ISSN 0280-5316
ISRN LUTFD2/TFRT--5676--SE

Vision Guided Force Control in Robotics

Tomas Olsson

Department of Automatic Control
Lund Institute of Technology
October 2001

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

Document name
MASTER THESIS

Date of issue
October 2001

Document Number
ISRN LUTFD2/TFR--5676--SE

Author(s)
Tomas Olsson

Superviso
Rolf Johansson, LTH
Johan Bengtsson, LTH
Henrik Malm, LTH

Sponsoring organization

Title and subtitle
Vision Guided Force Control in Robotics (Visuellt styrd kraftreglering inom robotik).

Abstract
One way to increase the flexibility of industrial robots in manipulation tasks is to integrate additional sensors in the control systems. Cameras are an example of such sensors, and in recent years there has been an increased interest in vision based control. However, it is clear that most manipulation tasks can not be solved using position control alone, because of the risk of excessive contact forces. Therefore, it would be interesting to combine vision based position control with force feedback.

In this thesis, we present a method for combining direct force control and visual servoing in the presence of unknown planar surfaces. The control algorithm involves a force feedback control loop and a vision based reference trajectory as a feed-forward signal. The vision system is based on a constrained image-based visual servoing algorithm, using an explicit 3D-reconstruction of the planar constraint surface. We show how calibration data calculated by a simple but efficient camera calibration method can be used in combination with force and position data to improve the reconstruction and reference trajectories.

The task chosen involves force controlled drawing on an unknown surface. The robot will grasp a pen using visual servoing, and use the pen to draw lines between a number of points on a whiteboard. The force control will keep the contact force constant during the drawing.

The method is validated through experiments carried out on a 6-degree-of-freedom ABB Industrial Robot 2000.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
49

Recipient's notes

Security classification

Contents

1. Introduction

1.1 Background

Visual servoing

When robots operate in industry, as is the case with the majority of the robots in use today, the environment can usually be controlled to suit the robots. For instance, assembly and other tasks are set up so that the robots know exactly where the different objects in the workspace can be found. However, if robots are to be used in an environment that can not be controlled in this way, for instance in a home, or in the control of mobile robots, much more flexibility is needed. This flexibility could also be useful in a traditional industrial environment, to avoid time consuming setup procedures.

One way to increase the flexibility would be to integrate more sensors in the robot systems. Cameras are examples of such sensors, and the use of visual information can increase the flexibility considerably in some tasks. By using data from the cameras it is possible to locate and control the interaction with different objects in the environment. Cameras can give a lot of information about the environment of the robot, but it could also be difficult and time consuming to extract all this information from the images.

Visual servoing is one example of the use of vision in robotics. A good background on visual servoing is given in [7], see also [4]. The main idea of visual servoing is to use the cameras as sensors, and use the data extracted from the images in a feedback loop. The goal is usually to align the manipulator with the object to be manipulated in a desired way, for instance so that the object can be grasped. It is convenient to describe this alignment as the desired relative pose, see Appendix B, between the manipulator and the object, given by

$$\mathbf{T}_g^o = \mathbf{T}_b^o \left(\mathbf{T}_b^g\right)^{-1}$$

where $\mathbf{T}_b^o$ and $\mathbf{T}_b^g$ are the poses of the object and the manipulator respectively, expressed in the robot base coordinate frame. The problem is of course that the object pose $\mathbf{T}_b^o$ is in general unknown.

One straightforward way of solving the alignment problem would be to do a full euclidean reconstruction of the object to be grasped, and then moving the robot manipulator to the pose

$$\hat{\mathbf{T}}_b^g = (\mathbf{T}_g^o)^{-1}\hat{\mathbf{T}}_b^o$$

where $\hat{\mathbf{T}}_b^o$ is the estimated pose of the object. The problem with this approach is that it requires a very accurate calibration of the camera system to be accurate, an assumption that is not realistic in most scenarios. If the setup of the camera system changes slightly, for instance if the cameras are moved, or the intrinsic parameters change, the system will have to be recalibrated or there will be errors in the euclidean reconstructions. Sufficiently

accurate calibration of the system can be a very time-consuming task, or even impossible if high accuracy is required. Therefore this approach is not suitable for many robotics tasks.

One solution to this problem is to use the camera measurements directly in a feedback loop, with a controller acting on the control error e, in its simplest form defined as

$$\mathbf{e} = \mathbf{y}_r - \mathbf{y}$$

where $\mathbf{y}_r$ is the setpoint and $\mathbf{y}$ is the measurements from the cameras. In this way the camera is considered to be just another sensor, and using feedback we can avoid the dependence on accurate calibration of the system. By using an appropriate controller it is possible to make the control error go to zero, which means that the alignment is, in theory, exact.

The quantities $\mathbf{e}$, $\mathbf{y}_r$ and $\mathbf{y}$ can be either poses in cartesian space, in which case we have a so called position-based servo, or image-space coordinates in an image-based servo. In the position-based servo the error signal $\mathbf{e}$ is calculated as the difference between the 3D-reconstruction of the manipulator $\mathbf{y}$, and the object to be grasped $\mathbf{y}_r$. Since this method requires 3D-reconstructions it is sensitive to calibration errors. The image-based servo solves this problem. Here, the signals $\mathbf{e}$, $\mathbf{y}_r$ and $\mathbf{y}$ are defined directly in image space. Therefore, no 3D-reconstruction is necessary. The camera measurements $\mathbf{y}$ are simply the pixel coordinates of some features on the end-effector that are to be aligned with some features on the object with image space coordinates $\mathbf{y}_r$. If a suitable set of features are selected, then $\mathbf{e} = 0$ will imply that the pose of the manipulator $\mathbf{T}_b^g$ is equal to the pose of the object $\mathbf{T}_b^o$. In this way we can achieve a positioning accuracy that is ideally independent of the calibration accuracy.

Force/vision control

When robots interact directly with objects in the environment, force sensing capabilities are crucial. Force control and hybrid force/position control have been studied for many years, and an introduction can be found in [5].

The nature and limited accuracy of vision based position control makes it less suitable for controlling interaction with objects in the environment. An interesting and obvious solution is to combine force control and visual servoing in a multi-sensor control system. In the last couple of years, some research on this subject has been presented [14, 10, 2, 11, 6].

Perhaps the most obvious approach to the problem is to combine the data from the cameras and force sensors using multi-sensor fusion methods. However, as many researchers have pointed out [10], the force- and visual sensors are fundamentally different, in that they measure very different physical phenomena, while the goal of most multi-sensor fusion methods is to obtain a single information from the sensor data. This makes such an approach less suitable in this case. Therefore, most force/vision control methods are based on a division of the workspace into force- and vision controlled directions. When a robot is interacting with a rigid surface, movement in the normal direction of the surface should be force controlled in order to accurately control the interaction. The remaining, unconstrained degrees of freedom could then be controlled by, for instance, a (constrained) visual servoing algorithm. A special problem is that the exact location and orientation of the surface may be unknown, and therefore has to be estimated.

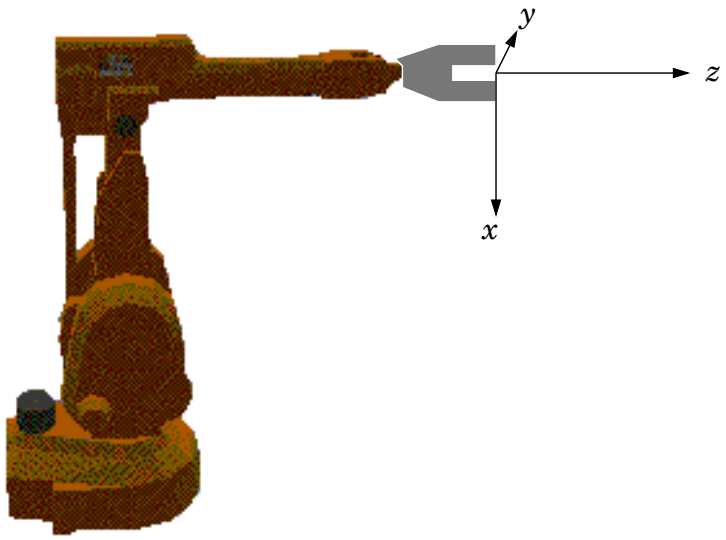

Figure 1.1 Irb 2000 and robot gripper frame.

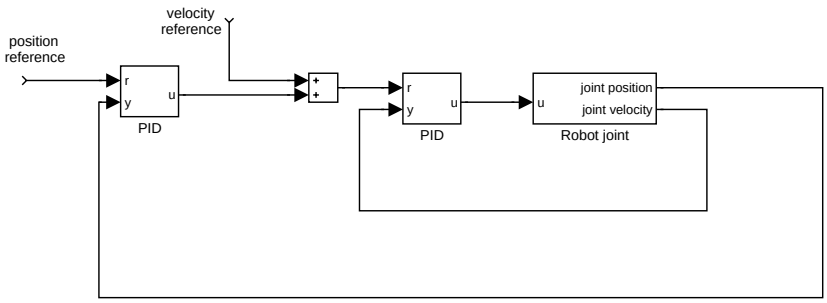

Figure 1.2 Structure of robot joint controllers.

1.2 The robot and camera system

In the robot lab at the Department of Automatic Control, Lund Institute of Technology, there are two industrial robots. The first one is an ABB Industrial Robot 2000, *Irb 2000*, which is the one used in these experiments. There is also an ABB Irb-6 robot, which holds the cameras. In these experiments the cameras are considered to be fixed in the workspace, and the Irb-6 is not moved during the experiments.

The Irb 2000 is built up by two large arms and a wrist, see Figure 1.1. The robot has 6 degrees of freedom, which means the end-effector can be moved to any desired position and orientation within the task space.

The robot has one built-in controller for each of the 6 joint angles. These controllers are cascaded PID controllers, with an outer position loop around an inner velocity loop according to the block diagram in Figure 1.2. The velocity signal used in the inner loop is obtained by differentiating and low-pass filtering the position signal. The robot can be controlled from Matlab/Simulink by sending trajectories of position- and velocity data to the robot through the network.

In the lab there are also two Sony DFW-V300 digital cameras. Both cameras are used in the experiments. The cameras are set up to take 30

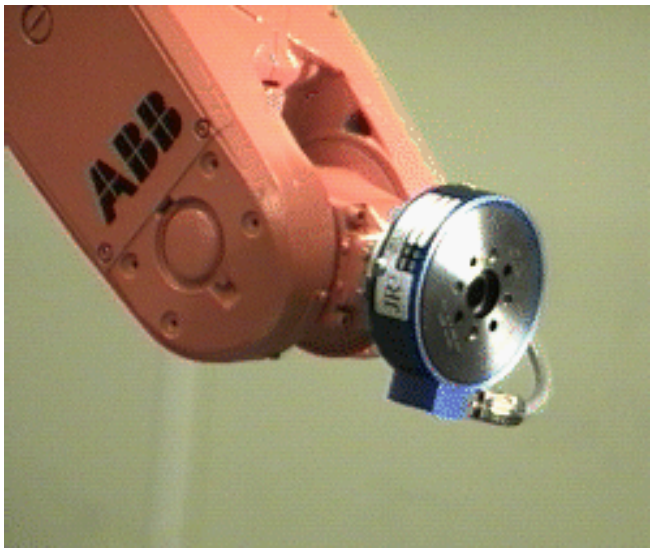

Figure 1.3 Robot with attached JR3 force/torque sensor.

images per second, and the images we get from the cameras are on the YUV 4:2:2 format. Only the grayscale (intensity) part of the images is used in the servoing, which makes the conversion from YUV very simple. The images are read into a computer using a IEEE-1394 (FireWire) connection, using the Fire-i API from UniBrain. The cameras are not synchronized, which means that the delay from image capture to control signal is not the same for both cameras.

We can also measure forces an torques on the robot gripper using a JR3 force sensor, shown in Figure 1.3. The JR3 is placed between joint six and the end-effector. It measures forces and torques in the x-, y- and z-directions of the end-effector coordinate system.

To protect the force sensor and robot from large contact forces at impact, the connection to the force sensor and end-effector is done by a pneumatic lock, which is connected to the emergency stop.

1.3 Problem formulation

The main goal of this work is to make the robot identify and interact with its environment, by using a combination of force measurements and computer vision. The chosen task is to make the robot pick up a pen, and use this to draw a picture on a whiteboard in the workspace. At the same time, it should accurately estimate the location and orientation of the whiteboard, and use this information to improve the control.

The drawing is specified by the user, who puts a number of dots on the whiteboard, and the robot will complete it by connecting the dots with straight lines. In order to complete this task it is necessary to use other sensors than just the cameras, because of the low accuracy of the visual information. The force control must make sure that the robot is in contact with the whiteboard, and that the contact force between the whiteboard and the gripper is constant. At the same time it must also make sure that

the pen is drawing a line between the specified points, which means that the tip of the pen should be moving across the whiteboard.

This task can be divided into a number of subtasks described below.

Camera and stereo system calibration

It is also intuitively obvious that the system needs some information on where in the workspace the cameras are located, and the camera setup. We can get this information by calibrating the cameras and the stereo system. The calibration process should give us information on where the two cameras are located relative to the robot, and the intrinsic parameters of the cameras, see Appendix A.

Grasping the pen

The goal is to guide the robot to the pen, and grasp it. The servo should be image-based, that is, no 3D-reconstruction of the scene should be required. The use of an image-based servo also means that only a coarse calibration of the camera system is needed.

The pen is placed on the table, the exact location is unknown. The goal here is to align the robot gripper with the pen in a pre-specified way, so that the pen can be grasped easily. The alignment between the gripper and the pen is specified as a relative pose in 3D, and is then transferred into image coordinates for the image-based servo. Here it is necessary to use 4 degrees of freedom, the position in the x-, y- and z-directions and the rotation around the z-axis of the robot.

Drawing on the board

After the pen has been grasped, the pen should be moved to the whiteboard, and lines should be drawn between the marker points on the board. During this phase, the orientation of the gripper will be kept constant, which means that this is a 3-degrees-of-freedom servoing task. The visual servoing algorithm will be used in combination with force control, to make sure that the contact forces between the pen and the whiteboard are kept constant while drawing.

Estimating the location of the board

The trajectories that the robot must follow are constrained to lie in the plane of the whiteboard. If the exact location of the board was known this could be used to improve the control. Here, the equation of the whiteboard plane must be estimated recursively from measurements of force, position and the image data.

