ISSN 0280-5316
ISRN LUTFD2/TFRT--5690--SE

6-DOF Visual Servoing Using the Lie Group of Affine Transformations

Nikolaus Correll

Department of Automatic Control
Lund Institute of Technology
June 2002

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

Document name
MASTER THESIS

Date of issue
June 2002

Document Number
ISRN LUTFD2/TFRT--5690--SE

Author(s)
Nicolaus Correll

Supervisor
Rolf Johansson and Anders Robertsson (LTH)
Manfred Morari Automatic Control Lab. Zürich

Sponsoring organization

Title and subtitle
6-DOF Visual Servoing Using the Lie Group of Affine Transformations (Robotstyrning i sex frihetsgrader med användning av datorseende och Lie-gruppen av Affina transformationer)

Abstract
In this thesis a visual servoing approach for uncalibrated systems with 6 degrees of freedom (DOF) is evaluated. Assuming weak perspective, the observed motions of the robot end-effector are constrained to affine trans-formations in the image space. This allows us to express the control error directly in terms of an affine transformation in the camera images. By this approach, visual servoing is realized without previous knowledge of the camera parameters or the kinematic properties of the robot. Similarly to conventional image based visual servoing, but without the usual drawbacks, such as suboptimal trajectories in cartesian space.
The system was evaluated for different camera configurations, regarding stability and precision, using both a single camera and a stereo rig config-uration. Also, a motion primitive to define complex pick and place tasks is proposed.
Applications of such an approach are in an industrial environment where a flexible behavior of the robot is expected, e.g. handling slightly changing parts in a manufacturing line. Additional applications are in harsh envi-ronments where conventional sensors are not suitable for some reasons or applications where the kinematics of the robot are changing, resp. unknown, for example in the growing field of reconfigurable robots.
Being an image based visual servoing approach, our method is more robust with respect to maintenance and repair then position based visual servoing. The entity of sensor, actor and controller breaks up and every system can be regarded in particular. A replacement of the camera, the end-effector or even the whole kinematic setup does not involve modifications on the other parts of the system. Some practical considerations were examined using a virtual robot envi-ronment and validated using a ABB 6-DOF industrial robot and a stereo camera system.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
55

Recipient's notes

Security classification

6-DOF Visual Servoing Using the Lie Group of Affine Transformations

Nikolaus Correll

Supervision

Prof. Rolf Johansson*
Prof. Manfred Morari‡
Dr. Anders Robertsson*

Department of Automatic Control, Lund Institute of Technology*
Automatic Control Laboratory, Swiss Federal Institute of Technology‡

June 29, 2002

Contents

Acknowledgements

I would like to thank Prof. Rolf Johansson from Department of Automatic Control, Lund, and Prof. Manfred Morari, Automatic Control Laboratory, Zurich, for the great opportunity to work on this thesis and the trust they had into me.
Also, I thank my supervisors Anders Robertsson, Ph.D., Tomas Olsson, Johan Bengtsson and Mathias Haage from whom I learnt a lot.
Finally, I would like to thank all the people of the department who provided me with a friendly environment.
Being an exchange student at Lund University has been a great experience for me and I am very happy to have been at the Department of Automatic Control.

Nikolaus

Abstract

In this thesis a visual servoing approach for uncalibrated systems with 6 degrees of freedom (DOF) is evaluated. Assuming weak perspective, the observed motions of the robot end-effector are constrained to affine transformations in the image space. This allows us to express the control error directly in terms of an affine transformation in the camera images. By this approach, visual servoing is realized without previous knowledge of the camera parameters or the kinematic properties of the robot. Similarly to conventional image based visual servoing, but without the usual drawbacks, such as suboptimal trajectories in cartesian space.
The system was evaluated for different camera configurations, regarding stability and precision, using both a single camera and a stereo rig configuration. Also, a motion primitive to define complex pick and place tasks is proposed.
Applications of such an approach are in an industrial environment where a flexible behavior of the robot is expected, e.g. handling slightly changing parts in a manufacturing line. Additional applications are in harsh environments where conventional sensors are not suitable for some reasons or applications where the kinematics of the robot are changing, resp. unknown, for example in the growing field of reconfigurable robots.
Being an image based visual servoing approach, our method is more robust with respect to maintenance and repair then position based visual servoing. The entity of sensor, actor and controller breaks up and every system can be regarded in particular. A replacement of the camera, the end-effector or even the whole kinematic setup does not involve modifications on the other parts of the system.
Some practical considerations were examined using a virtual robot environment and validated using a ABB 6-DOF industrial robot and a stereo camera system.

Chapter 1

Introduction

Automation is desired in environments which are unhealthy or for other reasons unsuitable for human beings. Furthermore, automation can provide immense time savings in certain applications, especially in system control where the complexity of the plant is far beyond human capabilities [CRW98]. In both cases, automation increases the gain of the application, either due to increased productivity or due to savings in personnel, not to mention the prevention of injuries which are highly undesired and expensive. Thus, automation is highly demanded in an industrial environment, and has already become very popular in manufacturing, as well as in domains which cope with repetitive setups. In such cases, the environment is usually contrived to suit the robot. This is not always feasible and makes automation difficult, for example outdoors or in changing setups as at home e.g.
The latter cases demand for more complex approaches, usually involving a higher number of sensors and a feedback control scheme to be more robust to changes in the task setup. Expanding the perceptional capabilities of the robot by vision is thereby an intuitive approach, not only due to the human exemplar but also due to big advantages in hardware in the last decades. Using vision in a feedback control scheme has become known as *visual servoing* and many efforts have been made in this field, see for example [ECR92], [HC94] or [HHC96] for a more comprehensive review.

1.1 Outline of the report

After a brief introduction to conventional visual servoing and our approach in the remainder of this chapter, chapter 3 begins with a description of the robot and camera system and the virtual environment used for simulation. After that, the principles of affine and Euclidian transformations are presented, giving also a brief introduction to Lie groups and the Lie algebra. After a description of the used control scheme, a brief outline to the used vi-

sion algorithms is given. Finally, the results with respect to different camera positions and control strategies are presented and discussed.

1.2 Visual Servoing

State-of-the-Art visual servoing can be usually divided into two groups which will be briefly described below. These are usually referred to as position based visual servoing (PBVS) and image based visual servoing (IBVS) [HHC96].

1.2.1 Position based visual servoing

In position based visual servoing, the control error needed for feedback is generally expressed in Cartesian space, mirroring industrial needs where the coordinates of the goal and the robot end-effector are usually known in cartesian coordinates. This usually requires a 3D-reconstruction of the world based on (stereo) camera images. Besides that the exact transformation between the camera and the end-effector has to be known, a linear approximation to the highly non-linear camera parameters has to be found, known as the *Camera Calibration* problem, see also [Ols01].
After determination of the Cartesian error, a control signal has to be generated in actuator space, requiring prior knowledge about the inverse kinematic of the robot. The inverse kinematic of the robot is the transformation from Euclidian 3D space to the actuator setup of the robot. This transformation is in most cases highly non-linear and often due to singularities not unique solvable [Cra89].
Finally PBVS approaches inherently depend on having an accurate 3D model of the target object [HHC96] what is not always available and makes PBVS ineligible to handle with slightly changing parts, e.g. in a manufacturing environment.

1.2.2 Image based visual servoing

In image based visual servoing, the control error in the feedback loop is expressed directly in image coordinates. The non-linear relation between the error in the image and the control signal for the robot is hereby approximated by a linear transformation, the so called *image Jacobian*, see section 3.4. An error could be for examples image features taken from two different camera positions, usually provided by a stereo rig, to introduce information about the three-dimensional setup of cameras and feature point.
The Jacobian can be generated either analytically, assuming a projective camera projection, see A, or by doing linear-independent test movements in actuator space [Jae97]. [Jae97] suggests also an adaptive update of the Jacobian, based on the performed motions of the robot.

A drawback of IBVS are the presence of singularities in the Jacobian, leading to possible unstable control [HHC96]. Another problem is that the performed trajectories are often not well behaved in Cartesian space. An extreme version of this problem has become known as the *Chaumette Conundrum*, and is depicted in Figure 1.1: The desired pose corresponds to a pure rotation about the optical axis. As IBVS chooses the shortest path in image space, which is a straight line, the end-effector, respectively the camera, performs an infinite retreat, corresponding to a singularity in the Jacobian [CM00].

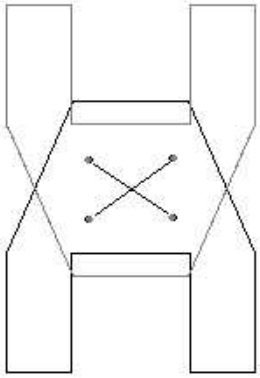


Figure 1.1: Robot end-effector with four feature points and the desired pose which is rotated by 180^o, illustrating the *Chaumette Conundrum*: As IBVS chooses the shortest path in image space, a straight line, the end-effector performs an infinite retreat.

1.2.3 Hybrid approaches

The pros and cons with PBVS and IBVS has lead to a number of hybrid approaches. For example, in [HC00] the z-axis rotational and translational components become decoupled from the remaining degrees of freedom, providing a solution to the *Chaumette Conundrum*. Chaumette [CM00] himself uses methods of 3D reconstruction, with the usual drawbacks as pointed out in section 1.2.1.

1.2.4 Visual servoing using affine transformations

In this project, we constrain the perceived motions of the end-effector to the Lie group (section 3.1.5) of affine transformations, consisting of translation, rotation, dilation, squash, shear and stretch (3.1.3). The group of affine transformations describes the transformations that the image of a planar object which moves in 3D-space can undergo when viewed under weak

perspective (3.1.4) from a camera. An important feature of this approach is, that the representation of 3D-motions by affine transformations in a 2D-single image embeds knowledge of the 3D-world configuration. The control error in the feedback loop is hereby the difference between a desired affine transformation of the robot and the performed transformation with respect to an initial position of the end-effector (section 3.3).
Feedback control of a 5-DOF robot using this approach was realized in [DC00]. The approach has none of the drawbacks of PBVS as it is not necessary to have any prior knowledge of the camera configuration. Also, the approach is not dependent on the kinematics of the robot, which makes it interesting for robots with changing kinematics [YZD02].
The *Chaumette Conundrum* as an extreme formulation for the drawbacks in IBVS does not play a role either, as the rotation is singled out in the affine mode of rotation.

1.3 Problem formulation

The goal of this project is to implement visual servoing using the Lie Group of affine transformations on a 6-DOF industrial robot. Thereby, it is assumed, that prior knowledge of the kinematics of the robot and the camera parameters are not available.
This involves computer vision as well as control problems: Feature points on the end-effector have to be detected at frame rate and an appropriate controller has to be found.
The involved systems, the robot kinematics and the camera projection, are highly non-linear and the influence of this nonlinearities have to be examined.
Finally, motion primitives to run simple tasks, such as pick and place applications, have to be defined and implemented. All above mentioned steps have first to be implemented in a virtual enviroment and then applied to the "real" system.

