf{e}$. The output $\mathbf{y}$ may be the entire image, or a vector of positions of key image features, such as corners, edges or other points of interest.

It is common to assume that the manipulator is velocity controlled, and to let the visual servoing system close the position loop. The open-loop system will then contain "integrating" action. It is then possible to partition the state vector, and in some cases to rewrite the dynamics (2.1)-(2.2) as

$$\dot{\mathbf{x}}_p = \mathbf{f}_p\left(\mathbf{x}_d, \mathbf{v}_p\right) \tag{2.3}$$

$$\dot{\mathbf{x}}_d = \mathbf{f}_d\left(\mathbf{x}_p, \mathbf{x}_d, \mathbf{u}, \mathbf{v}_d\right) \tag{2.4}$$

$$\mathbf{y} = \mathbf{h}\left(\mathbf{x}_p\right) + \mathbf{e} \tag{2.5}$$

where $\mathbf{x}_p$ is a parameterization of the position, and $\mathbf{x}_d$ describes additional dynamic states, for instance velocities, accelerations and the states of the actuator dynamics.

Control

Two fundamentally different choices for the controller structure exist, corresponding to *image-based* and *position-based* visual servoing [Hutchinson *et al.*, 1996]. In image-based visual servoing, the image space output $\mathbf{y}$ is controlled directly to a desired position $\mathbf{y}_r$ defined *in the image*. In position-based visual servoing, an additional pose estimation step is used in order to obtain an estimate of $\mathbf{x}_p$ or the entire state vector, which is used to compute the control signal $\mathbf{u}$.

Image-Based Control. In image based techniques, the dynamics of $\mathbf{x}_d$ is often assumed to be considerably faster than the closed-loop system. In this case, it is common use a model

$$\dot{\mathbf{x}}_p = \mathbf{f}(\mathbf{u}) \tag{2.6}$$

$$\mathbf{y} = \mathbf{h}\left(\mathbf{x}_p\right) + \mathbf{e} \tag{2.7}$$

where $\mathbf{u}$ is a commanded velocity. In many cases we can choose a parameterization of $\mathbf{u}$ such that $\mathbf{u} = \dot{\mathbf{x}}_p$ with $\mathbf{f}$ as the identity function. A controller

$$\mathbf{u} = \mathbf{f}^{-1}\left(\mathbf{K}\mathbf{J}^{\dagger}(\mathbf{x}_p)\left(\mathbf{y}_r - \mathbf{y}\right)\right) \tag{2.8}$$

where $\mathbf{J} = \partial \mathbf{h} / \partial \mathbf{x}_p$ is the *image Jacobian* or *interaction matrix*, and where $\mathbf{K}$ is a positive definite matrix, will in general drive the system so that $\mathbf{y} \rightarrow \mathbf{y}_r$, see [Hutchinson *et al.*, 1996; Espiau *et al.*, 1992].

The advantage of image-based techniques is that positioning accuracy is independent of calibration accuracy, especially if the image-space reference trajectory has been defined in a teach-by-showing approach [Hutchinson *et al.*, 1996]. Therefore, image-based techniques are suitable in situations where less information about the system parameters and environment is available. It should be noted however, that the image Jacobian $\mathbf{J}(\mathbf{x}_p)$ is in general a function of the task-space coordinates $\mathbf{x}_p$, specifically the depth (the Z-coordinate as shown in Fig. A.1) of the features with respect to the camera. Therefore, some task-space quantities need to be estimated or approximated, just as for the position-based methods, described in the next section. Additionally, in more complex servoing tasks the desired trajectories are often more naturally defined in task space, and Cartesian models of the environment may be required for proper task and trajectory planning.

Position-Based Control. In position based techniques, the control signal is computed based on a value of $\mathbf{x}_p$, computed from the image data $\mathbf{y}$ by for instance so called *pose estimation*. In some situations, such as for a small number of point features, there exist analytic solutions to the pose estimation problem. Alternatively, non-linear least-squares and other optimization based methods may be used. Usually, the controllers are implemented in discrete time, and the value of $\mathbf{x}_p$ from the previous step may be used as a starting point for the iteration.

For control purposes, or if predictive capabilities are desired for increased reliability of feature tracking, observer-based tracking are useful. Traditionally, such methods are based on the Extended Kalman Filter (EKF) or other linearized observers. In recent years, alternative methods based on Sequential Monte Carlo methods, usually referred to as *Particle Filters*, have generated substantial research interest [Doucet *et al.*, 2001]. In visual tracking, the most well known of these methods is the CONDENSATION algorithm [Isard and Blake, 1998].

The main advantage of position-based methods is that the reference trajectory, the measured position, as well as the control signal, can all be defined in task-space coordinates. This simplifies the control problem, as the dynamic model is usually expressed in task-space, usually

Cartesian space. The main drawback of position-based control is that it is less robust to calibration errors. Errors in camera calibration parameters will lead to errors in estimated pose, and consequent errors in trajectories [Hutchinson *et al.*, 1996].

Calibration

In almost all applications using vision based control, some calibration procedure is required in order to find estimates of system parameters. The most common examples of such parameters are camera positions and internal camera parameters. Many different techniques exist for off-line and online calibration of these parameters, here we will give a brief review of the calibration techniques used in this work.

Intrinsic Camera Parameters. The intrinsic, or internal, camera parameters describe the internal structure of the camera. For the pinhole camera, as described in Appendix A, the intrinsic camera parameters are the focal length, skew, principal point, aspect ratio, and possibly radial- and tangential distortion coefficients. Many methods for camera calibration use a number of images of a special *calibration object* to estimate the intrinsic and extrinsic camera parameters. The extrinsic, or external, parameters describe the object position with respect to the camera coordinate system. The calibration often consists of one or several planar surfaces, covered with accurately placed markers. The images are taken from several different positions and orientations. In [Zhang, 1999], a method is presented in which both intrinsic and extrinsic camera parameters are estimated using a linear method followed by a nonlinear minimization step, based on several images of a planar pattern. The method will find the set of parameters that minimizes the reprojected image errors in a least-squares sense.

Extrinsic Camera Parameters and Hand-Eye Calibration. In addition to the internal camera parameters, it is required to establish the geometric relations between the sensor- and actuation coordinate systems. The calculation of the relative position and orientation between the robot end-effector and a camera, which is mounted rigidly on the end-effector, is referred to as hand-eye calibration. Finding this relationship between the positions of the sensor- and actuator frames is a standard problem in vision guided robotics, and many different

methods exist. The problem is usually formulated as finding the unknown transformation $\boldsymbol{X}$ from the hand-eye equation

$$\boldsymbol{AX} = \boldsymbol{XB}, \tag{2.9}$$

which is a special case of the Sylvester equation [Andreff *et al.*, 2001], where $\boldsymbol{A} = \boldsymbol{A}_2^{-1}\boldsymbol{A}_1$ and $\boldsymbol{B} = \boldsymbol{B}_2\boldsymbol{B}_1^{-1}$ are given by measurements of the position and orientation of the robot hand, and of the camera with respect to some object in the world, see Fig. 2.1. Typically a number of movements are performed to get measurements of different $\boldsymbol{A}$ and $\boldsymbol{B}$, and use them to solve for $\boldsymbol{X}$ in Eq. (2.9). Both linear [Tsai and Lenz, 1989a; Daniilidis, 1999] and non-linear methods [Horaud and Dornaika, 1995] have been suggested. In [Tsai and Lenz, 1989a], it was shown that at least two motions with non-parallel rotation axes are required, and the problem was solved by dividing Eq. (2.9) into two equations for rotation and translation, respectively. The method of [Daniilidis, 1999] uses a dual quaternion representation of $\boldsymbol{A}$ and $\boldsymbol{B}$ to simultaneously solve for the rotation- and translation parts of $\boldsymbol{X}$ using linear methods.

For a multi-camera system, we also need to find the relative positions of the sensors. When the cameras are fixed in the workspace, we can attach the calibration object to the robot end-effector itself. In Appendix B, the method used for multi-camera calibration in this thesis is presented.

Robust Image Processing and Data Correspondence

In all visual servoing methods, the raw image data from the cameras need to be compressed into a more compact representation in order for the controller to be able to compute the control signal. The first step is the image processing, where relevant image data is extracted into the measurement $\mathbf{y}$.

The raw image data is usually obtained in several channels, each consisting of an array of data of dimensions height$\times$width, with height and width typically somewhere between 200 and 1600. For a digital camera, such as the ones considered in this thesis, the number of channels is usually one, for pure intensity or grayscale images, or three, for YUV or RGB color space images. The amount of data for a single

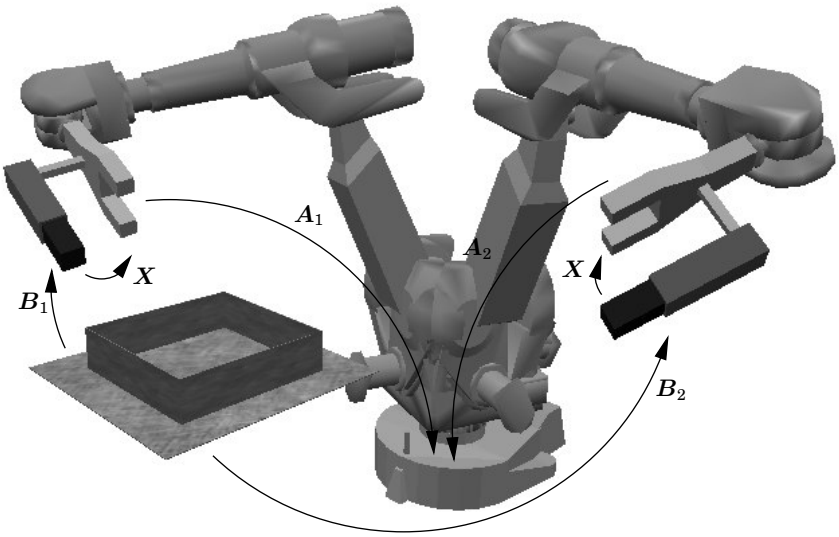

Figure 2.1 Two different positions of the robot, with the frames relevant to hand-eye calibration.

image is generally in the range between 100 kB and 4 MB, and processing the entire image at a sufficiently fast sample rate may require very large computational resources. Methods that process the entire image have been developed, such as methods based on optical flow [Allen *et al.*, 1991], and eigenspace methods, in which the image space is represented by a small number of eigenimages, see [Schuurman and Capson, 2004; Deguchi and Noguchi, 1996].

For real time applications however, most methods work by extracting positions of *image features*. Features are either parts of the image with sharp changes in intensity, such as edges, corners or markers, or textured and/or colored surface patches. The first type can often be detected and localized using convolutions with suitably chosen kernels, while textured surface patches can be found and tracked using area correlation based methods, such as the SSD (Sum of Squared Differences) method.

Robustness. In all visual processing systems, there exists a risk for false matches. In order to design the system to be robust to such image processing errors, methods such as RANSAC (Random Sampling Consensus) [Fischler and Bolles, 1981] or ICP (Iterated Closest Point) [Besl and McKay, 1992], are often used to match features and eliminate outliers from data sets. Such data association methods are useful for robust pose estimation and 3D-reconstruction in the presence of large numbers of outliers.

In order to minimize the number of false matches, feature based methods are often combined with window-based tracking, in which features are followed through the image sequence frame by frame. By searching for the feature only in a small image window around the predicted position, we increase efficiency and decrease the risk for false matches [Hutchinson *et al.*, 1996]. If an observer-based pose estimation technique is used, the position of the feature search windows can be obtained from the predicted position $\mathbf{x}_p$ computed at the previous sample.

Nonlinear Estimation for Visual Tracking

In dynamic vision problems, it is often desired to estimate the states of a general nonlinear system

$$\mathbf{x}(k+1) = \mathbf{f}_d\left(\mathbf{x}(k), \mathbf{u}(k), \mathbf{v}(k)\right) \tag{2.10}$$

$$\mathbf{y}(k) = \mathbf{h}_d\left(\mathbf{x}(k), \mathbf{e}(k)\right) \tag{2.11}$$

usually expressed in discrete time. The main difficulties in such problems are the nonlinearities of the system, together with potentially complex noise models and parameterizations of the state space.

Kalman Filtering Methods. Traditionally, the most common method for nonlinear estimation in dynamic vision problems has been the Extended Kalman Filter (EKF). The EKF is obtained by using a standard Kalman Filter and linearizing the process model around the current

estimates. The equations for the EKF are given by the equations

$$\begin{aligned}
\hat{\mathbf{x}}(k+1|k) &= \mathbf{f}_d(\hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{0}) && (2.12)\\
\mathbf{P}(k+1|k) &= \mathbf{A}_k \mathbf{P}(k) \mathbf{A}_k^T + \mathbf{W}_k \mathbf{Q}_k \mathbf{W}_k^T && (2.13)\\
\mathbf{K}_k &= \mathbf{P}(k|k\text{–}1)\mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}(k|k\text{–}1)\mathbf{H}_k^T + \mathbf{E}_k \mathbf{R}_k \mathbf{E}_k^T)^{-1} && (2.14)\\
\hat{\mathbf{x}}(k) &= \hat{\mathbf{x}}(k|k\text{–}1) + \mathbf{K}_k(\mathbf{y}(k) - \mathbf{h}_d(\hat{\mathbf{x}}(k|k\text{–}1), \mathbf{0})) && (2.15)\\
\mathbf{P}(k) &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}(k|k\text{–}1) && (2.16)
\end{aligned}$$

where $\mathbf{Q}_k$ and $\mathbf{R}_k$ are the covariance matrices for $\mathbf{v}(k)$ and $\mathbf{e}(k)$, and where

$$\begin{aligned}
\mathbf{A}_k &= \frac{\partial \mathbf{f}_d}{\partial \mathbf{x}}(\hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{0}) && (2.17)\\
\mathbf{W}_k &= \frac{\partial \mathbf{f}_d}{\partial \mathbf{v}}(\hat{\mathbf{x}}(k), \mathbf{u}(k), \mathbf{0}) && (2.18)\\
\mathbf{H}_k &= \frac{\partial \mathbf{h}_d}{\partial \mathbf{x}}(\hat{\mathbf{x}}(k), \mathbf{0}) && (2.19)\\
\mathbf{E}_k &= \frac{\partial \mathbf{h}_d}{\partial \mathbf{e}}(\hat{\mathbf{x}}(k), \mathbf{0}) && (2.20)\\
& && (2.21)
\end{aligned}$$

the Jacobians of $\mathbf{f}_d$ and $\mathbf{h}_d$ from the process model in Eq. (2.10) and Eq. (2.11).

Although the EKF works well in many vision applications, it has several important problems. A practical problem is that in many applications the measurement $\mathbf{y} \in I\!R^n$ is a very high-dimensional vector. The most computationally expensive part of the EKF computations is then the update of the Kalman gain $\mathbf{K}_k$ in Eq. (2.14), which will have a time complexity of $O(n^3)$. In [Wunsch and Hirzinger, 1997] it was suggested to use to EKF with the measurement defined in task space instead of image space, thereby decreasing the dimension of $\mathbf{y}$. For systems with linear state dynamics but a nonlinear measurement equation $\mathbf{h}$, it is often possible to use a Kalman filter with $O(n)$ that linearizes only the measurement equation and uses a covariance estimate $(\mathbf{H}_k^T \mathbf{H}_k)^{-1}\sigma^2$, as shown in Chapter 5.

The linearized and approximative nature of the EKF is also a problem in vision applications, where both the state dynamics and the

measurement equations are usually highly nonlinear. Recently, the so called *unscented transformation* has been introduced as a method to propagate mean and covariance information through nonlinear transformations. The resulting Unscented Kalman Filter (UKF) has been reported to perform significantly better than the EKF in many applications, both in accuracy and in computational efficiency [Julier and Uhlmann, 2004]. The UKF has also been interpreted as a special case of the so called *linear regression Kalman Filter* (LRKF), in which the nonlinear process and measurement functions $\mathbf{f}$ and $\mathbf{h}$ by statistical linear regression, see the note by [Lefebvre *et al.*, 2002] and the reply.

Particle Filter. Particle filtering was presented in [Gordon *et al.*, 1993] as an alternative to the EKF. The key idea in particle filtering is to use a sample based representation of the conditional probability density function $p(\mathbf{x}_k|\mathbf{y}_{1:k})$ of the state given the measurements. This pdf is represented by a set of random samples or particles with associated weights, which are updated based on the current measurements and particle likelihoods. As the number of particles increases, this characterization approaches that of the optimal Bayesian estimate [Arulampalam *et al.*, 2002].

The main advantage of particle filters is their ability to handle non-Gaussian, non-linear systems without explicit linearization. This makes them suitable for problems in computer vision, where nonlinear motion models and complex state and position representations are common. Handling multi-modal probability distributions is especially important for robustness in visual tracking. In many situations, severe background clutter can easily cause the EKF to lose track, due to the unimodal nature of the probability distributions [Isard and Blake, 1998]. A drawback is the large computational power required, as the number of particles needed in practice increases rapidly with the dimension of the state space. Fewer particles may be used if particles are chosen according to some suitable *proposal distribution* [Arulampalam *et al.*, 2002]. Better proposal distributions may be generated by combining the particle filter with Kalman filter or unscented filters, into a Kalman particle filter or unscented particle filter [Li *et al.*, 2003].

2.3 Force Control and Sensor Fusion

The nature and limited accuracy of vision based control makes it difficult to control interaction with objects in the environment. An interesting solution is to combine force control and visual servoing in a multi-sensor control system. Perhaps the most obvious approach to the problem is to combine the measurements from the cameras and force sensor using multi-sensor fusion methods. However, force- and visual sensors are fundamentally different in that they measure very different physical phenomena, which could make such an approach difficult to use in practice.

Over the last decade, some work on image based vision/force control has been presented. In [Nelson *et al.*, 1995] three different strategies are presented, traded control, hybrid control, and shared control. In traded control each degree of freedom is controlled by both force- and visual control, with switching between the sensors based on the sensor signals. In hybrid control the task space is divided into orthogonal force- and vision controlled directions, so that each degree of freedom is controlled by one sensor only. Finally, in shared control both sensors are used simultaneously in each degree of freedom. In this case, the stability analysis must take coupling effects between the force control and vision control actions into account, such as the effects of inertial forces on the force sensor [Nelson *et al.*, 1995].

In [Zhou *et al.*, 1998] an application of image-based vision/force control in micro-manipulation is demonstrated. They use a traded control law which switches between proportional force control and optimal image-based visual servoing. The method presented in [Baeten *et al.*, 1999] uses Mason's task frame and a high level task description to determine how to use each sensor in hybrid force/vision control. [Hosoda *et al.*, 1996] present a shared adaptive technique, where the image Jacobian and the slope of the constraint surface are estimated online from sensor data. Another hybrid and adaptive technique has been presented in [Pichler and Jägersand, 2000].

An application of position based force/vision control in flexible assembly is presented in [Jörg *et al.*, 2000], with a demonstration of mating of moving parts. The pose estimation was based on features extracted with the Hough Transform [Trucco and Verri, 1998], using a non-linear Kalman filter for estimation of the circular motion of the

moving target. Experimental results with a hybrid technique which takes the robot dynamics into account was presented in [Xiao *et al.*, 2000].

As an alternative to direct force control, impedance control aims to achieve a certain dynamical behavior of the end-effector position and orientation in response to external forces [Hogan, 1985; Siciliano and Villani, 1999]. Using an inner motion control structure, the motion controller is made to track the pose of the so called *compliant frame*, denoted by Σ_c. The impedance relation is a relation on the form

$$\mathbf{M}_I \frac{d^2\mathbf{x}}{dt^2} + \mathbf{D}_I \frac{d\mathbf{x}}{dt} + \mathbf{K}_I \mathbf{x} = \mathbf{f} \tag{2.22}$$

where $\mathbf{x}$ is the relative position of the compliant frame with respect to the reference frame Σ_d, $\mathbf{f}$ is the external force, and $\mathbf{M}_I$, $\mathbf{D}_I$ and $\mathbf{K}_I$ are positive definite matrices which can be interpreted as the effective mass, damping, and stiffness, respectively. In [Morel *et al.*, 1998] the use of vision/impedance control is proposed, and demonstrated in a peg-in-the-hole insertion task. The target impedance was chosen as a pure damping. It was shown that for low bandwidth a separation property holds, so that overall stability is guaranteed if the force- and impedance controllers are stable separately.

3

Interface for External Control of an Industrial Robot

3.1 Introduction

In general, industrial robots are designed to handle repetitive tasks in well-known, well structured environments. There are many types of tasks however, in which time-consuming calibration procedures or large workpiece variations make it difficult for standard industrial robots to operate efficiently. In several cases, external sensors and feedback can be used in order to overcome these difficulties. Examples of such tasks are polishing and other material removal procedures, in which some type of mechanical compliance is required in order to avoid damage to the workpiece. As an alternative to passive compliance, active force sensing and feedback control can be used in order to accurately control the interaction between the robot and the workpiece. Unfortunately, even in modern robot control systems, there are no interfaces which can be used for feedback control with sufficient performance from external sensors.

This chapter presents an interface developed for external control, designed by extending an ABB S4CPlus industrial robot control sys-

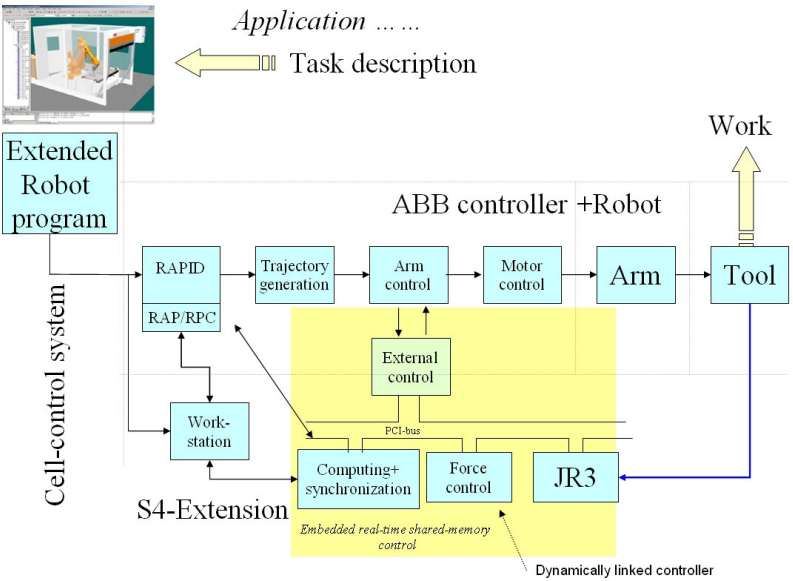

Figure 3.1 Structure of the extended ABB S4CPlus control system, with support for external control from JR3 force sensors. The references of the S4CPlus system are modified at a 4 ms sampling rate.

tem. Results from experiments, where the designed interface was used for force controlled grinding and deburring, are presented.

3.2 Structure of S4CPlus Extensions

In this section a brief description of the structure of the extended ABB S4CPlus control system is given. A more detailed desription of the complete system, together with a discussion of other issues considered in the design and implementation, is given in [Blomdell *et al.*, 2004].

The structure of the extended ABB S4CPlus control system can be seen in Fig. 3.1. In standard S4CPlus, the high-level task descrip-

tion is converted into program code written in the robot programming language RAPID. The RAPID program execution will then convert the instructions into trajectories for the internal arm and motor controllers responsible for the low-level motion control. The standard S4CPlus controller does not permit low-level external sensor feedback. On a high level there is a possibility to read sensors via customer IO, and influence the robot task according to instructions written in the RAPID language. However, since the sampling frequency is limited to around 10 Hz, and large time delays are present in the trajectory generation step, the achievable bandwidth on the RAPID level is far too low for most force control applications.

In the extended system, the RAPID program is extended with instructions for the external control, such as references, parameters, and activation/deactivation commands for the force controllers. The external instructions are encoded as xml-style tags and added as comments in the RAPID code. The part of the RAPID program without the extended instructions is then sent to and interpreted by the S4CPlus system and used in the trajectory generation, while the extended instructions are processed by a Master PC, responsible for communication and synchronization between the S4CPlus and the external controller.

For safety and efficiency reasons, the external controllers are implemented on a separate Motorola PPC-G4 PrPMC-800 processor board, mounted on an Alpha-Data PMC-to-PCI carrier board with a local PCI bus. In each sample, the references and parameters necessary for the external control are copied from the S4CPlus to the external controller board over the PCI bus, using a shared memory interface between the the built-in motion control and the external controller. The external controller modifies the references for position, velocity and torque according to sensor data and the active control algorithm, and values are copied back to the S4CPlus system, where safety checking is performed on the updated references.

To accomplish interrupt driven hard real-time execution of the controllers with shared memory communication, the external controller is run as a Linux kernel module on the PowerPC board. All parts of the force controller were implemented as Simulink block diagrams, see Fig. 3.7. The Simulink models were converted into C code using Real-Time Workshop from MathWorks, and cross compiled to the target computer and incrementally linked to form the Linux kernel module.

The Controller Interface

Fig. 3.2 shows the Simulink interface between the external controller and the S4CPlus system. This interface was developed primarily for force control applications, but can easily be used or extended for other types of sensor feedback. The following is a description of all input- and output signals currently defined. The inputs are obtained from the S4CPlus system at a 4 ms sampling rate, giving an effective sampling frequency of 250 Hz for the external control.

Irb2Ext_vect is the input vector signal from the S4CPlus system, generated in the interface to the controller. This signal is divided into a number of subsignals as described below.

parKp, parKv, parKi, parTrqMin, parTrqMax are internal controller parameters of the S4CPlus motion control, and are currently not used.

posRaw, posFlt are the raw and filtered motor angle position measurements, respectively.

velRaw, velFlt are the raw and filtered motor angular velocity measurements, respectively.

velOut is currently not used.

trqRaw, trqFlt are the raw and filtered motor torque measurements, respectively.

trqOut is currently not used.

posRef is the position reference of the internal motion controller.

velRef is the velocity reference of the internal motion controller.

trqRef is the motor torque reference of the internal motion controller.

trqFfw is a motor torque feedforward signal.

trqDis, rP, uP are currently not used.

Ext2Irb_vect is the output vector signal copied back to the S4CPlus system. This signal is divided into a number of subsignals as described below.

parKx are the updated internal parameters of the S4CPlus motion control.