1.4 Organization of the report

Chapter 2 describes the setup used, and the general methods used in the implementations. Section 2.2 presents the ideas behind the calibration method used to obtain the desired relative poses and camera matrices. In section 2.3 we present the image-based visual servoing system, used for constrained and unconstrained motion of the robot. We also present a method for specifying the end position $\mathbf{y}_r$ in image-space using cartesian object information, and a feature extraction and tracking method based on

Kalman filters. A simple error analysis is also presented, to give us an idea of the accuracy of the visual information. Section 2.4 presents a method of combining force control with visual servoing for drawing on the white-board, based on explicit estimation of the location of the rigid constraint surface.

In chapters 3 and 4 experimental results are presented, and the strengths and weaknesses of the proposed method are discussed.

Appendices A and B presents some basic material on the perspective camera model and cartesian transformations, necessary for the understanding of the presented calibration- and visual servoing methods.

2. Methods

2.1 Setup

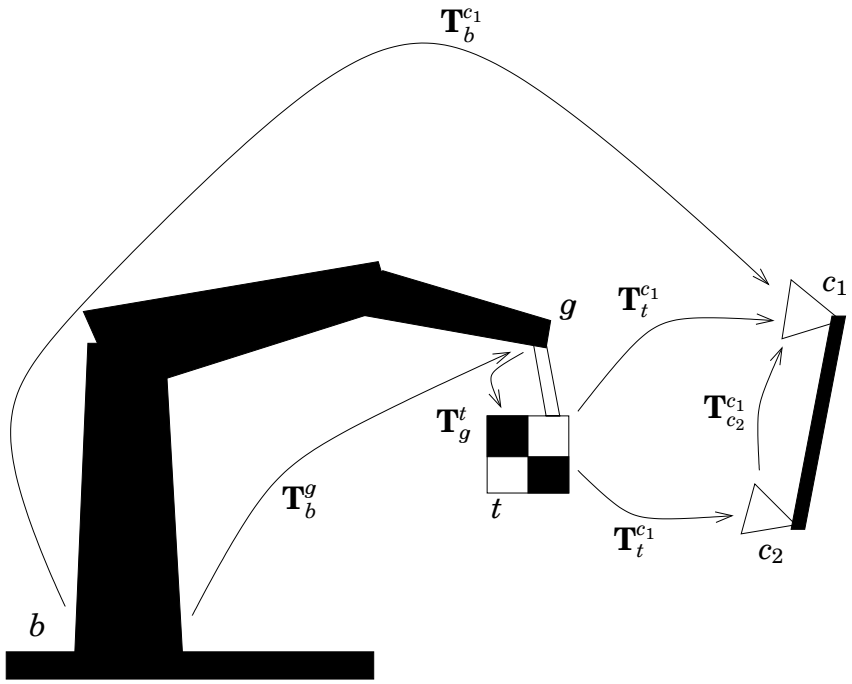

Figure 2.1 The most important frames and transformations

In the workspace we have a number of different frames, attached to different objects in the workspace, see Figure 2.1. The most important frames are the following:

c_1 **&** c_2 are the frames attached to the cameras, according to Figure A.1 in appendix A.

t is the frame attached to the calibration object. The coordinates of the model of the calibration object are expressed in this coordinate system.

g is attached to the end-effector, or gripper, on the robot.

b is attached to the base of the robot.

The transformations used to convert between frames are also shown in Figure 2.1. A background on cartesian coordinate transformations is given in Appendix B.

The different transformations used here are

$\mathbf{T}_t^{c_1}$ & $\mathbf{T}_t^{c_2}$ are the time-varying transformations between the cameras and the calibration object.

$\mathbf{T}_{c_2}^{c_1}$ describe how the cameras are oriented with respect to to each other.

$\mathbf{T}_{b}^{c_1}$ describe the transformation between camera 1 and the robot base.

$\mathbf{T}_{g}^{t}$ is the transformation between the gripper frame and the frame attached to the planar calibration object.

$\mathbf{T}_{b}^{g}$ is the transformation from the robot base to the gripper frame. This transformation is time-varying.

The transformations $\mathbf{T}_{t}^{c_1}$, $\mathbf{T}_{t}^{c_2}$, $\mathbf{T}_{c_2}^{c_1}$, $\mathbf{T}_{b}^{c_1}$ and $\mathbf{T}_{g}^{t}$ are estimated in the calibration, even if $\mathbf{T}_{g}^{t}$ is not used in the servoing. $\mathbf{T}_{b}^{g}$ can be accurately calculated from the measurements of the joint positions and kinematics of the robot.

2.2 Calibration of the camera system

Many methods for camera calibration use a number of images of a planar calibration object to estimate the camera parameters. The images are taken from many different positions and orientations, and the calibration estimates both the intrinsic and the extrinsic camera parameters, see Appendix A. One such method is described in [15].

In our case we attach the calibration object to the robot gripper itself, and from the kinematics of the robot we can get very accurate information about how the calibration object has moved between the images. Therefore, the extrinsic parameters are not completely unknown, and this extra information is used in the algorithm. Most importantly, this way we will need to estimate much fewer parameters, something that can be expected to increase the accuracy and robustness of the algorithm.

The setup of the camera/robot system is shown in Figure 2.1. The two cameras are fixed in the workspace, observing a calibration object (the target) that is rigidly attached to the gripper of the robot. The target is an A4 paper with a printed chessboard pattern. The goal of the calibration is to be able to calibrate the entire camera system using image information and cartesian robot gripper position information. The position and orientation of the gripper is obtained from measurement of the joint angles and the kinematics of the robot. Image information is obtained by taking a picture of the planar calibration object with each of the two cameras for n different robot gripper positions.

The calibration algorithm consists of three steps:

1. Calibration of intrinsic and extrinsic camera parameters
2. Hand-target calibration
3. Simultaneous optimization of all the calibration parameters in the entire camera/robot system

Steps 1 and 2 are used for obtaining suitable starting values for the nonlinear least squares optimization performed in step 3.

Estimation of intrinsic/extrinsic parameters

The method described here is based on [15]. The homogeneous image coordinates $\mathbf{m} = (u\ v\ 1)^T$ and calibration object coordinates $\mathbf{M} = (X\ Y\ Z\ 1)^T$ are related by the projective transformation $\mathbf{H}$ (see [15] for details):

$$\lambda \mathbf{m} = \mathbf{H}\mathbf{M}$$

where $\mathbf{H}$ is a 3x4 matrix

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 & \mathbf{h}_4 \end{pmatrix} = l \cdot \mathbf{K} \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{pmatrix} \tag{2.1}$$

where $\mathbf{K}$ is the matrix of intrinsic parameters, and l is an arbitrary scalar, since $\mathbf{H}$ is defined only up to a scale factor.

Using the fact that we can set $Z = 0$ in the planar calibration object, we have $\mathbf{M} = \begin{pmatrix} X & Y & 0 & 1 \end{pmatrix}^T$ and we can rewrite 2.1 as

$$\mathbf{H} = \begin{pmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_4 \end{pmatrix} = l \cdot \mathbf{K} \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{pmatrix} \tag{2.2}$$

Using the orthonormality of $\mathbf{r}_1$ and $\mathbf{r}_2$ we obtain two constraints on the intrinsic camera parameters as

$$\mathbf{r}_1^T \mathbf{r}_1 = 0 => \mathbf{h}_1^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_2 = 0 \tag{2.3}$$

$$\mathbf{r}_1^T \mathbf{r}_1 = \mathbf{r}_2^T \mathbf{r}_2 => \mathbf{h}_1^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{K}^{-T} \mathbf{K}^{-1} \mathbf{h}_2 \tag{2.4}$$

A closed form solution of the calibration problem is obtained by first estimating the projective transformation matrix $\mathbf{H}$ and then rewriting and solving these two equations. For estimation of $\mathbf{H}$ the method described in [15] is used.

If we define

$$\mathbf{B} = \mathbf{K}^{-T} \mathbf{K}^{-1} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{pmatrix}$$

and let the i:th column vector of $\mathbf{H}$ be

$$\mathbf{h}_i = \begin{pmatrix} h_{i1} & h_{i2} & h_{i3} \end{pmatrix}^T$$

we have

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{b}$$

with

$$\mathbf{v}_{ij} = \left(h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}\right)^T$$

and

$$\mathbf{b} = (B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33})^T$$

and equations 2.3 and 2.4 can be rewritten as

$$\begin{pmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{pmatrix} \mathbf{b} = \overline{\mathbf{0}}$$

By stacking $n \geq 3$ such equations and solving them using for instance Singular Value Decomposition we can obtain a unique solution. The estimated intrinsic camera parameters can easily be obtained directly from the solution b, see [15]. Then 2.2 gives us the estimation of the extrinsic camera parameters

$$\mathbf{r}_1 = \mathbf{K}^{-1}\mathbf{h}_1 / \|\mathbf{K}^{-1}\mathbf{h}_1\|$$
$$\mathbf{r}_2 = \mathbf{K}^{-1}\mathbf{h}_2 / \|\mathbf{K}^{-1}\mathbf{h}_1\|$$
$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$
$$\mathbf{t} = \mathbf{K}^{-1}\mathbf{h}_4 / \|\mathbf{K}^{-1}\mathbf{h}_1\|$$

In general, because of noise in the data, the matrix $\mathbf{R} = \begin{pmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{pmatrix}$ does not satisfy the property $\mathbf{R}^T\mathbf{R} = \mathbf{I}$ of a rotation matrix. The solution is to perform a singular value decomposition on $\mathbf{R} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, and setting $\mathbf{R}_{rot} = \mathbf{U}\mathbf{V}^T$. This gives the rotation matrix $\mathbf{R}_{rot}$ which minimizes the Frobenius norm of the difference

$$\|\mathbf{R} - \mathbf{R}_{rot}\|^2 = \text{trace}((\mathbf{R} - \mathbf{R}_{rot})^T(\mathbf{R} - \mathbf{R}_{rot}))$$

The estimation of $\mathbf{K}$ can be improved if we apply a ML-estimation by simply minimizing the error in the m reprojected pixels in all the n images

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\mathbf{m}_{ij} - \hat{\mathbf{m}}_{ij}(\mathbf{K}, \mathbf{r}_1^i, \mathbf{r}_2^i, \mathbf{t}_i, \mathbf{M}_j)\right)^2$$

with respect to $\mathbf{K}, \mathbf{r}_1^i, \mathbf{r}_2^i, \mathbf{t}_i$. This can easily be done with a suitable non-linear least squares method. This optimization is not necessary here, since the results will be used just to get a rough estimation of the parameters.

Hand-target calibration

The purpose is to obtain the rigid (cartesian) transformation $\mathbf{T}_g^t$, see Figure 2.1. This problem is solved by noting the similarity to a common problem in visual servoing called the hand-eye calibration problem. This is the problem of obtaining the cartesian transformation between the robot gripper and camera frames when the camera is mounted on the robot gripper. Most solutions to the hand-eye calibration problems work by solving a matrix equation of the form

$$\mathbf{A}\mathbf{X} = \mathbf{X}\mathbf{B} \tag{2.5}$$

where $\mathbf{A}$ is the 4x4 transformation matrix between two positions of the camera frame, and $\mathbf{B}$ the transformation between the corresponding positions of the gripper frame. $\mathbf{X}$ is the unknown hand-eye transformation. In our case we note that for the two different gripper poses the relation

$$\begin{aligned} (\mathbf{T}_t^{c_k})_i\,\mathbf{T}_g^t\,(\mathbf{T}_b^g)_i &= (\mathbf{T}_t^{c_k})_j\,\mathbf{T}_g^t\,(\mathbf{T}_b^g)_j \\ &\Longleftrightarrow \\ (\mathbf{T}_t^{c_k})_j^{-1}\,(\mathbf{T}_t^{c_k})_i\,\mathbf{T}_g^t &= \mathbf{T}_g^t\,(\mathbf{T}_b^g)_j\,(\mathbf{T}_b^g)_i^{-1} \end{aligned} \tag{2.6}$$

which is equivalent to 2.5 must hold for camera k, where $\mathbf{T}_t^{c_k}$ and $\mathbf{T}_b^g$ are known from the robot and image measurements, and $\mathbf{T}_g^t$ is the hand-target transformation.

Equation 2.5 can be decomposed into two equations, the first depending only on rotation:

$$\mathbf{R}_a\mathbf{R}_x = \mathbf{R}_x\mathbf{R}_b$$

Using the property that rotation matrices have one eigenvalue equal to 1 we can rewrite this as

$$\mathbf{R}_a\mathbf{R}_x\mathbf{n}_b = \mathbf{R}_x\mathbf{R}_b\mathbf{n}_b = \mathbf{R}_x\mathbf{n}_b$$

where $\mathbf{n}_b$ is the eigenvector of $\mathbf{R}_b$ corresponding to the eigenvalue 1. From this equation it can be concluded that

$$\mathbf{R}_x\mathbf{n}_b = \mathbf{n}_a \tag{2.7}$$

where $\mathbf{n}_a$ is the eigenvector of $\mathbf{R}_a$ corresponding to the eigenvalue 1. This can now be solved for the unknown $\mathbf{R}_x$. One method was introduced in [13]. The rotation $\mathbf{R}_x$ is represented by its unit eigenvector $\mathbf{R}_x$ and an angle θ_x. The rotations $\mathbf{R}_a$ and $\mathbf{R}_b$ are similarly represented by eigenvectors $\mathbf{n}_a$ and $\mathbf{n}_b$ and angles θ_a and θ_b. Equation 2.7 can be rewritten as (see [13]) as

$$(2\sin(\frac{\theta_a}{2})\mathbf{n}_a + 2\sin(\frac{\theta_b}{2})\mathbf{n}_b) \times \mathbf{n} = 2\sin(\frac{\theta_a}{2})\mathbf{n}_a - 2\sin(\frac{\theta_b}{2})\mathbf{n}_b$$

where $\mathbf{n} = \tan(\frac{\theta_x}{2})\mathbf{n}_x$. Since this equation can be shown to be rank deficient, at least two such equations are needed in order to obtain a unique solution. Therefore we need at least three different poses of the robot hand in order to obtain a least-squares solution, from which $\mathbf{R}_x$ can be computed [13].

Finally $\mathbf{t}_x$, the translation part of $\mathbf{X}$, can be calculated from the linear second equation

$$(\mathbf{R}_a - I)\mathbf{t}_x = \mathbf{R}_x\mathbf{t}_b - \mathbf{t}_a$$

obtained from equation 2.5.