Chapter 2

System description

For experiments, a 6-DOF industrial robot and a stereo camera system are available. Robot, controller and vision system form a distributed system, connected by conventional 100BaseT-ethernet. The full system is depicted in Figure 2.1 and is presented in the following sections.

2.1 The ABB IRB-2000

The ABB (Asea Brown Boveri) IRB 2000 is an industrial robot. The robot is built up by two large arms and a wrist (Fig. 2.2), having 7 links connected by 6 joints. Due to its 6 joints, the end-effector is able to reach any desired position and orientation within its task space [Cra89].
The robot has a built-in controller for each of its joints for position control. These controllers are cascaded PID[1] Controllers with an outer position loop around an inner velocity loop (Fig. 2.3). 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 a local network. It is also possible to send a position signal and the duration of the desired motion. In this case, the user has no influence on the resulting trajectory. This is usually not desired but simplifies experiments where the dynamics of the robot are not of interest.

2.2 Camera System

The camera system consists of two Sony DFW-V300 digital cameras. The cameras are operating with a frame rate of 30 images per second in 24Bit-color mode. The user has full control over external parameters such as

[1] Proportional-Integral-Derivative

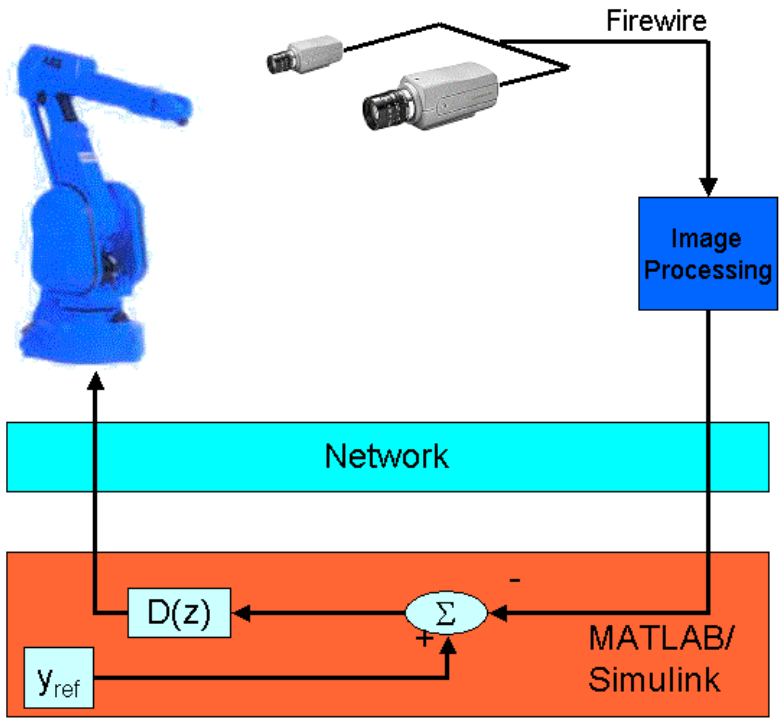


Figure 2.1: The IRB 2000 is observed by two Sony IFW-V300 Cameras which are connected using a Firewire connection to the image processing workstation. The Matlab/Simulink controller polls feature points of the robot end-effector and generates the new control signal for the robot. Robot, image processing unit and controller are connected via a 100MBit TCP/IP network connection.

orientation and pose in space and intrinsic parameters such as opening time, shutter and gain.
The images are read into a dedicated computer using a FireWire (IEEE-1394) connection. The frame rate of 30Hz is independent of the sampling rate of the robot controller, as they form a distributed system.

2.3 Vision System

The image processing is a multi-threaded application and runs on a workstation at frame rate. After capturing the images from the cameras, another thread extracts the feature points of the end-effector in both images. The former two steps are repeated continuously. A third thread performs a blocking communication with the robot controller: The thread waits until the controller requests new data and sends the latest feature points avail-

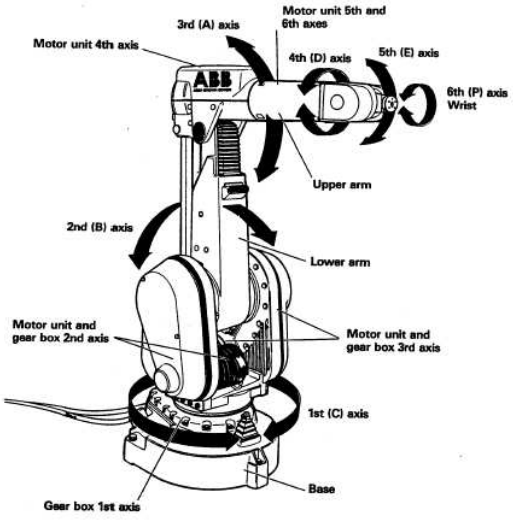


Figure 2.2: The six degrees of freedom of the ABB IRB 2000

able. This ensures that the robot controller does not get old data and can operate on a different sampling rate then the cameras.
If necessary, the resulting time lag due to the network transfer can be measured and compensated [Bou01].

2.4 Virtual environment description

As experiments on the real robot are time consuming and dangerous - not to mention that the robot usually is the "bottle-neck" device, resp. experiments are very expensive in a productive environment - a virtual environment was developed at the Department of Automatic Control, Lund Institute of Tech-

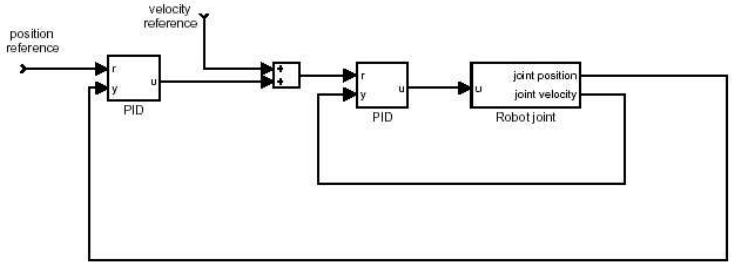


Figure 2.3: The internal joint controllers for the six joints of the ABB IRB 2000 [Ols01]

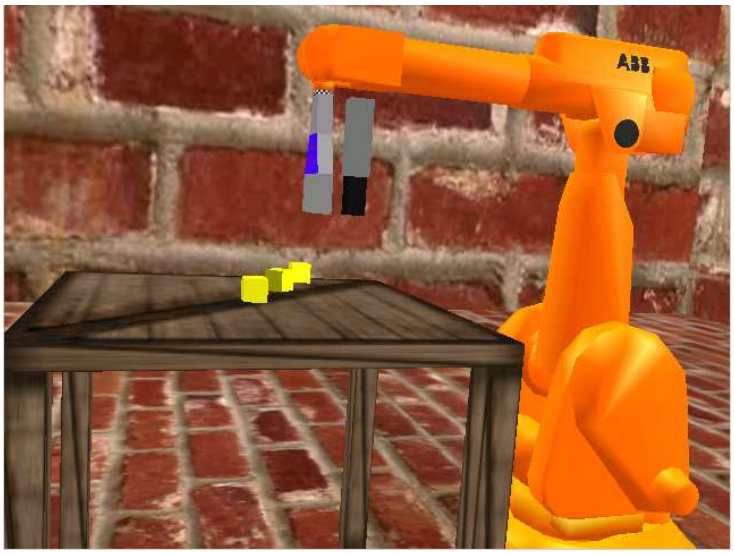


Figure 2.4: The "Virtual Robot". Here, the IRB 2000 has been set up for interaction with small dices lying on a table. The scene is observed by three cameras, one attached to the end-effector and two cameras in a stereo configuration in front of the robot (not shown in the Figure)

nology. The virtual environment can be controlled from Matlab/Simulink similarly to the real robot, by providing joint angle references. The system allows arbitrarily placement of an IRB-2000 and any three dimensional object with different textures (Fig. 2.4) in a 3D OpenGL[2] environment. The objects can be defined using an Extended Markup Language (XML) description. Furthermore, it is possible to place any number of cameras into the 3D environment (Fig. 2.5), whereby the intrinsic parameters of the camera, e.g. focal length, can be defined by the user. The whole system is depicted in Figure 2.6.

Providing this functionality, the Virtual Robot allows fast development of high-level control algorithms using vision. Thereby vision algorithms, robot controller, network communication and simulation can be run on the same machine or distributed.
Up to now, there are no dynamic effects implemented, the robot moves instantaneously to the desired position and new images can be obtained. To implement the robot dynamics, a system identification has to be made and applied to the environment. This has been already done in a previous version of the virtual robot based on Java3D[3] and was examined in [dM02].

[2]OpenGL is a cross-platform standard for 3D rendering and 3D hardware acceleration
[3]Java3D is a platform-independent 3D graphics application programming interface for

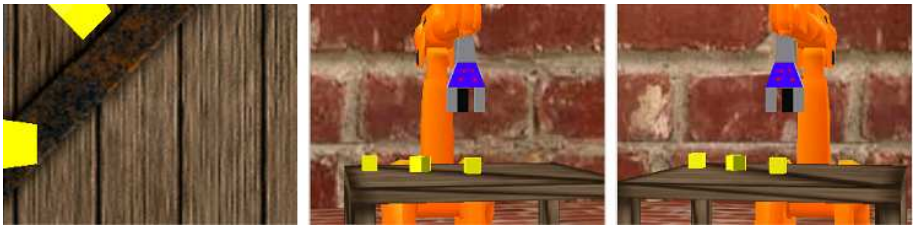


Figure 2.5: To the left, one can see the image produced by the camera mounted on the robot end-effector. To the right are the images taken from the stereo camera setup. The images are 320×240 pixels in 24Bit color. Compare with Figure 2.1.

Java^{TM}

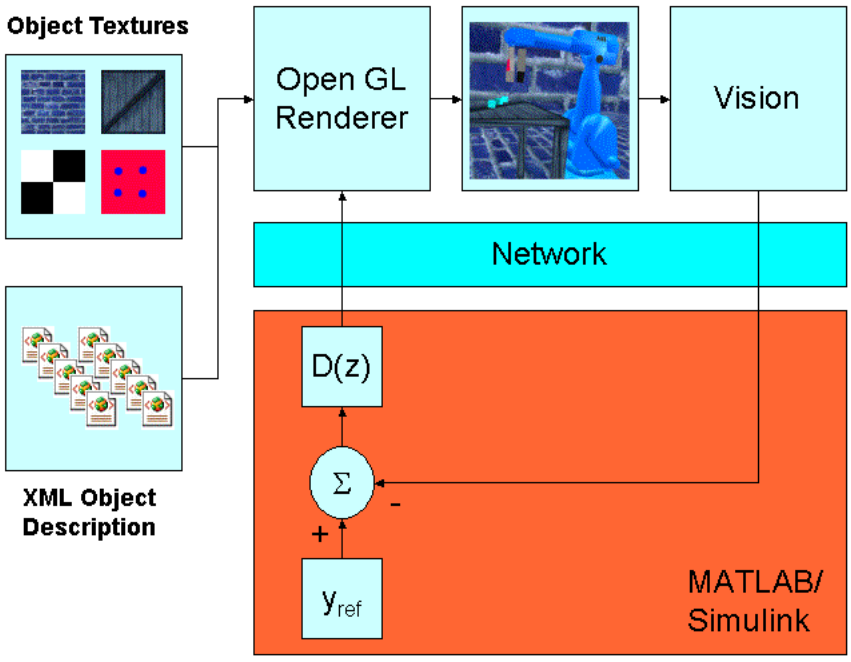


Figure 2.6: The core of the Virtual Robot is an OpenGL renderer engine. Provided with an XML description of the scene and joint angles from Matlab/Simulink over the network, images of the virtual environment are rendered and processed. If desired, results from the image processing can be used in a feedback loop with the controller.