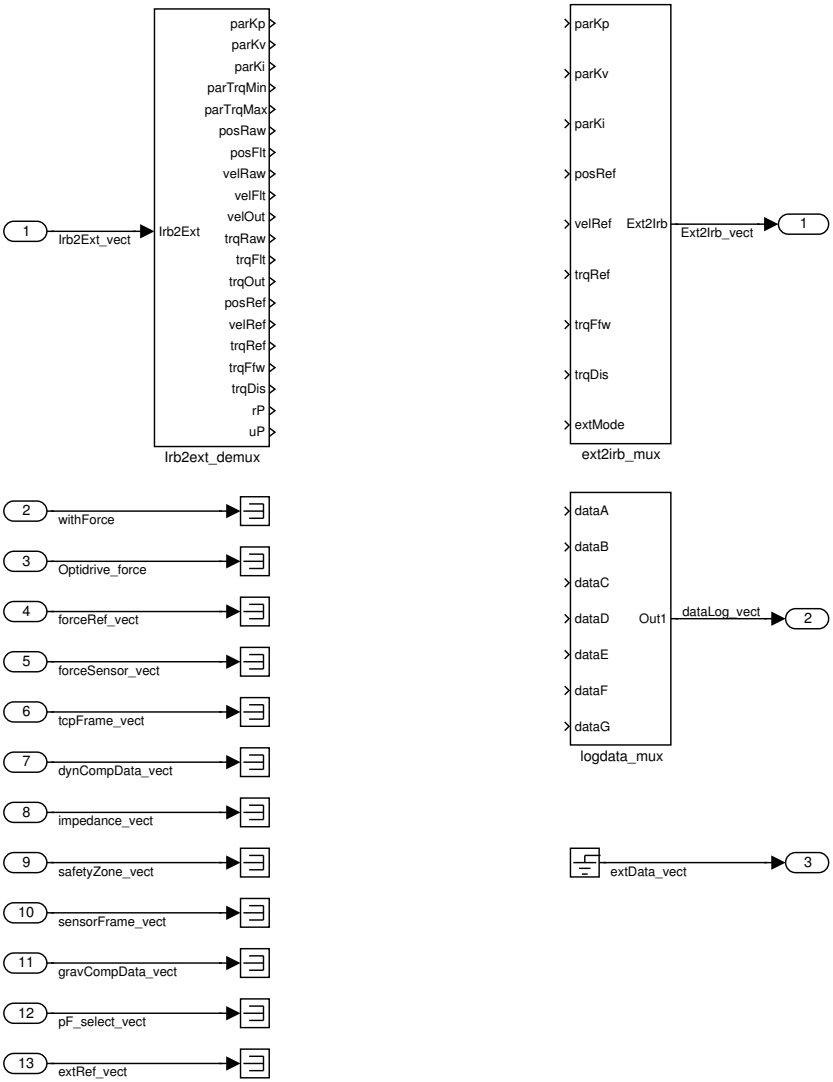

Figure 3.2 Structure of the signal interface between the external controller and the S4CPlus system.

posRef, velRef, trqRef are the updated references for the internal motion controller.

trqFfw is the updated motor torque feedforward signal.

trqDis is currently not used.

extMode is currently not used.

withForce is an software activation switch for the external control, set to zero when the external control is deactivated. This is normally used to allow the external controller to safely reset its states to safe values.

Optidrive_force is the force measurement from the optional compliant Optidrive force sensor.

forceRef_vect is the force reference, expressed in the coordinate system of the tool.

forceSensor_vect is the force measurement from the 6-axis JR3 force sensor, expressed in the internal coordinate system of the sensor.

tcpFrame_vect is the pose of the tool frame with respect to the flange of the robot hand, expressed as an orientation quaternion and a translation vector, and used for conversion of the force vector signals between the frames.

dynCompData_vect, gravCompData_vect contains data about the geometry, mass and moments of inertial of the tool. This is used for compensation of the effects of inertial- and gravity forces on the force measurement.

impedance_vect is a vector of impedance parameters for the controller, containing the diagonal elements of the matrices $\mathbf{M}_I$, $\mathbf{D}_I$ and $\mathbf{K}_I$ in Eq. (3.1) and (3.2).

safetyZone_vect is an optional vector of safety zone parameters, describing the zone around the nominal robot trajectory in which the references are allowed to be modified by the external controller.

sensorFrame_vect is the pose of the force sensor frame with respect to the flange of the robot hand, and is used for conversion of the force vector signals.

pF_select_vect is a vector describing the directions in the tool frame in which the external control is activated in a hybrid force/position controller.

extRef_vect is an external measurement signal from an optional higher level sensor, such as a vision system or a laser tracker.

dataLog_vect contains up to seven 6-vectors, which are sent to the Master PC for display or logging.

extData_vect is an output data vector, which is sent to the Master PC and used for synchronization.

3.3 Force Controller Structure

The Simulink model of the designed impedance controller is seen in Fig. 3.7. The central part of the controller is the impedance block. The impedance block is described by the rotational and translational impedance relations

$$\mathbf{M}_{I_{\text{trans}}} \frac{d^2\mathbf{t}_{dc}}{dt^2} + \mathbf{D}_{I_{\text{trans}}} \frac{d\mathbf{t}_{dc}}{dt} + \mathbf{K}_{I_{\text{trans}}} \mathbf{t}_{dc} = \mathbf{f} - \mathbf{f}_r \quad (3.1)$$

$$\mathbf{M}_{I_{\text{rot}}} \frac{d^2\phi_{dc}}{dt^2} + \mathbf{D}_{I_{\text{rot}}} \frac{d\phi_{dc}}{dt} + \mathbf{K}_{I_{\text{rot}}} \phi_{dc} = \mathbf{T}^T(\phi_{dc})(\tau - \tau_r) \quad (3.2)$$

where $\mathbf{f}$ and τ are the measured force and torque vectors, which has been compensated for gravity and expressed in the tool coordinate system, $\mathbf{f}_r$ and τ_r is the reference force/torque vector, $\mathbf{t}_{dc} = \mathbf{t}_c - \mathbf{t}_d$ and $\phi_{dc} = \phi(\mathbf{R}_c^T \mathbf{R}_d)$ are the translation vector and Euler XYZ angles describing the pose of the compliant frame Σ_c with respect to the reference frame Σ_d. $\mathbf{T}$ is a Jacobian matrix relating angular velocities to time derivatives of Euler angles [Siciliano and Villani, 1999].

In each sample, the contact force and torque are measured using a force/torque sensor and compensated for the effects of gravity. The position and velocity of the compliant frame are obtained from integrating Eqs. (3.1) and (3.2), and add them to the position and velocity of the reference frame. The corresponding positions and velocities of the robot joints are calculated through the inverse kinematics, and

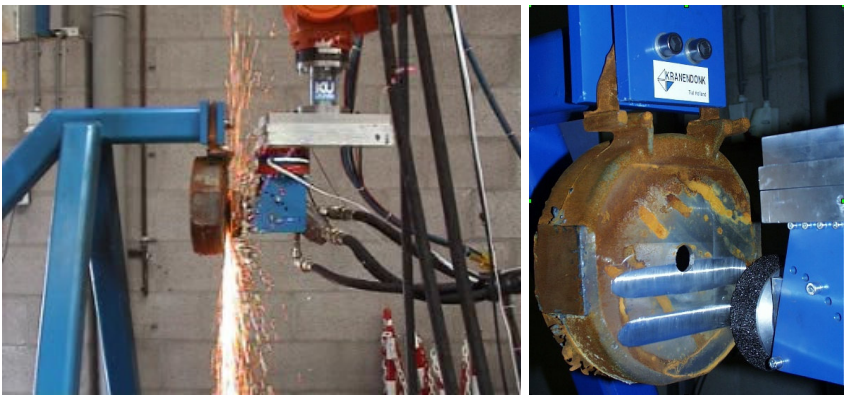

Figure 3.3 Grinding with Irb 6400 industrial robot, constant force reference of 150 N.

used as the new reference for the robot motion control. If necessary, the effect of the contact force on the motion control can be compensated for by feed-forward from the measured disturbance forces to the control torque.

Using the impedance controller, the dynamic behavior of the robot in contact can be influenced directly from the parameters given in a robot program. Many different types of behaviors can be achieved. By setting the stiffness parameters $\mathbf{K}_{I_{\text{trans}}}$ and $\mathbf{K}_{I_{\text{rot}}}$ to zero, we obtain a controller with integral action, which will dominate the position control in order to achieve the desired contact force [Siciliano and Villani, 1999]. Such a controller can be seen as a parallel position/force control structure, where all available degrees of freedom are controlled by both position and force control simultaneously. Alternatively, a hybrid position/force control structure could be used, where the contact force is accurately controlled in some degrees of freedom, while the remaining degrees of freedom are under position control.

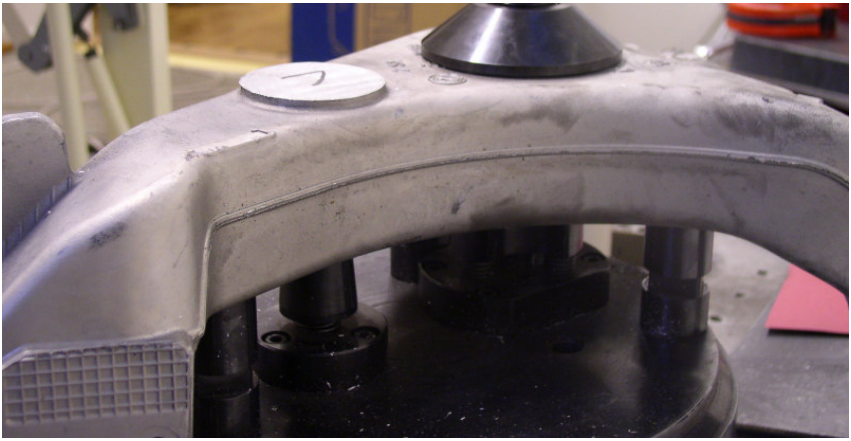

Figure 3.4 The workpiece of aluminum used in the deburring experiments. The burr height varied between 2 and 7 mm.

3.4 Experiments

The impedance controller was used in force controlled grinding and deburring as part of the Autofett project (www.autofett.org).

Force Controlled Grinding

The impedance controller was used in grinding experiments on an ABB Irb6400 robot at the company Kranendonk, the Netherlands. The special grinding tool used was developed at KU Leuven, Belgium. Instead of the 6-axis JR3 force sensor, a compliant Optidrive sensor was used to measure the contact forces. The results from an experiment with a force reference of 150 N in the normal direction of the surface, and a speed along the surface of 10 mm/s, is shown in Fig. 3.5. The disturbance that can be seen at time $t \approx 15$ s is caused by a resonance, which occurs when the grinding tool moves across the hole in the workpiece seen in Fig. 3.3.

Force Controlled Deburring

In order to illustrate the use of the stiff 6-axis JR3 force sensor on a workpiece with more complex geometry, a deburring experiment was

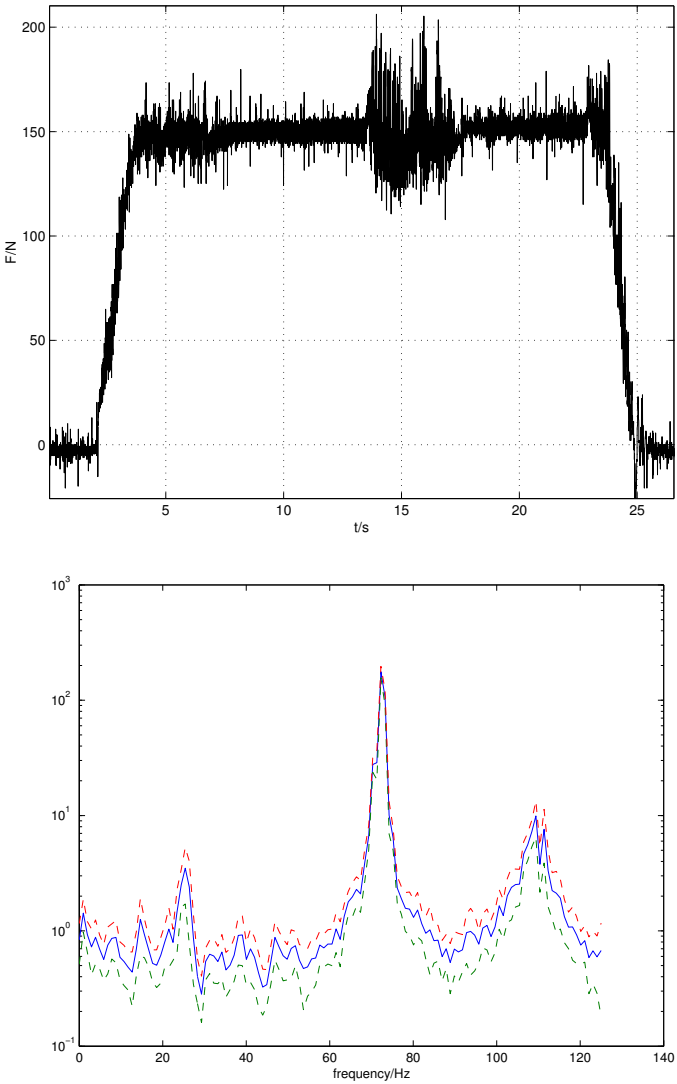

Figure 3.5 Top: Contact force during grinding experiment, with force reference 150 N. Bottom: Power spectrum estimate of the resulting force signal, clearly showing the rotation frequency of the grinding stone at 72 Hz. Additional resonances of the workpiece and fixture can also be seen.

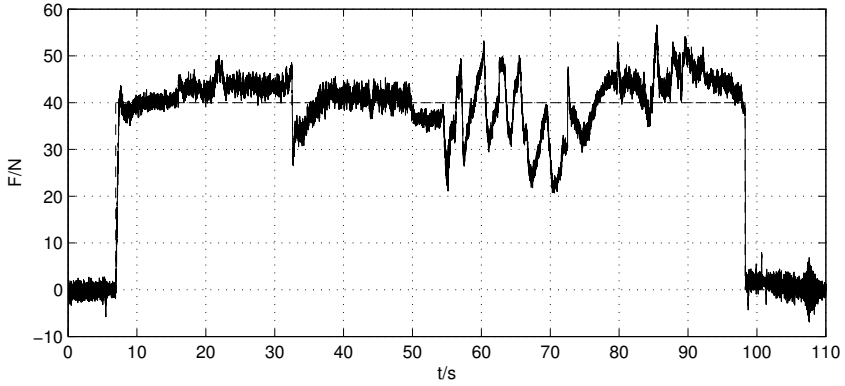

Figure 3.6 Contact force and during deburring experiment, with force reference 40 N.

performed using the workpiece shown in Fig. 3.4. The experiments were performed at the Robot Lab of Lund Institute of Technology, using an Irb 2400 industrial robot.

An approximate trajectory was programmed along the burr on the surface of the workpiece, and the force control was programmed to maintain a desired contact force of 40 N in the normal direction of the surface. In order to follow the trajectory, both the tool orientation and the force controlled direction had to be changed at certain points along the trajectory. The reorientations, together with the high stiffness of the contact and the roughness of the surface on which the milling tool is rolling, causes the variations in the measured contact force.

3.5 Summary

In this chapter the interface for external control has been described, designed by extending an ABB S4CPlus industrial robot control system. The system extensions make it possible to modify the references of the internal motion control on the 4 ms level. This makes it useful for high bandwidth feedback control, such as contact force control.

Using the extended system, force- and impedance controllers have

been designed and tested. Results from experiments where the designed controllers were used for force controlled grinding and deburring are presented.

Further experiments, where the force controllers are combined with feedback from a multi-camera system will be presented in Chapter 5.

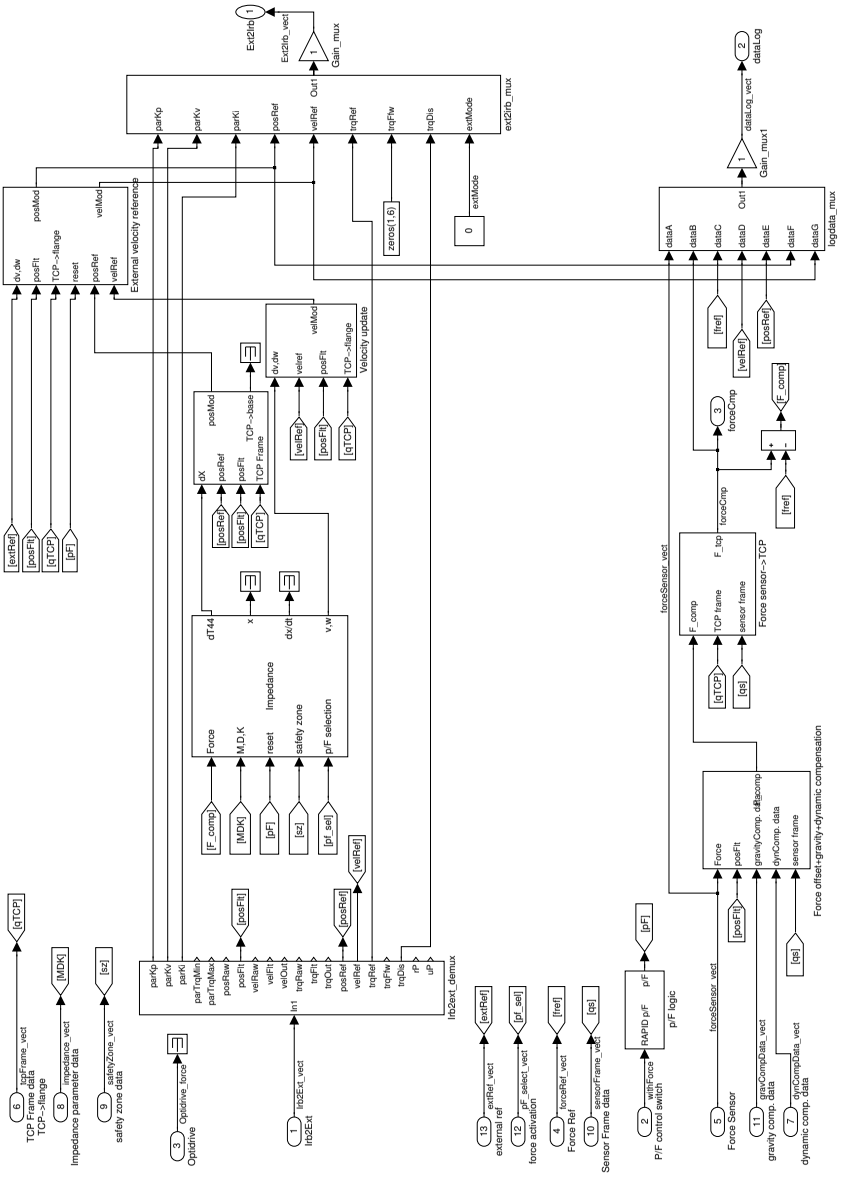

Figure 3.7 Structure of the impedance controller.

4

Image-Based Hybrid Vision/Force Control

4.1 Introduction

There are situations where a position-based visual tracking and servoing technique can not be used. One reason could be that the calibration accuracy of the camera system is insufficient for a reliable computation of task space position. In other situations the positioning task is naturally specified in image space, for instance when the robot tool is to follow a visible structure in the image, such as an edge or a pattern of markers. In these cases, a possible solution is to use an image based technique [Hutchinson *et al.*, 1996].

In this chapter, an image-based force/vision control algorithm is presented and demonstrated, based on a hybrid force/vision control structure. The control algorithm involves a force feedback control loop and a vision based reference trajectory as a feed-forward signal. Far away from any constraints, the robot can be controlled by unconstrained visual servoing only. When close to the constraint surface, one or several degrees of freedom should become force controlled in order to accurately control the interaction with the constraint surface. The remaining degrees of freedom should then be controlled by a constrained visual servoing algorithm. The unknown constraint equations can be estimated recursively from the available sensor data.

4.2 Image Based Visual Servoing

We assume a setup of the camera/robot system as described in Appendix B. Two fixed cameras are observing the tool, which is rigidly attached to the end-effector. We assume the dynamic model for a velocity controlled robot

$$\dot{\mathbf{x}}_p = \mathbf{f}(\mathbf{u}) \tag{4.1}$$

$$\mathbf{y} = \mathbf{h}(\mathbf{x}_p) + \mathbf{e} \tag{4.2}$$

where $\mathbf{x}_p$ is a parameterization of the robot end-effector position, $\mathbf{u}$ is a vector of the desired translational and angular velocity (a *velocity screw*), and $\mathbf{y}$ is a vector of image feature measurements.

In an image-based visual servo, the control error is defined directly in image space as $\mathbf{y}_r - \mathbf{y}$, where $\mathbf{y}_r$ is the desired position of the image features. A simple control law that would drive $\mathbf{y} \to \mathbf{y}_r$ is given by

$$\mathbf{u} = \mathbf{f}^{-1}\left(k_v \left[\mathbf{J}_v^{l,r}(\mathbf{x}_p)\right]^{\dagger} (\mathbf{y}_r - \mathbf{y})\right) \tag{4.3}$$

where k_v is a constant gain, and $\left[\mathbf{J}_v^{l,r}(\mathbf{x}_p)\right]^{\dagger}$ is the pseudo-inverse of the stereo *image Jacobian*, which relates image space velocities $\dot{\mathbf{y}}$ of the features to the corresponding end-effector velocity in Cartesian space by

$$\dot{\mathbf{y}} = \mathbf{J}_v^{l,r}(\mathbf{x}_p)\mathbf{u}. \tag{4.4}$$

Note that the image Jacobian is in general a function of the Cartesian coordinates $\mathbf{x}_p$, which means that we need some Cartesian information to calculate it exactly, usually the depth of the imaged points in the cameras. We can obtain approximate depth information by using the data from the calibration of the camera system and the robot kinematics. Alternatively, the depth information could be obtained from the stereo images. Some care needs to be taken however, as the robustness to depth estimation errors can be extremely poor in situations where the image Jacobian is poorly conditioned [Malis and Rives, 2003].

When using stereo cameras, the combined Jacobian for the stereo system is obtained by stacking the Jacobians for the individual cam-

eras [Martinet and Cervera, 2001]

$$\mathbf{J}_v^{l,r}(\mathbf{x}_p) = \begin{pmatrix} \mathbf{J}_v^l(\mathbf{x}_p)\mathbf{M}_b^l \\ \mathbf{J}_v^r(\mathbf{x}_p)\mathbf{M}_b^r \end{pmatrix} \tag{4.5}$$

where $\mathbf{M}_b^l$ and $\mathbf{M}_b^r$ are the transformation matrices for the screw, from the robot base frame to the left and right cameras respectively, and $\mathbf{J}_v^l(\mathbf{x}_p)$ and $\mathbf{J}_v^r(\mathbf{x}_p)$ are the Jacobians for the left and right cameras, expressed in the camera coordinate systems. The exact form of these Jacobians for point features can be found for instance in [Hutchinson *et al.*, 1996]. The screw transformation matrix for the cameras is given by

$$\mathbf{M}_b^i = \begin{pmatrix} \mathbf{R}_b^i & [\mathbf{t}_b^i\times]\,\mathbf{R}_b^i \\ \mathbf{0}_{3\times 3} & \mathbf{R}_b^i \end{pmatrix}, \quad i \in \{l, r\} \tag{4.6}$$

where $\mathbf{R}_b^i$ and $\mathbf{t}_b^i$ describe the relative position of camera i with respect to the robot base frame.

Constrained Motion

The constraint on the reference trajectories is that the motion should be in the plane $\hat{\mathbf{p}}$ in Cartesian space defined by

$$\mathbf{p}^T\mathbf{x}_p = 0 \tag{4.7}$$

where $\hat{\mathbf{p}} = (p_1 \;\; p_2 \;\; -1 \;\; p_4)^T$ and $\hat{\mathbf{x}}_p = (X \;\; Y \;\; Z \;\; 1)^T$. Differentiating this expression leads to an equation

$$\mathbf{p}^T\dot{\mathbf{x}}_p = 0 \tag{4.8}$$

for the constrained velocity $\dot{\mathbf{x}}_p = \begin{pmatrix}\dot{X} & \dot{Y} & \dot{Z}\end{pmatrix}^T$ of the end-effector, with $\mathbf{p} = (p_1 \;\; p_2 \;\; -1)^T$.

The purely translational motion on the surface of the plane is now given by

$$\begin{aligned} \dot{\mathbf{y}} &= \mathbf{J}_v^{l,r}(\mathbf{x}_p) \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ p_1 & p_2 \end{pmatrix} \begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} = \\ &= \mathbf{J}_{v,c}^{l,r}(\mathbf{x}_p) \begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} \end{aligned} \tag{4.9}$$

It is obvious that if $\mathbf{J}_v^{l,r}(\mathbf{x}_p)$ has full rank, so has the reduced image Jacobian $\mathbf{J}_{v,c}^{l,r}(\mathbf{x}_p)$. The constrained vision based control law becomes simply

$$\dot{\mathbf{x}}_c = k_v \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ p_1 & p_2 \end{pmatrix} \left[\mathbf{J}_{v,c}^{l,r}(\mathbf{x}_p)\right]^+ (\mathbf{y}_r - \mathbf{y}) \tag{4.10}$$

As the constraint itself, described by Eq. (4.7), is in general unknown, a method for its determination is required. A method based on local estimation of the constraint using measurements of forces and torques is suggested in [Xiao *et al.*, 1998]. Usually the low signal-to-noise ratio of the force sensor, in combination with large friction forces, makes this method less suitable. Instead we estimate the parameter vector $\hat{\mathbf{p}}$ using a recursive least-squares method and the equations

$$\begin{pmatrix} X_m & Y_m & (Z_m - F_z/k_z) & 1 \end{pmatrix} \hat{\mathbf{p}} = 0 \tag{4.11}$$

$$(p_1,\ p_2,\ -1) \left(\left[\mathbf{J}_v^{l,r}(\mathbf{r})\right]^+ (\mathbf{y}_r - \mathbf{y}) \right)^T = 0 \tag{4.12}$$

where X_m, Y_m and Z_m are the measured Cartesian coordinates for the end-effector obtained from the robot kinematics. Eq. (4.11) is derived from Eq. (4.7), where the z-coordinate is calculated from the measured force F_z and the stiffness k_z of the spring-mounted tool in the z-direction, the direction in which the tool is pointing. Eq. (4.12) is derived from Eq. (4.8) and the fact that in an accurately calibrated stereo system, the unconstrained Cartesian velocities $\dot{\mathbf{x}}_p$ obtained from the control law will produce trajectories that are in the direction of the target. If the system is moving between two points that both lie in the constraint plane, then all the velocity vectors $\dot{\mathbf{x}}_p(t)$ will be approximately parallel to the plane. The purpose of Eq. (4.12) is to use the predictive capability of the visual information in the estimation of the slope, which can be expected to speed up the convergence.