Full system calibration

As a final improvement of the calibration an optimization is performed to minimize the error of the m reprojected points in the n images from each camera, given by

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

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

$$\begin{aligned} \lambda_1\hat{\mathbf{m}}_{ij,1} &= \mathbf{K}_1\mathbf{T}_b^{c_1}(\mathbf{T}_b^g)_i^{-1}(\mathbf{T}_g^t)^{-1}\mathbf{M}_j \\ \lambda_2\hat{\mathbf{m}}_{ij,2} &= \mathbf{K}_2(\mathbf{T}_{c_2}^{c_1})^{-1}\mathbf{T}_b^{c_1}(\mathbf{T}_b^g)_i^{-1}(\mathbf{T}_g^t)^{-1}\mathbf{M}_j \end{aligned} \tag{2.8}$$

The minimization is done with respect to $\mathbf{K}_1$, $\mathbf{K}_2$, $\mathbf{T}_b^{c_1}$, $\mathbf{T}_{c_2}^{c_1}$ and $\mathbf{T}_g^t$.

The purpose of the last step is to use as much information about the physical setup of the camera/robot system as possible. Here we use the fact that the transformations $\mathbf{T}_{c_2}^{c_1}$, $\mathbf{T}_b^{c_1}$ and $\mathbf{T}_g^t$ are constant.

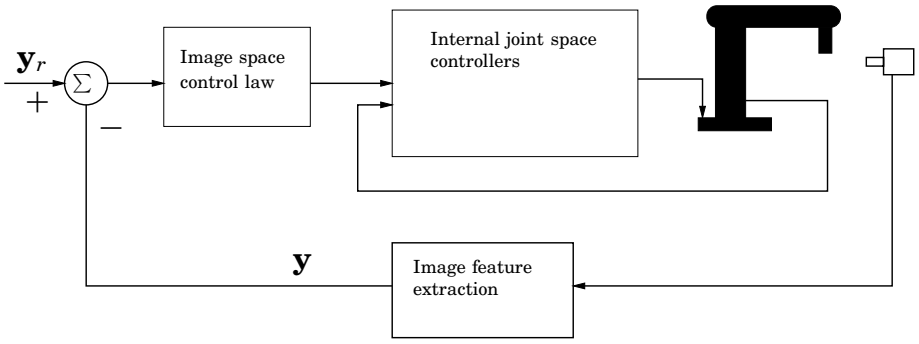

Figure 2.2 Image-based visual servo.

2.3 Visual servoing

In an image-based visual servo, see Figure 2.2 and section 1.1, the control is defined directly in image space quantities, by defining the control error $\mathbf{e}$ as

$$\mathbf{e} = \mathbf{y}_r - \mathbf{y}$$

where $\mathbf{y}_r$ and $\mathbf{y}$ are vectors of image space coordinates, $\mathbf{y}_r$ is the desired (end) position of the gripper in the images, and $\mathbf{y}$ is the measured current position. A simple control law in image-space that would drive the error $\mathbf{e}$ to zero is

$$\mathbf{u}_i = \dot{\mathbf{y}}_d = K_v \mathbf{e} \tag{2.9}$$

where K_v is a constant gain. The output $\dot{\mathbf{y}}_d$ from this controller is an image-space velocity vector, containing the desired velocities of the end-effector points in the image. Note that due to the rigidity of the end-effector, in general this set of velocities is not possible to achieve exactly.

The control signal sent to the robot is defined in task space, as a trajectory calculated from a so called velocity screw. The velocity screw is defined as a 6-vector of translational and angular velocities. Therefore the controller needs to relate the velocities in image space to those in task space, in order to achieve the desired movement of the end-effector in the image plane. If we let $\mathbf{r}$ be the coordinates of the end-effector in (m-dimensional) task space, and $\mathbf{y} = \mathbf{f}(\mathbf{r})$ be a k-vector of image coordinates, this can be done using the so called image Jacobian

$$\mathbf{J}_v(r) = \left[\frac{\partial \mathbf{f}}{\partial \mathbf{r}}\right] = \begin{pmatrix} \frac{\partial f_1(r)}{\partial r_1} & \cdots & \frac{\partial f_1(r)}{\partial r_m} \\ \vdots & & \vdots \\ \frac{\partial f_k(r)}{\partial r_1} & \cdots & \frac{\partial f_k(r)}{\partial r_m} \end{pmatrix}$$

and the relationship

$$\dot{\mathbf{y}} = \mathbf{J}_v(\mathbf{r})\dot{\mathbf{r}},$$

where $\dot{\mathbf{y}}$ is the image-space velocity of a feature point, and $\dot{\mathbf{r}}$ is the corresponding velocity screw in task space. Note that the image Jacobian is in general a function of the coordinates of the end-effector in task-space. The equation

$$\dot{\mathbf{y}}_d = \mathbf{J}_v(\mathbf{r})\dot{\mathbf{r}},$$

can be solved using the least-squares method, assuming that $\mathbf{J}_v$ is of full rank. This gives the velocity screw $\dot{\mathbf{r}}$ that will minimize $|\mathbf{J}_v(\mathbf{r})\dot{\mathbf{r}} - \dot{\mathbf{y}}_d|^2$. The velocity screw is usually expressed in the camera coordinate system. In order to generate the correct trajectories for the robot we also need to transform the velocity screws to the robot base coordinate system. Therefore we need the estimations of cartesian camera-robot transformations, obtained from the calibration of the stereo system described in section 2.2.

Image Jacobian

The image Jacobian for two cameras fixed in the workspace is derived as follows. First we derive the full Jacobian used for 6DOF servoing tasks, which can then be modified for the 4DOF and 3DOF servoing used here. Using the intrinsic camera matrices $\mathbf{K}_1$ and $\mathbf{K}_2$ we can normalize the cameras using the equations

$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} = \mathbf{x}_1^n = \mathbf{K}_1^{-1}\mathbf{x}_1,$$

and equivalently for camera 2

$$\mathbf{x}_2^n = \mathbf{K}_2^{-1}\mathbf{x}_2$$

giving the projection equations for a normalized camera

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{Z}\begin{pmatrix} X \\ Y \end{pmatrix} \tag{2.10}$$

Since the end-effector is a rigid object, its motion in the work space can be described by a velocity screw $\dot{\mathbf{r}}$, consisting of a translational and an angular velocity vector

$$\mathbf{T}(t) = \begin{pmatrix} T_x(t) & T_y(t) & T_z(t) \end{pmatrix}^T$$

$$\overline{\Omega}(t) = \begin{pmatrix} \omega_x(t) & \omega_y(t) & \omega_z(t) \end{pmatrix}^T$$

The time derivatives of the coordinates of a point on the end-effector $(X\ Y\ Z)^T$, expressed in the camera frame, can then be written as

$$\dot{X} = Z\omega_y - Y\omega_z + T_x \tag{2.11}$$
$$\dot{Y} = X\omega_z - Z\omega_x + T_y \tag{2.12}$$
$$\dot{Z} = Y\omega_x - X\omega_y + T_z \tag{2.13}$$

Differentiating the projection equations with respect to time and using these expressions we get

$$\begin{aligned} \dot{u} &= \frac{d}{dt}\left(\frac{X}{Z}\right) = \frac{\dot{X}Z - X\dot{Z}}{Z^2} = \frac{Z\omega_y - Y\omega_Z + T_x}{Z} - \\ &- X\frac{Y\omega_x - X\omega_Y + T_z}{Z^2} = [X = uZ,\ Y = vZ] = \\ &= \omega_y - v\omega_z + T_x/Z - uv\omega_x + u^2\omega_y - uT_z/Z \end{aligned} \tag{2.14}$$

$$\begin{aligned}\dot{v} &= \frac{d}{dt}\left(\frac{Y}{Z}\right) = \frac{\dot{Y}Z - Y\dot{Z}}{Z^2} = \frac{X\omega_z - Z\omega_X + T_y}{Z} - \\ &- Y\frac{Y\omega_x - X\omega_Y + T_z}{Z^2} = [X = uZ,\ Y = vZ] = \\ &= u\omega_z - \omega_x + T_y/Z - v^2\omega_x + uv\omega_y - vT_z/Z\end{aligned} \tag{2.15}$$

which can be written on matrix form

$$\begin{aligned}\begin{pmatrix}\dot{u}\\ \dot{v}\end{pmatrix} &= \begin{pmatrix}1/Z & 0 & -u/Z & -uv & 1+u^2 & -v\\ 0 & 1/Z & -v/Z & -1-v^2 & uv & u\end{pmatrix}\begin{pmatrix}T_x\\ T_y\\ T_z\\ \omega_x\\ \omega_y\\ \omega_z\end{pmatrix} = \\ &= \mathbf{J}(u,v,Z)\,\dot{\mathbf{r}}\end{aligned} \tag{2.16}$$

The image Jacobian is obtained by stacking the equations for both cameras and all points into one large matrix. One difficulty is that the velocity screws in the derivation above are expressed in the coordinate systems of the cameras, and are therefore not the same for the two different cameras. Therefore we need to express the velocity screws in the same coordinate system before stacking the equations for the two different cameras, preferably the robot base coordinate system, see [9]. This can be done using the equations

$$\begin{aligned}\dot{\mathbf{r}}^c &= \begin{pmatrix}\mathbf{R}_b^c\mathbf{T}^b - \mathbf{R}_b^c\overline{\Omega}^b \times \mathbf{t}_b^c\\ \mathbf{R}_b^c\overline{\Omega}^b\end{pmatrix} = \\ &= \begin{pmatrix}\mathbf{R}_b^c\mathbf{T}^b + \mathbf{t}_b^c \times \mathbf{R}_b^c\overline{\Omega}^b\\ \mathbf{R}_b^c\overline{\Omega}^b\end{pmatrix} = \\ &= \begin{pmatrix}\mathbf{R}_b^c & \mathbf{S}(\mathbf{t}_b^c)\mathbf{R}_b^c\\ \mathbf{0}_{3x3} & \mathbf{R}_b^c\end{pmatrix}\dot{\mathbf{r}}^b = \mathbf{M}_b^c\dot{\mathbf{r}}^b\end{aligned} \tag{2.17}$$

where b is the base frame, c is the camera and $\mathbf{S}(t)$ is the skew-symmetric matrix given by

$$\mathbf{S}(t) = \begin{pmatrix}0 & -t(3) & t(2)\\ t(3) & 0 & -t(1)\\ -t(2) & t(1) & 0\end{pmatrix}$$

This gives us the following expression for the Jacobian for two cameras, with n points in each camera

$$\mathbf{J}_v(r) = \begin{pmatrix} \mathbf{J}(u_1^{c_1}, v_1^{c_1}, Z_1^{c_1})\mathbf{M}_b^{c_1} \\ \vdots \\ \mathbf{J}(u_n^{c_1}, v_n^{c_1}, Z_n^{c_1})\mathbf{M}_b^{c_1} \\ \mathbf{J}(u_1^{c_2}, v_1^{c_2}, Z_1^{c_2})\mathbf{M}_b^{c_2} \\ \vdots \\ \mathbf{J}(u_n^{c_2}, v_n^{c_2}, Z_n^{c_2})\mathbf{M}_b^{c_2} \end{pmatrix} \tag{2.18}$$

and finally

$$\begin{pmatrix} \dot{u}_1^{c_1} \\ \dot{v}_1^{c_1} \\ \vdots \\ \dot{u}_n^{c_1} \\ \dot{v}_n^{c_1} \\ \dot{u}_1^{c_2} \\ \dot{v}_1^{c_2} \\ \vdots \\ \dot{u}_n^{c_2} \\ \dot{v}_n^{c_2} \end{pmatrix} = \mathbf{J}_v(r) \begin{pmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}^b \tag{2.19}$$

The unknown depths Z_i^c are estimated from the measurements of the robot end-effector pose and the calibration.

In practice the four equations given by each point includes some redundancy, since the point only contains three degrees of freedom. The redundant information is expressed by the epipolar constraint [12]. For our camera setup the epipolar constraint can be approximated by $v_1 = v_2$, meaning that the redundant information is in the vertical direction in the images. We remove this redundant information by simply deleting every fourth row in the Jacobian, more specifically the ones corresponding to the v-coordinate of the points in image 2. More general forms of the epipolar constraint could also be handled, see for instance [8].

4-degree-of-freedom Jacobian During the grasping phase the gripper could always be aligned so that the gripper z-axis points in the opposite direction of the z-axis of the robot base frame, that is, the gripper z-axis points straight down. This means that $\omega_x(t)$ and $\omega_y(t)$ are equal to 0, and that

$$\dot{y} = \mathbf{J}_v(r) \begin{pmatrix} T_x \\ T_y \\ T_z \\ 0 \\ 0 \\ \omega_z \end{pmatrix}^b = \mathbf{J}_v^4(r) \begin{pmatrix} T_x \\ T_y \\ T_z \\ \omega_z \end{pmatrix}^b \tag{2.20}$$

where $\mathbf{J}_v^4(r)$ denotes the matrix that consists of columns 1, 2 , 3 and 6 of the full 6DOF Jacobian $\mathbf{J}_v(r)$.

Constrained 3-degree-of-freedom Jacobian Similarly, during the drawing phase we can keep the orientation of the gripper constant, so that we can set $\omega_x(t)$, $\omega_y(t)$ and $\omega_z(t)$ to 0. This gives us

$$\dot{\mathbf{y}} = \mathbf{J}_v^3(r) \begin{pmatrix} T_x \\ T_y \\ T_z \end{pmatrix}^b$$

where $\mathbf{J}_v^3(r)$ is the matrix of the first three columns of $\mathbf{J}_v(r)$.

During this phase there is also the constraint that the motion should be in the board plane $\hat{\mathbf{p}}$ in cartesian space defined by

$$\hat{\mathbf{p}}^T \hat{\mathbf{r}} = 0 \tag{2.21}$$

where $\hat{\mathbf{p}} = (p_1,\ p_2, -1,\ p_4)^T$ and $\hat{\mathbf{r}} = (X,\ Y,\ Z,\ 1)^T$. This leads to an equation

$$\mathbf{p}^T \dot{\mathbf{r}} = 0 \tag{2.22}$$

for the constrained end-effector velocity $\dot{\mathbf{r}} = \left(\dot{X},\ \dot{Y},\ \dot{Z}\right)^T$, where we have $\mathbf{p} = (p_1,\ p_2,\ -1)^T$. The reduced image Jacobian on the surface of the plane is now given by

$$\begin{aligned} \dot{\mathbf{y}} &= \begin{pmatrix} \mathbf{J}_1 + p_1\mathbf{J}_3 & \mathbf{J}_2 + p_2\mathbf{J}_3 \end{pmatrix} \begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} = \\ &= \mathbf{J}_{v,c}^3(r) \begin{pmatrix} \dot{X} \\ \dot{Y} \end{pmatrix} \end{aligned} \tag{2.23}$$

where $\mathbf{J}_i$ denotes the i:th column in the Jacobian $\mathbf{J}_v^3(r)$ for the unconstrained motion. It is obvious that if $\mathbf{J}_v^3(r)$ has full rank, so has $\mathbf{J}_{v,c}^3(r)$.