Chapter 3

Methods

3.1 Mapping from task space to image space

To control the robot motion using cameras, we have to know how changes in the actuator space of the robot, e.g. the joint positions, affect the images taken from the camera(s). This problem is usually referred to as the *visual kinematics* of the robot [CG96].

3.1.1 Euclidean transformations in three dimensions

Regarding a rigid body in cartesian[1] space, any possible transformation is a combination of translation and rotation. In a three dimensional space we have translations along the axis X, Y and Z and rotations around them. Altogether, the object has 6 degrees of freedom (DOF). These modes of transformation on $\mathbb{R}^3$ together with the identity element, the transformation which maps each point to itself, build the $SE(3)$ group, the group of Euclidean[2] transformations. The $SE(3)$ group fully describes the possible transformations of a rigid body in 3D-space.

3.1.2 Projective transformations in two dimensions

Observing this transformations with a camera, the observed object becomes projected onto the image plane in $\mathbb{R}^2$. The group which describes all possible projective transformations of a rigid body in two dimensional space has 8 dimensions and is known as the $P(2)$ group. The number of dimensions can be explained as follows: Transformations on $\mathbb{R}^2$ can be described by a 3×3 matrix in homogeneous coordinates. Thereby, the third value of a point in $\mathbb{R}^2$ is always scaled to 1. This implies a loss of one degree of freedom, yielding $8 = 9 - 1$ modes of transformation.

[1] After *René Descartes*, French philosopher, 1596^* in France, $1650^\dagger$ in Sweden

[2] After *Euclid of Alexandria*, Egyptian mathematician, 325^* b.c- $265^\dagger$ b.c.

3.1.3 General affine transformations in two dimensions

If the observed object is planar and viewed under weak perspective (section 3.1.4), the projective transformations in the camera images can be completely described by the $GA(2)$ group, a 6-dimensional subgroup of the $P(2)$ group [DC00]. $GA(2)$ is the group of all affine transformations on two-dimensional space and can be broken down as follows:

1. x axis translation
2. y axis translation
3. Rotation about the origin
4. Dilation about the origin
5. Shear (squash y, stretch x)
6. Shear at 45^o to (5)

These transformations can be parameterized by α and represented by matrices in homogeneous coordinates, yielding pure transformations in each of the above mentioned six modes of deformation:

$$
\begin{aligned}
M_1 &= \begin{pmatrix} 1 & 0 & \alpha \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} & M_2 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & \alpha \\ 0 & 0 & 1 \end{pmatrix} \\
M_3 &= \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} & M_4 &= \begin{pmatrix} e^{\alpha} & 0 & 0 \\ 0 & e^{\alpha} & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
M_5 &= \begin{pmatrix} e^{\alpha} & 0 & 0 \\ 0 & e^{-\alpha} & 0 \\ 0 & 0 & 1 \end{pmatrix} & M_6 &= \begin{pmatrix} \cosh\alpha & \sinh\alpha & 0 \\ \sinh\alpha & \cosh\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}
\end{aligned}
\tag{3.1}
$$

3.1.4 Assumption of weak perspective

Considering a *pinhole* camera model, the perspective projection of a point $P = (X, Y, Z)^T$ in world coordinates to a point $p = (x, y, f)$ in camera coordinates, where f is the focal length of the camera, can be expressed as follows (see also Fig. 3.1):

$$
\begin{aligned}
x &= f\tfrac{X}{Z} \\
y &= f\tfrac{Y}{Z}
\end{aligned}
\tag{3.2}
$$

Equations (3.2) are non-linear due to the factor $1/Z$ and do not preserve distances between points or angles between lines [TV98]. To turn them into linear equations a classical approximation is the *weak perspective* camera model. This model requires, that distances between scene points are much smaller then the average distance of the scene points from the camera center.

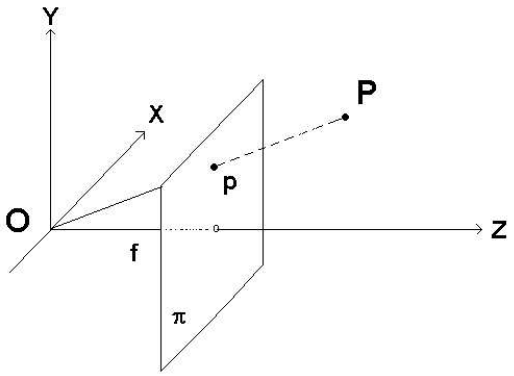


Figure 3.1: The perspective (pin-hole) camera model

This assumption is reasonable where small objects are viewed from two or three meters distance [HC94]. Equations (3.2) can then be approximated by

$$\begin{aligned} x &= f\frac{X}{Z} \approx \frac{f}{\hat{Z}}X \\ y &= f\frac{Y}{Z} \approx \frac{f}{\hat{Z}}Y \end{aligned} \tag{3.3}$$

where $\hat{Z}$ is the average distance of the scene points from the viewing camera. Note that the weak perspective camera model is a *scaled orthographic projection* and should not be confused with the *affine projection* camera model as the scaled orthographic projection is a special case of affine projection [HHC96].

3.1.5 Lie groups and Lie algebras

A Lie[3] group is a differentiable manifold obeying the group properties which also satisfies the additional condition that the group operations are continuous. Group properties are the existence of an identity element and an inverse element given an associative multiplication. A Lie algebra is a logarithm of a Lie group and thus a Lie group is an exponential of a Lie algebra. A more precise definition of Lie groups and algebras may be found in [BD85].
The matrices in Equation (3.1) are describing continuous one-dimensional transformations on $\mathbb{R}^2$, parameterized by α. Note that setting α to zero yields the identity element. Differentiating with respect to α and evaluating at $\alpha = 0$ yields generators G_i of a Lie Group and form a basis for a Lie

[3]After *Marius Sophus Lie*, Norwegian Mathematician, 1842*-1899†

algebra. For $GA(2)$ the generators are:

$$G_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad G_2 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$
$$G_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad G_4 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{3.4}$$
$$G_5 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad G_6 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

whereby

$$\widetilde{L}_i = \frac{dM_i(\alpha)}{d\alpha}\bigg|_{\alpha=0} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.5}$$

is a vector field and G_i is defined as

$$G_i = \frac{dM_i(\alpha)}{d\alpha}\bigg|_{\alpha=0} \tag{3.6}$$

Considering a Lie group as an exponential of a Lie algebra, an affine transformation matrix A can be obtained from a vector $\mathcal{A}$ by the exponential map:

$$A = e^{\sum_i \mathcal{A}_i G_i} \tag{3.7}$$

where

$$e^X = I + X + \frac{1}{2}X^2 + \frac{1}{6}X^3 + \dots$$

is the Taylor expansion of e^X and $i \in [1 \dots 6]$.

3.2 Determining the control signal

As was shown in section 3.1.3, the desired end-effector position in Euclidian space is directly related to an affine transformation in image space. Knowing a goal in Euclidian space, a straightforward approach is to move the robot to this position and measure the affine transformation it performs in image space. This simple proceeding is used throughout our experiments but is possibly not feasible in an industrial context. In this case, we have to teach once how end-effector and object "looks" in the desired position (*Teach by showing*).

Then we have to identify the goal object, by marking it manually in one image or detecting it with some higher level image processing, and determine the affine transformations for both the gripper and the goal object to the taught position, $\vec{\mathcal{A}}_{dice}$ and $\vec{\mathcal{A}}_{gripper}$. The desired transformation is then given by

$$\vec{\mathcal{A}}_{des} = \vec{\mathcal{A}}_{dice} \oplus \vec{\mathcal{A}}_{gripper}, \tag{3.8}$$

where

$$\vec{\mathcal{A}} \oplus \vec{\mathcal{B}} = e^{\sum_i \mathcal{A}_i G_i} e^{\sum_i \mathcal{B}_i G_i}, \tag{3.9}$$

considering that a Lie group is an exponential of a Lie algebra.

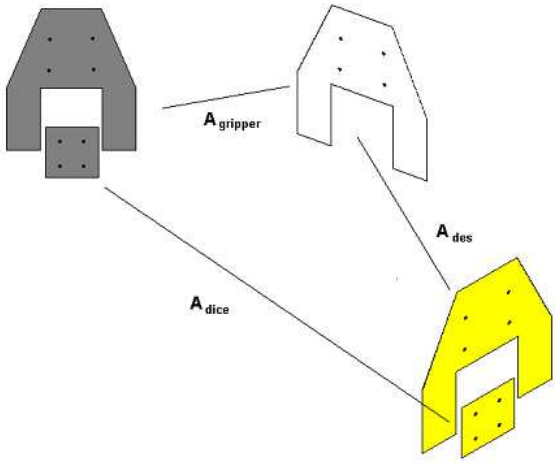


Figure 3.2: Identifying the dice and the gripper with higher level vision, we can get the affine transformations to a once taught "standard" position (dark-gray). The desired affine transformation is the "sum" of these both transformations.

3.3 Determining the control error

The control error in the visual servoing approach realized in this thesis is the difference between the desired and the measured affine transformation with respect to an initial position. For measuring the affine transformation of a planar object between two subsequent images, one need at least 3 unique feature points located on the object. The matrix A of the affine transformation can than be found by solving the following equation system:

$$A \begin{pmatrix} x_{1j} \\ y_{1j} \\ 1 \end{pmatrix} = \begin{pmatrix} x_{2j} \\ y_{2j} \\ 1 \end{pmatrix} \tag{3.10}$$

with

$$A = \begin{pmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

Here, x_{1j} denotes the jth featurepoint in the initial image and x_{2i} in the actual image, y respectively.
Using (3.7) yields:

$$\sum_i \mathcal{A}_i G_i = \log A \tag{3.12}$$

Looking at G_1 and G_2 in eq. (3.4) allows us to pick values for $\mathcal{A}_1$ and $\mathcal{A}_2$ directly from $\log A$. The other values of $\mathcal{A}$ can be found by solving four equations with four unknowns, as the last row of $\log A$ is always 0 (compare with equation (3.11)).
Knowing the desired transformation the end-effector has to perform in the image, the measured $\mathcal{A}$ can serve as control error.