Note that in Eq. (4.11) it is required that the stiffness k_z is known. However, it is possible to use just a rough estimation, since the force control will keep F_z approximately constant. Because of this the slope of the plane should still be estimated correctly.

4.3 Hybrid Force/Vision Control

We use a proportional motion rate controller

$$\dot{\mathbf{x}}_F = \begin{pmatrix} \mathbf{0}_{2\times 1} \\ k_F(F_r - F) \end{pmatrix} \tag{4.13}$$

where F_r is the reference for the force F in the z-direction. If we combine this with the vision based reference trajectory of Eq. (4.10), the hybrid control law becomes

$$\begin{aligned} \dot{\mathbf{x}}_H &= \dot{\mathbf{x}}_F + \dot{\mathbf{x}}_c = \begin{pmatrix} \mathbf{0}_{2\times 1} \\ k_F(F_r - F) \end{pmatrix} + \\ &+ k_v \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ p_1 & p_2 \end{pmatrix} \left[\mathbf{J}^{l,r}_{v,c}(\mathbf{x}_p)\right]^+ (\mathbf{y}_r - \mathbf{y}) \end{aligned} \tag{4.14}$$

4.4 Example - Vision Supported Drawing

We have set up an experimental system in the Robotics Lab at the department of Automatic Control at Lund Institute of Technology. The system consists of a 6-degree-of-freedom ABB Industrial Robot 2000 equipped with a 6DOF JR3 force/torque sensor, and two Sony DFW-V300 digital cameras working at a frame rate of 30 images/second. The cameras send image data through a 400 Mbps IEEE-1394 connection to a standard 450 Mhz WindowsNT PC where the image processing is performed. The extracted feature point locations are then sent to a Sun Ultra60 computer running a Matlab/Simulink version of the vision/force controller. The sampling period of the controller is 67 ms. The low level joint position control is handled by an open robot control system, which is described in [Nilsson, 1996].

A simple and illustrative task is chosen to test the performance of the presented methods. Two objects are placed in the view of both cameras, a pen and a white-board. The exact position and orientation of the objects is unknown, but the pen is standing in the vertical position.

On the white-board, a number of dots are drawn in random positions. The system should be able to align the end-effector with the pen and grasp it, and use it to connect the dots on the white-board with lines as in Fig. 4.5. The control should keep the contact force constant during the drawing phase. The exact location of the board should be estimated accurately using available sensor data.

The experiment is divided into an off-line and an online phase. Off-line, the robot, with the tool replaced by a planar calibration object, moves into a number of different positions, and the locations of 48 coplanar features on the object are measured in each image. We then use the calibration procedure described in Appendix B to find the intrinsic camera parameters, as well as the sensor-sensor and sensor-actuator transformations $\mathbf{T}_{c_2}^{c_1}$ and $\mathbf{T}_{b}^{c_1}$ shown in Fig. B.1. Experiments with real and simulated data show that $\mathbf{T}_{b}^{c_1}$ and $\mathbf{T}_{c_2}^{c_1}$ can be estimated with an error in translation of approximately 2 cm, and an orientation error of 1° in the Euler angles, using four different positions of the robot end-effector. In theory, more accurate calibration could be obtained by using more calibration positions. In practice, however, other limiting factors exist, such as the accuracy of the models of the calibration object and camera nonlinearities, such as radial distortion. An analysis of the effects such errors can be found in [Zhang, 1999].

Online, the robot end-effector is aligned with the pen using 4-degree-of-freedom visual servoing. Once the pen is grasped, we use visual servoing to guide the pen to the board, and once contact with the board is established, the force/vision control makes the robot connect the dots.

Results

In Fig. 4.1 we see the measured force in the force controlled direction. The force control is switched on at $t = 5.3$ s and contact is achieved at $t = 5.5$ s. Note the large initial influence of the inertial forces during the acceleration at $t = 0$.

The final estimation of the plane parameters is

$$\hat{\mathbf{p}} = (-0.0461 \quad 0.0128 \quad -1 \quad 1.235)^T,$$

which can be compared to the true values

$$\hat{\mathbf{p}}_r = (-0.0478 \quad 0.0155 \quad -1 \quad 1.237)^T$$

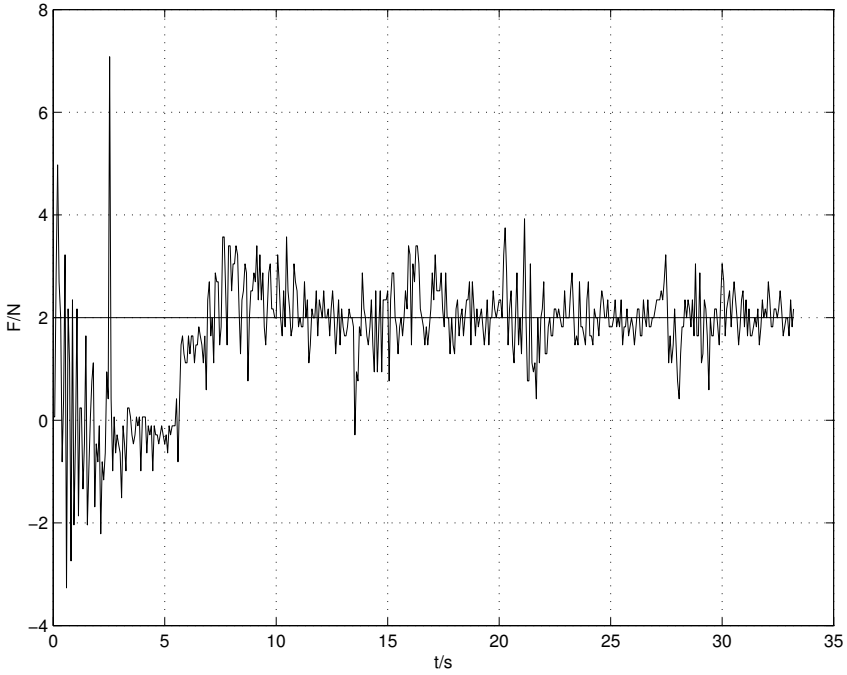

Figure 4.1 Measured force F and reference $F_r = 2$ N.

obtained from an accurate measurement where the robot was manually positioned so that it just made contact with the plane at five different locations. The recursively estimated parameters can be seen in Fig. 4.2. The estimation is started at $t = 6.7$ s. In Fig. 4.3 a), c), and d) we see the residuals for Eq. (4.11), their autocorrelation, and the histogram of the residuals. The residuals for Eq. (4.12) can be seen in Fig. 4.3 b). Note that $t = 0$ in Fig. 4.3 a) and b) corresponds to $t = 6.7$ s in Fig. 4.1.

The trajectory of the tip of the pen in Cartesian space can be seen in Fig. 4.4, and the corresponding image-space trajectories can be seen in Fig. 4.5.

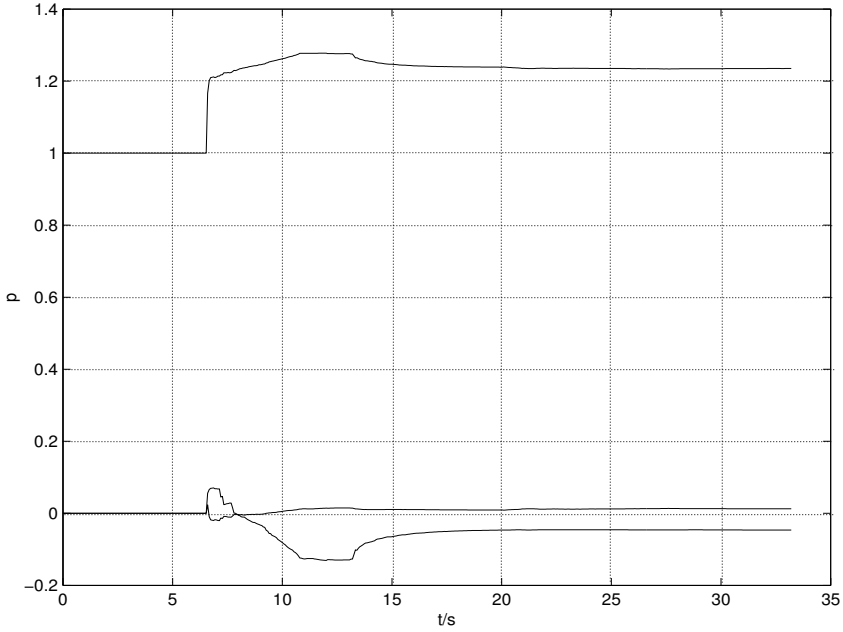

Figure 4.2 Plane parameters p_1, p_2 and p_4.

4.5 Discussion

We see from Fig. 4.1 that the force overshoots slightly at the beginning of the first line at $t \approx 7$s. The reason is that the estimate of the plane parameters $\hat{\mathbf{p}}$ has not yet converged, and the accuracy of the reference trajectories from the vision system is therefore limited by the relatively low accuracy of the cameras. The combined stiffness of the environment and the spring-mounted pen is estimated to 400 N/m, which means that the overshoot corresponds to an error of approximately 1.5 mm in the reference trajectory.

Other sources of error are the quantization effects and the noise resulting from errors in the image feature extraction, most clearly seen

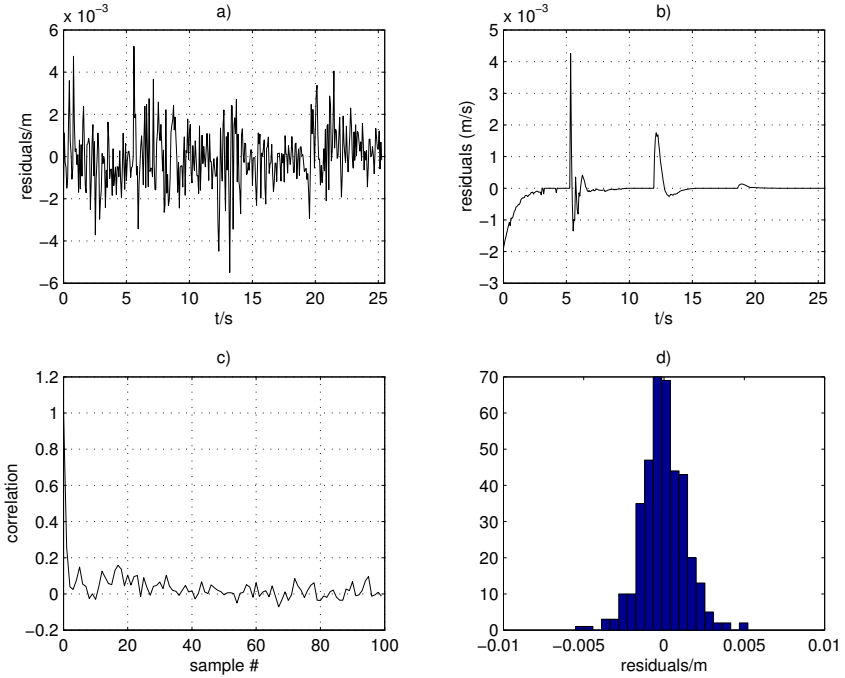

Figure 4.3 a) Residuals in Eq. (4.11). b) Residuals in Eq. (4.12). c) Autocorrelation of the residuals in Eq. (4.11). d) Histogram of the residuals in Eq. (4.11).

in Fig. 4.5. This will result in noise in the reference trajectories and the resulting contact forces, see Fig. 4.1.

The estimation of the plane parameters changes stepwise, with fast convergence to the final value at time $t = 13.3$ s, the start time for the drawing of the second line. The estimated values at $t < 13.3$ s reflect the slope of the plane along the first line. The small error in the estimation is caused by the noisy data from the force sensor, errors in the estimation of the stiffness of the spring, and calibration errors. Another reason is that the board is flexible, and is therefore deformed slightly by the contact forces.

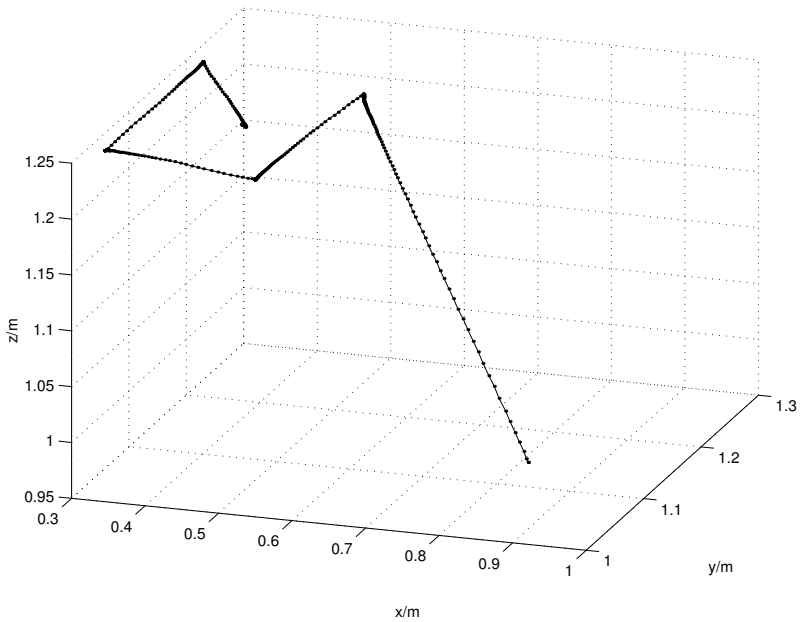

Figure 4.4 Trajectory of the pen, Cartesian space.

In Fig. 4.3 c) and d) we see that the residuals of Eq. (4.11) are approximately uncorrelated, Gaussian white noise. The residuals of Eq. (4.12) seen in Fig. 4.3 b) are of course not white noise, since they are affected not only by random image-space errors, but also by the constant errors in the Cartesian data estimated in the calibration. In general, the maximum error in $\dot{z}$ decrease to a value around $2 \cdot 10^{-3}$ m/s at a velocity in the x-y plane of 0.15 m/s, roughly corresponding to an error in the slope of the plane of around 1°. This agrees well with the estimated calibration accuracy.

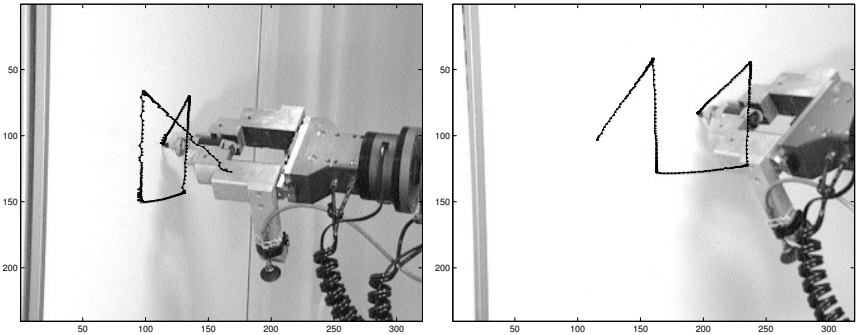

Figure 4.5 Pen trajectories in the image planes.

4.6 Conclusion

In this chapter, we have described a method for combining visual servoing methods with force control, based on explicit estimation of the position of an unknown planar constraint surface. The method differs from previous work [Xiao *et al.*, 1998] in that it does not rely on assumptions of negligible friction, or the possibility to recover the normal of the plane from accurate measurements of contact forces and torques. Instead, we use data from a calibrated robot and camera system to estimate the constraint location. The main drawbacks of this approach are that it requires the constraints to be (piecewise) planar surfaces, and that calibration is required. The method is shown to work in practice in experiments involving force controlled, vision guided drawing on a planar surface.

5

Position-Based Hybrid Vision/Force Control

5.1 Introduction

Position-based visual servoing techniques require some type of pose estimation, since the feedback law is defined in the workspace of the robot, rather than directly in the image. Accurate and robust tracking and estimation of the position of rigid objects using measurements from one or several cameras has been an active research topic for many years. Many methods for rigid body tracking work by minimizing some measure of the image space error as a function of the unknown position and orientation parameters, using standard non-linear optimization methods [Drummond and Cipolla, 1999; Martin and Horaud, 2002], or Kalman filtering techniques [Lippiello *et al.*, 2002; Olsson *et al.*, 2003]. In [Wunsch and Hirzinger, 1997] it was suggested that the output from the pose estimation should be used as input for the Kalman filter, in order to avoid the high computational complexity required when the output is a high-dimensional image-space vector. The position and orientation can be parameterized in different ways, such as roll-pitch-yaw angles [Lippiello *et al.*, 2002], quaternions or dual quaternions [Olsson *et al.*, 2003]. There are also various ways to measure the image space error, the most common measurements are the positions of point features [Lippiello *et al.*, 2002], lines, or point-

to-contour errors [Drummond and Cipolla, 1999; Martin and Horaud, 2002]. The point-to-contour method has a major advantage in that it does not require the exact matching of features, only the error in the normal direction at a number of points on a contour. This only requires a one-dimensional search for features (edges).

In this chapter we demonstrate how to achieve high performance six degree-of-freedom combined vision/force control for interaction with a stiff uncalibrated environment. It is shown how a process with linear dynamics in task space, together with a standard formulation for an edge-based rigid body tracker, can be used to design an observer with linear error dynamics, which avoids the high $O(n^3)$ time complexity of the Extended Kalman Filter described in Chapter 2.

5.2 Controller for Force/Vision Control

Modeling

Assume that M cameras are placed in fixed locations, viewing a target object whose position and orientation with respect to some fixed (world) coordinate system should be estimated. The position and orientation is parameterized as $\mathbf{x}_p \in I\!R^n$ where typically $n = 6$. The image data is compressed into a vector $\mathbf{y} \in I\!R^N$, usually the image space coordinates of corners, edges and other features. If the geometry of the target is known, $\mathbf{x}_p$ and $\mathbf{y}$ are related by the projection equations of the cameras

$$\mathbf{y} = \mathbf{h}(\mathbf{x}_p) \tag{5.1}$$

which is usually a very complex non-linear function. The most commonly used camera model is the homogeneous form pinhole camera projection equation, which in our case becomes

$$\mathbf{y}_i = \mathbf{h}_i(\mathbf{x}_p) = \frac{1}{Z_i}\mathbf{K}\mathbf{T}_{cw}\mathbf{T}_{wo}(\mathbf{x}_p)\mathbf{X}_i, \quad i \in [1, N] \tag{5.2}$$

where $\mathbf{K}$ is a matrix of intrinsic camera parameters, $\mathbf{X}_i$ is an object point expressed in the coordinate system of the object, Z_i is the depth of the point in the camera, and $\mathbf{T}_{cw}$ and $\mathbf{T}_{wo}(\mathbf{x}_p)$ are the homogeneous

coordinate transformation matrices between the target object and the world coordinate system, and between the world coordinate system and the camera, respectively. The parameterization $\mathbf{x}_p$ of $\mathbf{T}_{wo}$ is the unknown position/orientation to be estimated. In the following, the camera position $\mathbf{T}_{cw}$ is assumed to be known. This is not a restriction in situations where only the relative pose of two tracked objects is to be controlled, since the position of the world coordinate system is arbitrary. Only the relative positions of the cameras need to be accurately calibrated, in order to be able to relate measurements from different cameras.

We assume that the task space dynamics of the motion controlled manipulator can be modeled as a linear system, which together with the nonlinear measurement equation gives the Wiener-type model

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} \\ \mathbf{y} = \mathbf{h}(\mathbf{x}_p) \end{cases} \tag{5.3}$$

where $\mathbf{u}$ is the input, and $\mathbf{x}$ is the state vector typically consisting of the position $\mathbf{x}_p$ and velocity of the end-effector in the task space, and possibly other states depending on the model of the dynamics. For relatively low bandwidth systems, such as vision based controllers, the approximation of the complex closed loop robot dynamics with a linear system of relatively low order is a reasonable. The output $\mathbf{y}$ in Eq. (5.3) is the vector of image features obtained from the images, and $\mathbf{h}$ is given by Eq. (5.2), for each point.

For the pinhole camera, the task space position $\mathbf{x}_p$ could in general be obtained from a pose estimation as

$$\mathbf{x}_p = \mathbf{h}^{-1}(\mathbf{y}) \tag{5.4}$$

and used in a feedback control law in order to control the task space position [Wunsch and Hirzinger, 1997]. The pose estimation is typically performed using some type of iterative least-squares optimization algorithm, using the previous position as a starting point for the iteration. However, near singular configurations, where the Jacobian of $\mathbf{h}$ loses rank, the pose estimation becomes very inaccurate [Martin and Horaud, 2002]. An example of such a situation is when the relative depth of the object points is small, for instance when viewing a small object

at a long distance from the camera. In such cases, a very accurate estimation of the depth Z_i of each point may be required in order to maintain stability [Malis and Rives, 2003].

Vision-Based Observer

By exploiting the dynamic model in Eq. (5.3), we could obtain extra robustness and noise suppression. Since almost all real-time pose estimation algorithms work by updating an initial guess or prediction of the state, we use a state observer on the form

$$\frac{d\hat{\mathbf{x}}}{dt} = \mathbf{F}\hat{\mathbf{x}} + \mathbf{G}\mathbf{u} + \mathbf{K}\hat{\Delta}_p \tag{5.5}$$

where $\hat{\Delta}_p$ is an estimate of the relative pose between the prediction and the real pose. This is found from the linearization of the measurement equation in (5.3), and gives the estimator

$$\frac{d\hat{\mathbf{x}}}{dt} = \mathbf{F}\hat{\mathbf{x}} + \mathbf{G}\mathbf{u} + \mathbf{K}\mathbf{J}^{\dagger}\left(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})\right) \tag{5.6}$$

where $\mathbf{J}^{\dagger}$ is the pseudo inverse of the image Jacobian $\mathbf{J} = \partial\mathbf{h}/\partial\mathbf{x}$. If the Jacobian could be calculated exactly, Eq. (5.6) would reduce (locally) to

$$\frac{d\hat{\mathbf{x}}}{dt} = \mathbf{F}\hat{\mathbf{x}} + \mathbf{G}\mathbf{u} + \bar{\mathbf{K}}\mathbf{C}_n\left(\mathbf{x} - \hat{\mathbf{x}}\right) \tag{5.7}$$

where $\mathbf{C}_n = \begin{bmatrix} \mathbf{I}_{n\times n} & \mathbf{0} \end{bmatrix}$. A suitable value for the observer gain $\bar{\mathbf{K}}$ could be determined using techniques from linear control theory, and the estimated states could be used in a feedback law on the form

$$\mathbf{u} = \mathbf{L}(\mathbf{x}_{ref} - \hat{\mathbf{x}}). \tag{5.8}$$

Eq. (5.2) can be differentiated with respect to $\mathbf{x}$ and linearized around $\hat{\mathbf{x}}$, and the equations for multiple feature points can be stacked to give the linearized equation

$$\Delta\mathbf{y} = \mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}) \approx \mathbf{J}(\hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}}) \tag{5.9}$$

where $\mathbf{J}$ is the Jacobian of the projection equation, which can now be used directly in Eq. (5.6).

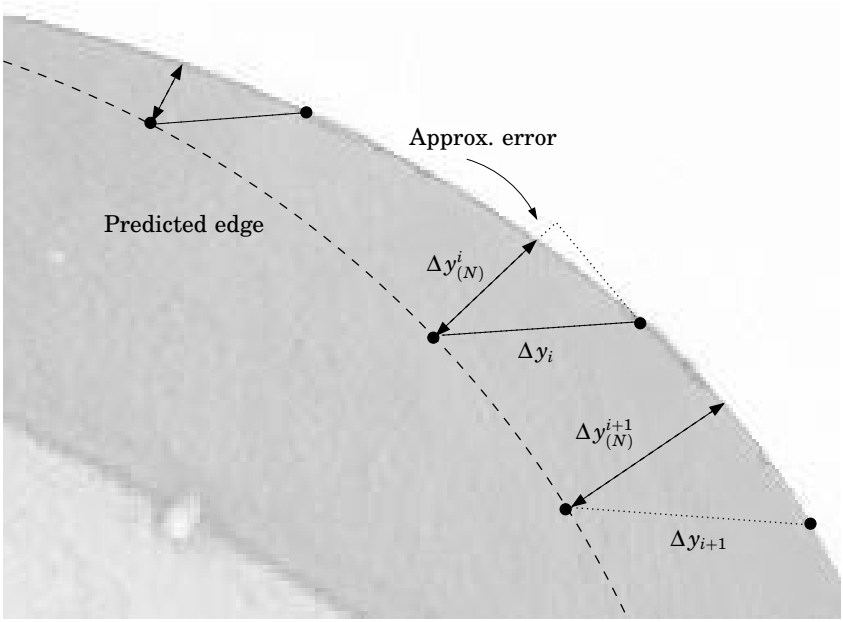

Figure 5.1 Edge detection in the normal direction of the predicted edges.

In the case of edge measurements, only the distance between the predicted and real edges in the normal direction of the contour is measurable. The corresponding equations are obtained by projecting the image space errors onto the normal as in [Drummond and Cipolla, 1999; Martin and Horaud, 2002], giving us the alternative equations

$$\Delta \mathbf{y}_{(N)} = \mathbf{N}^T(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}})) \approx \mathbf{N}^T \mathbf{J}(\hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}}) = \mathbf{J}_{(N)}(\hat{\mathbf{x}})(\mathbf{x} - \hat{\mathbf{x}}) \qquad (5.10)$$

where $\mathbf{N}$ is a block diagonal matrix, where the blocks are the edge normal directions at the n different measurement points along the contour (Fig. 5.1).