Implementation of control law

The controller in Figure 2.2 is implemented as a proportional controller

$$\mathbf{u}_i = K_v \mathbf{e} \tag{2.24}$$

where

$$\mathbf{e} = \mathbf{y}_r - \mathbf{y}.$$

The image-space control signal $\mathbf{u}_i$ is as an image-space velocity vector, which is then converted into a velocity screw using the pseudo inverse of the appropriate image Jacobian from section 2.3

$$\dot{\mathbf{r}} = \mathbf{J}_v^+(\mathbf{r})\mathbf{u}_i \tag{2.25}$$

This velocity screw is then converted into a corresponding one sample long trajectory segment, which can be sent to the robot.

Calculation of trajectories

The conversion of the velocity screw $\dot{\mathbf{r}}(t) = \begin{pmatrix} \mathbf{T}(t) \\ \overline{\Omega}(t) \end{pmatrix}$ into a trajectory segment is done by first using the equivalent angle-axis representation of a rotation matrix, see [5]. The angular velocity $\overline{\Omega}$ is first converted into a unit vector $\hat{\mathbf{K}} = \begin{pmatrix} k_x & k_y & k_z \end{pmatrix}^T$ and an angle θ_K, describing the angle and the axis of the rotation that the gripper should perform in the next sampling interval. This conversion is done with the simple equation

$$\mathbf{K}(t) = \theta_K(t)\hat{\mathbf{K}}(t) = \theta_K(t) \begin{pmatrix} k_x \\ k_y \\ k_z \end{pmatrix} = h\overline{\Omega}(t) \tag{2.26}$$

where h is the sampling period of the system. The rotation matrix corresponding to $\mathbf{K}$ is given by (see [5])

$$\mathbf{R}_\Omega(t) = \begin{pmatrix} k_x^2 v_\theta + c_\theta & k_x k_y v_\theta - k z s_\theta & k_x k_z v_\theta + k_y s_\theta \\ k_x k_y v_\theta + k z s_\theta & k_y^2 v_\theta + c_\theta & k_y k_z v_\theta - k_x s_\theta \\ k_x k_z v_\theta - k_y s_\theta & k_y k_z v_\theta + k_x s_\theta & k_z^2 v_\theta + c_\theta \end{pmatrix} \tag{2.27}$$

where $s_\theta = \sin\theta_K$, $c_\theta = \cos\theta_K$ and $v_\theta = 1 - \cos\theta_K$.

Then the translation is calculated as

$$\mathbf{t}(t) = h\mathbf{T}(t) \tag{2.28}$$

and the total transformation matrix is given by

$$\mathbf{T}_t^{t+h}(t) = \begin{pmatrix} \mathbf{R}_\Omega(t) & \mathbf{t}(t) \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{2.29}$$

Then the current joint angles $\mathbf{j}(t)$ of the robot are measured, and the current pose $\mathbf{T}_b^g(t)$ is calculated with the help of the kinematics of the robot. The desired pose in time t+h is given by

$$\mathbf{T}_b^g(t+h) = \mathbf{T}_t^{t+h}(t)\mathbf{T}_b^g(t) \tag{2.30}$$

which is converted into the corresponding joint angles $\mathbf{j}(t+h)$ using the inverse kinematics of the robot. Then the correct position trajectory is generated by linear interpolation between $\mathbf{j}(t)$ and $\mathbf{j}(t+h)$, and the (constant) joint space velocity trajectory is generated as

$$\frac{d\mathbf{j}}{dt}(t) = \frac{\mathbf{j}(t+h) - \mathbf{j}(t)}{h}. \tag{2.31}$$

Image feature detection/extraction

The features on the end-effector that are used are four circular black dots on the front of the gripper, each with an approximate radius of 2 mm. The position of the dots can be measured accurately. First, the tracker, see section 2.3, predicts where the marks will be in the next frame, and the black pixel closest to this location is selected. A pixel at location $(x\ y)^T$

is considered to be black if its value in the image satisfies the condition $I(x, y) < I_t$, where I_t is some pre-determined threshold. Then, all black pixels 4-connected to this pixel can be found using a simple algorithm, and the exact location is estimated by calculating the mass center of all the n black pixels

$$\mathbf{x}_m = \frac{\sum_{i=1}^{n}(I(\mathbf{x}_i)\mathbf{x}_i)}{\sum_{i=1}^{n} I(\mathbf{x}_i)}$$

which gives a sub-pixel estimation of the location of the black dot.

Grasp position specification

Before we do visual servoing we must specify the end position $\mathbf{y}_r$, which is a vector of image-space coordinates. The coordinates show where the gripper feature points should be when the gripper is aligned with the object to be grasped. One method of acquiring these points is by direct measurement of the end position in a "teach by showing" approach. Another more flexible method is to calculate $\mathbf{y}_r$ from the camera projection equations, and the position of the object in the images.

Here the end position is calculated using models of the gripper and the object in the aligned position, described in an object centered basis. Let $\mathbf{X}_i^o$ and $\mathbf{x}_i^o = (x_{i,1}^o,\ x_{i,2}^o,\ 1)^T$ be the model and image space homogeneous coordinates of a point i on the object, and $\mathbf{X}_i^g$ the model points of the gripper. From the projection equations in appendix A we know that the object points are related by a projective transformation $\mathbf{H}$

$$\lambda \mathbf{x}_i^o = \mathbf{H}\mathbf{X}_i^o, \tag{2.32}$$

expressed in homogeneous coordinates. Here we use the fact that the models of the gripper and object are planar, so we can set $Z = 0$ in $\mathbf{X}_i^o$ and $\mathbf{X}_i^g$. We can transform equation 2.32 for each point into a linear system of equations in the unknown matrix elements H_{ij}

$$\begin{pmatrix} (\mathbf{X}_i^o)^T & \mathbf{0}_{1\times 3} & -x_{i,1}^o(\mathbf{X}_i^o)^T \\ \mathbf{0}_{1\times 3} & (\mathbf{X}_i^o)^T & -x_{i2}^o(\mathbf{X}_i^o)^T \end{pmatrix} \begin{pmatrix} \hat{\mathbf{h}}_1 \\ \hat{\mathbf{h}}_2 \\ \hat{\mathbf{h}}_3 \end{pmatrix} = 0 \tag{2.33}$$

where $\hat{\mathbf{h}}_i$ is row i of $\mathbf{H}$. If we know at least 4 points $\mathbf{X}_i^o$ and $\mathbf{x}_i^o$ we can stack the equations and solve the resulting system using SVD, which gives $\mathbf{H}$ up to a scale factor. We can then reproject the gripper model points $\mathbf{X}_i^g$ onto the image using equation 2.32, which gives us a value for $\mathbf{y}_r$.

Tracking image features

Searching for features in an image can be very time consuming, because of the large data volumes involved. In our images there are $320 \times 240 = 76800$ pixels, and we have two images to process every sample. If we were to process all of the images every sample, we would introduce delays which would affect the performance of the controller. The most common way to avoid this delay is to choose only a part of each image to process, based on the location of the interesting features in the previous images. The window-based tracking techniques can decrease the time required for feature extraction considerably, but require that we can predict the location of the features

in each image accurately enough to place the search window correctly. A short review of window-based tracking and feature extraction is given in [7].

Here we use a Kalman filter together with a simple model of the dynamics and the image Jacobian to track the features. Simple experiments show that the robot dynamics can be described as a first order system with a time-constant of around 100 ms, giving us the following discretized equation for the velocity of a point in the image

$$\dot{\mathbf{y}}_i(k+1) = a\dot{\mathbf{y}}_i(k) + (1-a)\mathbf{J}'_v(\mathbf{r})\dot{\mathbf{r}}(k) \tag{2.34}$$

where $\dot{\mathbf{y}}_i(k)$ is the velocity at sample k, $a = 0.5134$, $\mathbf{J}'_v(r)$ are the rows in the image Jacobian corresponding to the point, and $\dot{\mathbf{r}}$ is the velocity screw sent to the robot at sample k. The full system model for the movement of an image feature point in the x- or y-direction in the image is

$$\begin{aligned}
\mathbf{x}(k+1) &= \begin{pmatrix} 1 & h \\ 0 & a \end{pmatrix}\mathbf{x}(k) + \begin{pmatrix} 0 \\ 1-a \end{pmatrix}\mathbf{J}''_v(r)\dot{\mathbf{r}}(k) + \varepsilon_x(k) \stackrel{def}{=} \\
&\stackrel{def}{=} \mathbf{A}\mathbf{x}(k) + \mathbf{B}u(k) + \varepsilon_x(k) \\
y(k) &= \begin{pmatrix} 1 & 0 \end{pmatrix}\mathbf{x}(k) + \varepsilon_y(k) \stackrel{def}{=} \mathbf{C}\mathbf{x}(k) + \varepsilon_y(k)
\end{aligned} \tag{2.35}$$

where the state vector $\mathbf{x} = (u\ \dot{u})^T$ is the position and velocity of the feature point coordinate, $\mathbf{J}''_v(r)$ is the row in the Jacobian that corresponds to the feature coordinate and ε_x and ε_y are Gaussian white noise processes with zero mean and estimated covariance matrices

$$\begin{aligned}
E\left(\varepsilon_x(k)\varepsilon_x^T(k)\right) &= \mathbf{R}_1 = \begin{pmatrix} 0.5 & 0 \\ 0 & 10 \end{pmatrix} \\
E\left(\varepsilon_y(k)\varepsilon_y^T(k)\right) &= \mathbf{R}_2 = 0.5 \\
E\left(\varepsilon_x(k)\varepsilon_y^T(k)\right) &= \mathbf{R}_{12} = (0\ 0)^T
\end{aligned} \tag{2.36}$$

The feature can now be tracked using the equations for an ordinary Kalman filter, see [1]

$$\mathbf{K}(k) = \left(\mathbf{A}\mathbf{P}(k)\mathbf{C}^T + \mathbf{R}_{12}\right)\left(\mathbf{R}_2 + \mathbf{C}\mathbf{P}(k)\mathbf{C}^T\right)^{-1} \tag{2.37}$$

$$\hat{\mathbf{x}}(k+1) = \mathbf{A}\hat{x}(k) + \mathbf{B}u(k) + \mathbf{K}(k)\left(y(k) - \mathbf{C}\hat{\mathbf{x}}(k)\right) \tag{2.38}$$

$$\mathbf{P}(k+1) = \mathbf{A}\mathbf{P}(k)\mathbf{A}^T + \mathbf{R}_1 - \mathbf{K}(k)\left(\mathbf{C}\mathbf{P}(k)\mathbf{A}^T + \mathbf{R}_{12}^T\right) \tag{2.39}$$

The size of the search window is 15×15 pixels, chosen to be large enough to cover the tracked feature at every sample. The features are then extracted with sub-pixel precision using the method in section 2.3.

Handling of time delays

Error analysis

As we have already mentioned above, one of the most important advantages of image-based visual servoing is that the error will go to zero even if there are calibration errors. An error of zero in the image space will also generally mean that the objects are aligned exactly in cartesian space. However, in practice the specification of the end position is not exact. If we specify the end point using the "teach by showing" approach there will always be a small error in the estimated position of the features, and if we instead calculate the end point using the projection equations there will be additional errors in the calculations.

The question is what these image-space errors will translate to in cartesian space. Here we analyze a slightly simplified case, which will give a rough approximation of the errors in cartesian space for the 3DOF visual servoing.

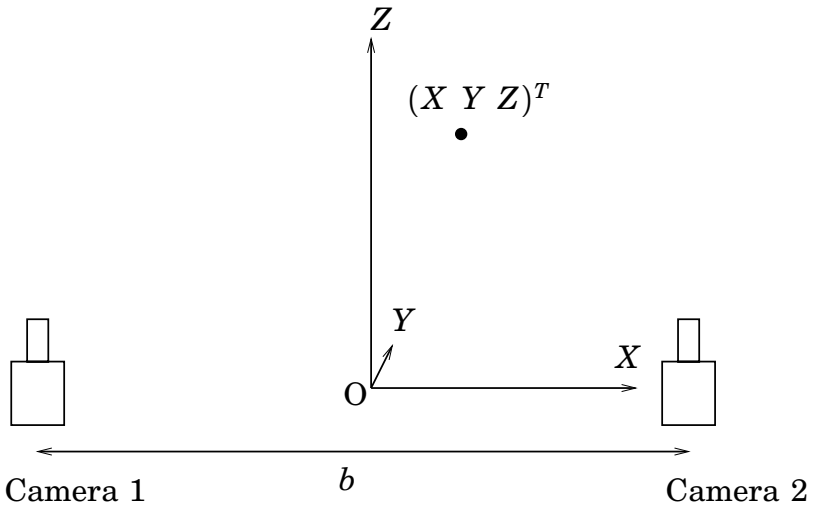

Figure 2.3 The simplified stereo rig.

We assume that we have two identical cameras, arranged as in Figure 2.3. The distance between the cameras is b, and the focal lengths are f. The cameras observe the desired end point $(X,\ Y,\ Z)^T$ measured in the basis O, which is located at the exact midpoint between the cameras. This point is projected in the two cameras as (see Appendix A)

$$\begin{pmatrix} u_1 \\ v_1 \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X + b/2 \\ Y \end{pmatrix} \tag{2.40}$$

$$\begin{pmatrix} u_2 \\ v_2 \end{pmatrix} = \frac{f}{Z} \begin{pmatrix} X - b/2 \\ Y \end{pmatrix} \tag{2.41}$$

in the error-free case.