3.4 Calibration of the Jacobian

After determination of the control error we need to find a control signal in robot actuator space. This is usually done by considering a function $f(d\vec{x})$ which maps positions $\vec{x}$ in robot-control space to values in sensor space $\vec{s}$. This can be expressed as

$$\vec{s} = f(\vec{x}), \tag{3.13}$$

where $f(\vec{x})$ is a (usually) nonlinear function and therefore hard to estimate, especially if the the sensor setup is unknown. Assuming that all partial derivatives for $f(\vec{x})$ exist, the relation in (3.13) can be approximated by a first-order Taylor expansion, yielding:

$$\begin{aligned} \vec{s}(\vec{x}) &\approx \vec{s}(\vec{x}_0) + J(\vec{x} - \vec{x_o}) \\ \Rightarrow \underbrace{\vec{s}(\vec{x}) - \vec{s}(\vec{x}_0)}_{d\vec{s}} &= J\underbrace{(\vec{x} - \vec{x}_0)}_{d\vec{x}} \end{aligned}$$

$$\Rightarrow d\vec{s} = \mathbf{J} \cdot d\vec{x}, \tag{3.14}$$

where $\mathbf{J}$ is the Jacobian[4] Matrix of $f(\vec{x})$. The Jacobian is a matrix build up by the derivative of each input to each output and thus is an indicator for how much an output, a sensor value, is affected by a certain input, a motion of the robot.
In IBVS $\mathbf{J}$ has become known as *image Jacobian* (section 1.2.2) and is an approximation of the non-linear camera calibration and the robot kinematics. An analytical approach to the image Jacobian is given in A.
Here, the Jacobian relates motions of the robot, given by a velocity in jointspace $d\vec{j}$, to the perceived motion in image space, given by their affine transformation $d\mathcal{A}$. Thus,

$$d\mathcal{A} = \mathbf{J}d\vec{j}. \tag{3.15}$$

3.4.1 One Camera

Using only one camera, $d\vec{x}$ and $d\vec{s}$ are vectors in $\mathbb{R}^6$ according to the 6 degrees of freedom of the robot and the 6 degrees of freedom in affine space (see section 3.1.3). The Jacobian indicates in this case how much a change in joint space affects a changes in affine space.
If the transformation from camera space to robot space is unknown, in this case by using uncalibrated cameras in unknown positions, the Jacobian can only be estimated by performing a series of test motions, yielding the following matrix equation:

$$\begin{pmatrix} d\vec{\mathcal{A}}_1 & \dots & d\vec{\mathcal{A}}_n \end{pmatrix} = \mathbf{J}_{est} \cdot \begin{pmatrix} d\vec{x}_1 & \dots & d\vec{x}_n \end{pmatrix}, \tag{3.16}$$

where n is the number of test movements and $d\vec{x}$ and $d\vec{\mathcal{A}}$ are column vectors in joint space, respectively affine space. To get a unique solution to (3.16), it is obvious that both matrices must have rank 6. Therefore, it is a necessary condition that there is a minimum of 6 test movements to perform, where every joint has to be excited at least once. On the right hand side of (3.16), this is guaranteed by choosing $d\vec{x} = ke_i$ where e_i is a basis vector of $\mathbb{R}^6$ with $i \in \{1 \dots 6\}$ and $k \neq 0$ a weight. In this configuration, each joint will be excited separately and the resulting

[4] After *Carl Gustav Jacobi*, German mathematician, $1804^* - 1851^\dagger$

affine transformation can be measured in image space.
Unfortunately, this proceeding does not guarantee a well conditioned matrix on the left hand side of (3.16). Bad conditioment occurs then, when motions in joint space generate only little changes in the image. Then, the measurement noise caused by the computer vision may be greater than the actual transformation, resulting in an almost random entry in the Jacobian for this particular joint to affine mode relation. Some of those cases are more obvious than others. For example one may consider a situation where the image plane is parallel to the end-effector and the end-effector moves towards the camera. In affine space this causes only to a change in scaling but, due to the bad depth resolution of the camera image this change will be relatively small. In the worst case, the scale mode of the affine transformation becomes equally weighted as transformations in other modes, due to image noise. This leads to an ill-conditioned Jacobian and unstable control. A similar problem was already pointed out in [DC00] and referred to as *extrinsic degeneracy*. Hereby, the rank of the Jacobian drops from 6 to 4.
To get rid of the measurement noise, an approach is to change the weight k for each movement to emphasize some joints more then others, and though the change in image space. In this case, a prior knowledge of the camera setup is necessary. Changing k dynamically with respect to a maximum change in affine space may be feasible in certain configurations but can not assure equal excitation in every affine mode with finite test movements.
Finally, if the range between the minimum and the maximum weight k becomes to big, the Jacobian becomes ill-conditioned as well, as changing k does not influence the matter that, due the camera setup, the Jacobian predicts wrong relation between certain combinations of joint movements and affine transformations.

3.4.2 Multiple Cameras

A work-around to the problems mentioned in the former section is to use more than one camera. If the cameras are well placed (see section 5.2), movements in joint space will lead to k linear independent transformations in affine space, where k denotes the number of cameras used. Formula (3.16) will then expand to

$$\begin{pmatrix} d\vec{\mathcal{A}}_1, 1 & \dots & d\vec{\mathcal{A}}_n, 1 \\ \vdots & \ddots & \vdots \\ d\vec{\mathcal{A}}_1, k & \dots & d\vec{\mathcal{A}}_n, k \end{pmatrix} = \begin{pmatrix} \mathbf{J}_{est_1} \\ \vdots \\ \mathbf{J}_{est_k} \end{pmatrix} \cdot \begin{pmatrix} d\vec{x}_1 & \dots & d\vec{x}_n \end{pmatrix}. \tag{3.17}$$

For a setup with two cameras ($k = 2$) and 6 test movements ($n = 6$), the matrix of affine transformations will be a (12×6) matrix, the Jacobian a (12×6) and the matrix of changes in joint space a (6×6) matrix. Now eq. (3.17) is over-constrained, promising a well-conditioned Jacobian. As can be seen in section (5.2) one has to be cautious: If the cameras are badly placed, the affine transformations A_i, k for the different cameras may become linear dependent in all modes which yields to the same problems as in section (3.4.1), as the second camera becomes obsolete. Last but not least, the rank of the matrix of stacked affine transformations will be reduced by one because one mode of deformation will always be linear dependent. This phenomenon can be observed as well in "classical" visual servoing where the Jacobian maps directly to translations in the image space. Assuming a stereoscopic setup, a featurepoint will perform a 4-DOF translation in image space, whereby

one translation is linear dependent on one of the other three (see [HHC96]). This can be expressed via the *epipolar constraint* [TV98].

3.4.3 From image space to joint space - the inverse Jacobian

In the previous section we showed how to relate motion of the robot end-effector to perceived motion in the images. However, as we want to control the robot joints using camera information we need a rule to map from image to joint space. In the simple case where $\mathbf{J}$ is quadratic and non-singular (section 3.4.1) this is given by

$$d\vec{x} = \mathbf{J}^{-1} d\vec{\mathcal{A}} \tag{3.18}$$

Using a multiple camera setup to estimate the Jacobian (section 3.4.2) yields an over-determined Jacobian. Then, the (pseudo)-inverse is given by [HHC96]

$$\mathbf{J}^{+} = (\mathbf{J}^{T}\mathbf{J})^{-1}\mathbf{J}^{T} \tag{3.19}$$

and equation (3.18) becomes

$$d\vec{x} = \mathbf{J}^{+} d\vec{\mathcal{A}}. \tag{3.20}$$

3.5 Controller

The robot is controlled using a PID-controller with additional feed-forward as suggested in [CG96]. Furthermore, the controller gain P is enhanced by a diagonal gain matrix $\mathbf{G}$.
The control scheme is depicted in Fig. 3.3. Latencies in the system, depending on network transfer and image processing, are depicted with latency blocks in the control scheme.

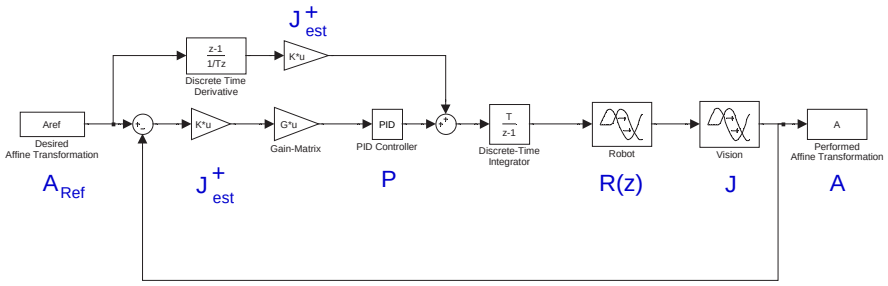


Figure 3.3: The full control scheme with additional feed-forward path. Latencies in the control path are depicted with latency blocks.

Neglecting the latencies and reducing the PID controller to a proportional controller, yields the continuous time "transfer function" (MIMO[5] system) for the

[5]Multiple-Input-Multiple-Output

feedback loop (no feed-forward):

$$\vec{A} = \frac{1}{s}\mathbf{JG}P\mathbf{J}^{+}_{est}\mathbf{R}(s)(\vec{\mathcal{A}}_{ref} - \vec{\mathcal{A}})$$

$$\Leftrightarrow \vec{\mathcal{A}} = (\mathbf{JG}P\mathbf{J}^{+}_{est}\mathbf{R}(s))(\mathbf{I}s + \mathbf{JG}P\mathbf{J}^{+}_{est}\mathbf{R}(s))^{-1}\vec{\mathcal{A}}_{ref} \tag{3.21}$$

Note, that $\mathbf{J}_{est}$ is the estimated Jacobian from section 3.4 and $\mathbf{J}$ is the real world Jacobian. $\mathbf{J}^{+}_{est}$ is the (Pseudo)-Inverse according to section 3.4.3. Finally, $\vec{\mathcal{A}}$ is a vector of the Lie Algebra of $GA(2)$ according to section 3.1.5 and $\mathbf{I}$ is the identity matrix.
If we assume that the robot has negligible dynamics, that is

$$\mathbf{R}(s) \approx \mathbf{I} \tag{3.22}$$

and that the estimated Jacobian equals the real world Jacobian,

$$\mathbf{J}\mathbf{J}^{+}_{est} \approx \mathbf{I} \tag{3.23}$$

equation 3.21 simplifies to

$$\vec{\mathcal{A}} = \frac{\mathbf{G}P}{s + \mathbf{G}P}\vec{\mathcal{A}}_{ref} \tag{3.24}$$

The step response to (3.24) can be seen in Figure 3.4. One can observe, that the time until a steady state is reached is considerable and depends on the controller gain $\mathbf{G}P$.
Including the feed-forward into (3.21) yields