It is clear that in general the accuracy of the estimation will improve with the number of image measurements N. If we assume that the errors in the image measurements $\Delta \mathbf{y}_{(N)}$ can be modeled as Gaussian, spatially uncorrelated white noise with variance σ^2, a useful ap-

proximation of the effective measurement error covariance $\tilde{\mathbf{x}} = (\mathbf{x} - \hat{\mathbf{x}})$, can be obtained as

$$\begin{aligned} E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] &= E[\mathbf{J}^{\dagger}_{(N)} \Delta\mathbf{y}_{(N)} (\mathbf{J}^{\dagger}_{(N)} \Delta\mathbf{y}_{(N)})^T] = \\ &= (\mathbf{J}^T_{(N)} \mathbf{J}_{(N)})^{-1} \sigma^2 = (\sum_{i=1}^{M} \mathbf{J}_i^T \mathbf{J}_i)^{-1} \sigma^2 \end{aligned} \tag{5.11}$$

where the Jacobian has been partitioned into the individual Jacobians for each of the M cameras as $\mathbf{J}^T_{(N)} = [\mathbf{J}_1^T, \mathbf{J}_2^T, \cdots \mathbf{J}_M^T]^T$. In Chapter 7 a method which attempts to minimize the measurement error covariance in Eq. (5.11) by a proper selection of active cameras is presented. In addition, the covariance in Eq. (5.11) could be used as the measurement error covariance parameter, for instance in a Kalman filter.

Combined Vision/Force Controller

The force/vision controller combines the force- or impedance controller described in Section 3.3 with feedback from the cameras. The block diagram for the system under vision/force control is shown in Fig. 5.2. The desired trajectory of the tool is defined relative to the target object, whose position is estimated from the image data. The visual feedback controller generates a reference position and velocities in order to follow the desired trajectory, based on the estimated relative position of the end-effector and the target. The force controller updates the position and velocity according to Eqs. (3.1) and (3.2), and the new references are sent to the built-in robot motion control.

We assume a decoupled dynamic model of a velocity controlled manipulator

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{v}_c \\ \Delta\mathbf{y} = \mathbf{h}(\mathbf{C}\mathbf{x}, \mathbf{C}\hat{\mathbf{x}}) \approx \mathbf{J}(\mathbf{C}\hat{\mathbf{x}})(\mathbf{C}\mathbf{x} - \mathbf{C}\hat{\mathbf{x}}) \end{cases} \tag{5.12}$$

where $\mathbf{x} = [\, \mathbf{x}_p^T \quad \dot{\mathbf{x}}_p^T \,]^T$ is the state vector, $\mathbf{v}_c$ the commanded velocity, $\Delta\mathbf{y}$ contains the normal distances between the search points and the image edges as calculated from the projection equation, $\hat{\mathbf{x}}$ is the estimated state, and the system matrices are given by

$$\mathbf{F} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\omega\mathbf{I} \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \mathbf{0} \\ \omega\mathbf{I} \end{bmatrix}, \quad \mathbf{C} = [\, \mathbf{I} \quad \mathbf{0} \,]. \tag{5.13}$$

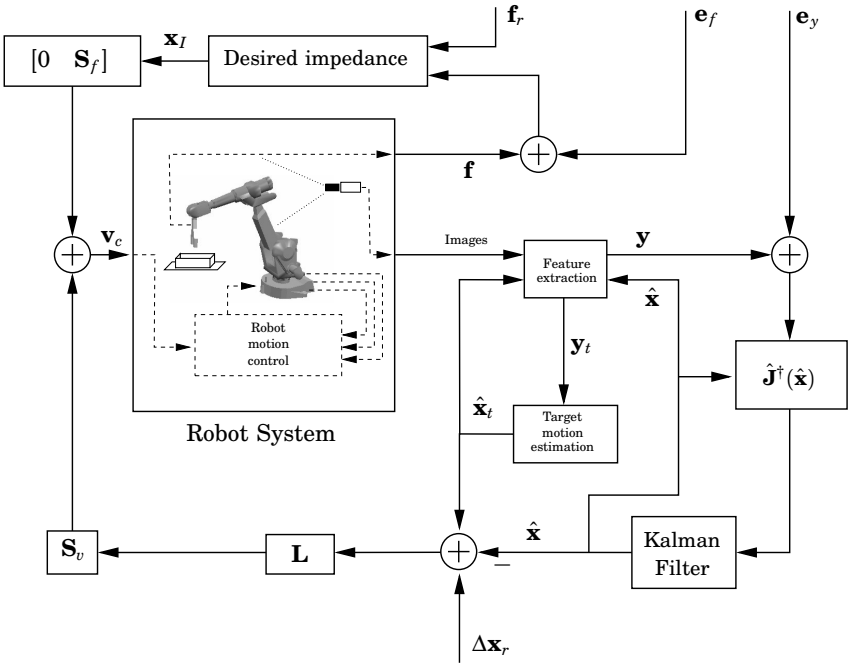

Figure 5.2 Block diagram showing the structure of the combined force/vision control system. Inputs are the reference signals $\Delta\mathbf{x}_r$ and $\mathbf{f}_r$ and the output disturbances $\mathbf{e}_y$ and $\mathbf{e}_f$. The motion of the target is estimated using a nonlinear least-squares estimator.

By assumption $\tilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$ is small, and the approximation in Eq. (5.12) holds locally. A state observer on the form of Eq. (5.6) is given by

$$\frac{d\hat{\mathbf{x}}}{dt} = \mathbf{F}\hat{\mathbf{x}} + \hat{\mathbf{G}}\mathbf{v}_c + \mathbf{K}\hat{\mathbf{J}}^\dagger\Delta\mathbf{y} = \mathbf{F}\hat{\mathbf{x}} + \hat{\mathbf{G}}\mathbf{v}_c + \mathbf{KC}(\mathbf{x} - \hat{\mathbf{x}}) \quad (5.14)$$

where the estimation $\hat{\mathbf{J}}$ is assumed to be equal to the true image Jacobian $\mathbf{J}$, and the difference $\Delta\mathbf{G} = \mathbf{G} - \hat{\mathbf{G}}$ models the calibration errors in the geometric model of the manipulator object frame. Using a force

controller given by

$$\frac{d\mathbf{x}_I}{dt} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & -\mathbf{M}_I^{-1}\mathbf{D}_I \end{bmatrix} \mathbf{x}_I + \begin{bmatrix} \mathbf{0} \\ \mathbf{M}_I^{-1} \end{bmatrix} \mathbf{S}_f(\mathbf{f} - \mathbf{f}_r) = \\ = \mathbf{F}_I\mathbf{x}_I + \mathbf{G}_I\mathbf{S}_f(\mathbf{f} - \mathbf{f}_r) \tag{5.15}$$

together with the hybrid vision/force controller

$$\mathbf{v}_c = \mathbf{S}_v\mathbf{L}(\mathbf{x}_r - \hat{\mathbf{x}}) + \mathbf{S}_f \begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix} \mathbf{x}_I, \tag{5.16}$$

where $\mathbf{S}_v$ and $\mathbf{S}_f$ are the hybrid *selection matrices*, whose diagonal elements will be 1 or 0 depending on which degrees of freedom are activated, and a model of the forces in contact with a surface through the origin, given by

$$\mathbf{f} = -\mathbf{k}_e\mathbf{C}\mathbf{x}, \tag{5.17}$$

we can write down the equations for the closed loop system as

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{v}_c = \mathbf{F}\mathbf{x} + \mathbf{G}(\mathbf{S}_v\mathbf{L}(\mathbf{x}_r - \hat{\mathbf{x}}) + \begin{bmatrix} \mathbf{0} & \mathbf{S}_f \end{bmatrix} \mathbf{x}_I) = \\ = (\mathbf{F} - \mathbf{G}\mathbf{S}_v\mathbf{L})\mathbf{x} + \mathbf{G}\mathbf{S}_v\mathbf{L}\tilde{\mathbf{x}} + \mathbf{G} \begin{bmatrix} \mathbf{0} & \mathbf{S}_f \end{bmatrix} \mathbf{x}_I + \mathbf{G}\mathbf{S}_v\mathbf{L}\mathbf{x}_r \tag{5.18}$$

$$\dot{\tilde{\mathbf{x}}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{v}_c - (\mathbf{F} - \mathbf{K}\mathbf{C})\hat{\mathbf{x}} - \hat{\mathbf{G}}\mathbf{v}_c - \mathbf{K}\mathbf{C}\mathbf{x} = \\ = (\mathbf{F} - \mathbf{K}\mathbf{C})\tilde{\mathbf{x}} + (\mathbf{G} - \hat{\mathbf{G}})\mathbf{v}_c = (\mathbf{F} - \mathbf{K}\mathbf{C} + \Delta\mathbf{G}\mathbf{S}_v\mathbf{L})\tilde{\mathbf{x}} \\ - \Delta\mathbf{G}\mathbf{S}_v\mathbf{L}\mathbf{x} + \Delta\mathbf{G} \begin{bmatrix} \mathbf{0} & \mathbf{S}_f \end{bmatrix} \mathbf{x}_I + \Delta\mathbf{G}\mathbf{S}_v\mathbf{L}\mathbf{x}_r \tag{5.19}$$

$$\dot{\mathbf{x}}_I = \mathbf{F}_I\mathbf{x}_I + \mathbf{G}_I\mathbf{S}_f(\mathbf{f} - \mathbf{f}_r) = \mathbf{F}_I\mathbf{x}_I - \mathbf{G}_I\mathbf{S}_f\mathbf{k}_e\mathbf{C}\mathbf{x} - \mathbf{S}_f\mathbf{G}_I\mathbf{f}_r \tag{5.20}$$

or equivalently

$$\frac{d}{dt} \begin{bmatrix} \mathbf{x} \\ \tilde{\mathbf{x}} \\ \mathbf{x}_I \end{bmatrix} = \mathbf{F}_c \begin{bmatrix} \mathbf{x} \\ \tilde{\mathbf{x}} \\ \mathbf{x}_I \end{bmatrix} + \mathbf{G}_c \begin{bmatrix} \mathbf{x}_r \\ \mathbf{f}_r \end{bmatrix} \tag{5.21}$$

with

$$\mathbf{F}_c = \begin{bmatrix} \mathbf{F} - \mathbf{G}\mathbf{S}_v\mathbf{L} & \mathbf{G}\mathbf{S}_v\mathbf{L} & \mathbf{G} \begin{bmatrix} \mathbf{0} & \mathbf{S}_f \end{bmatrix} \\ -\Delta\mathbf{G}\mathbf{S}_v\mathbf{L} & \mathbf{F} - \mathbf{K}\mathbf{C} + \Delta\mathbf{G}\mathbf{S}_v\mathbf{L} & \Delta\mathbf{G} \begin{bmatrix} \mathbf{0} & \mathbf{S}_f \end{bmatrix} \\ -\mathbf{G}_I\mathbf{S}_f\mathbf{k}_e\mathbf{C} & \mathbf{0} & \mathbf{F}_I \end{bmatrix} \tag{5.22}$$

and

$$\mathbf{G}_c = \begin{bmatrix} \mathbf{G}\mathbf{S}_v\mathbf{L} & \mathbf{0} \\ \Delta\mathbf{G}\mathbf{S}_v\mathbf{L} & \mathbf{0} \\ \mathbf{0} & -\mathbf{G}_I\mathbf{S}_f \end{bmatrix} \tag{5.23}$$

If the case of perfect calibration $\Delta\mathbf{G} = \mathbf{0}$, and the system matrices reduce to

$$\mathbf{F}_c = \begin{bmatrix} \mathbf{F} - \mathbf{G}\mathbf{S}_v\mathbf{L} & \mathbf{G}\mathbf{S}_v\mathbf{L} & \mathbf{G}\begin{bmatrix}\mathbf{0} & \mathbf{S}_f\end{bmatrix} \\ \mathbf{0} & \mathbf{F} - \mathbf{K}\mathbf{C} & \mathbf{0} \\ -\mathbf{G}_I\mathbf{S}_f\mathbf{k}_e\mathbf{C} & \mathbf{0} & \mathbf{F}_I \end{bmatrix} \tag{5.24}$$

and

$$\mathbf{G}_c = \begin{bmatrix} \mathbf{G}\mathbf{S}_v\mathbf{L} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{G}_I\mathbf{S}_f \end{bmatrix} \tag{5.25}$$

Stability

When $\Delta\mathbf{G} = \mathbf{0}$ and $\hat{\mathbf{J}} = \mathbf{J}$, for small $\tilde{\mathbf{x}}$ the observer error dynamics is locally described by the stable system $\dot{\tilde{\mathbf{x}}} = (\mathbf{F} - \mathbf{K}\mathbf{C})\tilde{\mathbf{x}}$. In the force controlled directions, a stable and well-damped response in the measured contact force is obtained by proper tuning of $\mathbf{M}_I$ and $\mathbf{D}_I$. In practice, the possible choices of $\mathbf{M}_I$ and $\mathbf{D}_I$ are also limited by sensor noise, unmodeled dynamics, and resonances in the tool and workpiece.

In the case where the estimation of the Jacobian in Eq. (5.14) is not exact, the state observer (5.14) and the resulting closed loop system may become unstable. For a purely kinematic robot model $\dot{\mathbf{x}}_p = \mathbf{v}_c$ with $\mathbf{x} = \mathbf{x}_p$, $\mathbf{F} = \mathbf{0}$ and $\mathbf{G} = \mathbf{I}$, local stability of the observer with $\mathbf{K} = \mathbf{I}$ is guaranteed as long as the matrix $\hat{\mathbf{J}}^\dagger\mathbf{J}(\hat{\mathbf{x}})$ is positive definite. Near singularities, this is satisfied only if a very accurate estimation $\hat{\mathbf{J}}$ is available, and small errors in the intrinsic camera parameters or point depth distribution can cause the system to become unstable [Malis and Rives, 2003]. However, an observer for the dynamical system in Eq. (5.12) will need additional constraints on $\hat{\mathbf{J}}$ and $\mathbf{J}$ for stability, as stability can not be guaranteed even if $\hat{\mathbf{J}}^\dagger\mathbf{J}(\hat{\mathbf{x}})$ is positive definite.

Implementation

The vision controller and observer are designed as a stationary LQG controller, based on a discretized version of the dynamic model in

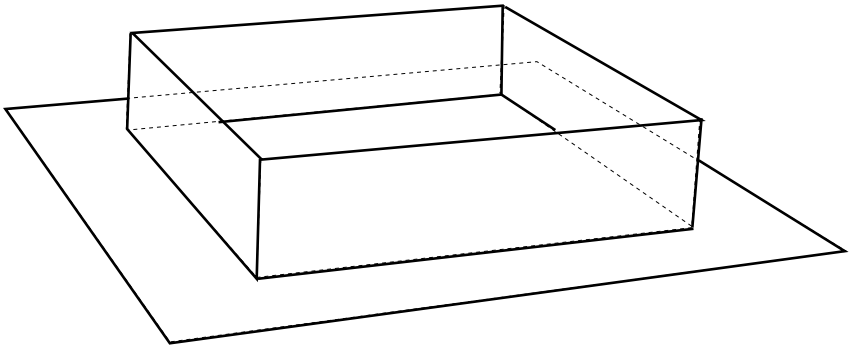

Figure 5.3 Object model with hidden features removed.

Eq. (5.12) sampled at 33 ms. The force controller in Eq. (5.15) is discretized at a sampling period of 4 ms. The force controller runs on a PowerPC G4 processor, connected to the internal robot motion controller over the PCI bus [Blomdell *et al.*, 2004]. The image processing, and calculation and inversion of the image Jacobian runs on a separate 2 GHz Pentium 4, which communicates with the controller on the PowerPC using Ethernet.

The tracking algorithm running on the PC is summarized in Fig. 5.4. Two objects are tracked, the stationary target and the manipulator object, assumed to be rigidly attached to the robot hand. The tracker states are initialized through an optimization-based pose estimation algorithm, using only the locations of four or more corners. At each sample time all images are read from the cameras, the control signal which was pre-calculated at the previous sample is sent to the main controller using an Ethernet connection, measurement vectors are obtained and Jacobians are calculated. The total hybrid control signal is then read back from the PowerPC, and is used to calculate the state estimate and vision-based part of the control signal. A fast hidden-line removal technique is used to predict locations of visible edges in the next set of images, using an object model consisting of a number of planar surfaces connected at their edges, see Fig. 5.3. No assumptions are made about the shape of the planar surfaces, although in the experi-

ments we use an object with only straight edges. Visible object edges are selected based on the predicted object pose and a pre-generated Binary Search Partitioning (BSP) tree description of the object, see [van Dam *et al.*, 1991]. The BSP tree recursively divides the surfaces in the object into "in front" and "behind", until we have a perfect front-to-back ordering. The surfaces are then processed front-to-back, with each surface clipped against all surfaces in front of it, in order to determine visible edges. A number of search points are selected on each visible edge. The image position measurements are then obtained from a one-dimensional edge localization in the local edge normal direction at each point. The edges are found from the convolution with a differentiated Gauss kernel at three different scale spaces, in order to get a robust detection/localization. To increase robustness only points where a clear single edge is detected are used by the tracker.

5.3 Experiment - Surface Following

By combining force control with visual feedback as described in Section 5.2, we could achieve surface following that is independent of the workpiece calibration accuracy. Experiments with this scenario have been performed using an ABB Irb2400 industrial robot equipped with a rolling tool, in contact with a metal box with dimensions $40 \times 40 \times 10$ cm. Experiments were first performed using only two Sony digital cameras, and later repeated with an extra camera, using the resource allocation algorithm presented in Chapter 7, see Fig. 5.5.

The robot makes stable contact with the workpiece under vision guided impedance control, and when contact has been established the control switches to parallel vision/force control as described in Section 5.2, while the tool moves across the surface at around 10 mm/s. The resulting force can be seen in Fig. 5.6. At time $t = 3$ s the force reference was changed from 15 N to 25 N in the x-direction of the tool. At time $t = 17$ s the tool reaches a corner, and the force reference changes to 15 N in the negative y-direction. The combined stiffness of the robot and surface was approximately 10 kN/m, and the translational controller parameters were chosen as $M_I = 0.1$, $D_I = 1.5$ and $K_I = 0$.

Fig. 5.7 shows the estimated position of the tool with respect to the

1. Initialize state and data structures
2. Send pre-computed visual command velocity $\mathbf{u}_v$ to the main controller on the PowerPC
3. Capture images from each camera and perform image preprocessing
4. Search for image edges around the predicted edges of the manipulator and target, and build measurement vectors $\Delta\mathbf{y}$ and $\Delta\mathbf{y}_t$
5. For each edge measurement, build one row of the corresponding Jacobian $\mathbf{J}$ or $\mathbf{J}_t$
6. Read effective control signal from PowerPC, given by

$$\mathbf{v}_c := \mathbf{S}_v\mathbf{u}_v + \mathbf{S}_f\,[\,\mathbf{0} \quad \mathbf{I}\,]\,\mathbf{x}_I$$

7. Update state estimate for the manipulator using the discrete time dynamic model as

$$\hat{\mathbf{x}} := \mathbf{F}_d\hat{\mathbf{x}} + \hat{\mathbf{G}}_d\mathbf{v}_c + \mathbf{K}\mathbf{J}^{\dagger}\Delta\mathbf{y}$$

8. Update estimated target position using one Gauss-Newton iteration

$$\hat{\mathbf{x}}_t := \hat{\mathbf{x}}_t + \mathbf{J}_t^{\dagger}\Delta\mathbf{y}_t$$

9. Predict visible edges during the next sample, by performing hidden line removal based on the predicted positions $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}_t$
10. Calculate $\mathbf{x}_r$ using the estimated target position $\hat{\mathbf{x}}_t$ and the desired relative position $\Delta\mathbf{x}_r$, and pre-calculate the vision based part of the control signal

$$\mathbf{u}_v := \mathbf{L}(\mathbf{x}_r - \hat{\mathbf{x}})$$

11. Wait for next sample time and repeat from Step 2.

Figure 5.4 Algorithm for tracking and control of relative position.

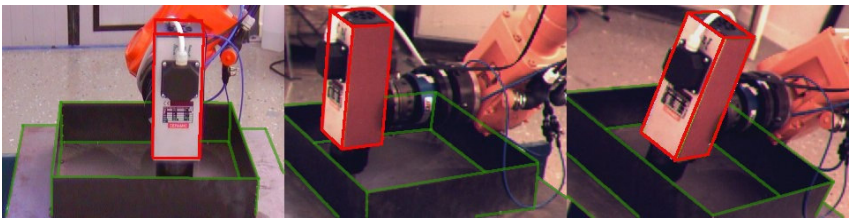

Figure 5.5 Simultaneous tracking of tool and workpiece without force control, using two Sony DFW-V300 and one Basler A602fc digital cameras.

target frame during the same experiment. Fig. 5.8 shows the corresponding estimated velocities. A small control error in the force controlled directions is caused by the force control action, which makes the position deviate from the nominal trajectory.

Effect of Calibration Errors. In the presence of calibration errors $\Delta\mathbf{G}$ in Eqs. (5.21)–(5.23), the system properties will change. We assume that the error can be modeled as

$$\Delta\mathbf{G} = \mathbf{G} - \hat{\mathbf{G}} = \mathbf{G} \begin{bmatrix} \mathbf{I} - \mathbf{R}_\Delta(\delta) & \mathbf{0} \\ \mathbf{0} & \mathbf{I} - \mathbf{R}_\Delta(\delta) \end{bmatrix}, \tag{5.26}$$

where the rotation matrix $\mathbf{R}_\Delta(\delta)$ corresponds to an orientation error δ between the tracked frame and the actuated frame. In practice, the stability of the system is preserved for all reasonably small δ, but the servo properties of the system may degrade considerably. Particularly, in the common situation when the position trajectory is a ramp along the surface, large force errors may occur in the force controlled directions, due to the high stiffness of the surface.

We have simulated this effect in a typical scenario, where a hybrid controller with bandwidth 5 rad/s in the vision controlled direction has been used, together with an observer bandwidth of 10 rad/s. The force controller bandwidth was 15 rad/s, and the surface stiffness was 10 N/mm. Force control is applied in the x-direction, while the remaining degrees of freedom are vision controlled. The calibration error $\mathbf{R}_\Delta(\delta)$ is given by a rotation of $\delta = 1^\circ$ around the z-axis. $\mathbf{x}_r$ was given by a constant velocity of $\mathbf{v}_y = 10$ mm/s in the y-direction. The resulting

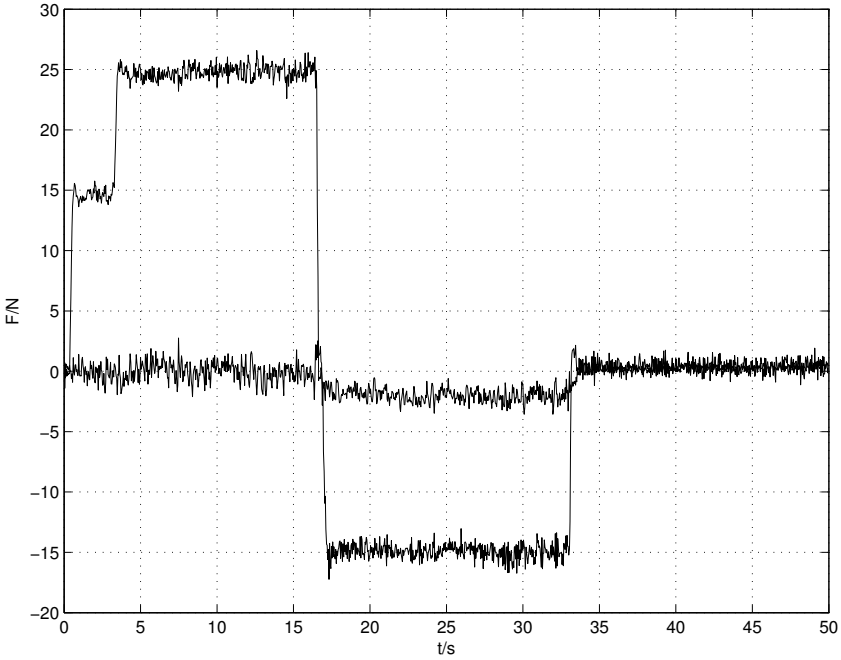

Figure 5.6 Measured contact force during vision guided force control. The force reference was first changed from 15 N to 25 N in the x-direction, and finally to 15 N in the negative y-direction.

stationary force error was 0.18 N, an error that scales approximately linearly with δ and $\mathbf{v}_y$.

5.4 Discussion

The combination of force- and visual feedback is ideal for handling environments with geometric uncertainties on different scales, where the force controller is responsible for accurate control of the contact force, while the visual control takes care of the overall guidance of the tool. Experiments show that the system is able to follow low speed

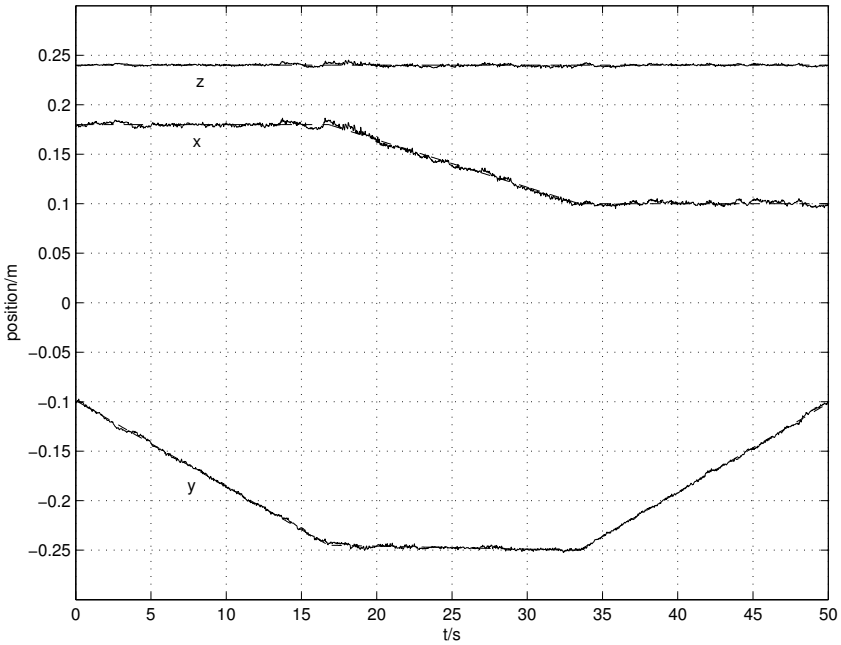

Figure 5.7 Estimated tool translation (solid) and reference (dashed) during vision guided force control.

trajectories with an accuracy of around 1 mm, while accurately controlling the contact force. The force controller achieves tracking with rise times of under 0.2 s in stiff environments, so that the force controller can quickly compensate for deviations from the nominal geometry. At higher speeds along the surface, calibration errors may cause large stationary errors in the contact force, and the effects of geometrical deviations in the workpiece will become larger.