We now assume that, because of measurement errors, the servoing will instead converge to the point $(X+\Delta X,\ Y+\Delta Y,\ Z+\Delta Z)^T$. The corresponding exact image points are now $(u_1 + \Delta u_1,\ v_1 + \Delta v_1)^T$ and $(u_2 + \Delta u_2,\ v_2 + \Delta v_2)^T$, where Δu_1, Δv_1, Δu_2 and Δv_2 can be interpreted as the measurement

errors in the image. The projection equations for $(X + \Delta X,\ Y + \Delta Y,\ Z + \Delta Z)^T$ can be written as

$$\begin{pmatrix} u_1 + \Delta u_1 \\ v_1 + \Delta v_1 \end{pmatrix} = \frac{f}{Z + \Delta Z} \begin{pmatrix} X + \Delta X + b/2 \\ Y + \Delta Y \end{pmatrix} \tag{2.42}$$

$$\begin{pmatrix} u_2 + \Delta u_2 \\ v_2 + \Delta v_2 \end{pmatrix} = \frac{f}{Z + \Delta Z} \begin{pmatrix} X + \Delta X - b/2 \\ Y + \Delta Y \end{pmatrix} \tag{2.43}$$

We now simplify the problem further by assuming that $\Delta v_2 = \Delta v_2$.[1] Equations 2.42 and 2.43 can then be rewritten as a linear system of 3 equations in the unknowns $\Delta X,\ \Delta Y$ and ΔZ.

$$\begin{pmatrix} -f & 0 & u_1 + \Delta u_1 \\ 0 & -f & v_1 + \Delta v_1 \\ -f & 0 & u_2 + \Delta u_2 \end{pmatrix} \begin{pmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{pmatrix} = \begin{pmatrix} (\frac{b}{2} + X)f - (u_1 + \Delta u_1)Z \\ Yf - (v_1 + \Delta v_1)Z \\ (-\frac{b}{2} + X)f - (u_2 + \Delta u_2)Z \end{pmatrix}$$

Straightforward calculations gives the solution

$$\begin{aligned} \Delta Z &\approx \frac{\Delta u_2 - \Delta u_1}{fb} Z^2 \\ \Delta X &\approx \frac{\Delta u_1 + \Delta u_2}{2f} Z \end{aligned} \tag{2.44}$$

We see that the errors in Z increases rapidly with the distance, and decreases when the baseline b is increased. Therefore, in order to achieve a good positioning accuracy we want to observe the scene from a short distance, using a stereo rig with a long baseline. This is not always possible, due to the constraint that the entire scene must be visible in both cameras at the same time.

We also see that ΔZ depends only on the difference between the errors in the two cameras, while ΔX is proportional to the sum of the errors.

2.4 Drawing on the whiteboard

The force control must control the force in the z-direction of the gripper to a constant value, see Figure 1.1, while the visual servo is moving the gripper and pen across the board. One way to approach this task is to divide the degrees of freedom of the problem into force- and vision controlled directions, which in our case would mean that the z-direction of the gripper would be controlled using only force control. However, then we would not use all the available information on the state of the system in the z-direction, and it would therefore be a good idea to use a combination of force and vision control to control this degree of freedom. Imagine for instance that the pen loses contact with the board, which means that the force in the z-direction $F_z = 0$. Then the only information on the system

[1] With this particular camera setup, this condition could be satisfied if we use the epipolar constraint $v_1 = v_2$ in the end point specification and feature extraction.

state from the force sensor is that $Z \leq Z_0$. From the vision system however, we can still get information on the position of the pen, and use this to control the pen back to the board.

A method that combines force control with the visual servoing described above is presented below. This method is based on the observation that the force constant of the spring mounted pen and the environment combined is relatively stiff, meaning that a small change in the position of the pen will lead to a large change in the contact force. Therefore it is appropriate to use the force measurements to adjust the trajectories from the visual servo so that the correct force is obtained. It also means that since the force control typically makes very small adjustments to the trajectories, these adjustments will not affect the results of the visual servo part of the algorithm.

Force control

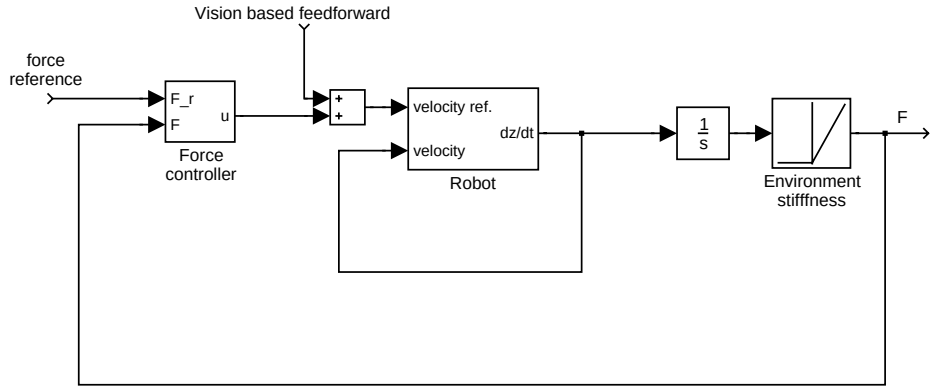

Figure 2.4 Force controller structure.

The block diagram for the closed loop system can be seen in Figure 2.4. The outer force loop provides the setpoint for the inner velocity control loop, which is the position/velocity control system already implemented in the robot. The resulting robot dynamics is approximated with a first order system

$$G_{rob} = \frac{1}{sT_{rob} + 1} \tag{2.45}$$

where T_{rob} can be estimated from a simple step response experiment.

The environment is approximated by an integrator and a spring with the nonlinear characteristics

$$F = \begin{cases} k_z(Z - Z_0) & Z > Z_0 \\ 0 & Z \leq Z_0 \end{cases} \tag{2.46}$$

where Z_0 is the z-position of the board surface.

When the pen is in contact with the system will be like a normal spring, otherwise the force is zero. In this case the control must quickly steer the pen back into contact with the board.

The spring constant k_z is estimated to approximately 400 N/m in a simple experiment. This means that a force of 1 N will correspond to a

change in z-position of just 2,5 mm. Compare this with the estimations of the visual positioning accuracy for the stereo rig in section 3.3.

The main problem with the force sensor is the very low signal-to-noise ratio. The forces used in the drawing on the board are quite small, usually 1–3 N is enough, which unfortunately means that the amplitude of the signal from the force sensor and the noise are of the same magnitude. Using the internal digital low-pass filters in the JR3 force sensor increases the SNR, but also introduces extra lag into the system.

The force control law is chosen as a simple proportional motion rate controller

$$\dot{Z} = K_F(F_r - F) \tag{2.47}$$

where K_F is a constant gain and F_r is the reference.

Force control with vision based feed-forward

The force control described will now control the movement in the z-direction of the gripper, leaving the other degrees of freedom for the visual servoing. However, the z-axis is not exactly orthogonal to the whiteboard plane. We can rewrite the plane equation 2.22 as $Z = f(X,Y) = p_1X + p_2Y + p_4$. If $p_1 \neq 0$ or $p_2 \neq 0$, movement in the x- and y-directions will affect also the z-position of the pen relative to the board, and unless the force control is fast, this will cause noticeable errors in the contact force.

This can however be compensated for, if we use the z-part of the vision based control signal as a feed-forward signal for the force control loop. The constrained velocity in the z-direction is calculated from equation 2.21 as

$$\dot{Z} = p_1\dot{X} + p_2\dot{Y} \tag{2.48}$$

where $\begin{pmatrix}\dot{X} & \dot{Y}\end{pmatrix}^T$ is the velocity screw obtained from equation 2.25, using the constrained Jacobian from equation 2.23.

By combining equations 2.23, 2.24, 2.25, 2.47 and 2.48, we see that the combined force/vision control law becomes

$$\dot{\mathbf{r}}_H = \begin{pmatrix} 0 \\ 0 \\ K_F(F_r - F) \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ p_1 & p_2 \end{pmatrix} \left(K_v \left[\mathbf{J}^3_{v,c}(\mathbf{r})\right]^{-1} (\mathbf{y}_r - \mathbf{y}) \right) \tag{2.49}$$

Estimating the location of the board

The constraint, described by equation 2.21, is in general unknown. [14] presents a method based on local estimation of the constraint using measurements of forces and torques. In our case the low signal-to-noise ratio of the force sensor, in combination with considerable friction effects, makes this method less suitable. Instead we estimate the parameter vector $\hat{p}$ using a recursive least-squares method and the equations

$$\begin{pmatrix} X_m & Y_m & \left(Z_m - \frac{F_z}{k_z}\right) & 1 \end{pmatrix} \hat{\mathbf{p}} = 0 \tag{2.50}$$

$$\begin{pmatrix} p_1 & p_2 & -1 \end{pmatrix} \left(\left[\mathbf{J}^3_v(\mathbf{r})\right]^{-1} (\mathbf{y}_r - \mathbf{y}) \right)^T = 0 \tag{2.51}$$

where X_m, Y_m and Z_m are measured cartesian coordinates for the end-effector obtained from the robot kinematics, and F_z and k_z are the measured force and the force constant in the z-direction. $\mathbf{J}^3_v(r)$ is the unconstrained 3DOF Jacobian from section 2.3. Equation (2.51) comes from the

fact that in an accurately calibrated stereo system, the unconstrained cartesian velocities $\dot{\mathbf{r}}$ obtained from the control law will produce trajectories that are straight lines. If the system is moving between two points that both lie in the constraint plane, then all the velocities $\dot{\mathbf{r}}(t)$ will be approximately parallel to the plane. Including this in the plane estimation, we will benefit from the "look-ahead" capabilities of the vision system, which can be expected to improve the convergence rate of the estimation. A drawback is that it makes the estimation more sensitive to calibration errors.

Note that the equation (2.50) require that the force constant 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. The value of k_z could also be estimated online.

3. Results

The experiments are performed in the robot lab at the Department of Automatic Control, Lund Institute of Technology.

3.1 Calibration

The calibration object used is a checkered pattern with 48 corner features, and the size of the object is approximately 10×15 cm. The resulting output from the calibration of the stereo system are the intrinsic camera parameter matrices $\mathbf{K}_1$ and $\mathbf{K}_2$, and the relative poses $\mathbf{T}_b^{c_1}$ and $\mathbf{T}_{c_2}^{c_1}$. As a by-product in the calculations we also get $\mathbf{T}_g^t$, the relative pose of the calibration object relative to the gripper frame. The matrices $\mathbf{T}_{c_2}^{c_1}$ and $\mathbf{T}_g^t$ can be checked against their measured values, to test if the results are reasonable.

The results from the calibration of the system used in the experiments are

$$\mathbf{K}_1 = \begin{pmatrix} 816.5573 & -1.2822 & 78.2626 \\ 0 & 854.2631 & 96.6541 \\ 0 & 0 & 1.0000 \end{pmatrix}$$

$$\mathbf{K}_2 = \begin{pmatrix} 814.9845 & 2.2944 & 175.0039 \\ 0 & 846.9191 & 139.9104 \\ 0 & 0 & 1.0000 \end{pmatrix}$$

$$\mathbf{T}_b^{c_1} = \begin{pmatrix} -0.9996 & 0.0089 & 0.0286 & 1.2825 \\ -0.0275 & 0.1107 & -0.9935 & 1.2439 \\ -0.0120 & -0.9938 & -0.1104 & 1.6403 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix}$$

$$\mathbf{T}_{c_2}^{c_1} = \begin{pmatrix} 0.9156 & 0.0240 & -0.4013 & 0.7192 \\ 0.0056 & 0.9974 & 0.0725 & 0.0128 \\ 0.4020 & -0.0686 & 0.9131 & 0.1944 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix}$$

$$\mathbf{T}_g^t = \begin{pmatrix} -0.7762 & 0.6303 & -0.0151 & 0.0747 \\ -0.0088 & 0.0130 & 0.9999 & -0.3087 \\ 0.6304 & 0.7763 & -0.0045 & -0.0245 \\ 0 & 0 & 0 & 1.0000 \end{pmatrix}$$

To test the calibration we compare these results with the results we get when we use only four of the images, and the results obtained from a calibration done with a well known camera calibration toolbox [3]. The results can be seen in tables 3.1–3.4.

Calibration	α	β	γ	u_0	v_0
Images 1–8	816.5573	854.2631	-1.2822	78.2626	96.6541
Images 1, 3, 5, 7	810.3757	851.1468	-1.2700	66.3426	99.1179
Images 2, 4, 6, 8	828.7196	865.8866	-2.7935	88.5513	88.9727
Toolbox	806.4±5.8	829.2±6.0	0	79.3±10.5	85.5±9.0

Table 3.1 Intrinsic parameters, camera 1

Calibration	α	β	γ	u_0	v_0
Images 1–8	814.9845	846.9191	2.2944	175.0039	139.9104
Images 1, 3, 5, 7	816.6615	848.8810	4.9501	150.6089	112.3556
Images 2, 4, 6, 8	811.9553	843.3814	-1.0145	171.2718	154.0182
Toolbox	804.4±5.8	829.0±7.0	0	123.4±10.9	87.1±9.9

Table 3.2 Intrinsic parameters, camera 2

Error analysis

We test the calibration by creating eight "virtual images" of the calibration object for each camera, by using the estimated transformations and intrinsic parameters from a normal calibration, and reprojecting them back on the images with the equations

$$\mathbf{x}^{c_1} = \frac{1}{\lambda_1}\mathbf{K}_1\mathbf{T}_b^{c_1}\mathbf{T}_g^b\mathbf{T}_t^g\mathbf{X}^t + \varepsilon_1 \tag{3.1}$$

$$\mathbf{x}^{c_2} = \frac{1}{\lambda_2}\mathbf{K}_2\mathbf{T}_{c_2}^{c_1}\mathbf{T}_b^{c_1}\mathbf{T}_g^b\mathbf{T}_t^g\mathbf{X}^t + \varepsilon_2 \tag{3.2}$$

where ε_1 and ε_2 is normally distibuted white noise in the pixel coordinates with mean 0 and standard deviation σ. The calibration is run 10 times for each of a number of different noise levels, and the mean absolute errors of the parameters are calculated. The corner detection algorithm used in the calibration is capable of calculating the position with a standard deviation σ of less than 0.1 pixels, of course depending on image quality.

Intrinsic parameters The errors in the estimated intrinsic parameters are plotted against σ, see Figures 3.1–3.2. We see that the only parameter estimation that is improved considerably in the final optimization step is the image skew γ.