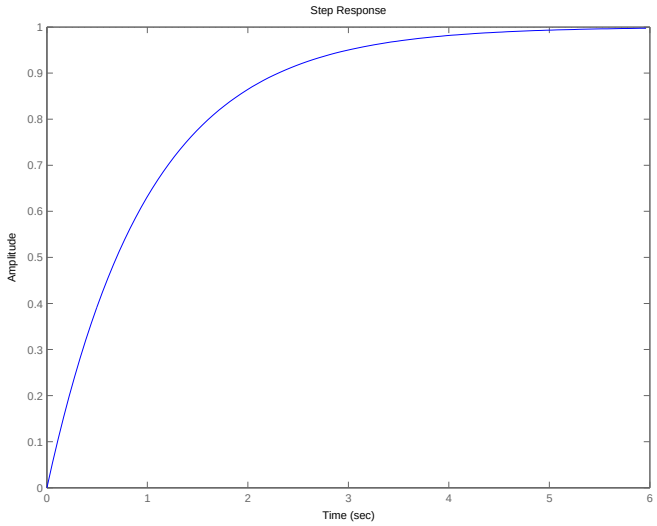


Figure 3.4: The step response for eq. (3.24) and $\mathbf{G}P = 1$

$$\vec{\mathcal{A}} = \frac{1}{s}\mathbf{JR}(s)\Big(s\mathbf{J}^{+}_{est}\vec{\mathcal{A}}_{ref} + \mathbf{G}P\mathbf{J}^{+}_{est}(\vec{\mathcal{A}}_{ref} - \vec{\mathcal{A}})\Big)$$

$$\vec{\mathcal{A}} = (s\mathbf{J}\mathbf{J}^{+}_{est}\mathbf{R}(s) + \mathbf{G}P\mathbf{J}\mathbf{J}^{+}_{est}\mathbf{R}(s))(\mathbf{I}s + \mathbf{G}P\mathbf{J}\mathbf{J}^{+}_{est}\mathbf{R}(s))^{-1}\vec{\mathcal{A}}_{ref} \tag{3.25}$$

Employing the same assumptions as in eq. (3.22) and eq. (3.23) simplifies the transfer function for the system with both feed-back and feed-forward (eq. (3.25)) to

$$\vec{\mathcal{A}} = \vec{\mathcal{A}}_{ref} \tag{3.26}$$

This behavior is obviously preferable to a control scheme without feed-forward because the trajectory following becomes better.

3.5.1 Stability of the controller

Looking at the control signal in Figure 3.3, velocities in joint space, and disregard the feed-forward, as it has no influence to stability (compare the "poles" of (3.25) and (3.21)), we have

$$\begin{aligned} \frac{d\vec{j}}{dt} &= \mathbf{G}P\mathbf{J}^{+}_{est}(\vec{\mathcal{A}}_{ref} - \mathcal{A}) \\ \mathbf{J}\frac{d\vec{j}}{dt} &= J\mathbf{G}P\mathbf{J}^{+}_{est}(\vec{\mathcal{A}}_{ref} - \mathcal{A}) \\ \mathbf{J}\frac{d\vec{j}}{dt} &= -\mathbf{G}P\mathbf{J}\mathbf{J}^{+}_{est}\vec{\mathcal{A}} + \mathbf{G}P\mathbf{J}\mathbf{J}^{+}_{est}\vec{\mathcal{A}}_{ref} \end{aligned} \tag{3.27}$$

Remembering the state-space description form [FPEN95]

$$\begin{aligned} \dot{\vec{x}} &= \mathbf{F}\vec{x} + \mathbf{G}\vec{u} \\ \vec{y} &= \mathbf{H}\vec{x} + J\vec{u} \end{aligned}$$

and comparing with 3.27 yields to a stable system if all eigenvalues of $\mathbf{F} = -\mathbf{G}P\mathbf{J}\mathbf{J}^{+}_{est}$ are in the left half plane. This is equivalent to the well known sufficient condition to ensure global asymptotic stability in visual servoing [CM00]

$$\mathbf{J}^{+}_{est}\mathbf{J} > 0 \tag{3.28}$$

for positive gain ($\mathbf{G}P > 0$).
In our approach to determine $\mathbf{J}^{+}_{est}$ (section 3.4) we have a fixed Jacobian, that is $\mathbf{J}^{+}_{est} = \mathbf{J}^{+}_{est}(t)$. In that case, it is impossible to demonstrate when condition (3.28) is ensured. However, this condition is difficult to check and exploit in practise.
Here, the stability of the system seems to be independent of the controller gain, but with increasing gain, the assumption of no robot-dynamics does not hold any longer and the system becomes unstable.

3.6 Discrete time controller

In discrete time, the process given by equation (3.15) becomes

$$\triangle\vec{\mathcal{A}}_k = J\triangle\vec{j}_k, \tag{3.29}$$

where

$$\triangle\vec{\mathcal{A}}_k = \vec{\mathcal{A}}_{k+1} - \vec{\mathcal{A}}_k$$

and

$$\triangle\vec{j}_k = \vec{j}_{k+1} - \vec{j}_k.$$

3.6.1 Stability of the discrete time controller

Looking at the control signal in discrete time, we get

$$\frac{\vec{j}_{k+1} - \vec{j}_k}{h} = \mathbf{G}P\mathbf{J}^+_{est}(\vec{\mathcal{A}}_{ref} - \vec{\mathcal{A}}_k), \tag{3.30}$$

where h is the sampling interval. Plugging now equation (3.30) into (3.29), yields

$$\begin{aligned} \vec{\mathcal{A}}_{k+1} - \vec{\mathcal{A}}_k &= \mathbf{J}(\vec{j}_{k+1} - \vec{j}_k) \\ \mathbf{J}(\vec{j}_{k+1} - \vec{j}_k) &= \mathbf{J}h\mathbf{G}P\mathbf{J}^+_{est}(\vec{\mathcal{A}}_{ref} - \vec{\mathcal{A}}_k) \\ \vec{\mathcal{A}}_{k+1} &= \vec{\mathcal{A}}_k + h\mathbf{G}P\mathbf{J}\mathbf{J}^+_{est}(\vec{\mathcal{A}}_{ref} - \vec{\mathcal{A}}_k) \end{aligned} \tag{3.31}$$

Considering now the state-space description for discrete time [FPEN95]

$$\begin{aligned} \vec{x}_{k+1} &= \Phi\vec{x}_k + \Gamma\vec{u}_k \\ \vec{y}_k &= \mathbf{H}\vec{x}_k + J\vec{u}_k \end{aligned} \tag{3.32}$$

we can rewrite equation (3.31) to

$$\vec{\mathcal{A}}_{k+1} = (\mathbf{I} - h\mathbf{G}P\mathbf{J}\mathbf{J}^+_{est})\mathcal{A}_k + h\mathbf{G}P\mathbf{J}\mathbf{J}^+_{est}\vec{\mathcal{A}}_{ref}. \tag{3.33}$$

A discrete-time, linear system is stable, if all eigenvalues of $\Phi = \mathbf{I} - h\mathbf{G}P\mathbf{J}\mathbf{J}^+_{est}$ are inside the unit circle. If we get back to our assumption for perfect calibration (3.23), Φ becomes

$$\Phi = \mathbf{I} - h\mathbf{G}P\mathbf{I}, \tag{3.34}$$

being stable for

$$0 \leq \mathbf{G}P \leq \frac{2}{h}\mathbf{I}. \tag{3.35}$$

Thus, the system stability is dependent on the sampling interval.

3.7 Image processing

To determine an affine transformation it is necessary to keep track of some features in image space, at least three (section 3.3). In our experiments four feature points were attached to the face of the gripper (Figure 3.5) by four red dots on a blue trapezoid. The marked feature points are then detected at frame-rate by the following algorithm.

After lowpass filtering of the Image using Gaussian Pyramids (3.7.1) to smooth the image and thresholding (3.7.2) to get rid of image components which do not have the expected intensity, respectively color, the blue plane of the RGB[6] image will be searched for contours (3.7.3), respectively connected areas with similar intensity/color. Then, contours which match certain criteria on size will be passed to a polygon approximation algorithm (3.7.4). Polygons, which are convex and have exactly four vertices may be the outer contour of the trapezoid corners.
Unfortunatly the vertices of the polygon are, especially for estimating the Jacobian, not sufficiently accurate. To gain sub-pixel accuracy, it is necessary to find

[6] Red-Green-Blue color model

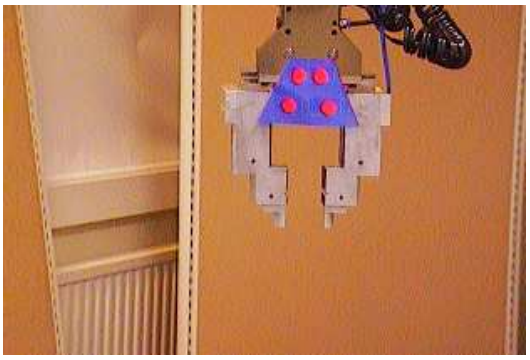


Figure 3.5: The robot end-effector. Four feature points are marked with red dots on a blue trapezoid.

the feature points which are manually attached to the gripper, where the polygon found in the previous step constrains the search window (3.7.5). Using this search window, the red plane will be searched for contours and the contours will be drawn in an empty image. In the next step, this binary image will be segmented using a floodfilling algorithm (3.7.6). The segmented areas are considered as feature points and their center of gravity yields an observation of there coordinates with sub-pixel accuracy. If four feature points are found, they are sorted clock-wise with the upper-left featurepoint first and returned to the controller after IIR-Filtering and outlier-rejection (3.7.7).
The algorithms below are implemented using the Open Source Computer Vision Library (OpenCV) and the Intel® Image Processing Library (IPL). OpenCV is mainly developed by the Intel Corporation and supports the user with a wide variety of tools for image interpretation. Building on IPL that implements low-level operations on digital images, OpenCV provides high-level algorithms for calibration techniques, feature detection and tracking, shape analysis, motion analysis, 3D reconstruction and object segmentation and recognition. Most of the functions have been assembler optimized to take advantage of Intel® architecture [Cor01].

3.7.1 Gaussian Pyramids

First of all, we have to get rid of acquisition noise. Therefore, the image becomes convoluted with a 5×5 pixels gaussian kernel. After that, it will be down-sampled by rejecting even rows and columns, reducing the image to a fourth of its original size. Then, the image will be up-sampled by injecting even zero rows and columns and convolution with with the same kernel as before multiplied by 4 for interpolation [Cor01]. As the gaussian kernel is separable, the 2D-convolution can be replaced by first convolving all rows and then all columns with a 1D gaussian kernel having the same σ [TV98]. The 1D gaussian kernel can be built by sampling a continuous Gaussian:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.36}$$

where σ is the desired standard deviation, the "width" of the gaussian shape and μ is half the filter width.

3.7.2 Thresholding

As we are only interested in certain intensities, respectively colors, we *threshold* the color plane of interest. Hereby, a pixel value y remains unchanged if it is above a certain threshold θ, otherwise it will be set to zero [Cor01]:

$$y = (y > \theta \qquad y : 0). \tag{3.37}$$

The result of such a thresholding operation is depicted in Figure 3.6.