By tracking multiple objects and controlling the relative position, we can theoretically achieve surface following with an accuracy that is independent of the calibration accuracy of the work cell. The price we pay is the use of external sensors such as cameras for position control. The robustness of camera sensing is still problematic, since phenomena such as occlusions, reflections, poor lighting or limited fields of view

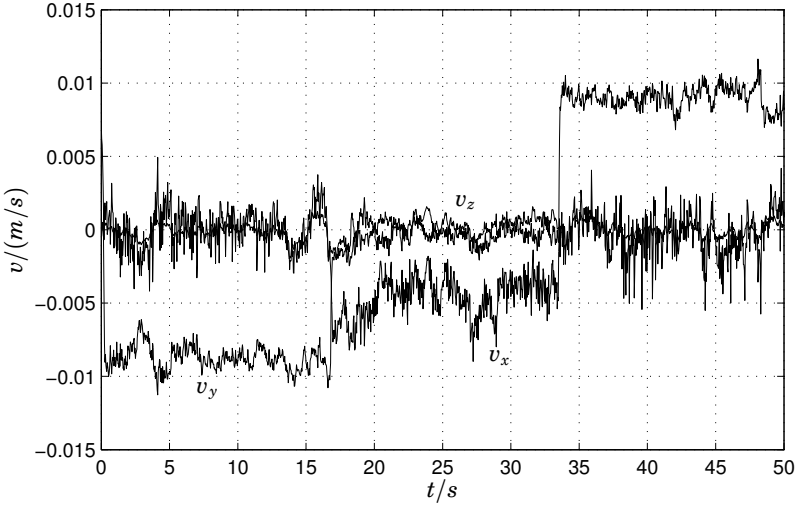

Figure 5.8 Estimated tool velocity during vision guided force control, with respect to the target frame.

could cause the system to lose track and signal loss, or degradation of measurement accuracy. Robustness increases considerably with the number of cameras [Martin and Horaud, 2002], and using multiple low-cost cameras could potentially be a cost-effective solution for certain tasks, especially in poorly known or uncalibrated environments.

5.5 Conclusions

In this chapter we have demonstrated how to achieve high performance six degree-of-freedom combined vision/force control for interaction with a stiff uncalibrated environment. A process with linear dynamics in task space, is used together with a standard edge-based rigid body tracker, which gives a locally stable observer with linear error dynamics. The effect of error sources such as image measurement noise and geometrical calibration errors are considered. Finally, experiments and simulations were used to validate the approach.

6

Rigid-Body Tracking Using Dual Quaternions

6.1 Introduction

In [Drummond and Cipolla, 1999] a method to recursively estimate not only the position and orientation, but also the intrinsic parameters of the camera (focal length, aspect ratio and principal point), is presented. In [Martin and Horaud, 2002], on the other hand, it is pointed out that the problem of simultaneously tracking position and intrinsic parameters is ill-conditioned when the points of the object lie on a plane parallel to the image plane, which causes the Jacobian matrix, relating errors in position- and intrinsic parameters to image errors, to lose rank. Because of noise, this problem extends also to positions where the relative depth of the object points in the camera is small. A multi-camera tracking system is suggested as a possible solution to this problem.

In this chapter we develop methods for real-time rigid body tracking with simultaneous calibration and tracking of intrinsic parameters. We intend to show that a dual quaternion parameterization of the object pose can be used to formulate hand-eye constraints on the estimated motion, which are expressed as linear equations in the states. We also show that the data from the tracker can be used for hand-eye calibration, also when intrinsic camera parameters are allowed to vary during

the motion sequence.

Quaternions and Dual Quaternions

In this section we briefly introduce the notation and properties of dual quaternions used in this chapter. For a more detailed description and introduction to the theory and properties of quaternions and screws, see for instance [Murray *et al.*, 1994; Daniilidis, 1999; Goddard, 1997].

Quaternions were invented by Hamilton [Hamilton, 1853]. Unit quaternions are useful for representing rotations in three dimensions. Quaternions can be represented as a pair $\boldsymbol{q} = (q^0, \vec{\boldsymbol{q}})$, where $q^0 \in \mathbb{R}$ and $\vec{\boldsymbol{q}} \in \mathbb{R}^3$, with the operations

$$\boldsymbol{q}_1 + \boldsymbol{q}_2 = (q_1^0 + q_2^0, \vec{\boldsymbol{q}}_1 + \vec{\boldsymbol{q}}_2) \tag{6.1}$$

$$k\boldsymbol{q} = (kq^0, k\vec{\boldsymbol{q}}) \tag{6.2}$$

$$\boldsymbol{q}_1\boldsymbol{q}_2 = (q_1^0 q_2^0 - \vec{\boldsymbol{q}}_1^T \vec{\boldsymbol{q}}_2, q_1^0\vec{\boldsymbol{q}}_2 + q_2^0\vec{\boldsymbol{q}}_1 + \vec{\boldsymbol{q}}_1 \times \vec{\boldsymbol{q}}_2) \tag{6.3}$$

where $k \in \mathbb{R}$. A quaternion has a norm given by $\|\boldsymbol{q}\|^2 = \boldsymbol{q}\bar{\boldsymbol{q}}$, where $\bar{\boldsymbol{q}} = (q^0, -\vec{\boldsymbol{q}})$ is the conjugate quaternion. It is well known that every rigid rotation (element of the special orthogonal group SO(3)) with angle θ about an axis $\vec{\boldsymbol{n}}$ with $\|\vec{\boldsymbol{n}}\| = 1$ can be represented as a unit quaternion

$$\boldsymbol{q} = (\cos(\theta/2), \sin(\theta/2)\vec{\boldsymbol{n}}), \tag{6.4}$$

which rotates a vector $\vec{\boldsymbol{x}} \in \mathbb{R}^3$ to the vector $\boldsymbol{q}(0, \vec{\boldsymbol{x}})\bar{\boldsymbol{q}}$.

Dual Quaternions. Similarly to real quaternions, dual quaternions are defined as $\check{\boldsymbol{q}} = (\check{q}^0, \check{\vec{\boldsymbol{q}}})$, where $\check{q}^0 = q^0 + \varepsilon q'^0$ is a dual number with $\varepsilon^2 = 0$, and where $\vec{\boldsymbol{q}} = \vec{\boldsymbol{q}} + \varepsilon\vec{\boldsymbol{q}}'$ is a dual vector. Dual numbers were invented by Clifford [Clifford, 1873]. The dual quaternion operations are

$$\check{\boldsymbol{q}}_1 + \check{\boldsymbol{q}}_2 = (\check{q}_1^0 + \check{q}_2^0, \check{\vec{\boldsymbol{q}}}_1 + \check{\vec{\boldsymbol{q}}}_2) \tag{6.5}$$

$$k\check{\boldsymbol{q}} = (k\check{q}^0, k\check{\vec{\boldsymbol{q}}}) \tag{6.6}$$

$$\check{\boldsymbol{q}}_1\check{\boldsymbol{q}}_2 = (\check{q}_1^0\check{q}_2^0 - \check{\vec{\boldsymbol{q}}}_1^T\check{\vec{\boldsymbol{q}}}_2, \check{q}_1^0\check{\vec{\boldsymbol{q}}}_2 + \check{q}_2^0\check{\vec{\boldsymbol{q}}}_1 + \check{\vec{\boldsymbol{q}}}_1 \times \check{\vec{\boldsymbol{q}}}_2). \tag{6.7}$$

We will often write the dual quaternion as the sum of the real and dual parts $\boldsymbol{q} + \varepsilon\boldsymbol{q}'$. Its norm is given by $\|\check{\boldsymbol{q}}\|^2 = \check{\boldsymbol{q}}\check{\bar{\boldsymbol{q}}}$ with $\check{\bar{\boldsymbol{q}}} = \bar{\boldsymbol{q}} + \varepsilon\bar{\boldsymbol{q}}'$,

and the unity conditions become

$$q\bar{q} = 1 \tag{6.8}$$
$$\bar{q}q' + \bar{q}'q = 0. \tag{6.9}$$

Unit dual quaternions can be used to represent general rigid transformations including translations, similarly to the way rotations can be represented by real quaternions. In [Daniilidis, 1999], it is shown that the rigid transformation of a line through the point $\vec{p}$, represented by its direction $\vec{n}$ and moment $\vec{m} = \vec{p} \times \vec{n}$, is given by $\breve{q}(n + \varepsilon m)\breve{\bar{q}}$, where $\vec{n}$ and $\vec{m}$ are expressed as quaternions $n = (0, \vec{n})$ and $m = (0, \vec{m})$, respectively. The dual quaternion itself is $q + \varepsilon q'$, where q is the quaternion describing the rotation, and where $q' = tq/2$ with $t = (0, \vec{t})$ being the translation.

Screws and Hand-Eye Constraints

Screws. According to Chasles' theorem [Murray *et al.*, 1994] a general rigid transformation can be modeled as a rotation about an axis not through the origin and a translation along the rotation axis. The parameters of the screw are the direction $\vec{n}$ and the moment $\vec{m}$ of the screw axis line, the rotation angle θ, and the translation (pitch) d along $\vec{n}$. Together with the constraints $\vec{n}^T\vec{n} = 1$ and $\vec{n}^T\vec{m} = 0$ these parameters constitute the six degrees of freedom of a rigid transformation. It can be shown that the dual quaternion corresponding to the screw with parameters $\vec{n}$, $\vec{m}$, θ, and d can be written as

$$\breve{q} = (\cos(\breve{\theta}/2), \sin(\breve{\theta}/2)\breve{\vec{l}}), \tag{6.10}$$

where the dual angle is $\breve{\theta} = \theta + \varepsilon d$, and the line is given by $\breve{\vec{l}} = \vec{n} + \varepsilon\vec{m}$.

Hand-Eye Constraints. The hand-eye equation (Eq. (2.9) in Section 2.2) can be written using dual quaternions as

$$\breve{a} = \breve{q}\breve{b}\breve{\bar{q}}. \tag{6.11}$$

In [Daniilidis, 1999], it is shown that the scalar parts of $\check{a}$ and $\check{b}$ are equal, which can easily be shown as follows

$$\begin{aligned} Sc(\check{a}) &= \frac{1}{2}(\check{a} + \bar{\check{a}}) = \frac{1}{2}(\check{q}\check{b}\bar{\check{q}} + \check{q}\bar{\check{b}}\bar{\check{q}}) = \\ &= \frac{1}{2}\check{q}(\check{b} + \bar{\check{b}})\bar{\check{q}} = Sc(\check{b})\check{q}\bar{\check{q}} = Sc(\check{b}). \end{aligned} \tag{6.12}$$

From the expression for the dual quaternion in Eq. (6.10), and using the fact that a function of a dual number can be rewritten as

$$f(a + \varepsilon b) = f(a) + \varepsilon b f'(a), \tag{6.13}$$

we can write Eq. (6.12) as

$$\cos\frac{\theta_a}{2} - \varepsilon\frac{d_a}{2}\sin\frac{\theta_a}{2} = \cos\frac{\theta_b}{2} - \varepsilon\frac{d_b}{2}\sin\frac{\theta_b}{2}. \tag{6.14}$$

Dividing this equation into real and dual parts, we can see that the angle and pitch of the camera screw and the robot end-effector screw must be equal. This is known as the Screw Congruence Theorem, see [Chen, 1991]. In [Daniilidis, 1999], this equality is used to rewrite the hand-eye equation using only the vector parts of $\check{a}$ and $\check{b}$. Each motion of the robot and camera will give us six linear equations in the unknowns q and q', the real and dual parts of the unknown hand-eye dual quaternion. A minimum of two motions together with the constraints

$$q^T q = 1, \quad q^T q' = 0 \tag{6.15}$$

from Eqs. (6.8)–(6.9) are generally enough to solve for the eight unknowns. The solution can be obtained by finding the vectors spanning the null space of the linear system using SVD, and then finding the linear combination which satisfies the unity conditions (6.15), see [Daniilidis, 1999] for details.

6.2 Modeling

Consider a manipulator with a single camera attached to its end-effector, viewing a stationary object as in Fig. 6.1. Only a very rough

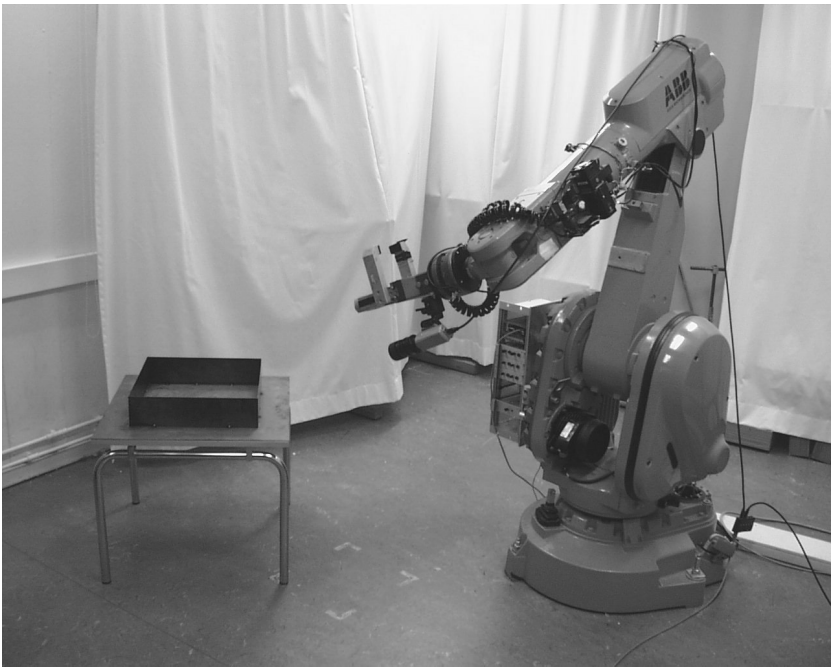

Figure 6.1 ABB Irb2000 industrial robot with digital camera used in the experiments.

initial estimation of the intrinsic camera parameters and the position/orientation of the objects are known, but we assume that a CAD model of the object is available. The motion of the robot end-effector is related to the motion of the camera through the hand-eye equation (2.9), where the relative sensor-actuator pose $\boldsymbol{X}$ is unknown. We assume that the camera can be modeled as a four parameter pinhole camera

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & u_0 \\ 0 & \gamma f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{t} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{6.16}$$

with λ corresponding to the depth of the point with respect to the camera, see Appendix A. The parameters to be estimated are f, γ, u_0, v_0, and some parameterization of $\boldsymbol{R} \in \mathrm{SO}(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$.

Extended Kalman Filtering (EKF)

The motion of the system can be written as a non-linear discrete-time dynamic system

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{f}(\boldsymbol{x}_k) && (6.17) \\ g(\boldsymbol{p}_k, \boldsymbol{x}_k) &= 0 && (6.18) \end{aligned}$$

with $\boldsymbol{x}_k \in \mathbb{R}^n$ the state of the system, $\boldsymbol{p}_k \in \mathbb{R}^m$ a vector of measured outputs, $\boldsymbol{f}$ a known vector-valued function describing the system dynamics, and $\boldsymbol{g}$ a known function relating the state to the output. The state vector is chosen as

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{q} & \boldsymbol{q}' & f & \gamma & u_0 & v_0 \end{pmatrix}^T \tag{6.19}$$

where $\boldsymbol{q}, \boldsymbol{q}' \in \mathbb{R}^4$ is the vector representation of the object-camera dual quaternion $\check{\boldsymbol{q}} = \boldsymbol{q} + \varepsilon \boldsymbol{q}'$.

Measurement Model. In the function $\boldsymbol{g}(\boldsymbol{p}_k, \boldsymbol{x}_k)$ we have all the measurement equations, and the constraints on the state vector. For a point feature, denoted by index i, the measurement would be image position $\boldsymbol{p}_{k,i} = (u_i, v_i)^T \stackrel{\text{def}}{=} \boldsymbol{h}_i(\boldsymbol{x})$, which is known from Eq. (6.16) to be related to the states by the equations

$$\begin{aligned} \hat{\boldsymbol{h}}_i(\boldsymbol{x}) &= \begin{pmatrix} \hat{u}_i \\ \hat{v}_i \\ \hat{w}_i \end{pmatrix} = \boldsymbol{K} \begin{pmatrix} \boldsymbol{r}_x^T(\boldsymbol{q}) & t_x(\check{\boldsymbol{q}}) \\ \boldsymbol{r}_y^T(\boldsymbol{q}) & t_y(\check{\boldsymbol{q}}) \\ \boldsymbol{r}_z^T(\boldsymbol{q}) & t_z(\check{\boldsymbol{q}}) \end{pmatrix} \begin{pmatrix} \boldsymbol{X}_i \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} f(\boldsymbol{r}_x^T(\boldsymbol{q})\boldsymbol{X}_i + t_x(\check{\boldsymbol{q}})) + u_0(\boldsymbol{r}_z^T(\boldsymbol{q})\boldsymbol{X}_i + t_z(\check{\boldsymbol{q}})) \\ \gamma f(\boldsymbol{r}_y^T(\boldsymbol{q})\boldsymbol{X}_i + t_y(\check{\boldsymbol{q}})) + v_0(\boldsymbol{r}_z^T(\boldsymbol{q})\boldsymbol{X}_i + t_z(\check{\boldsymbol{q}})) \\ \boldsymbol{r}_z^T(\boldsymbol{q})\boldsymbol{X}_i + t_z(\check{\boldsymbol{q}}) \end{pmatrix} \end{aligned} \tag{6.20}$$

where $\boldsymbol{K}$ is the matrix of intrinsic parameters in Eq. (6.16). The image space coordinates are given by

$$\boldsymbol{h}_i(\boldsymbol{x}) = \begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} \hat{u}_i/\hat{w}_i \\ \hat{v}_i/\hat{w}_i \end{pmatrix}. \tag{6.21}$$

The rotation matrix is calculated directly from the unit quaternion $\boldsymbol{q}$, see for instance [Goddard, 1997] for details, and the translation can be obtained from $\breve{\boldsymbol{q}}$ as

$$\boldsymbol{t} = 2\boldsymbol{q}'\bar{\boldsymbol{q}}. \tag{6.22}$$

If $\boldsymbol{p}_{k,i}$ are the measured image coordinates, we can write the measurement equation for this point as

$$\boldsymbol{g}_i(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k) = \boldsymbol{p}_{k,i} - \boldsymbol{h}_i(\boldsymbol{x}_k) = 0 \tag{6.23}$$

This equation can be linearized around the predicted state $\boldsymbol{x}_k^{(p)}$, which gives the approximation

$$\begin{aligned} \boldsymbol{g}_i(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k) &\approx \boldsymbol{g}_i(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k^{(p)}) + \tfrac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{x}}(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k^{(p)})(\boldsymbol{x}_k - \boldsymbol{x}_k^{(p)}) \\ &= \boldsymbol{p}_{k,i} - \boldsymbol{h}_i(\boldsymbol{x}_k^{(p)}) + \tfrac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{x}}(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k^{(p)})(\boldsymbol{x}_k - \boldsymbol{x}_k^{(p)}) \approx 0 \end{aligned} \tag{6.24}$$

In our system however, the only image measurements available are the point-to-contour error in the predicted (local) normal direction of the contour, which can be approximated with the normal component of the error

$$\boldsymbol{g}_i^{(n)} = \boldsymbol{n}_i^T(\boldsymbol{p}_{k,i} - \boldsymbol{h}_i(\boldsymbol{x}_k^{(p)})) \tag{6.25}$$

Eq. (6.24) can then be rewritten as

$$\begin{aligned} 0 \quad &= \quad \boldsymbol{g}_i^{(n)}(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k) \approx \boldsymbol{n}_i^T(\boldsymbol{p}_{k,i} - \boldsymbol{h}_i(\boldsymbol{x}_k^{(p)})) + \\ &+ \quad \boldsymbol{n}_i^T \frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{x}}(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k^{(p)})(\boldsymbol{x}_k - \boldsymbol{x}_k^{(p)}) \end{aligned} \tag{6.26}$$

which can be expressed on linear form as

$$\boldsymbol{y}_{k,i} = \boldsymbol{C}_{k,i}\boldsymbol{x}_k \tag{6.27}$$

with

$$\boldsymbol{C}_{k,i} = \boldsymbol{n}_i^T \frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{x}}(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k^{(p)}) = -\boldsymbol{n}_i^T \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{x}}(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k^{(p)}) \tag{6.28}$$

$$\boldsymbol{y}_{k,i} = -\boldsymbol{n}_i^T \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{x}}(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k^{(p)}) \boldsymbol{x}_k^{(p)} - \boldsymbol{n}_i^T (\boldsymbol{p}_{k,i} - \boldsymbol{h}_i(\boldsymbol{x}_k^{(p)})) \tag{6.29}$$

The Jacobian

$$\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{x}}(\boldsymbol{p}_{k,i}, \boldsymbol{x}_k^{(p)}) \tag{6.30}$$

can be calculated by direct differentiation of Eq. (6.20) with respect to the elements of $\boldsymbol{x}$, combined with the equations

$$\begin{pmatrix} u_i' \\ v_i' \end{pmatrix} = \begin{pmatrix} \dfrac{\hat{u}_i'}{\hat{w}_i} - \dfrac{\hat{u}_i \hat{w}_i'}{\hat{w}_i^2} \\ \\ \dfrac{\hat{v}_i'}{\hat{w}_i} - \dfrac{\hat{v}_i \hat{w}_i'}{\hat{w}_i^2} \end{pmatrix} \tag{6.31}$$

where $\hat{u}_i'$ denotes the differentiation of $\hat{u}_i$ with respect to the relevant quantity.

The constraints on the dual quaternion in Eq. (6.15) can be included by linearizing around the prediction $\boldsymbol{x}_k^{(p)}$, which gives us two linear equations in $\boldsymbol{q}$ and $\boldsymbol{q}'$

$$1 \approx \boldsymbol{q}^{(p)T} \boldsymbol{q}^{(p)} + 2\boldsymbol{q}^{(p)T} (\boldsymbol{q} - \boldsymbol{q}^{(p)}) \tag{6.32}$$

$$0 \approx -\boldsymbol{q}^{(p)T} \boldsymbol{q}'^{(p)} + \boldsymbol{q}'^{(p)T} \boldsymbol{q} + \boldsymbol{q}^{(p)T} \boldsymbol{q}', \tag{6.33}$$

which can be included in the output $\boldsymbol{y}_k$.

Including the hand-eye constraints from Eq. (6.14) is also straightforward. Consider two different robot poses, represented by the dual quaternions $\check{\boldsymbol{q}}_{B_1}$ and $\check{\boldsymbol{q}}_{B_2}$, and the corresponding relative object-camera poses $\check{\boldsymbol{q}}_{A_1}$ and $\check{\boldsymbol{q}}_{A_2}$. We know from Eq. (6.12) that the scalar parts of $\check{\boldsymbol{q}}_A = \bar{\check{\boldsymbol{q}}}_{A_2} \check{\boldsymbol{q}}_{A_1}$ and $\check{\boldsymbol{q}}_B = \check{\boldsymbol{q}}_{B_2} \bar{\check{\boldsymbol{q}}}_{B_1}$ must be equal. Define the scalar part of the relative robot pose $\check{\boldsymbol{q}}_B$ as $\check{q}_B^{(0)} = q_B^{(0)} + \varepsilon q_B'^{(0)}$, which can be calculated directly from the forward kinematics of the robot. The scalar part of

$\check{q}_A$ can be seen from Eq. (6.7) to be $q_{A_1}^T q_{A_2} + \varepsilon(q_{A_1}'^T q_{A_2} + q_{A_1}^T q_{A_2}')$, with the quaternions written on vector form. Setting the scalar parts equal gives us two more linear equations in the states

$$q_{A_1}^T q_{A_2} = q_B^{(0)} \tag{6.34}$$

$$q_{A_1}'^T q_{A_2} + q_{A_1}^T q_{A_2}' = q_B'^{(0)}, \tag{6.35}$$

which can also be added to the system measurement equation, which can now be formulated as

$$y_k = C_k x_k + \delta_k \tag{6.36}$$

where δ_k is a sequence of uncorrelated Gaussian noise, and where the vector y_k and time-varying matrix C_k are obtained by stacking equations (6.27) for each edge search point, and adding constraints from Eqs. (6.32)–(6.33) and (6.34)–(6.35). Any number of robot motion constraints can be added to the measurement equation. In general each position used gives us two independent constraints on the pose, meaning that three positions are sufficient to completely constrain the estimated pose. This can be compared to the problem of hand-eye calibration, where it is well known that three positions are necessary for the calculation of the hand-eye transformation [Daniilidis, 1999].

State Dynamics Model. We choose to investigate two different versions of the function f in the state update equation (6.17). First, we assume the state equation

$$x_{k+1} = x_k + \epsilon_k, \tag{6.37}$$

where ϵ_k is an uncorrelated Gaussian noise sequence. The second option is to extend the state vector x_k with velocity $\vec{v}_k \in \mathbb{R}^3$ and angular velocity $\vec{\omega}_k \in \mathbb{R}^3$, and update the estimate of the dual quaternion using the equations

$$\dot{q} = \frac{1}{2}\omega q = \frac{1}{2}(0, \vec{\omega})q \tag{6.38}$$

and

$$\dot{q}' = \frac{1}{2}\dot{t}q + \frac{1}{2}t\dot{q} = \frac{1}{2}vq + \frac{1}{4}t\omega q \tag{6.39}$$

where the details can be found in [Goddard, 1997]. By discretizing Eqs. (6.38) and (6.39) using sample time h, we obtain the noise-free state update equation

$$\begin{pmatrix} \boldsymbol{q}_{k+1} \\ \boldsymbol{q}'_{k+1} \\ \tilde{\boldsymbol{K}}_{k+1} \\ \vec{\boldsymbol{\omega}}_{k+1} \\ \vec{\boldsymbol{v}}_{k+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \frac{h}{2}\boldsymbol{Q}_k & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \frac{h}{4}\boldsymbol{T}_k\boldsymbol{Q}_k & \frac{h}{2}\boldsymbol{Q}_k \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{q}_k \\ \boldsymbol{q}'_k \\ \tilde{\boldsymbol{K}}_k \\ \vec{\boldsymbol{\omega}}_k \\ \vec{\boldsymbol{v}}_k \end{pmatrix} \tag{6.40}$$

where $\tilde{\boldsymbol{K}} = (f, \gamma, u_0, v_0)^T$, and where the matrices

$$\boldsymbol{Q}_k = \begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{pmatrix} \tag{6.41}$$

and

$$\boldsymbol{T}_k = \begin{pmatrix} 0 & -t_x & -t_y & -t_z \\ t_x & 0 & -t_z & t_y \\ t_y & t_z & 0 & -t_x \\ t_z & -t_y & t_x & 0 \end{pmatrix} \tag{6.42}$$

correspond to the quaternion products with $\boldsymbol{q}_k = (q_0, q_1, q_2, q_3)$ and $\boldsymbol{t}_k = (0, t_x, t_y, t_z) = 2\boldsymbol{q}'_k\bar{\boldsymbol{q}}_k$ in Eqs. (6.38) and (6.39). With noise, Eq. (6.40) can be written as

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k\boldsymbol{x}_k + \boldsymbol{\epsilon}_k. \tag{6.43}$$

State Estimation

The linearized equations in Eqs. (6.40) and (6.36) will be used to recursively update the state estimate using an Extended Kalman Filter. The EKF based on Eqs. (6.40) and (6.36) uses image-space measurements to update the state, and will therefore be referred to as the *image based EKF*.