Calibration	t_x/m	t_y/m	t_z/m	$\theta_x/^\circ$	$\theta_y/^\circ$	$\theta_z/^\circ$
Images 1–8	1.2825	1.2439	1.6403	-96.3400	0.6883	-178.4269
Images 1, 3, 5, 7	1.3130	1.2375	1.6167	-96.2044	-0.0548	-178.6448
Images 2, 4, 6, 8	1.2577	1.2616	1.6617	-95.9078	1.4203	-178.2152

Table 3.3 $\mathbf{T}_b^{c_1}$, translation and Euler angles

Calibration	t_x/m	t_y/m	t_z/m	$\theta_x/^\circ$	$\theta_y/^\circ$	$\theta_z/^\circ$
Images 1–8	0.7192	0.0128	0.1944	-4.2973	-23.7007	0.3522
Images 1, 3, 5, 7	0.7212	0.0117	0.1828	-2.2894	-24.5660	0.3090
Images 2, 4, 6, 8	0.7159	0.0162	0.2207	-5.8154	-24.7093	0.7018

Table 3.4 $\mathbf{T}_{c_2}^{c_1}$, translation and Euler angles

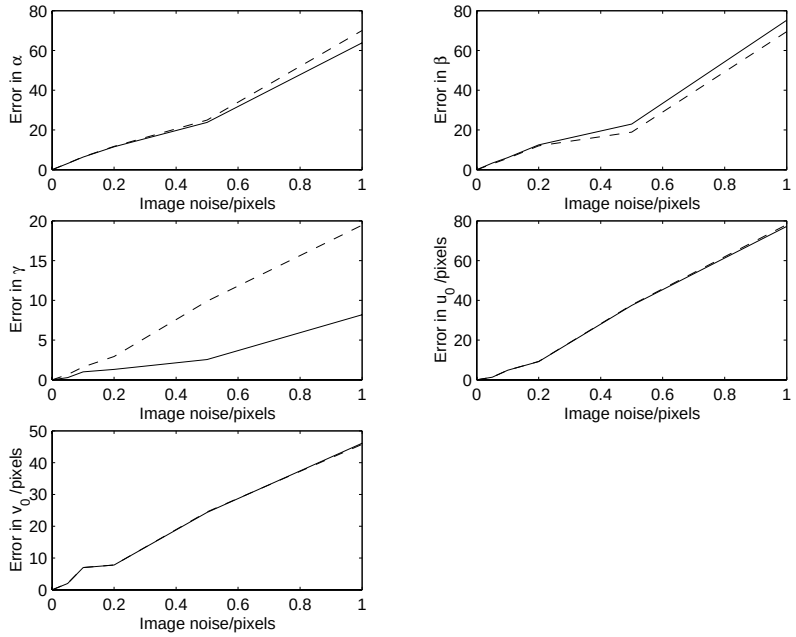

Figure 3.1 Errors in estimated intrinsic parameters for camera 1, before (dashed) and after (solid line) the final optimization step.

Extrinsic parameters Errors in the extrinsic parameters can be seen in Figures 3.3–3.5. Note the very large improvement in the final optimization step for the estimation of the hand-target pose $\mathbf{T}_g^t$. The estimations of the camera-camera and camera-robot transformations $\mathbf{T}_{c_2}^{c_1}$ and $\mathbf{T}_b^{c_1}$ are also improved considerably.

Other sources of error not included in the analysis are measurements errors in the model target points, systematic non-planarity of the calibration object, kinematic errors caused by errors in the calibration of the robot, and various distorsions in the camera. The errors due to the distorsions can be reduced by including a model of the radial distorsion in the estimation [15].

Robustness of the algorithm

The convergence of the final optimization depends on the accuracy of the initial values obtained in step 1 and 2 above. Especially the hand-target calibration in step 2 is sensitive to the experimental setup, especially the different end-effector poses. This shows that it is important to choose these poses in a suitable way, see [13]. In practice the convergence is not a problem as long as a number of poses with different orientations are used for the gripper.

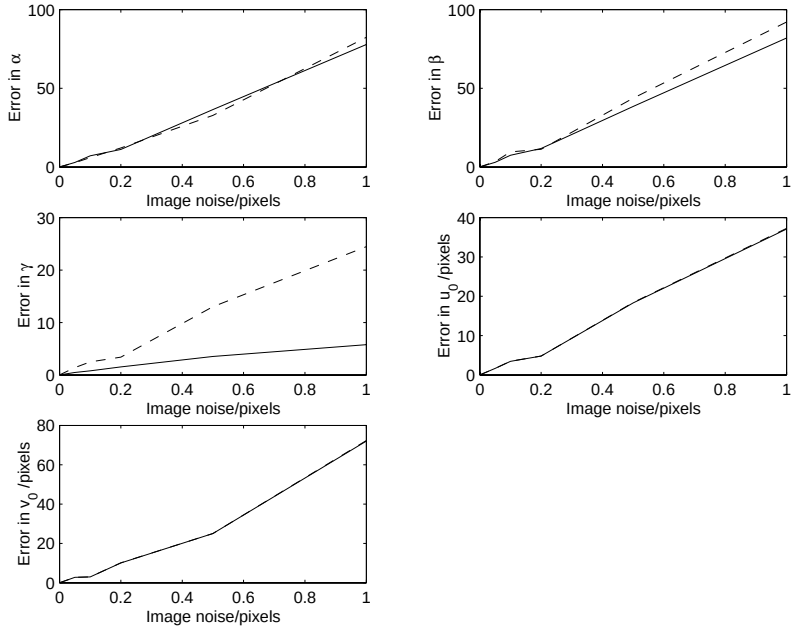

Figure 3.2 Errors in estimated intrinsic parameters for camera 2, before (dashed) and after (solid line) the final optimization step.

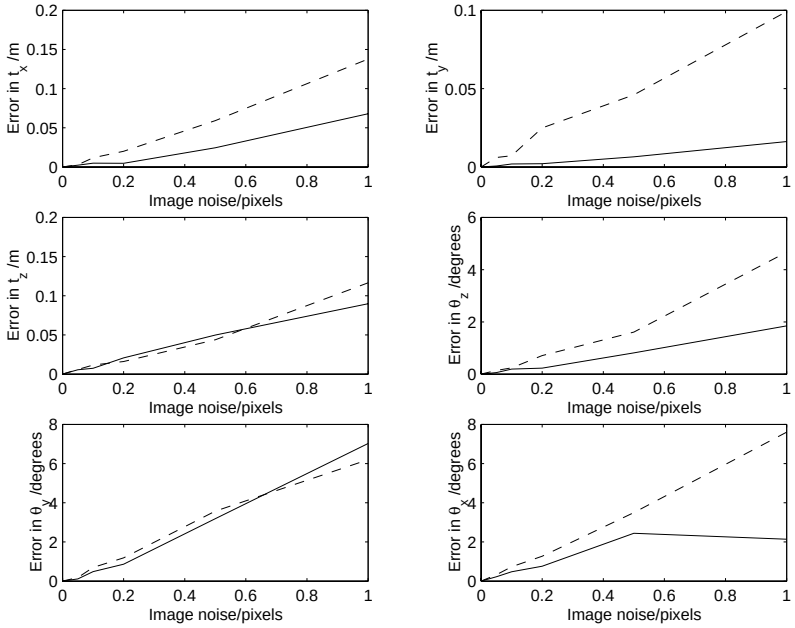

Figure 3.3 Errors in $\mathbf{T}_{c_2}^{c_1}$, translation $\mathbf{t}$ and Euler angles $\overline{\theta}$, before (dashed) and after (solid line) the final optimization step.

3.2 Picking up the pen

In Figures 3.6–3.7 we see the image-space trajectories of the features on the end-effector during the 4DOF visual servoing phase. The trajectories are not exactly straight lines in the image, which is due to the fact that the gripper is a rigid body, and the solution to equation 2.20 therefore gives

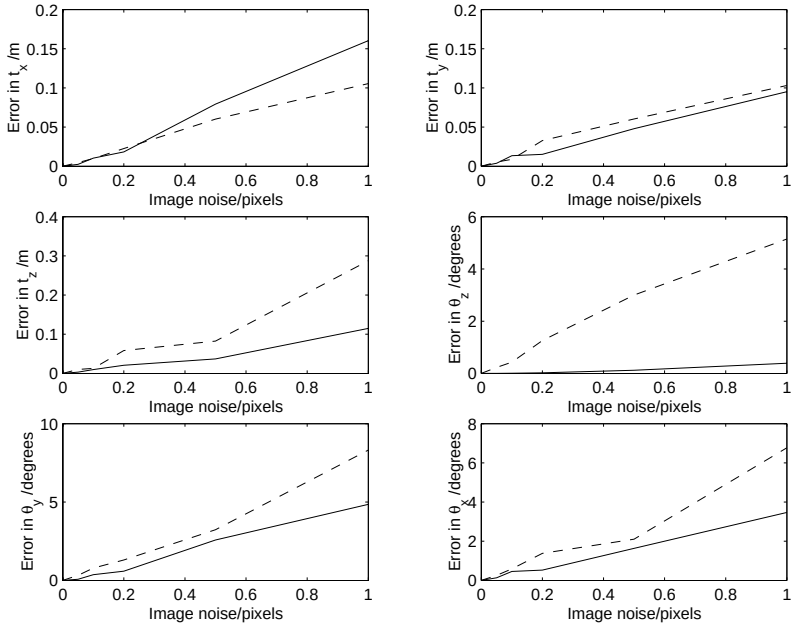

Figure 3.4 Errors in $\mathbf{T}_b^{c_1}$, translation $\mathbf{t}$ and Euler angles $\overline{\theta}$, before (dashed) and after (solid line) the final optimization step.

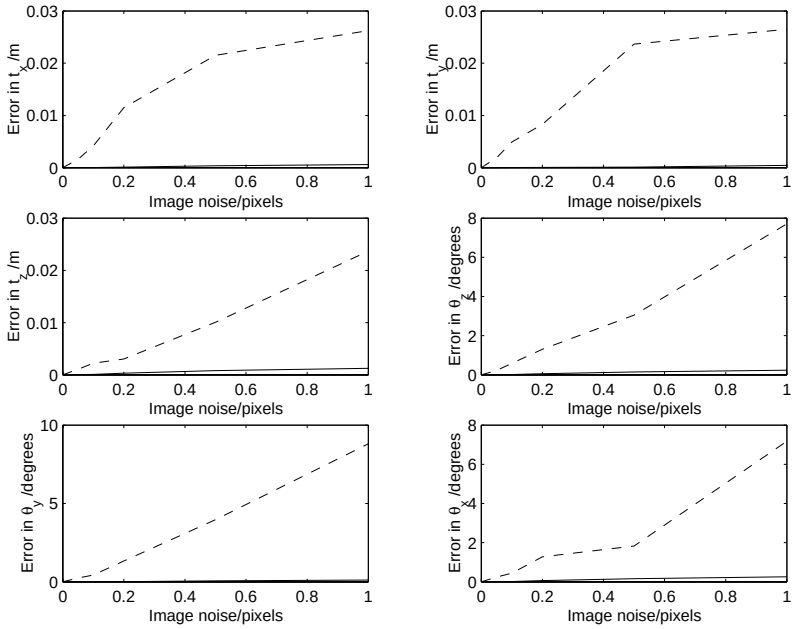

Figure 3.5 Errors in $\mathbf{T}_g^t$, translation $\mathbf{t}$ and Euler angles $\overline{\theta}$, before (dashed) and after (solid line) the final optimization step.

the best approximation to the straight-line trajectory, in the least-squares sense, that satisfies this geometric constraint. We see that the output of the system converges to a value close to the desired feature parameter vector. The mean absolute error between the desired and measured image coordinates at convergence is 1.7 pixels, mostly because of large errors in the specification of the end point. In the workspace the error in an object

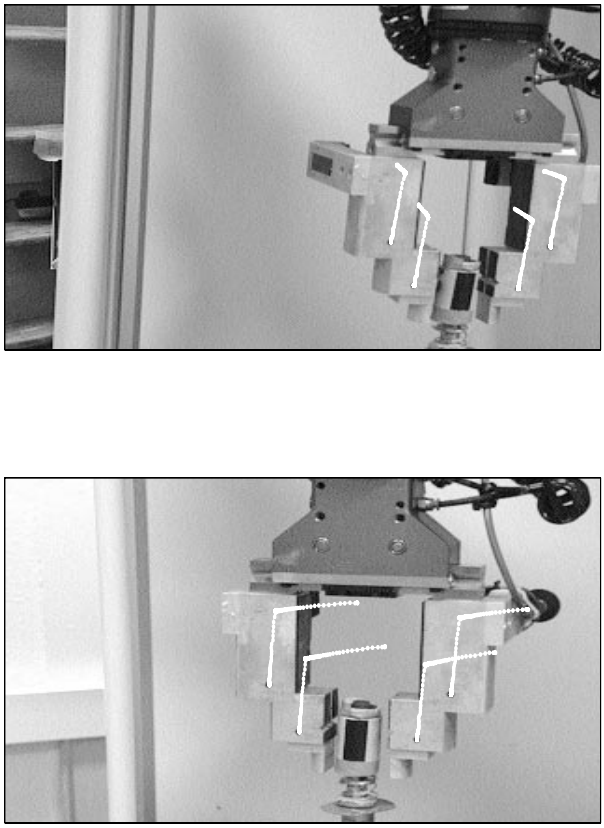

Figure 3.6 Image space trajectories, camera 1 and 2

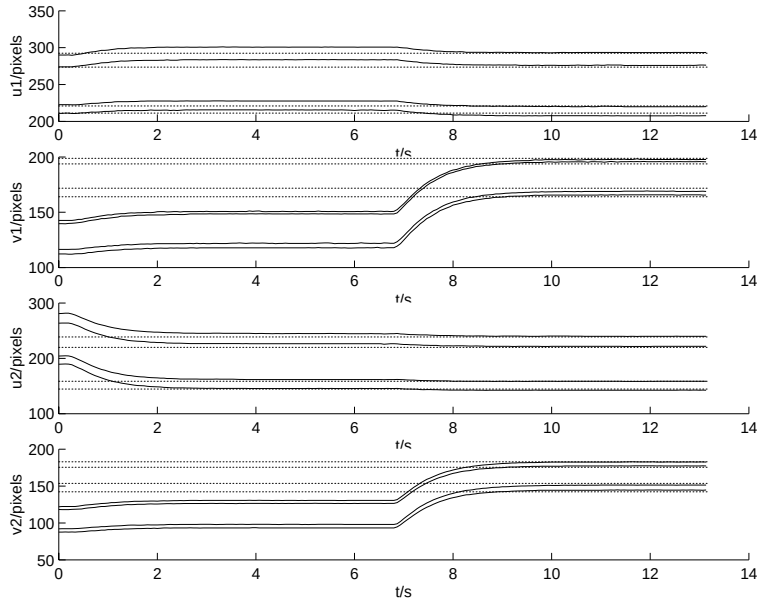

Figure 3.7 Image space feature positions and references, camera 1 and 2.