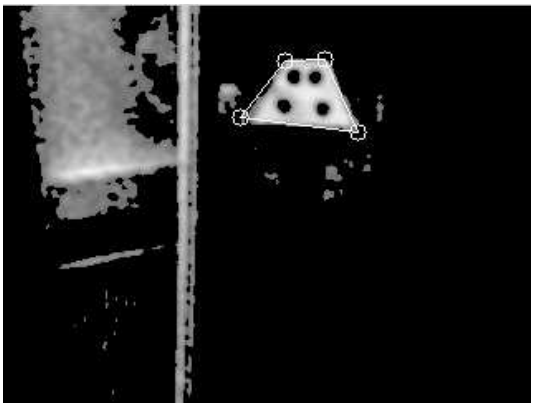


Figure 3.6: The blue color-plane of an image of the robot end-effector after thresholding. As the blue trapezoid is much brighter then the background it is clearly to spot. The detected vertices of the trapezoid are marked with circles.

3.7.3 Contour detection

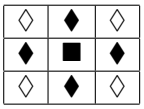
To find the outer contour of the robot end-effector, respectively the blue trapezoid, the choice of an appropriate threshold in 3.7.2 is necessary. After that, the image is treated as a *binary* image, pixels with value zero are treated as *0-pixels*, pixels with values different from zero are treated as *1-pixels*. Then, the algorithm find connected areas of 0-pixels and 1-pixels. There are two common sorts of connectivity, the *4-connectivity* and the *8-connectivity*. Two pixels with coordinates (x', y') and (x'', y'') are called 4-connected if

$$|x' - x''| + |y' - y''| = 1$$

and 8-connected if

$$\max(|x' - x''|, |y' - y''|) = 1.$$

Graphically, these relations are looking like that



where ⧫ are 4- and 8-connected to ■, and ◊ are 8-connected to ■.
Using this relationship, the image is broken into several non-overlapped 4-connected and 8-connected components, whereby 8-connectivity is used with 1-pixels and 4-connectivity with 0-pixels to avoid topological contradiction [Cor01].
0-components making up the background, a 0-component directly surrounded by a 1-component is called the *hole* of the 1-component. An example can be seen in Figure 3.7 where 1-components and holes are depicted by their borders.

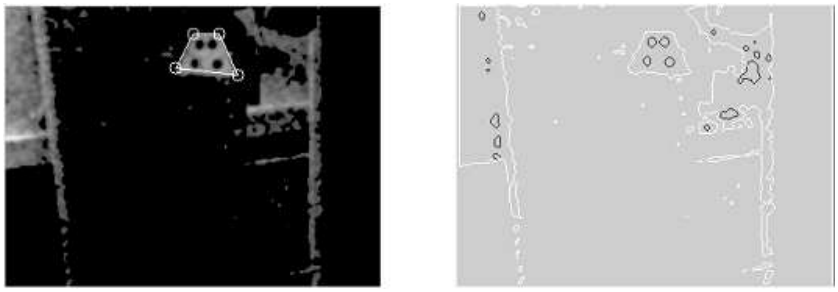


Figure 3.7: To the left the blue color-plane of an image of the end effector is shown. After thresholding with a suitable value, 1-component borders (white) and holes (black) have been marked (right image).

3.7.4 Polygonal approximation

As we have an a-priori knowledge about the expected geometry of the contour, we try now to find contours, which are quadrangular and neither to large nor to small. As the latter conditions are easily satisfied, the first condition requires a polygonal approximation.
The main idea behind a polygonal approximation is to find and keep only dominant points of the contour, for example inflection points or local maxima and connect them by straight lines. The underlying algorithms are too extensive to be covered in this report, for a brief introduction the reader may refer to [Cor01].
An example of a successful polygonal approximation of the end-effector contour can be seen in Figure 3.8.

3.7.5 Point-in-polygon test

As Figure 3.8 shows, the polygon vertices are determined with low accuracy. As the edges of the trapezoid are not very reliable after thresholding due to lighting effects, accuracy can only be increased by finding the red dots on the blue trapezoid. Therefore, we threshold the red color plane of the image with a suitable threshold and search for contours (section 3.7.3) within the trapezoid. To determine whether a point is within the trapezoid or not, we use the following algorithm:
Considering a polygon with N vertices p_i, we connect every point p_i with the test-point. This divides the polygon into slices with angles α_i, where α_i is always determined for subsequent points (compare Figure 3.9). If the sum $\sum_{i=1}^{N} \alpha_i = 2\pi$ the point is within the polygon, if the sum equals zero, it lies outside.

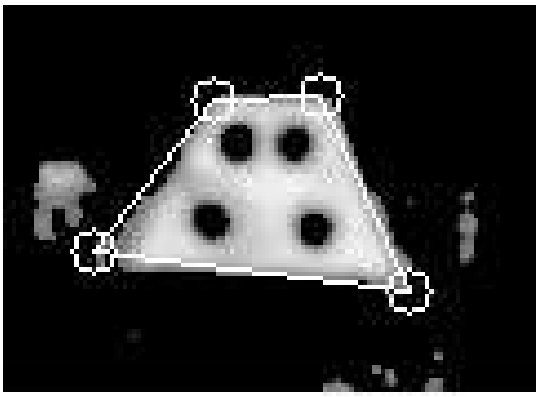


Figure 3.8: Polygonal approximation of the trapezoid contour. Outer vertices are marked with circles. It can be observed, that, due to thresholding, the vertices of the polygon are not very accurate.

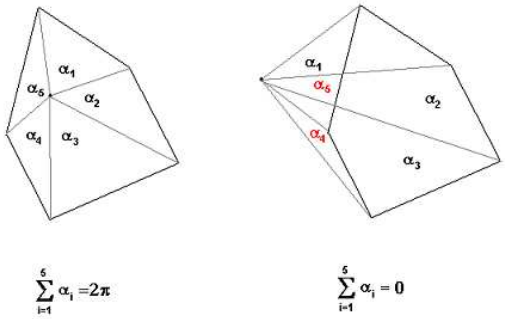


Figure 3.9: Point-in-polygon test: If the sum $\sum_{i=1}^{N} \alpha_i = 2\pi$ the point is within the polygon (left), if the sum equals zero, it lies outside (right).

3.7.6 Floodfill

To segment areas in a binary image, floodfilling them is a suitable method. Whenever a pixel value is 1, it will be taken as seed-pixel for the floodfilling algorithm and marked with an increasing color. Then, the the flood filling process propagates and analyzes the 8-connectivity of the seed point until it reaches the image borders or cannot find any new pixels to fill. The result are connected components in the image, each marked with an individual color.

3.7.7 Outlier rejection and IIR-filtering

A measurement of the four feature points is considered as an outlier, when the Euclidian distance of at least one featurepoint to his predecessor in the former image is greater than a certain threshold. In this case, the feature point detector holds the last value.

Otherwise, the detected feature points are IIR (Infinite Impulse Response)[7] filtered and returned to the controller. This is done by an *Exponential Averager*:

$$\vec{F}_{new} = \vec{F}_{old} + \alpha\big(\vec{F}_{new} - \vec{F}_{old}\big), \tag{3.38}$$

where $\vec{F}$ is a vector of feature points. Hereby, a suitable value for α has to be chosen, preferably in the interval of

$$1 \geq \alpha \geq \frac{\text{Sampling rate of the robot controller}}{\text{Sampling rate of the image processing}} \tag{3.39}$$

as a trade-off between tracking performance (optimum with $\alpha = 1$) and noise reduction ($\alpha \ll 1$).

[7] An IIR filter produces an output that is the weighted sum of the current and last inputs and the past outputs

Chapter 4

Results

Most experiments in this thesis were carried out using the virtual environment (2.4). We assume, that experiments with unsatisfying results in the virtual world, such as instability or lack of precision, become worse in the real world. In such cases, we waived the experiments on the real robot.
Furthermore, we assume for our particular experiments, that methods we found as being better then others in the simulation are also better in the real world.

4.1 Image processing

The image processing emerged as being the most delicate part in visual servoing. As the feature point detection is very reliable in the virtual environment - the feature points are detected in almost every possible orientation and position of the end-effector, as long as they are visible - in the real world, a rotation of the end-effector requires an adaption of the chosen thresholds what is not feasible in real time control. However, under good lighting conditions, experiments are possible. An example of a successful detection of the end-effector feature points is shown in Figure 4.1.

4.2 Experiments on the Virtual Robot

As pointed out in section 3.4, the first and most important step in visual servoing is the calibration of the Jacobian (4.2.1). Using calibration method that showed the best performance, we run experiments with only one camera (4.2.2). Using one camera yields good results, but the system is not stable in all camera configurations. Therefore, we try different setups using two cameras (4.2.3). This yields to robust visual servoing and is further examined regarding camera placement (4.2.4) and open-loop control error (4.2.6).

4.2.1 Calibration

For calibration of the Jacobian the best results were obtained using the method described in section 3.4.1. That is, moving the robot end-effector in every possible direction in joint space by a fixed angle. It is also possible to move in arbitrary,

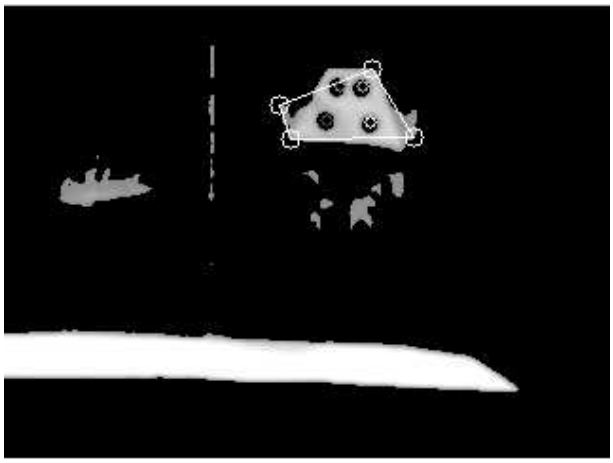


Figure 4.1: Using the real cameras, we are able to detect the feature points on the robot end-effector. Here, the outer contour of the gripper was not accurately identified, but the feature points are detected correctly.

linear independent directions in joint-space.
An obviously drawback of our approach is, that some joints cause more change in the images than others. For example, one may consider joint 1 and joint 6 moved by 5^o degrees. While joint 1 provokes a big change in the image, the influence of joint 6 is only very small. A work around to this problem is to define the motions in linear independent directions in Cartesian space, but this does not match with the philosophy of this thesis, visual servoing unaware of robot kinematics and vision setup.

4.2.2 Single camera experiments

Under the condition of weak perspective (3.1.4), 6-DOF visual servoing is theoretically possible, because a three point pose configuration in image space corresponds to an unique location in Euclidian space [HBHN95]. In Figure 4.3 results are shown for a trajectory from a position according to Figure 2.4 to a grasping position for the rear dice (Figure 4.2). The camera was hereby placed at an angle of 22.5^o relative to the initial end-effector normal. Setups with smaller angles yield to highly unstable control. This behavior was also examined in [DC00], referred to as *extrinsic degeneracy*. Instability occurs then, when the camera plane is parallel to the plane of the end-effector.

A trajectory in Cartesian space where the robot goes instable for a camera angle of 11^o degree is depicted in Figure 4.4. One can observe, that the robot tries to perform the desired trajectory to grasp the middle dice (Figure 4.2), but quickly becomes unstable. For reference, the achieved trajectory with a camera angle of 22.5^o is depicted as well. One can observe a control error which is due the dominating feed-forward but is corrected by the feed-back when the feed-forward becomes zero.