The computational complexity of the EKF grows rapidly with the dimension of the output vector $\boldsymbol{y}_k$. Similarly to Chapter 5, an EKF could instead be used to estimate the states of the system

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_k \boldsymbol{x}_k + \boldsymbol{\epsilon}_k \tag{6.44}$$

$$\boldsymbol{z}_k = \hat{\boldsymbol{C}}_k^{\dagger} \boldsymbol{y}_k = \hat{\boldsymbol{C}}_k^{\dagger} \boldsymbol{C}_k \boldsymbol{x}_k + \hat{\boldsymbol{C}}_k^{\dagger} \boldsymbol{\delta}_k \approx \boldsymbol{H}_n \boldsymbol{x}_k + \hat{\boldsymbol{C}}_k^{\dagger} \boldsymbol{\delta}_k \tag{6.45}$$

where $\hat{\boldsymbol{C}}_k$ is an approximation of the Jacobian $\boldsymbol{C}_k$ at the estimated pose $\hat{\boldsymbol{x}}_k$, and the new output $\boldsymbol{z}_k \in \mathbb{R}^n$ with $n = 8$ corresponds to a measurement of the object-camera pose, and $\boldsymbol{H}_n = \begin{bmatrix} \boldsymbol{I}_{n \times n} & \boldsymbol{0} \end{bmatrix}$. Assuming that the image space noise covariance matrix $\boldsymbol{E}\boldsymbol{\delta}_k \boldsymbol{\delta}_k^T = \sigma^2 \boldsymbol{I}$ as before, the new effective output noise vector $\hat{\boldsymbol{\delta}}_k = \hat{\boldsymbol{C}}_k^{\dagger} \boldsymbol{\delta}_k$ has covariance matrix

$$\boldsymbol{E}\left[\hat{\boldsymbol{\delta}}_k \hat{\boldsymbol{\delta}}_k^T\right] = \boldsymbol{E}\left[\hat{\boldsymbol{C}}_k^{\dagger} \boldsymbol{\delta}_k \boldsymbol{\delta}_k^T \hat{\boldsymbol{C}}_k^{\dagger T}\right] = \sigma^2 (\boldsymbol{C}_k^T \boldsymbol{C}_k)^{-1} \tag{6.46}$$

which can be used in the EKF. This EKF will be referred to as the *position based EKF*.

6.3 Experiments

The algorithm is evaluated in experiments using an image-generation tool, which generates semi-realistic images of a typical robot workcell using OpenGL 3D rendering. This makes it possible to simulate phenomena such as occlusion, specular reflections and noise due to a cluttered background.

The object model consists of a number of planar surfaces connected at their edges, see Fig. 6.2. At each step in the tracking visible object edges are selected and image edges are localized as described in Chapter 5.

Experiments

The experiments are performed in two steps. The tracker is initialized with a poor initial guess for the intrinsic camera parameters, which is used to get a very rough estimate of the object-camera pose. We then run the tracker for a little over a second with the robot stationary

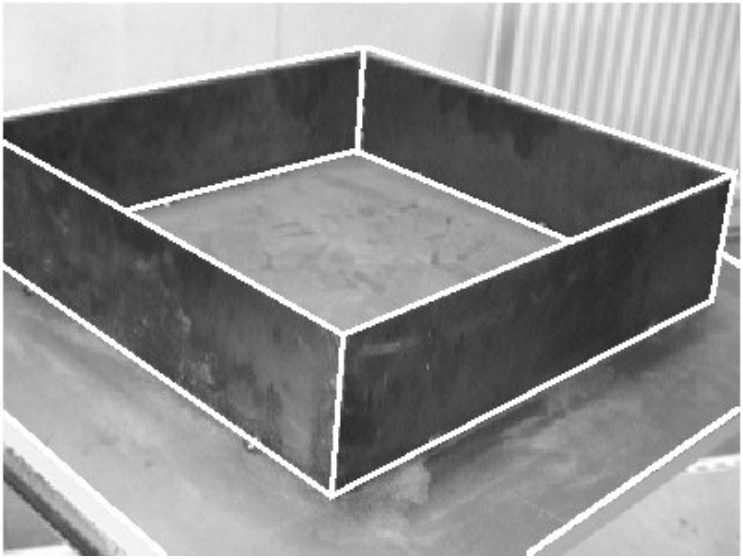

Figure 6.2 Example of image with superimposed object model, where hidden features have been removed.

to get a good initial estimate of the state. During the initialization phase the hand-eye constraints are not used, since they would require a good initial guess for the object pose. When the state estimate has converged, the tracker is started, using the initial estimate of the state as $\check{q}_{A_1}$ to constrain the position estimate according to Eqs. (6.34)–(6.35).

In this section the presented methods will be validated. There will be a comparison between when only using Eqs. (6.32) and (6.33) and when also using Eqs. (6.34) and (6.35). The first will be referred to as the *two constraints case* and the latter as the *four constraints case*. In the study we have looked at the mean of the absolute estimation error, which will be denoted with Δ. The number of search points used in the edge detections varied between 100 and 250 during the motion.

Visual Position Tracking

Figs. 6.3 and 6.4 show the result of tracking θ and t. The tracking of the position is satisfactory, both with four and with two constraints.

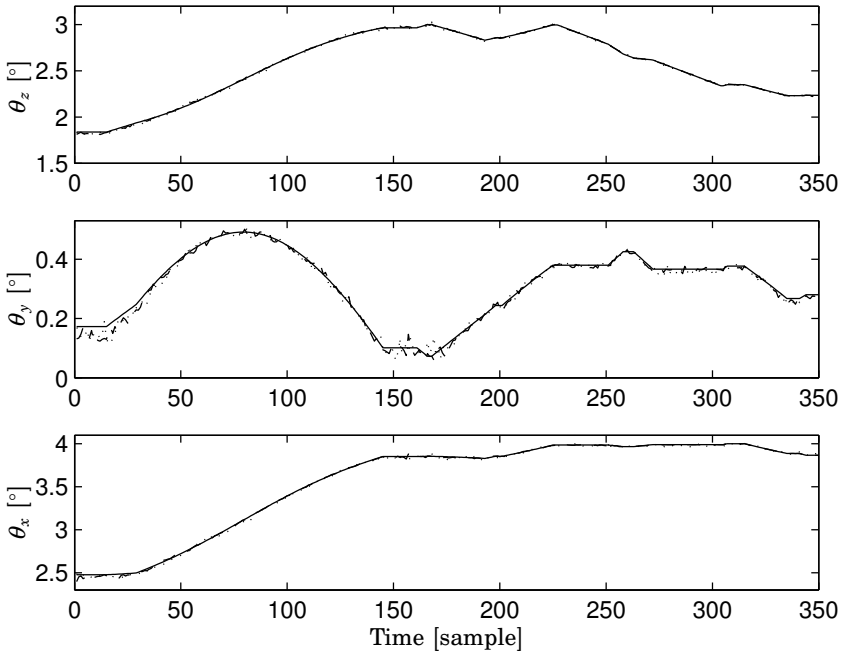

Figure 6.3 Tracking of θ. The diagram shows the real orientation (*solid*), estimated orientation using four constraints (*dashed*), and estimated orientation using two constraints (*dotted*).

There are some differences in the tracking accuracy, see Fig. 6.5. We see that with four constraints both the error and the variance of the estimation of θ and t are reduced.

Table 6.1 shows results using different conditions and number of constraints in the estimation of the object pose. For case 1 no extra noise was added to the measured output, but even so we have some noise due to the image processing. For case 2 extra noise $\in N(0,3)$ was added. Case 3 is the same as case 1, but with velocity estimation using the model in Eq. (6.43) instead of the model in Eq. (6.37). Case 4 is the same as case 2, but using velocity estimation. The initial state

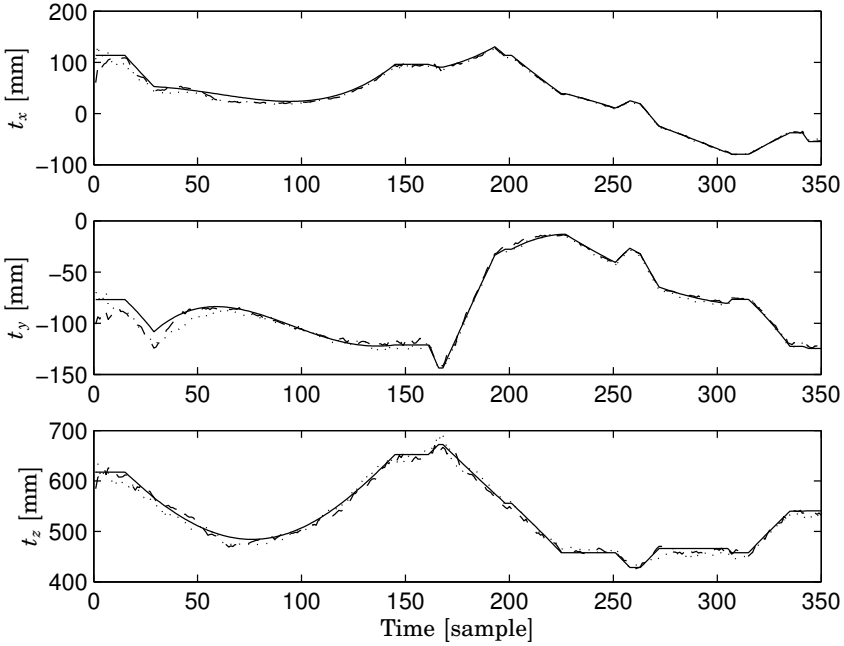

Figure 6.4 Tracking of t. The diagram shows the real translation (*solid*), estimated translation using four constraints (*dashed*), and estimated translation using two constraints (*dotted*).

covariance, P_0, and state noise covariance, Q, for case 1 and 2 were set to

$$\begin{aligned} P_0 &= \text{diag}(0.1^2 \cdot \mathbf{1}_{1\times 8} \;\; 50^2 \;\; 0.1^2 \;\; 40^2 \;\; 40^2) \\ Q &= \text{diag}(0.1^2 \cdot \mathbf{1}_{1\times 8} \;\; 3^2 \;\; 0.01^2 \;\; 0.2^2 \;\; 0.2^2). \end{aligned}$$

For case 3 and 4, the initial values were set to

$$\begin{aligned} P_0 &= \text{diag}(0.1^2 \cdot \mathbf{1}_{1\times 8} \;\; 50^2 \;\; 0.1^2 \;\; 40^2 \;\; 40^2 \;\; \mathbf{0}_{1\times 6}) \\ Q &= \text{diag}(\mathbf{0}_{1\times 8} \;\; 3^2 \;\; 0.01^2 \;\; 0.2^2 \;\; 0.2^2 \;\; 0.1^2 \cdot \mathbf{1}_{1\times 6}). \end{aligned}$$

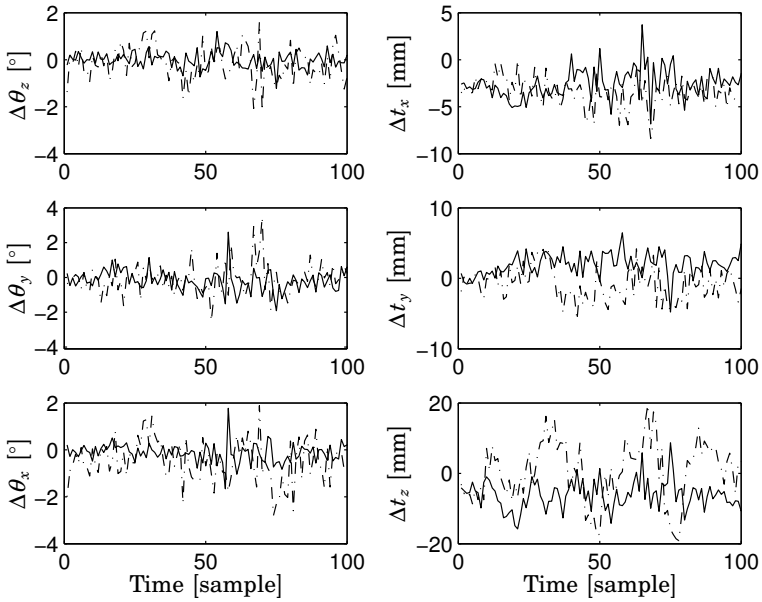

Figure 6.5 Error in the estimation of θ and t when using four constraints (*solid*) and when using two constraints (*dashed*).

The output noise variance was set to $E(\delta_k^2) = 1$ in case 1 and 3 and to $E(\delta_k^2) = 3^2$ in case 2 and 4.

Table 6.2 shows the prediction errors using the two different dynamical models described in Section 6.2.

Varying Focal Length.

Fig. 6.6 shows results of when the focal length was varied between 300 and 600 pixels during a different motion sequence. Still the tracking of the focal length was successful, and the effect on the depth estimation was small.

Incorrect Initial Values

Figs. 6.7 and 6.8 show results of when the tracker starts with incorrect initial values, both for the camera matrix and for the pose of the object. After approximately 30 samples the camera matrix and the position have converged to their correct value. The intrinsic camera parameters in this experiment were $f = 400$, $\gamma = 1.0$, $u_0 = 320$, and $v_0 = 240$.

Hand-Eye Calibration

In Table 6.3 the result of the hand-eye calibration is shown. The estimation is successful even if noise is added, and when the focal length is varying during the experiment.

Position-Based EKF

In order to compare the performance of the position based and image based EKF approaches, twenty different simulations were performed with different random motion of the camera. The resulting estimation errors in orientation and translation for the two methods are shown in Fig. 6.9. The covariance matrices and weights on the constraints were identical for the two methods, and synthetic independent noise

Table 6.1 Comparison between two and four constraints, and with or without velocity estimation.

Case	1		2		3	4
# constr.	4	2	4	2	4	4
$\Delta\theta_z$ (°)	0.122	0.216	0.292	0.541	0.120	0.184
$\Delta\theta_y$ (°)	0.317	0.313	0.529	0.632	0.335	0.346
$\Delta\theta_x$ (°)	0.130	0.381	0.221	0.777	0.140	0.136
Δt_x (mm)	2.105	3.406	2.434	3.310	2.456	2.086
Δt_y (mm)	2.390	2.093	2.322	2.878	2.456	1.867
Δt_z (mm)	4.857	4.926	5.704	7.228	5.148	5.178
$\|\Delta\boldsymbol{\theta}\|$ (°)	0.364	0.538	0.643	1.138	0.382	0.415
$\|\Delta\boldsymbol{t}\|$ (mm)	5.808	6.345	6.622	8.455	6.210	5.887

$\in N(0,3)$ was added to the image space measurement vector. The number of edge search points varied between 500 and 700 in the experiments. For this number of features, the execution time for the image based EKF equations was approximately 8–10 times longer than for the position based EKF in a C implementation. The total execution time, including image processing and computation of Jacobians, was around three times longer for the image based EKF.

Real World Experiments.

Fig. 6.10 shows the results of an experiment using images from a Sony DFW-V300 640x480 pixels digital camera mounted on an ABB Irb2000 industrial robot in an eye-in-hand configuration. The setup is shown in see Fig. 6.1, see also Fig. 6.2 for an example image. The top figures in Fig. 6.10 show the estimated focal length and principal point, which should be compared to the values $f = 1020$ pixels, $u_0 = 344$ pixels and $v_0 = 215$ pixels from an off-line camera calibration. The lower figure shows the estimated position of the camera, where the lines indicate the direction of the camera z-axis.

6.4 Discussion

The use of the hand-eye constraints showed an improvement in the estimation of the parameters, even though the hand-eye transformation was unknown. Additionally, it is a reasonable assumption that the extra constraints improve the robustness of the tracking against other error sources, such as errors due to the edge detector locking on to false edges.

Table 6.2 Errors between measurements and prediction. Case 1 is with no velocity estimation, case 2 with velocity estimation using the model in Eq. (6.43).

Case	$\Delta\theta_z$	$\Delta\theta_z$	$\Delta\theta_x$	Δt_x	Δt_y	Δt_z
1	0.113	0.151	0.072	1.864	1.210	1.097
2	0.040	0.036	0.024	0.462	0.391	0.519

The estimated values showed a small improvement when using the dynamical model in Eq. (6.43), especially in the case with added noise. Additionally, the prediction was improved, which makes it possible to track faster motions.

The system is capable of performing a total calibration of all relevant parameters, based on only rough initial values. The use of dual quaternions and hand-eye constraints in this type of tracker has not previously been demonstrated. The fact that we are able to track even during changes in the intrinsic parameters is an advantage, for instance in vision-based control. This gives the practical advantage of allowing the system to dynamically change its field of view, allowing a wider range of motions.

One apparent drawback with our EKF-based algorithm is that the updating of the state covariance estimates is very time consuming when the number of outputs is large. Effectively, this limits the feasible number of edge search points to a value which is lower than for optimization based methods. As previously pointed out [Wunsch and Hirzinger, 1997], the position based EKF algorithm is more suitable for real time tracking when the number of image features is large. In our C implementation, a total computation time in each time step of around 10 ms has been achieved on a Pentium 4 2GHz workstation, for around 500 search points, making it possible to track multiple objects independently in multiple cameras, at camera frame rate.

Table 6.3 Hand-eye calibration, with and without additional noise $\in N(0,3)$.

Errors	No noise	Noise	No noise, f varying
$\Delta\theta_z$ (°)	0.125	0.190	0.411
$\Delta\theta_y$ (°)	0.504	1.474	0.356
$\Delta\theta_x$ (°)	1.067	1.491	0.851
Δt_x (mm)	0.632	9.624	1.112
Δt_y (mm)	5.617	13.72	5.249
Δt_z (mm)	3.263	12.96	7.506

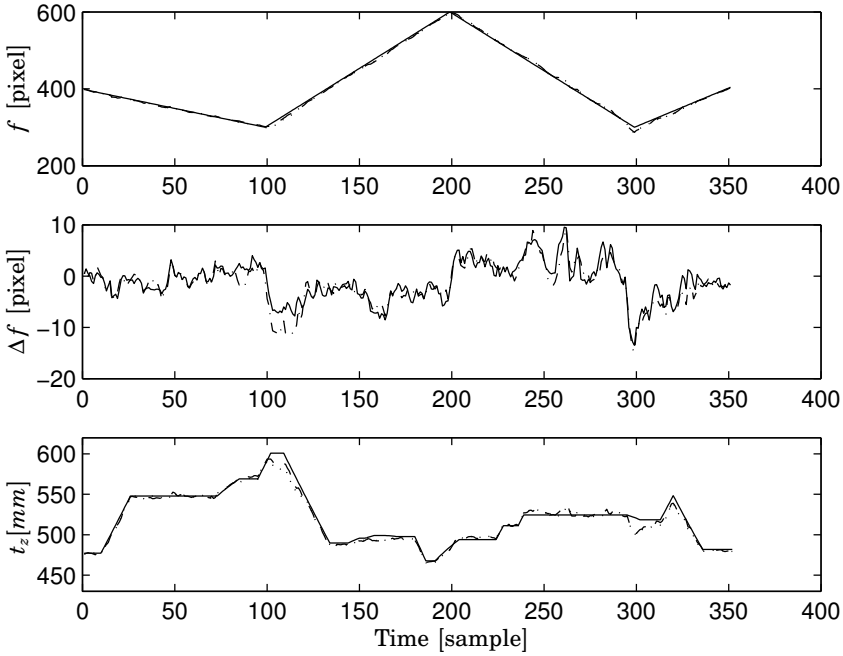

Figure 6.6 Experiment where f is varying between 300 and 600 pixels. The diagram shows the real values (*solid*), estimation using four constraints (*dashed*), and estimation using two constraints (*dotted*.)

6.5 Conclusion

In this chapter we have developed methods for real-time rigid body tracking, with simultaneous calibration and tracking of intrinsic parameters. A dual quaternion parameterization can be used to formulate linear hand-eye constraints on the estimated states, and these constraints help to reduce the tracking error. Additionally, it has been shown how the high computational complexity of the Extended Kalman Filter can be avoided by defining a new output vector in task space.

The data from the tracker could be used for hand-eye calibration,

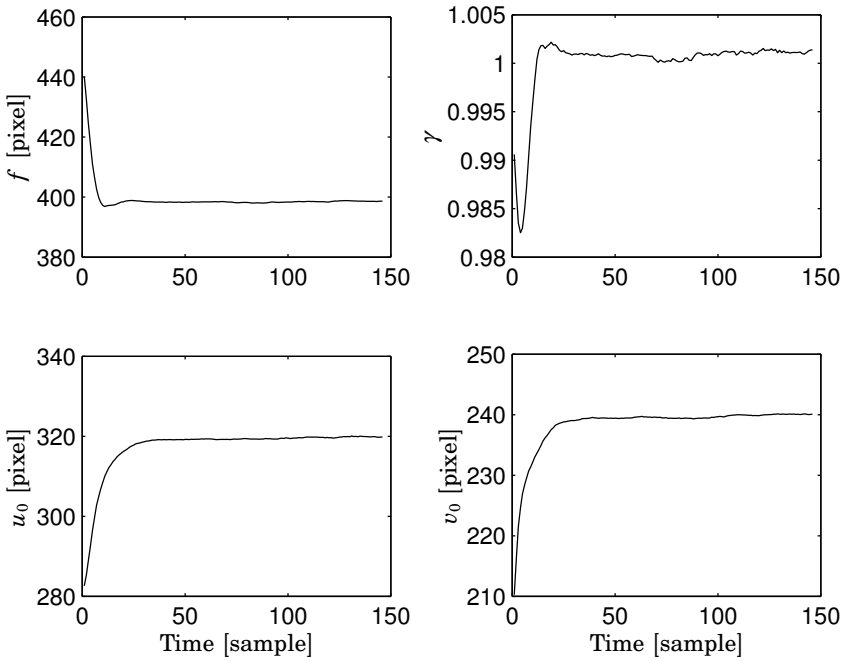

Figure 6.7 Transient responses of estimates of the focal length f, aspect ratio γ, and principal point (u_0, v_0) from incorrect initial estimates of the position and camera matrix.

with an error of approximately 1° in orientation and $5 - 10$ mm in translation. This is true also when intrinsic camera parameters are allowed to vary during the ten second long motion sequence.

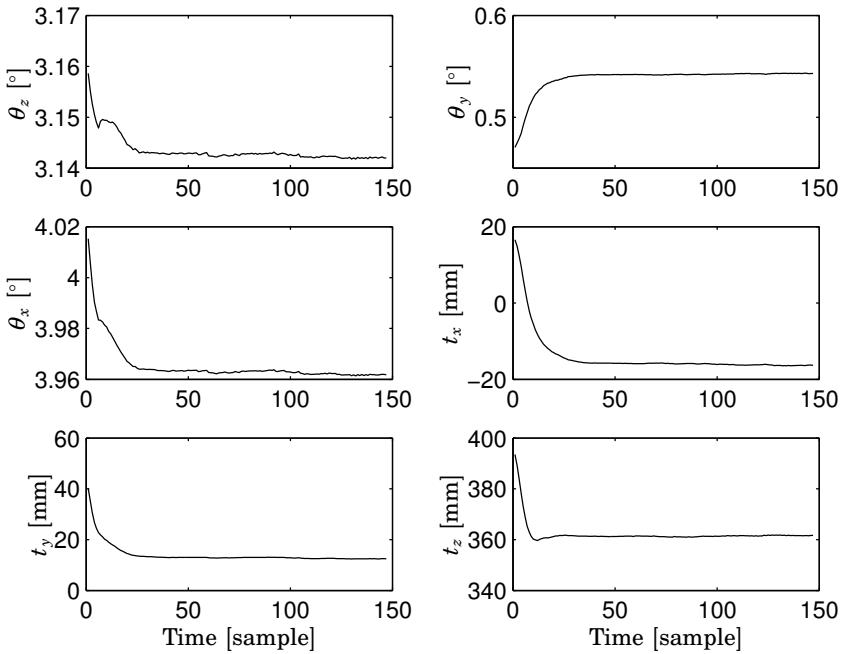

Figure 6.8 Transient responses of estimates from an incorrect initial position and camera matrix.

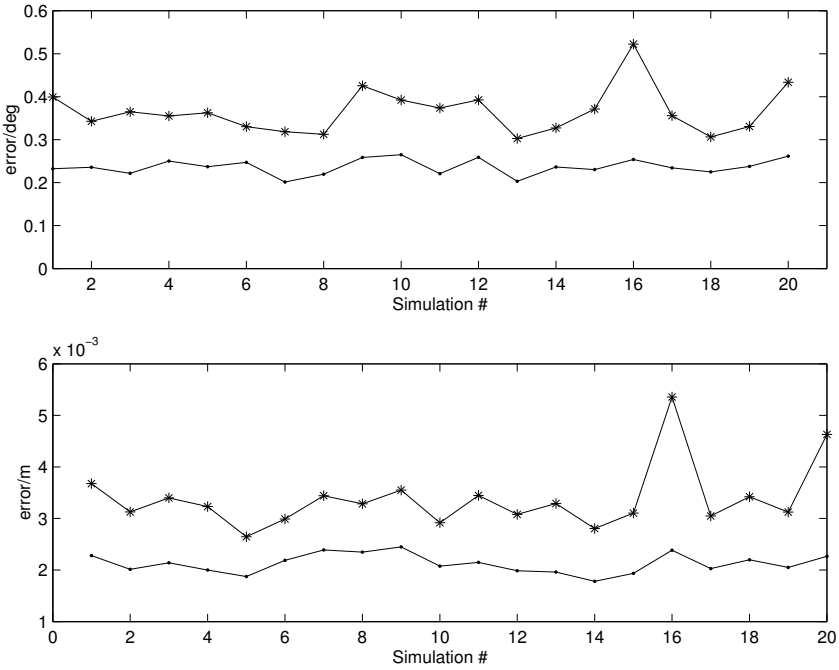

Figure 6.9 Mean of norm of errors in orientation and translation for position based (dots) and image based (stars) EKF.