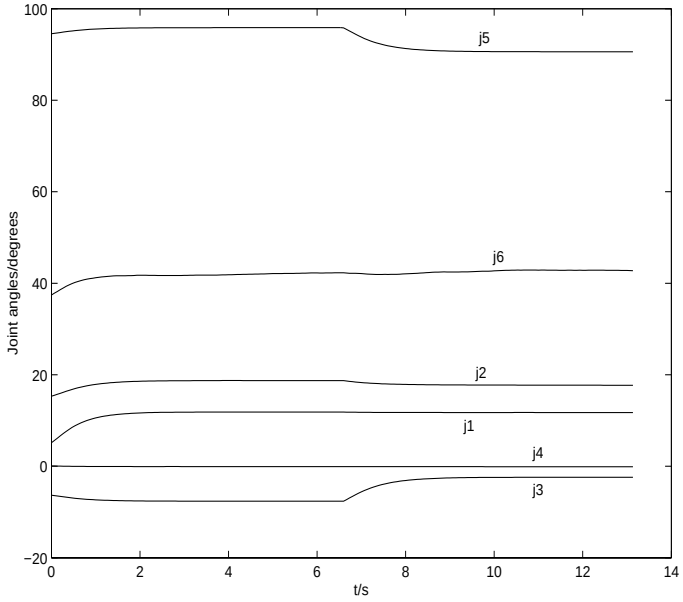

Figure 3.8 Joint space trajectories.

centered coordinate system is measured to be approximately

$$\begin{aligned} \Delta X &\approx 3\ mm \\ \Delta Y &\approx 1\ mm \\ \Delta Z &\approx 2\ mm \\ \Delta\theta &\approx 5^\circ \end{aligned} \tag{3.3}$$

We can see that the error ΔZ which corresponds roughly to the depth in the cameras is relatively small, thanks to the relatively large baseline of the system, see section 2.3. The error in the orientation $\Delta\theta$ is larger, because of image-space measurement errors in the grasp position specification.

The resulting joint space trajectory sent to the robot is plotted in Figure 3.8. The cartesian position and orientation of the gripper during servoing is plotted in Figure 3.9. Note the slightly irregular behaviour of the orientation angle θ.

3.3 Drawing on the board

Here only one feature point in each image is used in the feedback loop. In the experiments we use the internal low-pass filter of the force sensor, with a cutoff frequency of 125 Hz. We see from the Figures 3.10–3.12 that the trajectories are now almost straight lines in the image, since there is of course no rigidity constraint on the solution when we are using a single point. The velocity screw sent to the robot is plotted in Figure 3.13.

The contact force in the z-direction of the end-effector is plotted in Figure 3.14.

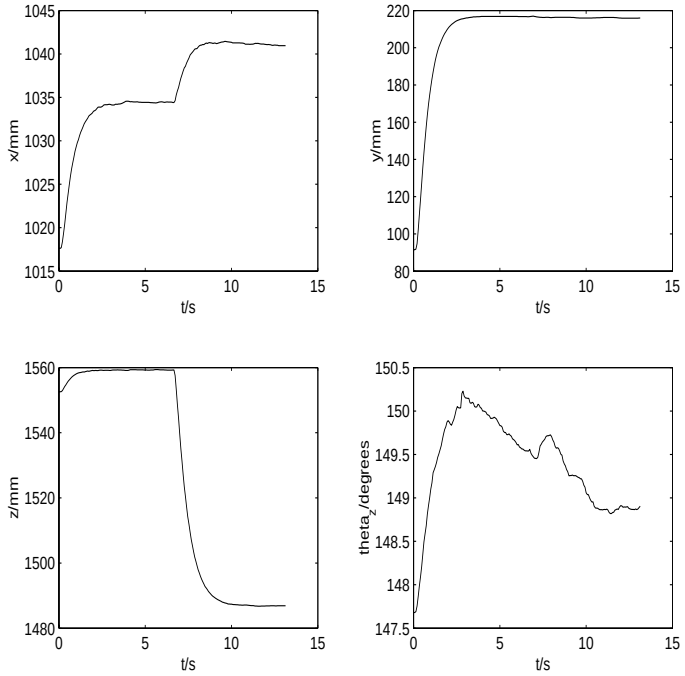

Figure 3.9 Cartesian position/orientation of the end-effector.

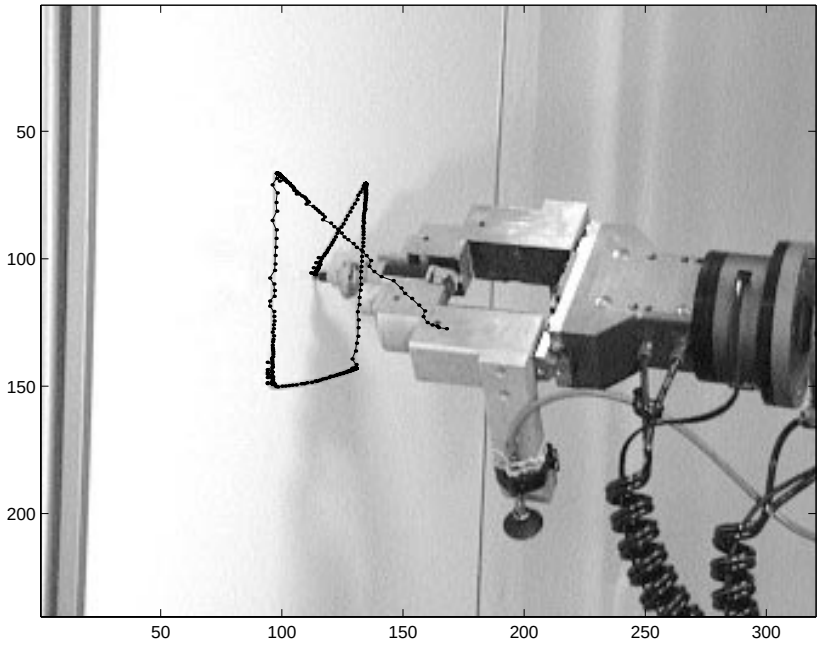

Figure 3.10 Pen trajectory, camera 1.

The stationary cartesian positioning error can be estimated using the results from section 2.3. In the camera system used in the experiments[1] we have $b \approx 0{,}7$ m, $(X,\ Y,\ Z)^T \approx (0,\ 0,\ 2{,}0)$ m, $f \approx 850$ and the estimated worst-case image error $\Delta u \approx 1{,}0$. This gives us the worst case estimations

[1]In the real stereo rig, the two cameras are tilted slightly inwards, corresponding to a rotation of about 12° around the y-axis of each camera. This is ignored in this analysis.

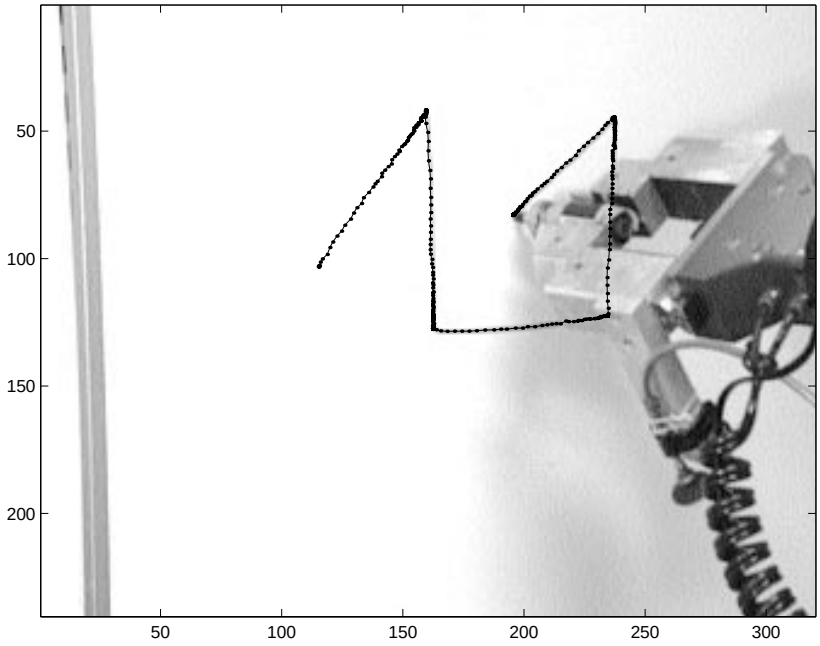

Figure 3.11 Pen trajectory, camera 2.

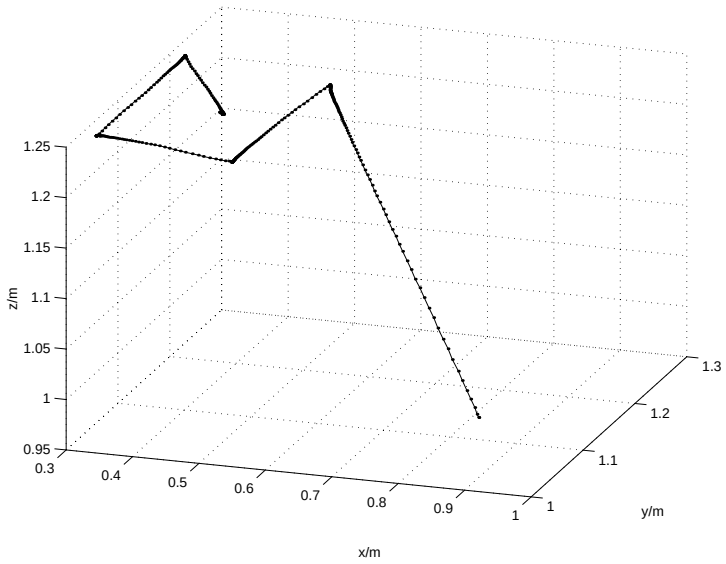

Figure 3.12 Pen trajectory, cartesian space.

$\Delta X = 3$ mm and $\Delta Z = 14$ mm.

3.4 Board equation estimation

The final estimation for the parameters of the plane is

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

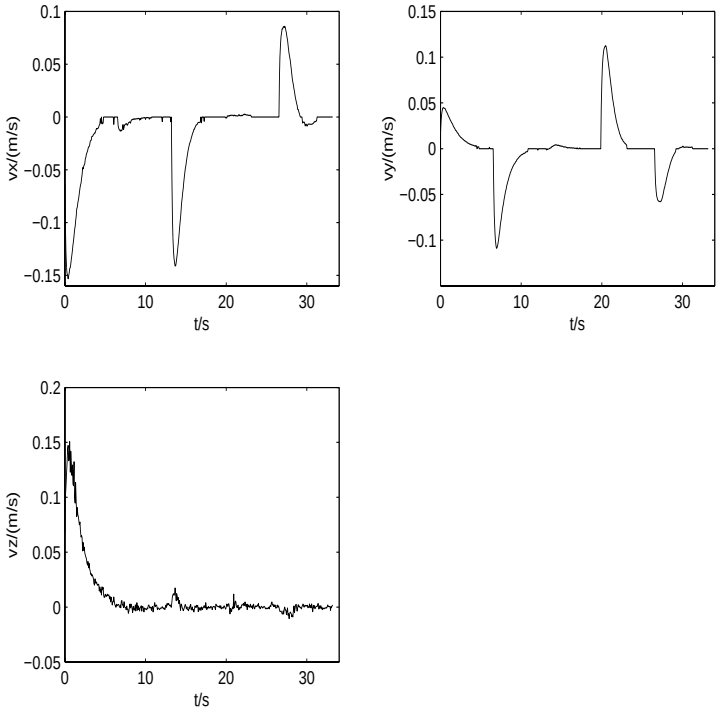

Figure 3.13 3DOF velocity screw.

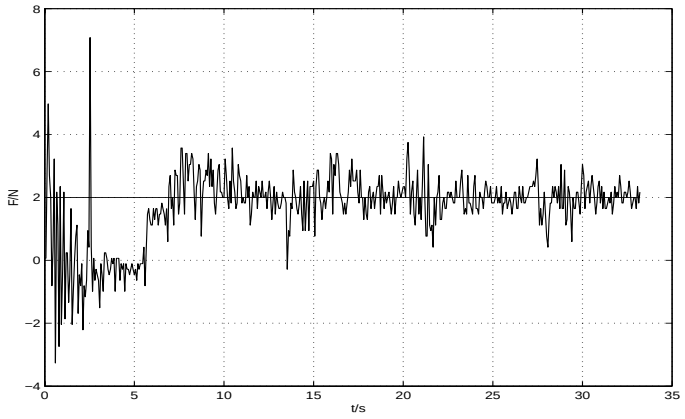

Figure 3.14 Measured force F and reference $F_r = 2$ N.

which should be compared to the correct values

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

obtained from an accurate measurement using the robot. The recursively estimated parameters can be seen in Figure 3.15. The estimation is started at $t = 6.7$ s.

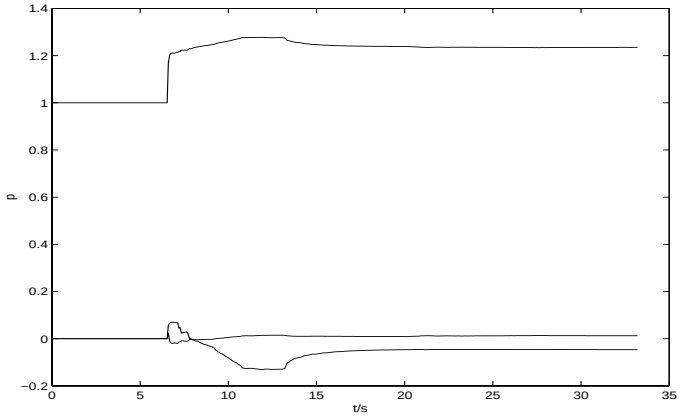

Figure 3.15 Plane parameters p_1, p_2 and p_4

4. Discussion

We see in Figures 3.1–3.5 that at our estimated noise level of $\sigma = 0.1$ pixels, the errors in the estimated intrinsic camera parameters are small, with a relative error in α and β of around 0.7%.

The relative error in the principal point is much larger, around 5%. For image-based visual servoing purposes, this error can still be considered to be very small. The error is due to the very small relative depth in the images. A small relative depth will make it difficult to distinguish between translations of the cameras in cartesian space, and translations of the principal point in the cameras, causing the calibration problem to become very ill-conditioned. The calibration object is small, 10×15 cm, and the images are taken from a distance of around 1.5–2 meters. Ideally, we would want to use a large calibration object, and take the images from a shorter distance and from many different angles. These requirements are however difficult to satisfy in practice, since we have the constraint that the entire calibration object must be visible in both images at once. One solution would be to use fixed values for the principal point, perhaps obtained from an accurate calibration of one camera at a time.