Figure 4.2: The robot is grasping the rear dice. This position was achieved using a single-camera setup. In the upper left of the image, a camera image after image processing is depicted. The detected feature points are marked.

4.2.3 Stereo camera experiments

Using two cameras we obtain more robust visual servoing. The system was stable over the whole field of view, using two cameras with a distance of 3m form the initial end-effector position and angles between the cameras between $15^o - 90^o$. With increasing distance, the precision and stability behavior became worse. In particular, a distance of 4m yields to unstable behavior in the outer regions of the field of view, whereby the Jacobian was calibrated in the center of the field of view. A response to a feed-forward trajectory is depicted in Figure 4.5. The trajectory is describing a motion from initial position (Figure 2.4) to one of the dices (Figure 4.6), depicted in Figure 4.7.

Also by using two cameras, the control error does not diminishes when the feed-forward becomes zero. This is due to the non-linearity of the transformation from joint space to affine space, which is only approximated by the Jacobian (3.4). This effect becomes smaller for higher gain in the feed-back loop but on cost of "shaky" behavior.
For a controller gain near the theoretical limit according to (3.35), the robot becomes unstable. A trajectory in cartesian space is shown in Figure 4.8. Hereby, the sampling rate of the controller is $\frac{1}{20}s$ and the gain was set to 35.

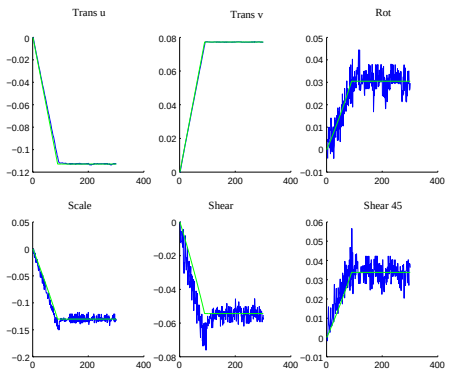


Figure 4.3: The reference trajectory and the achieved trajectory for feed-forward with feed-back are shown for every mode of affine transformation according to (3.1.3). The controller gain was set to 3, the camera was installed at an angle of 22.5^o in front of the robot.

4.2.4 Camera placement

A straightforward approach to increase the depth information in a two cameras setup, is to increase the angle between them. To measure the performance of the different camera setups, we measure the mean error between reference and actual affine transformation for each mode of two cameras. This is depicted in Figure 4.9. It can be observed, that the mean of the error decreases with an greater angle between the two cameras. Hereby, stereo setups with very small angles yield to unstable behavior, similar to the setup with a single camera. Setups with large angles become problematic in terms of image processing, because the feature points are hardly visible due to the strong shear of the end-effector.

4.2.5 Weighting of controller gain

For high controller gains, the motion of the robot is shaky. Although the noise in the shear modes (compare Figure 4.5 and Figure 4.9) is much higher than the noise level of the translation modes (1,2,6 and 7 in Figure 4.9), the translations are the main transformation in our application. By changing the gain matrix $\mathbf{G}$ as introduced in section 3.5 from the identity matrix to for example

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & .2 & 0 & 0 & 0 \\ 0 & 0 & 0 & .2 & 0 & 0 \\ 0 & 0 & 0 & 0 & .2 & 0 \\ 0 & 0 & 0 & 0 & 0 & .2 \end{pmatrix},$$

respectively a diagonal stacked version for multiple cameras, the affine modes of translation become emphasized. The result for such a gain matrix is shown in Figure 4.10. One can observe that the noise level of the translation modes has become very

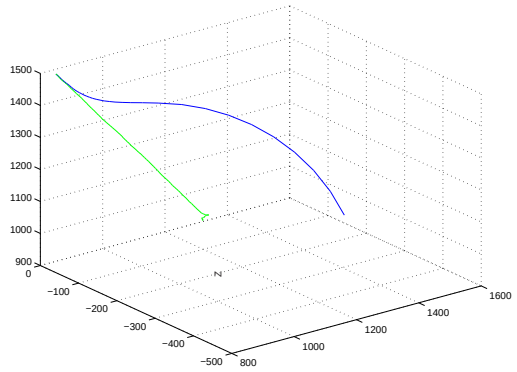


Figure 4.4: End-effector trajectory in Cartesian space. The robot goes unstable for an camera angle of 11^o degrees. A trajectory achieved with a camera angle of 22.5^o is depicted as reference. Axes are in millimeters.

little. Observing the motion of the robot, the shaky behavior diminishes.
The trade-off of such an approach is decreased performance in the under-emphasized affine modes, yielding to higher rise-times for this transformations. However, a smart choice of $\mathbf{G}$ will allow a fast motion to the desired position with maintainable delay in end-effector orientation.

4.2.6 Open-loop control

As we use only a linear approximation of the highly non-linear transformation from robot actuator to affine image space, we examine the emerging error. This can be done for example by disabling the feed-back loop in the controller (3.5). A ramp response for the same trajectory as used in Figure 4.5 is depicted in Figure 4.11. Experiments have shown, that the error increases with growing distance from the point of calibration, due to the non-linear effects.
Looking at Figure 4.11 one can observe, that corresponding translations, for example the first translation modes, differ significantly from each other. This is due the end-effector is near the epipolar line of one camera, yielding only small error in image space. Compare Figure B.1 in Appendix B.

4.2.7 Closed-loop without feed-forward

The system was simulated using only a feed-back loop. The results for $\mathbf{G}P = 5\mathbf{I}$ are shown in Figure 4.12 for the ramp response and Figure 4.13 for the trajectory in Cartesian space. As expected according to equation (3.24), the trajectory following is worse then with feed-forward. On the other hand, the end position is reached in a "smooth" motion without the observable correction in a feed-forward/feed-back configuration (compare Figure 4.7 The stability properties of the system are identical to the case with feed-forward (3.5.1

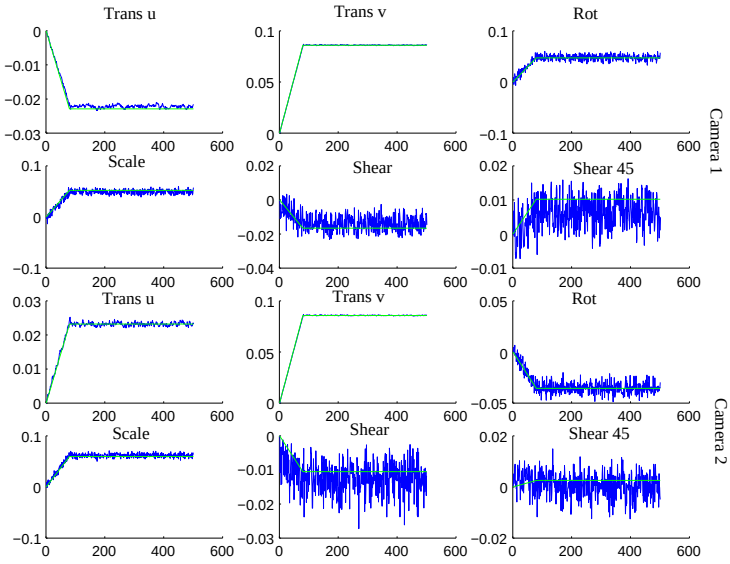


Figure 4.5: Ramp response to a trajectory in affine space using feedback and feed-forward. The controller gain was set to 5 for all affine modes. The first two rows correspond to the 6 affine transformations in the left camera, the last two rows correspond to the right camera.

4.3 Experiments on the real robot

Using the same Jacobian estimation as in the virtual environment, we did experiments on the real robot with a stereo camera setup. We observed good trajectory following and transient behavior in a neighborhood of the estimation point of the Jacobian (compare Figure 4.14 and Figure 4.15). Farer away, the ramp response showed considerable over-shoot (compare Figure 4.16). This behavior was also observed during simulation with inaccurate definition of the end-position in affine space: if the desired affine transformation is only roughly estimated, the trajectory is possibly not feasible for the robot. In such cases, the feedback will cause oscillating behavior or instability.

To increase the transient behavior, an option is to send a "non-causal" trajectory to the feed-forward, that is

$$\vec{\mathcal{A}}_{ref,feed-forward}[t] = \vec{\mathcal{A}}_{ref,feed-back}[t + \triangle t],$$

where $\triangle t$ is the time the feed-forward is looking ahead.

4.4 Motion primitives

We are able to define desired trajectories in Euclidian space by their projection to affine space (3.1.3) using a *Teach by showing* approach or by the knowledge of the

Figure 4.6: The robot is grasping a dice. The camera images are projected to the upper left, their angle is 45^o and the distance is 3m.

objects we want to interact with, described in section 3.2. Hence, an appropriate motion primitive could be defined as follows:

```
move(object, affinetrans, time, J)
```

Provided with the knowledge, where the gripper is, given by `object`, `move` performs an affine transformation in image space, given by `affinetrans` in the given time `time` and using the Jacobian `J`.
This motion primitive was implemented and used to define a complex task, collecting dices and put them on top of each other to construct a tower (Figure 4.17). An example for a subset of such a task is given as pseudo code below:

```
10 J=do_testmovements
20 object=vision_find_gripper
30 dice=vision_find_dice
40 affinetrans=determine_affine_transformation(object, dice)
50 time=10s
60 move(object, affinetrans, time, J)
```

After determination of the Jacobian for a neighborhood of the actual end-effector position (10), a high-level vision algorithm has to determine the end-effector (20) and dice (30) position in image space. Then, the affine transformation from the end-effectors actual position to the dice has to be calculated (40). After setting of the duration of the motion (50) it is executed by (60).

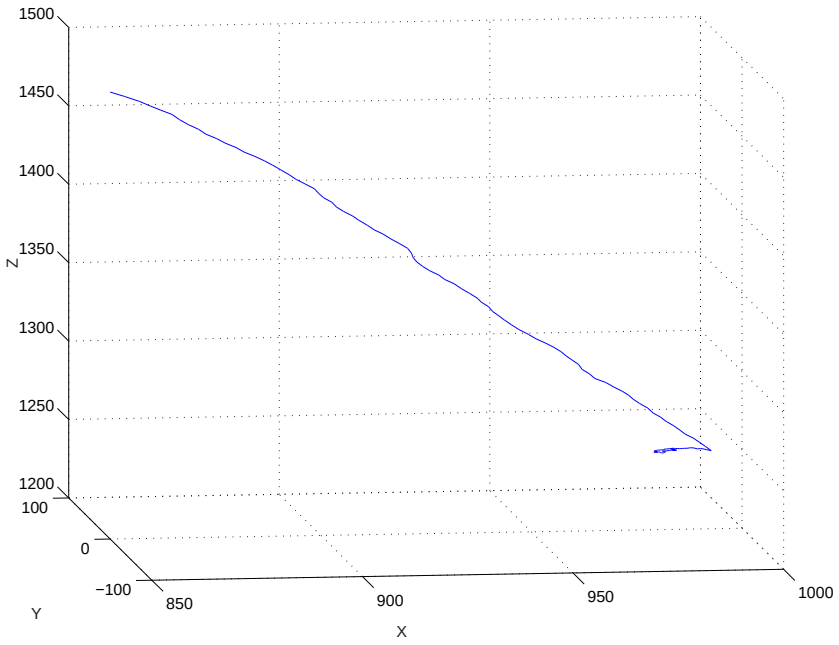


Figure 4.7: Trajectory in cartesian space. The robot is grasping the dice. Axes are in millimeters.