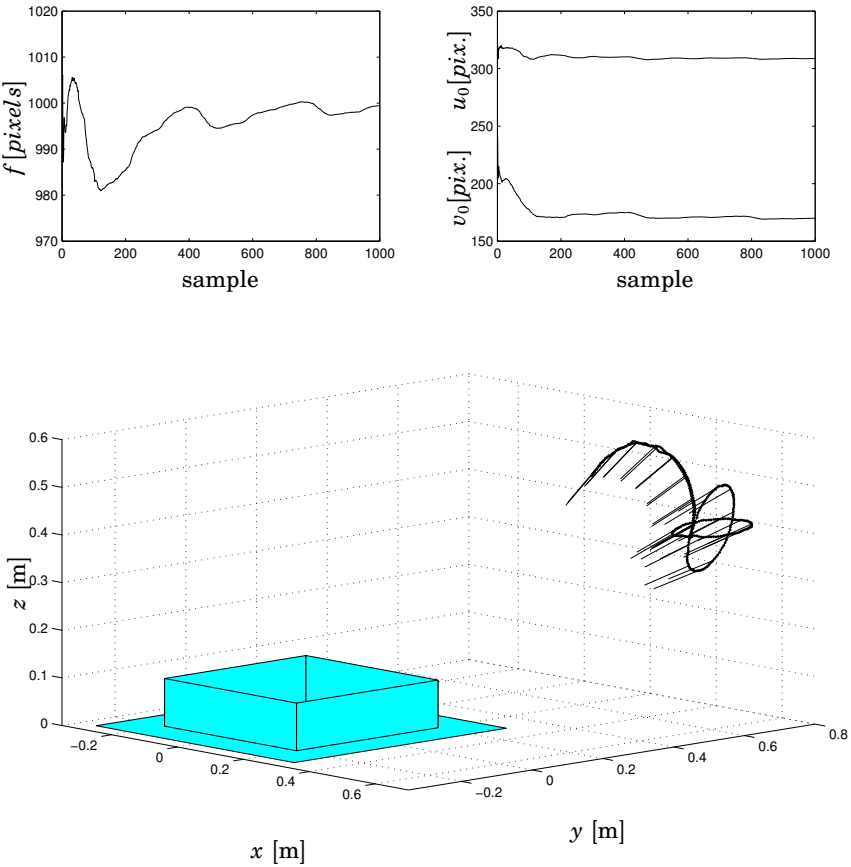

Figure 6.10 Estimated focal length, principal point, and trajectory in the real world experiment.

7

Multi-Camera Feedback Control with Time Constraints

7.1 Introduction

For real-time control applications in general, the importance of minimizing the input-output latency, i.e., the delay from the reading of the sensors to the generation of the control output, is well-known. Unless compensated for, the input-output latency will compromise the performance of the control system, and may even cause instability. In vision-based control systems the latency is dominated by the image processing. This chapter presents a method for multi-camera visual servoing, that aims at maximizing the achieved accuracy of the estimated vision-based feedback information during a predictable computation time.

In this chapter the problem of on-line resource allocation is considered. That is, to within a limited time frame maximize the estimation quality by a proper choice of active cameras and distribution of the feature points between these cameras. The camera selection algorithm is based on successive minimization of the measurement error covariance, by adding more cameras to the active camera set, until no more

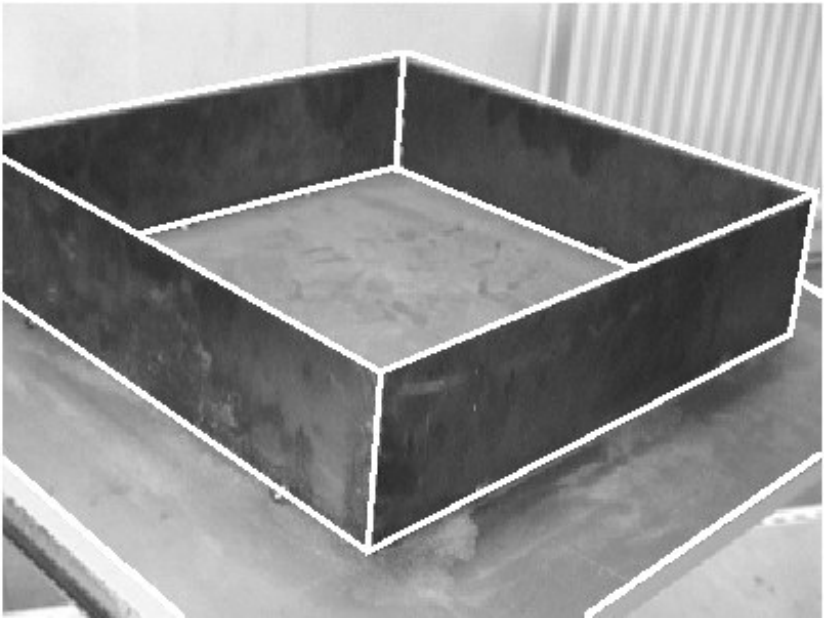

Figure 7.1 Measurements for estimating position/orientation from edges by updating predicted position (solid lines)

improvements of the estimation accuracy can be made. The suggested algorithm is evaluated in an extensive simulation study, using a setup of six cameras. The scheme is evaluated both in terms of estimation variance and control performance using a delay-compensating LQG-controller.

7.2 Estimation of Object Position and Orientation

We assume that M cameras are placed in fixed locations, viewing a target object whose position and orientation with respect to some fixed (world) coordinate system should be estimated. The position and orientation is parameterized as an n-vector $\mathbf{x}$ where typically $n = 6$ or

$n = 7$, depending on the selected parameterization of the orientation. The position $\mathbf{x}$ and the image-space position vector $\mathbf{y}$ are related by the projection equations of the cameras

$$\mathbf{y} = \mathbf{h}(\mathbf{x}) \tag{7.1}$$

in our case given by the homogeneous form pinhole camera projection equation

$$\mathbf{y}_i = \mathbf{h}_i(\mathbf{x}) = \frac{1}{Z}\mathbf{K}\mathbf{T}_{cb}\mathbf{T}_{be}(\mathbf{x})\mathbf{X}_i \tag{7.2}$$

where $\mathbf{K}$ is a matrix of internal camera parameters, $\mathbf{X}_i$ is the coordinates of the point in an object centered coordinate system, Z is the depth of the point in the camera, $\mathbf{T}_{cb}$ and $\mathbf{T}_{be}(\mathbf{x})$ are the homogeneous coordinate transformation matrices between the robot base the camera, and end-effector and robot base, respectively.

The estimated position $\hat{\mathbf{x}}_k$ at sample k can be updated iteratively as

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + \mathbf{J}^{\dagger}(\hat{\mathbf{x}}_{k-1})(\mathbf{y} - \mathbf{h}(\hat{\mathbf{x}}_{k-1})) \tag{7.3}$$

where $\mathbf{J}^{\dagger} = (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T$ is the pseudo inverse of $\mathbf{J} = d\mathbf{h}/d\mathbf{x} \in I\!R^{2N\times n}$. In the case of point-to-edge measurements, as explained in Chapter 5, we get the modified update equation

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k-1} + (\mathbf{N}^T\mathbf{J}(\hat{\mathbf{x}}_{k-1}))^{\dagger}\Delta\mathbf{y}_{(N)} = \hat{\mathbf{x}}_{k-1} + \mathbf{J}^{\dagger}_{(N)}(\hat{\mathbf{x}}_{k-1})\Delta\mathbf{y}_{(N)} \tag{7.4}$$

where $\mathbf{N}$ is a sparse $N \times 2N$ matrix, where the "diagonal" blocks are the normal directions at the N different measurement points along the edge, as shown Fig. 5.1. $\mathbf{J}_{(N)} \in I\!R^{N\times n}$ is the new Jacobian for point-to-edge measurements. When using edge features, we have significant freedom in how we choose which features to measure, since any number of edge searches can be performed anywhere along any object edge, in each camera. Usually, when using point-to-edge measurements we will have $N \gg n$.

Algorithm for Object Tracking

The algorithm for rigid body tracking is summarized in Fig. 7.2. The image pre-processing step involves all necessary image conversions and filtering necessary for each camera. The position at the next sample

```
while (true) {
     Read camera images;
     Perform image pre-processing;
     for i=each search point {
         dy[i] = edge distance;
     }
     x_est=x_est+invert(J)*dy;
     predict x_est one step ahead;
     determine visible features;
     place search points;
     calculate jacobians;
}
```

Figure 7.2 Algorithm for object tracking.

is predicted, and the predicted position is used to determine where interesting image features will be visible next sample. Visible features are determined as described in Chapter 5, and a large number of search points are divided between the cameras as described in Section 7.3 and placed along the predicted edges of the object. Finally, the Jacobians for each camera are computed.

Timing

The total computation time required in each sample depends on the number of cameras, M, and the total number of feature search points, N. The time required for reading and pre-processing all images is proportional to the number of cameras used, whereas the total time used for finding edges, placing search points, updating the estimation and building the Jacobians, is proportional to the total number of search points. The total time T_{tot} from sampling the cameras until the new estimation is obtained can therefore be modeled by the equation

$$T_{tot} = T_0 + T_c M + T_f N \tag{7.5}$$

where T_0 is a constant time required for image capture and image data transfer. The values of the time coefficients depends on many factors, such as camera sensor type and interface, camera shutter speed,

platform and implementation. Experimental values of the computation time have verified the timing model of Equation 7.5. Approximate values in our implementation have been determined experimentally to $T_0 = 6.0$ ms, $T_c = 1.0$ ms and $T_f = 0.01$ ms.

The implications of the timing model are twofold. First, as the computation time is deterministic and roughly constant, it can be compensated for by the control algorithm. Second, the relation between T_c and T_f shows a potential of gaining accuracy by switching cameras on and off, thereby allowing more features search points in the tracking. Assuming a desired computational delay, T_{comp}, and M_k active cameras at sample k, the number of features search points to distribute between the M_k cameras is given by

$$N_k = \frac{T_{comp} - T_o - T_c M_k}{T_f} \tag{7.6}$$

The delay T_{comp} is typically chosen in relation to the dynamics of the controlled system and the closed-loop bandwidth. With a camera frame rate of 30 Hz, corresponding to a sample period of 0.033 s, simple rules-of-thumb [Åström and Wittenmark, 1997] give that a realistic closed-loop bandwidth should lie between 6 and 18 rad/s.

A delay of 15 ms would then correspond to a phase lag of 5-15 degrees, which can be compensated for without too much performance degradation. Using the estimated camera timing parameters in our implementation, $T_{comp} = 15$ ms corresponds to a total of 300 image features points when using $M_k = 6$ cameras, and 800 points when using only one camera.

Thus, depending on the complexity of the scene and the dynamics of the control system, the relation between computational delay and number of features search points can be chosen arbitrarily off-line. For instance, in robotics applications with highly non-linear dynamics it is non-trivial to compensate for input-output latency. In this case it may be beneficial to have a small number of feature points and a short delay.

Estimation Accuracy

In general, it is clear that the accuracy of the estimation will improve with the number of image measurements N. If we assume that the

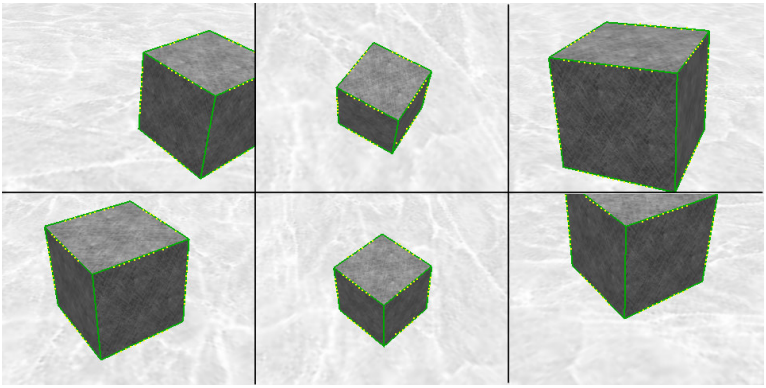

Figure 7.3 Example of six images from he simulated cameras with wireframe object superimposed, and search point locations indicated.

errors on the image measurements $\Delta \mathbf{y}_{(N)}$ can be modeled by Gaussian, independent noise with variance σ^2, the covariance of the estimation error $\tilde{\mathbf{x}} = (\mathbf{x} - \hat{\mathbf{x}})$ can be approximated as in Chapter 5 by

$$E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = \sigma^2(\mathbf{J}_{(N)}^T \mathbf{J}_{(N)})^{-1} = \sigma^2(\sum_{i=1}^{M} \mathbf{J}_i^T \mathbf{J}_i)^{-1} \tag{7.7}$$

where the Jacobian has been partitioned into M individual Jacobians for each camera as $\mathbf{J}_{(N)}^T = [\mathbf{J}_1^T \mathbf{J}_2^T \cdots \mathbf{J}_M^T]^T$.

The Jacobian for each camera is a function of the current position $\mathbf{x}$, as well as of the number of search points N_i for that camera, and how they are distributed. If search points are distributed evenly along the visible edges of the object, we can use the approximation

$$\mathbf{J}_i^T \mathbf{J}_i \approx N_i \Phi_i(\mathbf{x}) \tag{7.8}$$

where Φ_i is a positive semidefinite $n \times n$ matrix which does not depend on N_i. Using Eq. (7.8) in Eq. (7.7) we get

$$E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = \sigma^2 \left(\sum_{i=1}^{M} N_i \Phi_i(\mathbf{x}) \right)^{-1} \tag{7.9}$$

which shows that the covariance of the estimation error is a function of the number N_i of search points placed in each camera.

As can be seen from Eq. (7.7), the estimation error is also a function of the Jacobians, and therefore also a function of the object position $\mathbf{x}$. For one camera, $\Phi_i(\mathbf{x})$ is large and the resulting estimation error is small when the entire object is visible and close to the camera. Conversely, poorly conditioned situations occur when only part of the object is visible, or when the object is very far from the camera, where in extreme cases the problem could become very poorly conditioned. A common example is when all visible image features lie on a straight line, causing rotations of the object around this line to become unobservable from the image feature data.

7.3 Resource Allocation

When cameras are used for positioning, for instance in robotics, cameras need to be distributed so that the entire workspace is covered. Because of the limited resolution and field of view of each camera, it is usually beneficial to place the cameras so that each camera covers only a part of the available workspace. Some cameras may be placed so that they cover a large part of the workspace, giving rough information on the location of the object, while other cameras cover only part of the workspace for a more accurate localization. If the object is moving, different cameras will give more or less useful or accurate information at different times, depending on the current object position.

In general, the most accurate estimation of the position is obtained when using a subset of the available cameras. When timing is important, for instance when the estimated position is to be used for feedback control, it would be an advantage to use only the 'best' subset of cameras. The reason is the extra processing time required for each camera, as described in Section 7.2. In addition, each edge detection takes time, and therefore the edge search point should be distributed among these well-placed cameras.

Using the covariance of the estimation error as a measure of the estimation accuracy, we see from Eq. (7.9) that for a given object position $\mathbf{x}$, we must choose the numbers of features N_i in each camera in

the subset $\{C_k\}$ of all available cameras so that

$$E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T] = \sigma^2 \left(\sum_{i \in \{C_k\}} N_i \Phi_i(\mathbf{x}) \right)^{-1} \tag{7.10}$$

is minimized. In general, the covariance decreases with increasing N_i. Assuming a desired constant control delay T_{comp} as in Section 7.2 and the timing model in (7.5), we see that the total number of search points N, and the accuracy, decreases with the number M of cameras used.

Finding the optimal camera set and search point distribution from Eq. (7.10) for a general $\mathbf{x}$ is a non-trivial task. Simple heuristic choices for the optimal camera set, such as the best individual camera or all cameras, are possible, but can be very far from the optimum. Additionally, using a single camera or a low number of cameras could cause the problem to become ill-conditioned, for instance in situations where only a small part of the object is visible in each camera.

Algorithm

Since timing is important, a fast algorithm for selecting a suitable camera set and feature distribution has been developed. The algorithm is outlined in Fig. 7.4.

The algorithm updates the active set of cameras and the distribution of edge search points among the active cameras. This is done by testing if we can decrease the estimation error covariance by adding a camera and recomputing the distribution between search points in the current active set and the added camera. If the covariance can be decreased, the active set and distributions are updated with the new camera, and the algorithm tries to decrease the covariance further by adding another camera. If there is nothing that can be gained by adding another camera, the algorithm will stop and the current active set will be used in the next sample.

The algorithm is very robust and easy to implement. It takes negligible time to execute, since all information about the relative accuracy of the cameras is contained in the small $n \times n$-matrices Φ_i. The algorithm will in general not achieve the optimum covariance, but will find a small subset of cameras which together give a significantly lower covariance than for the heuristic choices.

```
N_feat[N_cam] = number of edge detection points using N_cam cameras;
for i = all cameras
  Φ[i] = transpose(J[i])*J[i]/N[i];
clear selected set of cameras and distribution of edge detection points;
set Φ_max = 0, best_val = 0, N_cam = 1, stop_flag = false;
while stop_flag == false {
  for k = all cameras not in currently selected set
    find the λ∈[0,1] minimizing new_val[k] = inv(λ*Φ_max+(1-λ)*Φ[k]);
  set best_new_cam = the camera cam with smallest new_val[cam];
  set best_new_ratio = the minimizing λ for camera best_new_cam;
  set Φ_new = best_new_ratio*Φ_max + (1-best_new_ratio)*Φ[best_new_cam];
  if (inverse(N_feat[N_cam]*Φ_new) < best_val) {
    update selected set by adding best_new_cam;
    redistribute edge detection points according to best_new_ratio;
    set Φ_max = Φ_new, best_val = inverse(N_feat[N_cam]*Φ_new);
    set N_cam = N_cam + 1;
  } else stop_flag = true;
}
```

Figure 7.4 Algorithm for selection of best camera set and edge detection point distribution.

7.4 Simulations

The accuracy of the tracking is evaluted in simulations using six cameras, where the images are generated using the standard graphics API OpenGL. The object being tracked is a textured box of dimensions 18 cm×18 cm×18 cm which is moved around in front of a textured background. Fig. 7.3 shows example images taken from a test sequence. We have assumed the timing model and values given in Section 7.2, the sampling period $h = 33$ ms and a maximum desired control delay $T_{comp} = 15$ ms.

Tracking Accuracy

The tracking accuracy for a stationary target is evaluated by measuring the estimation error variance for different image sequences, taken from different camera positions. Three different algorithms for resource allocation between the cameras are investigated.

1. Choosing the best camera, i.e., the camera for which Φ_i is 'largest', according to some chosen criterion.
2. Using all cameras, with equal distribution of edge search points.
3. Choosing the best set of cameras, using the algorithm in Section 7.3.

The tracking accuracy was measured for image sequences taken with several different camera configurations as given by Table 7.1. The estimated standard deviations for the error in estimated orientation and translation is shown in Table 7.2.

The estimation using the single-camera method did not converge for Sequence 2, since the problem becomes very poorly conditioned for any choice of a single camera. In Sequence 1, the minimum translation error was obtained by using all cameras, but the price is a significantly larger orientation error.

Control Performance

The visual feedback was applied in a feedback control setting, where the textured box was controlled one-dimensionally along the y coordinate axis. The estimated y-position was used as feedback information to the controller. The simulated dynamics was described by a second order system, which after discretization [Åström and Wittenmark, 1997]

Table 7.1 Camera configurations, distances z (in mm unless otherwise indicated) to target, and percentage of object visible in each camera.

Camera 1		Camera 2		Camera 3		Camera 4		Camera 5		Camera 6	
z	Vis.	z	Vis.	z	Vis.	z	Vis.	z	Vis.	z	Vis.
840	100	840	100	880	100	890	100	870	100	910	100
320	20	4 m	100	4 m	100	445	20	5 m	100	4 m	100
510	100	4 m	100	5 m	100	500	0	6 m	100	7 m	100
550	30	360	10	330	5	450	10	380	5	280	5
600	100	600	100	400	0	600	100	1 m	0	1 m	0
650	100	280	30	450	0	700	0	900	0	940	0
300	30	330	35	500	70	700	0	1 m	0	1 m	0
400	30	350	30	280	20	400	40	370	15	300	25

with the sampling interval $h = 33$ ms was given by

$$\begin{aligned} x(k+1) &= \begin{bmatrix} 1 & 0.033 \\ 0 & 0.97 \end{bmatrix} x(k) + \begin{bmatrix} 0.55 \\ 32.8 \end{bmatrix} u(k) \\ y(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x(k) \end{aligned} \tag{7.11}$$

The controller was a delay-compensating LQG-controller [Åström and Wittenmark, 1997], designed to maximize the continuous-time cost function

$$J(u) = \int \left(\begin{bmatrix} x(t) & u(t) \end{bmatrix} Q \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} \right) dt \tag{7.12}$$

with

$$Q = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.001 \end{bmatrix} \tag{7.13}$$

As seen by the process model in Eq. (7.11), only the first state is measurable. Therefore an observer was designed to reconstruct the state

Table 7.2 Tracking error standard deviations, orientation and translation, when using the best single camera, all cameras and the best selection of cameras, respectively.

#	Orientation error [°]			Translation error [mm]		
	Single	All	Sel.	Single	All	Sel.
1	0.18	0.36	0.18	0.75	0.49	0.68
2	∞	0.49	0.17	∞	0.37	0.25
3	0.12	1.23	0.10	0.26	1.64	0.21
4	0.22	0.20	0.16	0.97	0.28	0.26
5	0.12	0.29	0.10	0.35	0.36	0.31
6	0.14	0.20	0.08	0.36	0.46	0.17
7	0.22	0.25	0.07	1.10	0.28	0.16
8	0.08	0.09	0.05	0.24	0.14	0.12

vector. The state and output noise variances used in the design of the observer where chosen as $R_1 = 10 \cdot I$ and $R_2 = 0.01$.

The real-time simulations of the control system were conducted in MATLAB/Simulink using the TrueTime simulator [Henriksson *et al.*, 2002]. This simulator allows for co-simulation of continuous plant dynamics and discrete controller implemented as tasks in a computer. The simulation environment communicated with the 3D-visualization environment over TCP.

The true position of the continuous-time plant model was used to update the camera images. The controller was implemented as a periodic task with the sampling interval 33 ms. In the beginning of the sample, only the images of the cameras in the active set are read and processed. The position estimate is then fed into the LQG-control algorithm, whereafter the computed control signal is actuated. After the control signal actuation, the remaining images are read and the resource allocation algorithm is executed to obtain the camera set and point distribution for next sample.

Two separate simulations were performed, corresponding to scenario 4 and scenario 6 in Table 7.1, and the objective of the control was steady-state regulation of the position around $y = 0$, with no external load disturbances. Simulation results are shown in Figs. 7.5–7.8, showing the control signal and the controlled position variable for the two scenarios. It is seen that the suggested resource allocation algorithm results in better control performance than the heuristic choices for both scenarios.

7.5 Discussion

As can be seen from Table 7.2, the proposed method for choosing the best camera set works well for all camera configurations in Table 7.1. The heuristic selection of the best single camera works well in situations where we can find a single camera that gives sufficient information, but will fail in many common configurations such as the one in Sequence 2. Using all cameras works well in most situations, but is rarely necessary, and may not even be feasible if there are a large number of available cameras. The best selection algorithm will find a small set, typically consisting of 1–3 cameras, that will give suffi-

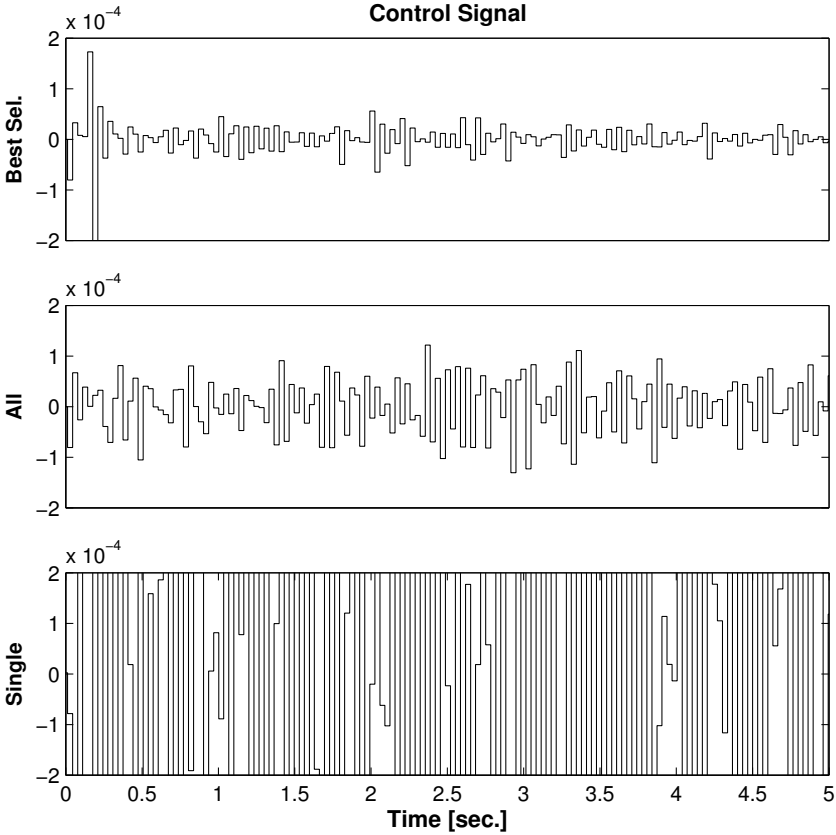

Figure 7.5 Control signal using the three different tracking algorithms in scenario #4 of Table 7.1.

cient information to accurately and robustly estimate the position and orientation. It also scales well with the number of available cameras.