The errors in the estimations of the poses $\mathbf{T}_{c_2}^{c_1}$ and $\mathbf{T}_{b}^{c_1}$ are also very small, with absolute errors in the translation of less than 1 cm, and less than 1° in orientation. Again, this is much less than what is required for image-based visual servoing, where a coarse calibration is sufficient [7].

The simulated results of Figures 3.1–3.5 should be compared to the experimental results in Tables 3.1–3.4. Note that the experimental results are affected by not only image noise, but also errors in the model of the calibration object and the robot. The estimated errors in the model points is in our case over 1%, which explains the small difference in our values of α and β from to the reference values. We also see that the errors in the principal points are large, showing the effects of poor depth information described above.

During the writing phase, we see from Figure 3.14 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 equation $\hat{\mathbf{p}}$ has not yet converged, and the accuracy of the reference trajectories from the vision system is therefore limited. 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. Figure 3.14 also shows the stick-slip effect due to friction at $t \approx 16$ s. This effect is even more visible in the force plots when a dry pen with a larger friction coefficient is used.

Another source of error is the noise resulting from errors in the image feature extraction, most clearly seen in Figure 3.10. This will result in noise in the reference trajectories and the resulting contact forces, see Figure 3.14.

The estimated plane parameters change in steps, 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 reflects 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.

5. Conclusions and future work

5.1 Conclusions

The subject of this thesis has been the combinination of visual servoing methods with force control. The task chosen to illustate this involves force controlled, vision guided drawing on a planar surface. The method is based on an explicit estimation of the position of an the unknown planar constraint surface, and a constrained 3DOF visual servoing algorithm. The method differs from previous work in that it does not rely on assumptions of negligible friction forces, or the possibility to locally recover the normal of the plane from accurate measurements of contact forces and torques [14, 6, 11]. 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 the accuracy depends on the calibration.

The camera calibration method uses a number of images of a planar calibration object to estimate both the intrinsic camera parameters and the location of the camera frames relative to the robot. We use a nonlinear least squares method to estimate the parameters, and [15] and [13] to obtain initial values. The calibration object is attached to the robot end-effector, and therefore the extrinsic camera parameters are not completely unknown. This extra information is used in the algorithm, in order to increase accuracy and robustness. It is shown that with simulated data, using this information will improve the estimation of the cartesian poses considerably, while the estimations in the camera parameters are improved very little or not at all.

Despite very poor depth information in the image sequence used, the results from the calibration are far more accurate than what is required for visual servoing purposes.

The pen will be grasped using an unconstrained image based visual servoing method. The reference position is specified using the projection equations. Inexact measurements and calculations will cause errors in the specifications of almost 2 pixels. Despite this, the robot will grasp the object with an error of only a few millimeters, due to the relatively large baseline of the stereo system.

During the drawing phase, the 3DOF visual servoing method uses an explicit planar constraint on the possible velocity screws $\dot{\mathbf{r}}$ used to generate reference trajectories. The values of the constraint parameters are estimated with a recursive least squares algorithm. Combining the reference trajectories with a proportional force control law leads to the combined

control law

$$\dot{\mathbf{r}}_H = \begin{pmatrix} 0 \\ 0 \\ K_F(F_r - F) \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ p_1 & p_2 \end{pmatrix} \left(K_v \left[\mathbf{J}^3_{v,c}(\mathbf{r}) \right]^{-1} (\mathbf{y}_r - \mathbf{y}) \right) \quad (5.1)$$

The method is implemented in the robot lab at the Department of Automatic Control, Lund Institute of Technology. The system is run at a sampling frequency of 15 Hz, which could easily be increased to the the video rate of 30 Hz. The experiments show that the system will maintain a constant contact force during drawing, despite large friction forces and limited accuracy of the force/torque sensor. The slope of the constraint plane is estimated with an error of around 0.2°.

5.2 Future work

Currently, the calibration is performed off-line using an attached calibration object. One straightforward improvement would be to remove the calibration object, and instead use a model of the gripper itself. This way the calibration could be run on-line, using data obtained during servoing.

The constrained visual servoing could be modified in a very straightforward way to full 6DOF servoing. The orientation of the gripper would then be changed during the drawing, where the reference could be for instance the estimated normal vector of the planar constraint.

In some tasks, an interesting alternative to direct force control would be to use impedance control in combination with visual servoing, see [10].

Further, many improvements can be made to the image processing, so that more complex features can be used. The tracking can also be improved, which will increase robustness and allow faster control.

By including time-stamping and synchronization in the real-time system used for acquiring the images, we can compensate for the time delays and the asynchronous nature of the cameras. The real-time system can also be made faster, so that a higher sampling rate can be obtained.

A. Camera models

A.1 Perspective

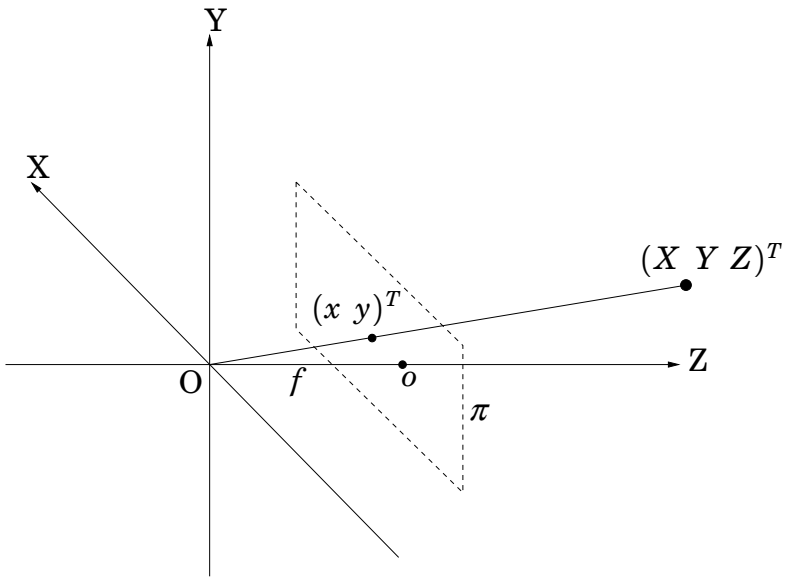

Figure A.1 The pinhole camera model.

The most common camera model is the pinhole or perspective camera, which is used in the calibration method used above, and it is therefore described briefly here. For more information about this and other camera models, see [12]. The perspective camera model consists of a point O, called the center of projection, and a plane π, the image plane. The origin of the camera centered coordinate system is in O, see Figure 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 called the principal point o. 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.

To transform between image-plane coordinates $(x\ y)^T$ to pixel-coordinates in the camera we need to introduce a number of extra so called 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 [12] for details. The new camera model becomes

$$\begin{aligned}\lambda\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} &= \begin{pmatrix} \alpha & \gamma & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \\ &= \begin{pmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \\ &\stackrel{def}{=} \mathbf{K}\mathbf{R}_{3\times 4}\mathbf{X} \end{aligned} \tag{A.1}$$

The matrix $\mathbf{K}$ is called the intrinsic camera matrix, and $\mathbf{R}_{3\times 4}$ is the extrinsic camera matrix. The intrinsic parameters α and β describes focal length and aspect ratio, gamma is the skew (usually close to 0), and $(u_0\ v_0)^T$ is the principal point, which is often located near the center of the image.

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

A.2 Weak perspective

An approximation to the perspective camera model that is useful if the depth of the scene ΔZ is small in comparison to the depth of the object points Z is the weak perspective camera model. This is described by the projection equations

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

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

where Z_m is the average depth of the points imaged in the camera.

B. Cartesian coordinate transformations

In robotics, most tasks are specified with respect to a specific coordinate system, or frame. In our case we have frames attached to for instance the cameras, the robot base, the end-effector, the calibration object and the object to be grasped. We will often need to relate the position and orientation of the frames/objects in the workspace to each other, or transform coordinates from one frame to another. If we want to change coordinate system from frame b to frame a we use a so called pose, denoted by $\mathbf{T}_b^a$, which describes the location of frame a relative to frame b. The pose $\mathbf{T}_b^a$ consists of a 3-vector $\mathbf{t}_b^a$ and a 3×3 orthonormal matrix $\mathbf{R}_b^a$, satisfiying

$$(\mathbf{R}_b^a)^T \mathbf{R}_b^a = I$$
$$det(\mathbf{R}_b^a) = 1$$

The vector $\mathbf{t}_b^a$ is the vector from the origin of a to the origin of b, expressed in the coordinate system of a, see Figure B.1. The column vectors in the matrix $\mathbf{R}_b^a$ are the unit x-, y- and z-vectors of the frame b, expressed in the coordinate system of a:

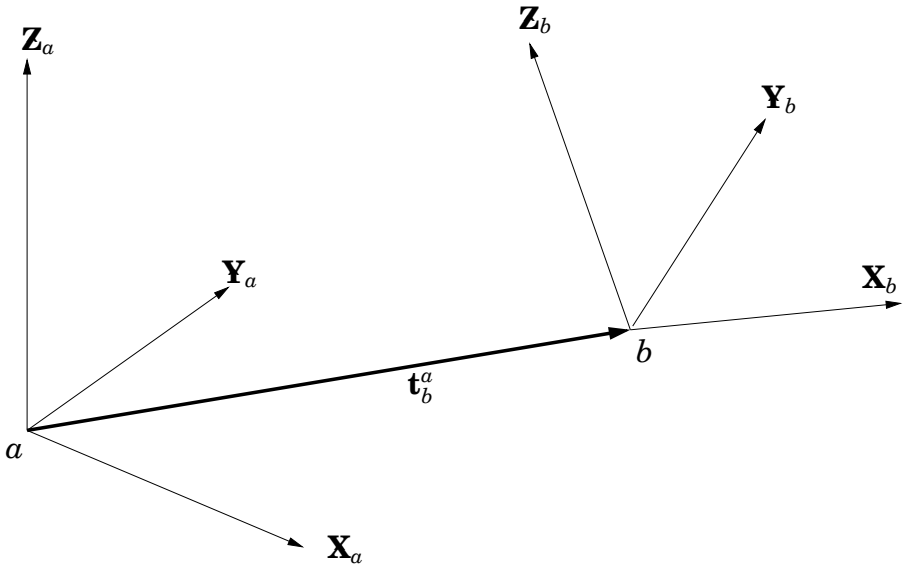

Figure B.1 Transformation between frames.

$$\mathbf{R}_b^a = \left((\hat{\mathbf{X}}_b)^a \ (\hat{\mathbf{Y}}_b)^a \ (\hat{\mathbf{Z}}_b)^a \right)$$

If we want to change coordinates from frame b to frame a we can now do so by applying the equation

$$\begin{pmatrix} X^a \\ Y^a \\ Z^a \end{pmatrix} = \mathbf{R}_b^a \begin{pmatrix} X^b \\ Y^b \\ Z^b \end{pmatrix} + \mathbf{t}_b^a \tag{B.1}$$

We now define the transformation matrix

$$\mathbf{T}_b^a = \begin{pmatrix} \mathbf{R}_b^a & \mathbf{t}_b^a \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}$$

By using homogeneous coordinates we can now write equation B.1 as

$$\begin{pmatrix} X^a \\ Y^a \\ Z^a \\ 1 \end{pmatrix} = \mathbf{T}_b^a \begin{pmatrix} X^b \\ Y^b \\ Z^b \\ 1 \end{pmatrix}$$

or simply

$$\mathbf{X}^a = \mathbf{T}_b^a \mathbf{X}^b$$

We can also combine several coordinate transformations in series

$$\mathbf{X}^a = \mathbf{T}_b^a \mathbf{X}^b = \mathbf{T}_b^a \mathbf{T}_c^b \mathbf{X}^c = \mathbf{T}_c^a \mathbf{X}^c$$

or invert the transform

$$\mathbf{X}^b = \mathbf{T}_a^b \mathbf{X}^a = (\mathbf{T}_b^a)^{-1} \mathbf{X}^a$$

Due to the special structure of the matrix $\mathbf{T}_b^a$ the inverse is easily computed

$$(\mathbf{T}_b^a)^{-1} = \begin{pmatrix} (\mathbf{R}_b^a)^T & -(\mathbf{R}_b^a)^T \mathbf{t}_b^a \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix}$$

C. Bibliography

[1] Karl Johan Åström and Björn Wittenmark. *Computer Controlled Systems—Theory and Design*. Prentice-Hall, Englewood Cliffs, New Jersey, second edition, 1990.

[2] J. Baeten, H. Bruyninckx, and J. De Schutter. Combining eye-in-hand visual servoing and force control in robotic tasks using the task frame. In *Proceedings IEEE International Conference on Multisensor Fusion*, pages 141–146, Taipei, Taiwan, R.O.C., August 1999.

[3] J-.Y Bouguet. Camera calibration toolbox for matlab.

[4] P.I. Corke. *Visual Control of Robots: high-performance visual servoing*. Research Studies Press Ltd., 1996.

[5] John J. Craig. *Introduction to robotics*. Addison-Wesley, 2nd edition, 1986.

[6] K. Hosoda, K. Igarashi, and M. Asada. Adaptive hybrid visual servoing/force control in unknown environment. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1097–1103, 1996.

[7] S. Hutchinson, G. Hager, and P. Corke. A tutorial on visual servo control. In *IEEE Transactions on Robotics and Automation*, volume 12, pages 651–670, October 1996.

[8] B. Lamiroy, B. Espiau, N. Andreff, and R. Horaud. Controlling robots with two cameras: How to do it properly. In *Proceedings IEEE International Conference on Robotics and Automation*, volume 3, pages 2100–2105, San Francisco, CA, April 2000.

[9] P. Martinet and E. Cervera. Stacking jacobians properly in stereo visual servoing. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 717–722, Seoul, Korea, May 2001.

[10] G. Morel, E. Malis, and S. Boudet. Impedance based combination of visual and force control. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1743–1748, Leuven, Belgium, May 1998.

[11] A. Pichler and M. Jägersand. Uncalibrated hybrid force-vision manipulation. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 1866–1871, 2000.

[12] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, New Jersey, 1998.

[13] R. Tsai and R. Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. In *IEEE Transactions on Robotics and Automation*, volume 5, pages 345–358. June 1989.

[14] D. Xiao, B. Ghosh, N. Xi, and Tarn T.J. Intelligent robotic manipulation with hybrid position/force control in an uncalibrated workspace. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1671–1676, Leuven, Belgium, May 1998.

[15] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 666–673, 1999.