It is important to note that we need the image data in order to correctly calculate the matrices Φ_i and the Jacobians $\mathbf{J}_i$. The reason is that not all image searches will be able to find a clear single edge, due to occlusions, specular reflections, noise, object texturing, etc. There-

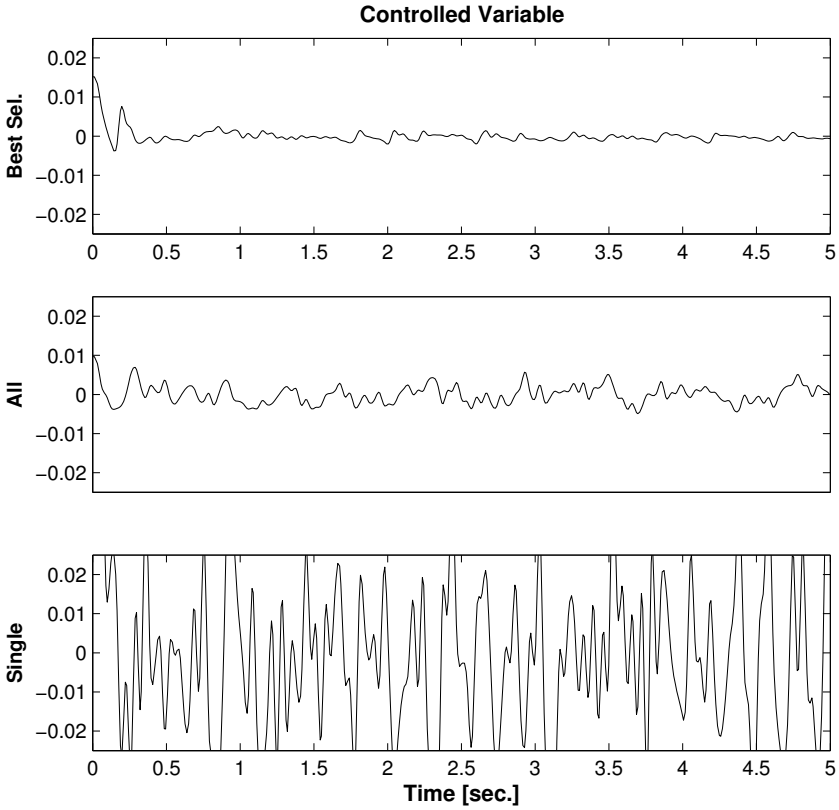

Figure 7.6 The controlled output using the three different tracking algorithms in scenario #4 of Table 7.1.

fore, the pre-processing and edge detection steps in the algorithm in Section 7.2 need to be performed for every image in every sample. However, for the cameras not used in the estimation, these steps can be performed *after* the estimated position has been obtained, and will therefore not affect the control delay T_{tot}.

The assumed model of the image noise as independent and Gaussian is usually not realistic, since there is a spatial correlation between

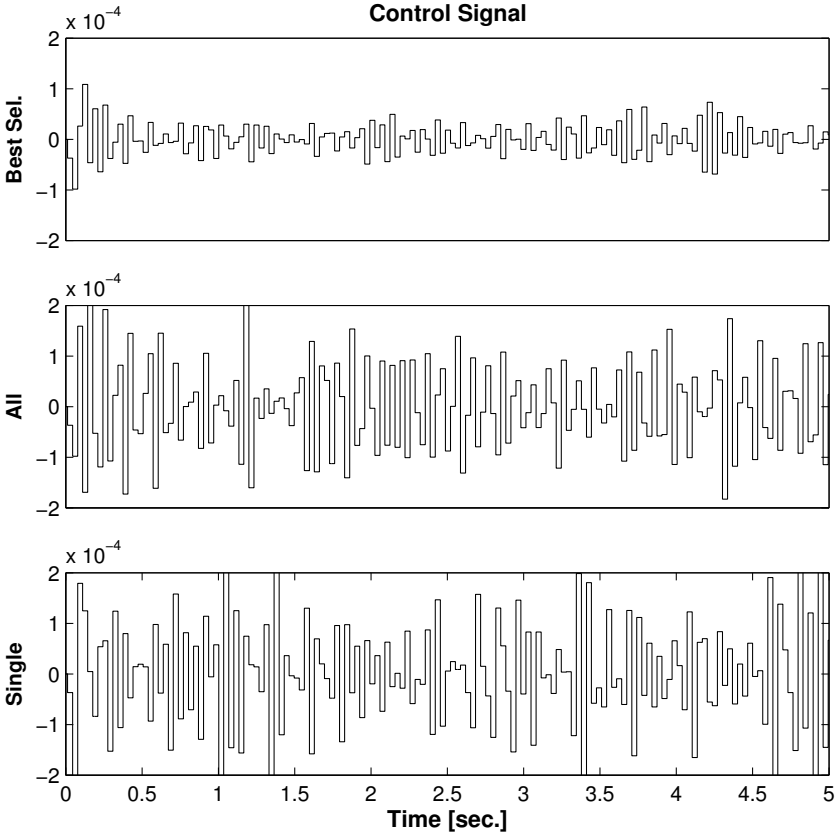

Figure 7.7 Control signal using the three different tracking algorithms in scenario #6 of Table 7.1.

the measurements, particularly for large N_i when the distance between search points will be small. However, the resource allocation algorithm will still give good results, as seen in Table 7.2. Another approximation involves the timing model in Eq. (7.5) and the approximation of the covariance given from Eq. (7.8), since not all N edge searches will result in an edge being found. Edges that are not found will not be

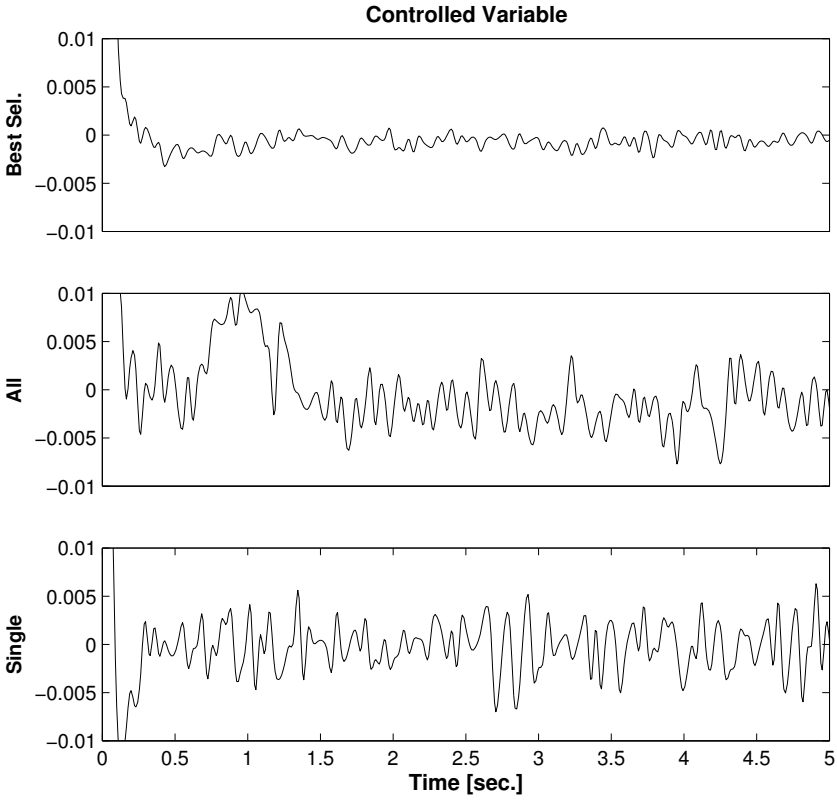

Figure 7.8 The controlled output using the three different tracking algorithms in scenario #6 of Table 7.1.

used in the state estimation update. Therefore, the computation time in Eq. (7.5) is a function of the number of searches, rather than the number of obtained measurements. This could be compensated for by measuring the percentage of successful edge detections, and assuming that this ratio is independent of N_i.

7.6 Conclusions

This chapter has presented a method for dynamic resource allocation in multi-camera based feedback control. It is shown how the covariance of the position estimation error depends on the set of cameras used and the number of edge detection points in each camera image. These parameters in turn affect the timing properties of the tracking algorithm.

An experimentally verified timing model was used to quantify the relation between the number of active cameras and the number of edge detection points. Using this timing model, it was possible to obtain a nearly constant input-output latency of the control loop. The latency was then compensated for by the controller.

The objective of the resource allocation algorithm was to minimize the variance of the position estimate within the time specified by the desired input-output latency. This was obtained by a proper choice of active cameras and point distribution between these cameras.

Real-time simulations have been conducted to demonstrate the effectiveness of the algorithm. It was shown that the resource allocation scheme significantly outperformed the heuristic choices of using only the best camera, and distributing all edge detection points evenly between all cameras. The algorithm also scales well with the number of cameras used.

A

Camera Model

The most common camera model is the *pinhole* or *perspective camera*, and its structure and parameters are therefore described briefly here. For more information about this and other camera models, see for instance [Trucco and Verri, 1998]. The perspective camera model consists of a point **O**, the center of projection, and a plane π, the image plane. The origin of the camera centered coordinate system is in **O**, see Fig. A.1. The distance between **O** and π is the focal length f. The line perpendicular to π that goes through **O** is the optical axis, and the intersection of this line with π is the origin **o** of the image coordinate system, the principal point. The projection equations for a point $(X\ Y\ Z)^T$ in Cartesian space in the perspective camera are given by

$$x = f\frac{X}{Z}$$

$$y = f\frac{Y}{Z}$$

This can be written using homogeneous coordinates as

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

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

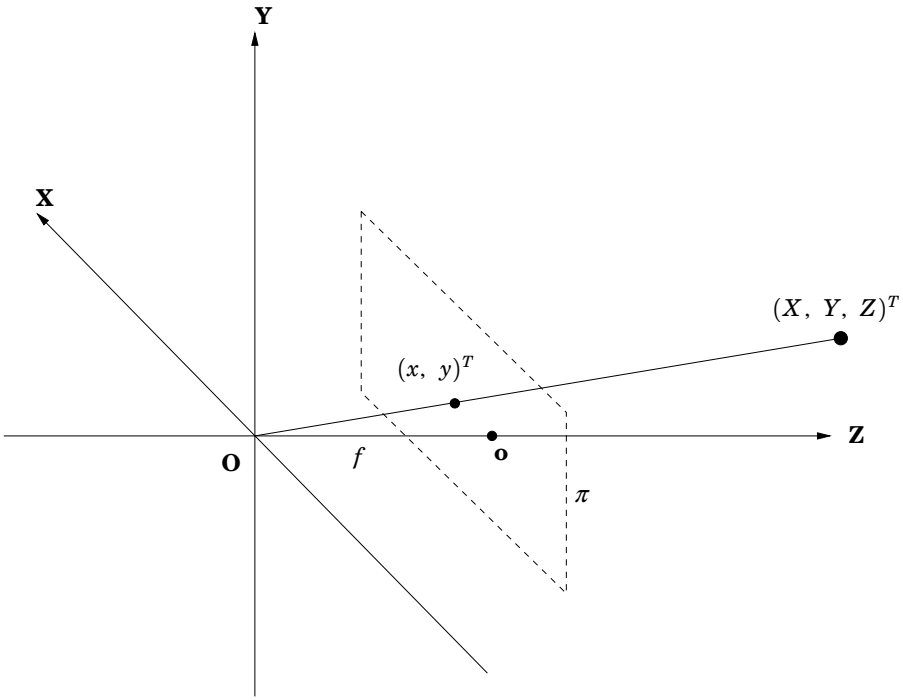

Figure A.1 The pinhole camera model.

To transform between image-plane coordinates $(x\ y)^T$ to pixel coordinates in the camera we need to introduce a number of intrinsic parameters that describe the CCD in the camera. These parameters allow us to describe non-quadratic pixels (aspect ratio$\neq$1), skew, and a principal point that is not located at the origin in the pixel grid, see [Trucco and Verri, 1998] for details. The new camera model becomes

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

$$= \begin{pmatrix} f & s & u_0 \\ 0 & \gamma f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} =$$

$$\stackrel{def}{=} \mathbf{K}\mathbf{R}_{3\times 4}\mathbf{X} \tag{A.1}$$

The matrix $\mathbf{K}$ is called the intrinsic camera matrix, and $\mathbf{R}_{3\times 4}$ is the extrinsic camera matrix. The intrinsic parameters f and γ describes focal length and aspect ratio, s is the skew (usually close to 0), and $(u_0\ v_0)^T$ is the principal point.

The matrix $\mathbf{R}_{3\times 4}$ can be used to change coordinate system in the world, usually to a coordinate system attached to some object in the scene:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \mathbf{K} \begin{pmatrix} \mathbf{R} & \mathbf{t} \end{pmatrix} \mathbf{X}^o,$$

where $\mathbf{R}$ is an orthogonal rotation matrix and $\mathbf{t}$ is a vector, describing the orientation and position of the camera and object frames respectively. $\mathbf{X}^o$ are the object points in the object-centered coordinate system.

B

Multi-Camera Calibration

For a multi-camera system, in addition to the intrinsic camera parameters, we need to find the relative positions of the sensors. When the cameras are fixed in the workspace, as depicted in Fig. B.1, we can attach the calibration object to the robot end-effector itself. From the kinematics of the robot we can get accurate information on the movement of the end-effector between the images. Therefore, the extrinsic parameters are partially known, and this is used in the algorithm in order to increase the accuracy and decrease the number of parameters that need to be estimated.

A calibration procedure in three steps can then be used, which find all of the necessary internal and external camera parameters shown in Fig. B.1:

1. Estimate of intrinsic and extrinsic camera parameters according to [Zhang, 1999].

2. Calculation of $\mathbf{T}_g^t$ by hand-target calibration according to [Tsai and Lenz, 1989b].

3. Calculation of all system parameters simultaneously using non-linear least squares optimization.

The last step is not strictly necessary, but have been found to improve the accuracy of the calibration. If we choose to perform this optimization, steps 1 and 2 may be used to obtain suitable starting values.

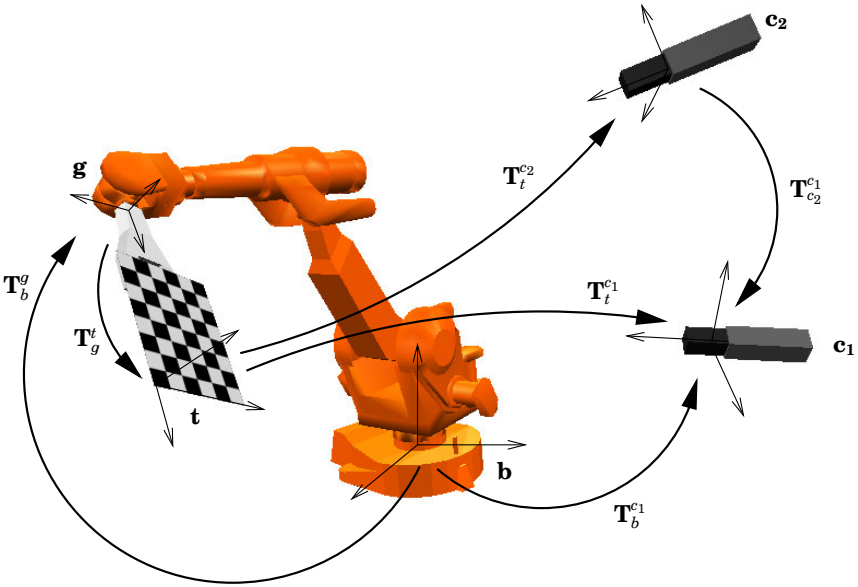

Figure B.1 The most important frames and transformations.

In step 2, a hand-eye calibration method [Tsai and Lenz, 1989b] can be used if we note that for two different end-effector positions, denoted by numbers p and q, and camera c_k, there holds a relationship

$$\mathbf{T}_t^{c_k}(q)^{-1}\,\mathbf{T}_t^{c_k}(p)\,\mathbf{T}_g^t = \mathbf{T}_g^t\,\mathbf{T}_b^g(q)\,\mathbf{T}_b^g(p)^{-1} \tag{B.1}$$

which can be solved for the unknown constant $\mathbf{T}_g^t$.

The final optimization is used to minimize, for both cameras, the errors between the m measured and reprojected image points in each of the n images

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{2}\left(\mathbf{x}_{ijk} - \hat{\mathbf{x}}_{ijk}(\mathbf{K}_1, \mathbf{K}_2, \mathbf{T}_g^t, \mathbf{T}_{c_2}^{c_1}, \mathbf{T}_b^{c_1})\right)^2$$

where k is the camera number, and $\hat{\mathbf{x}}_{ijk}$ is given by the projection

equations

$$\begin{aligned} \lambda_1 \hat{\mathbf{x}}_{i,j,1} &= \mathbf{K}_1 \mathbf{T}_b^{c_1} \mathbf{T}_b^g(i)^{-1} (\mathbf{T}_g^t)^{-1} \mathbf{X}_j \\ \lambda_2 \hat{\mathbf{x}}_{i,j,2} &= \mathbf{K}_2 (\mathbf{T}_{c_2}^{c_1})^{-1} \mathbf{T}_b^{c_1} \mathbf{T}_b^g(i)^{-1} (\mathbf{T}_g^t)^{-1} \mathbf{X}_j \end{aligned} \tag{B.2}$$

where $\mathbf{X}_j$ are the model points of the calibration object in its local coordinate system. $\mathbf{K}_1$ and $\mathbf{K}_2$ are matrices of intrinsic camera parameters, see [Zhang, 1999]. The minimization is performed with respect to $\mathbf{K}_1$, $\mathbf{K}_2$, $\mathbf{T}_b^{c_1}$, $\mathbf{T}_{c_2}^{c_1}$ and $\mathbf{T}_g^t$. Multiplying Eq. (B.2) with the inverses of $\mathbf{K}_1$ and $\mathbf{K}_2$ we get projection equations on the form

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} X \\ Y \end{pmatrix}, \tag{B.3}$$

for the point $(X \quad Y \quad Z)^T$ in camera coordinates.

C

Bibliography

Allen, P., B. Yoshimi, and A. Timcenko (1991): "Real-time visual servoing." In *IEEE International Conference on Robotics and Automation*, pp. 851–856.

Andreff, N., R. Horaud, and B. Espiau (2001): "Robot hand-eye calibration using structure from motion." *International Journal of Robotics Research*, **20:3**, pp. 228–248.

Arulampalam, M., S. Maskell, N. Gordon, and T. Clapp (2002): "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking." *IEEE Transactions on Signal Processing*, **50:2**, pp. 174–188.

Åström, K. J. and B. Wittenmark (1997): *Computer-Controlled Systems*. Prentice Hall, Upper Saddle River, NJ.

Baeten, J., H. Bruyninckx, and J. De Schutter (1999): "Combining eye-in-hand visual servoing and force control in robotic tasks using the task frame." In *Proceedings IEEE International Conference on Multisensor Fusion*, pp. 141–146. Taipei, Taiwan.

Besl, P. and N. McKay (1992): "A method for registration of 3-d shapes." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14:2**, pp. 239–256.

Blomdell, A., G. Bolmsjö, T. Brogårdh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, T. Olsson, A. Robertsson, and J. J. Wang (2004): "Extending an industrial robot controller

with a fast open sensor interface — implementation and applications." *IEEE Robotics and Automation Magazine*. to appear.

Chen, H. (1991): "A screw motion approach to uniqueness analysis of head-eye geometry." In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 145–151.

Clifford, W. (1873): "Preliminary sketch of bi-quaternions." *Proc. London Math. Soc.*, **4**, pp. 381–395.

Daniilidis, K. (1999): "Hand-eye calibration using dual quaternions." *International Journal of Robotics Research*, **18:3**, pp. 286–298.

Deguchi, K. and T. Noguchi (1996): "Visual servoing using eigenspace method and dynamic calculation of interaction matrices." In *Proceedings of the 13th International Conference on Pattern Recognition*, vol. 1, pp. 302–306.

Doucet, A., N. de Freitas, and N. Gordon (2001): *Sequential Monte Carlo Methods in Practice*. Springer-Verlag.

Drummond, T. and R. Cipolla (1999): "Real-time tracking of complex structures with on-line camera calibration." In *British Machine Vision Conference*, vol. 2, pp. 574–583.

Espiau, B., F. Chaumette, and P. Rives (1992): "A new approach to visual servoing in robotics." *IEEE Transactions on Robotics and Automation*, **8:3**, pp. 313–326.

Fischler, M. and R. Bolles (1981): "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM*, pp. 381–395.

Goddard, J. (1997): *Pose and motion estimation from vision using dual quaternion-based extended Kalman filtering*. PhD thesis, University of Tennessee, Knoxville, TN.

Gordon, N., D. Salmond, and A. Smith (1993): "Novel approach to nonlinear/non-gaussian bayesian state estimation." In *IEE Proc. F Radar and Signal Processing*, vol. 140, pp. 107–113.

Hamilton, W. R. (1853): *Lectures on Quaternions*. Hodges Smith & Co., Dublin.

Henriksson, D., A. Cervin, and K.-E. Årzén (2002): "TrueTime: Simulation of control loops under shared computer resources." In *Proceedings of the 15th IFAC World Congress on Automatic Control*. Barcelona, Spain.

Hogan, N. (1985): "Impedance control: An approach to manipulations parts I, II, III." *ASME Journal of Dynamic Systems, Measurement, and Control*, **107:1**.

Horaud, R. and F. Dornaika (1995): "Hand-eye calibration." *International Journal of Robotics Research*, **14:3**, pp. 195–210.

Hosoda, K., K. Igarashi, and M. Asada (1996): "Adaptive hybrid visual servoing/force control in unknown environment." In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1097–1103.

Hutchinson, S., G. Hager, and P. Corke (1996): "A tutorial on visual servo control." *IEEE Transactions on Robotics and Automation*, **12:5**, pp. 651–670.

Isard, M. and A. Blake (1998): "CONDENSATION-conditional density propagation for visual tracking." *International Journal of Computer Vision*, **29:1**, pp. 5–28.

Jörg, S., J. Langwald, J. Stelter, G. Hirzinger, and C. Natale (2000): "Flexible robot-assembly using a multi-sensory approach." In *IEEE Int. Conf. on Robotics and Automation*, pp. 3687–3694. San Francisco, CA, USA.

Julier, S. and J. Uhlmann (2004): "Unscented filtering and nonlinear estimation." *Proceedings of the IEEE*, **92:3**, pp. 401–422.

Lefebvre, T., H. Bruyninckx, and J. De Schutter (2002): "Comment on "A new method for the nonlinear transformation of means and covariances in filters and estimators" [with authors' reply]." *IEEE Transactions on Automatic Control*, **47:8**, pp. 1406 – 1409.

Li, P., T. Zhang, and A. Pece (2003): "Visual contour tracking based on particle filters." *Image and Vision Computing*, **21:1**, pp. 111–123.

Lippiello, V., B. Siciliano, and L. Villani (2002): "Objects motion estimation via BSP tree modeling and Kalman filtering of stereo

images." In *IEEE Int. Conference on Robotics and Automation*, pp. 2968–2973. Washington D.C.

Malis, E. and P. Rives (2003): "Robustness of image-based visual servoing with respect to depth distribution errors." In *IEEE Int. Conf. on Robotics and Automation*, pp. 1056–1061. Taipei, Taiwan.

Martin, F. and R. Horaud (2002): "Multiple camera tracking of rigid objects." *International Journal of Robotics Research*, **21:2**, pp. 97–113.

Martinet, P. and E. Cervera (2001): "Stacking jacobians properly in stereo visual servoing." In *Proceedings IEEE International Conference on Robotics and Automation*, pp. 717–722. Seoul, Korea.

Morel, G., E. Malis, and S. Boudet (1998): "Impedance based combination of visual and force control." In *Proceedings IEEE International Conference on Robotics and Automation*, pp. 1743–1748. Leuven, Belgium.

Murray, R., Z. Li, and S. Sastry (1994): *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL.

Nelson, B., J. Morrow, and P. Khosla (1995): "Improved force control through visual servoing." In *Proc. American Control Conf.*, pp. 380–386.

Nilsson, K. (1996): *Industrial Robot Programming*. PhD thesis ISRN LUTFD2/TFRT--LUTFD2/TFRT–1046–SE--SE, Lund Institute of Technology.

Olsson, T., J. Bengtsson, A. Robertsson, and R. Johansson (2003): "Visual position tracking using dual quaternions with hand-eye motion constraints." In *IEEE Int. Conference on Robotics and Automation*, pp. 3491–3496. Taipei, Taiwan.

Pichler, A. and M. Jägersand (2000): "Uncalibrated hybrid force-vision manipulation." In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1866–1871.

Schuurman, D. and D. Capson (2004): "Robust direct visual servo using network-synchronized cameras." **20:2**, pp. 319–334.

Siciliano, B. and L. Villani (1999): *Robot Force Control*. Kluwer Academic Publishers.

Trucco, E. and A. Verri (1998): *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, New Jersey.

Tsai, R. and R. Lenz (1989a): "A new technique for fully autonomous and efficient 3D robotics hand/eye calibration." In *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 345–358.

Tsai, R. and R. Lenz (1989b): "A new technique for fully autonomous and efficient 3d robotics hand/eye calibration." In *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 345–358.

van Dam, A., L. Foley, S. Feiner, and J. Hughes (1991): *Computer Graphics: Principles and Practice*, 2nd edition. Addison-Wesley, Boston, MA.

Wunsch, P. and G. Hirzinger (1997): "Real-time visual tracking of 3d objects with dynamic handling of occlusion." In *IEEE Int. Conf. on Robotics and Automation*, pp. 2868–2873. Albuquerque, NM, USA.

Xiao, D., B. Ghosh, N. Xi, and T. Tarn (1998): "Intelligent robotic manipulation with hybrid position/force control in an uncalibrated workspace." In *Proceedings IEEE International Conference on Robotics and Automation*, pp. 1671–1676. Leuven, Belgium.

Xiao, D., B. Ghosh, N. Xi, and T. Tarn (2000): "Sensor-based hybrid position/force control of a robot manipulator in an uncalibrated environment." *IEEE Trans. on Control Systems Technology*, **8:4**, pp. 635–645.

Zhang, Z. (1999): "Flexible camera calibration by viewing a plane from unknown orientations." In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1, pp. 666–673.

Zhou, Y., B. Nelson, and B. Vikramaditya (1998): "Fusing force and vision feedback for micromanipulation." In *IEEE Int. Conference on Robotics and Automation*, pp. 1220–1225.