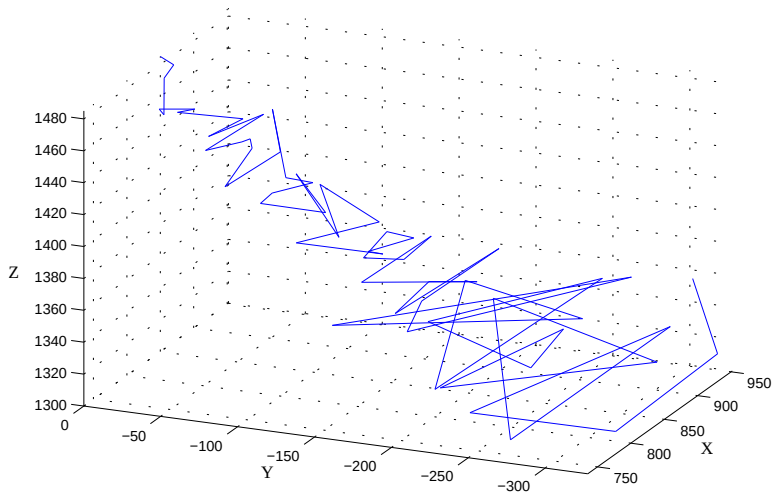


Figure 4.8: Trajectory in cartesian space. For a controller gain near the theoretical limit according to (3.35), the robot becomes unstable.

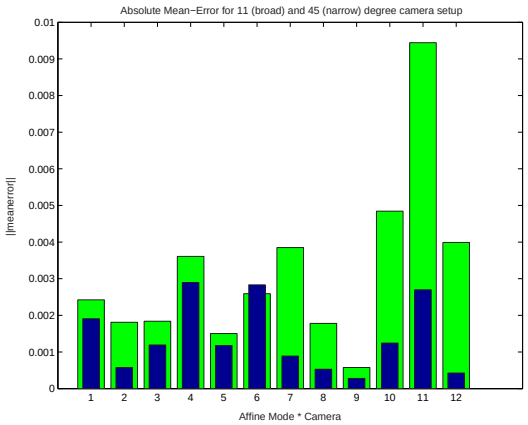


Figure 4.9: The mean error for a given trajectory in affine space is shown. The narrow blue bars are corresponding to a camera setup with 45^o (compare with Figure 4.5), the bold green bars correspond to a 11^o setup. One can observe a significantly higher mean error for the 11^o degree setup.

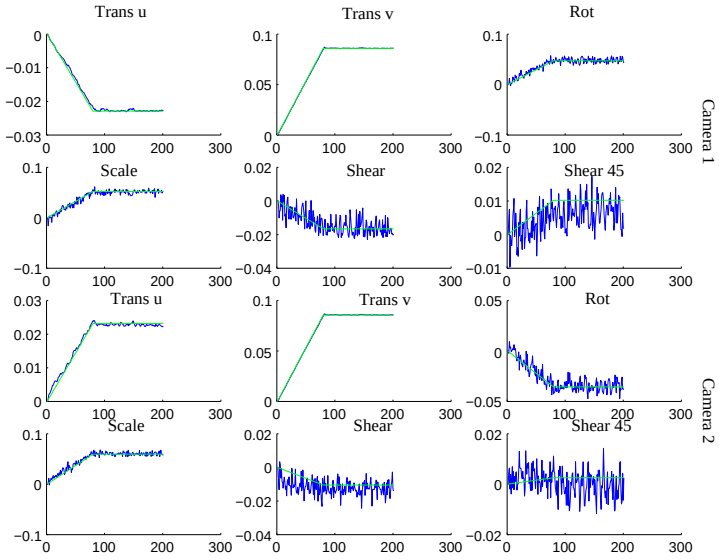


Figure 4.10: For the same trajectory as in figure 4.5 the controller gain was changed. The translation modes are weighted with 10 and the other with 2. One can observe a decrease in noise in the translation modes. Observing the motion of the robot, the "shaky" behavior diminishes.

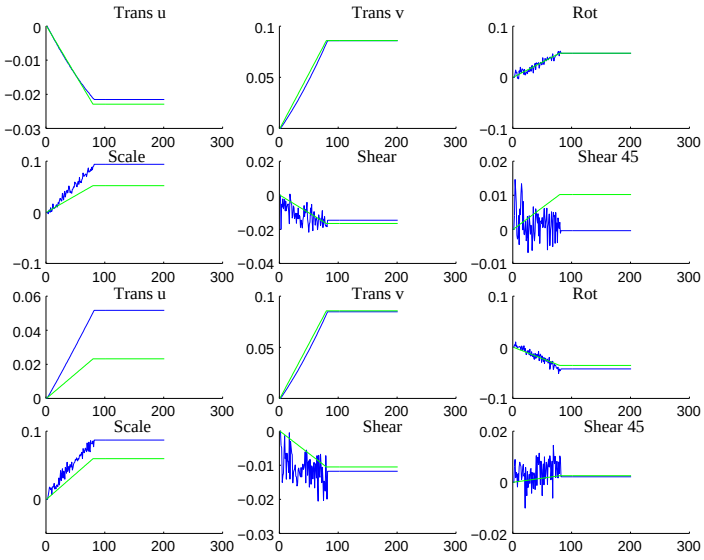


Figure 4.11: Ramp response using only feed-forward. Compare with 4.5. The steady-state error rises with growing distance from the point of calibration.

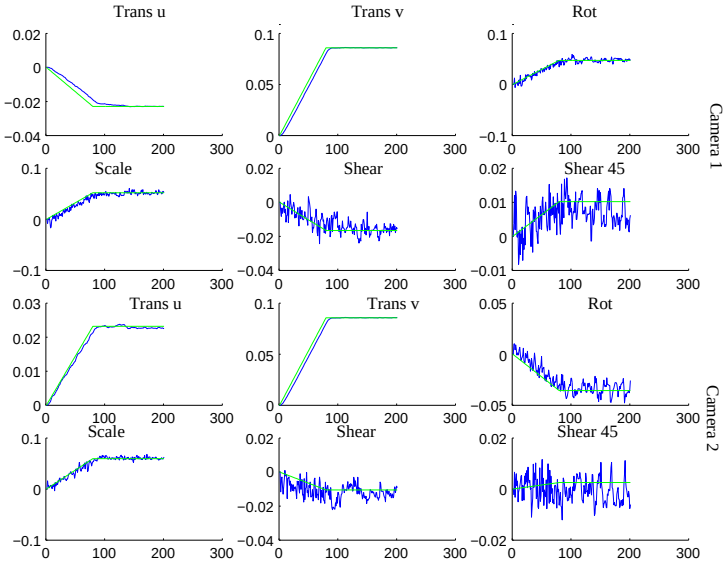


Figure 4.12: Ramp response for a controller using only feed-back (2 Cameras). $\mathbf{G}P = 5\mathbf{I}$.

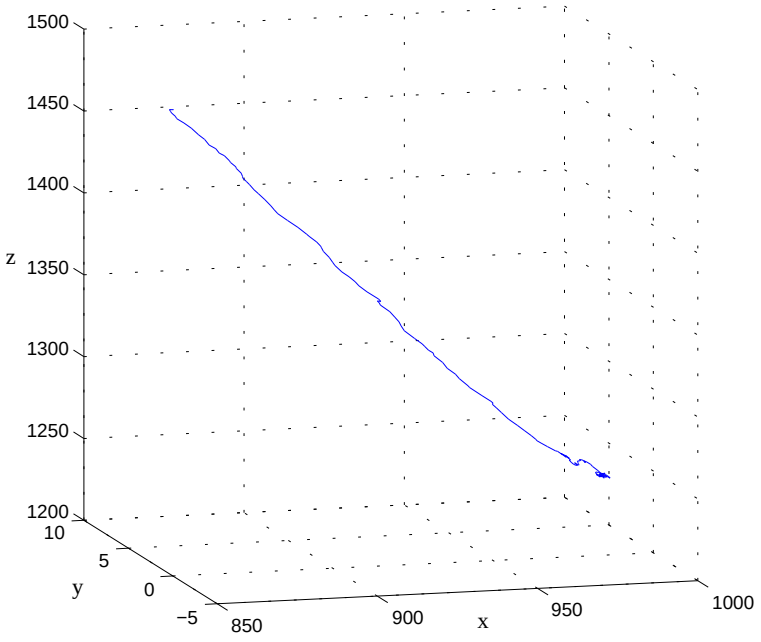


Figure 4.13: Trajectory in Cartesian space for a controller using only feedback (2 Cameras). $\mathbf{G}P = 5\mathbf{I}$.

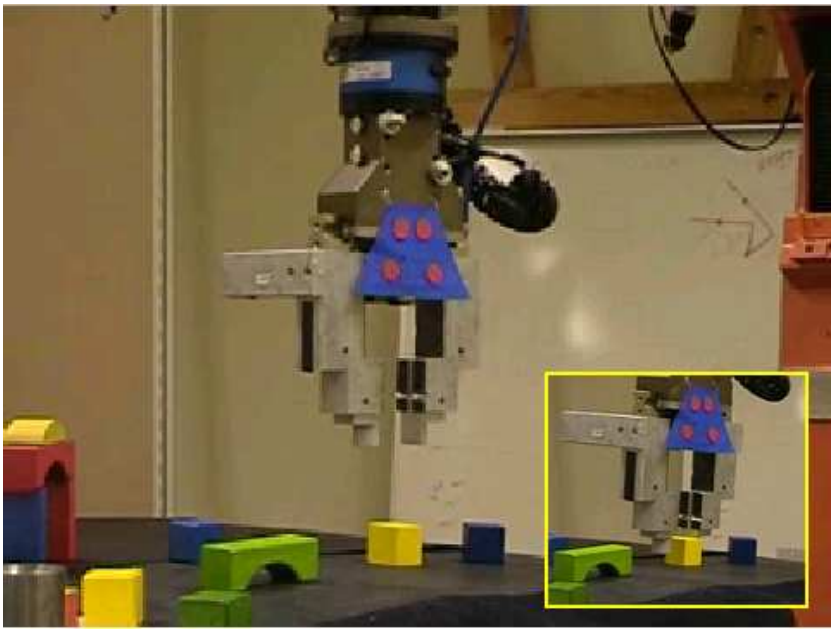


Figure 4.14: In a neighborhood of the estimation point of the Jacobian (big picture), the robot showed good trajectory following and transient behavior (the small picture shows the achieved end-position).

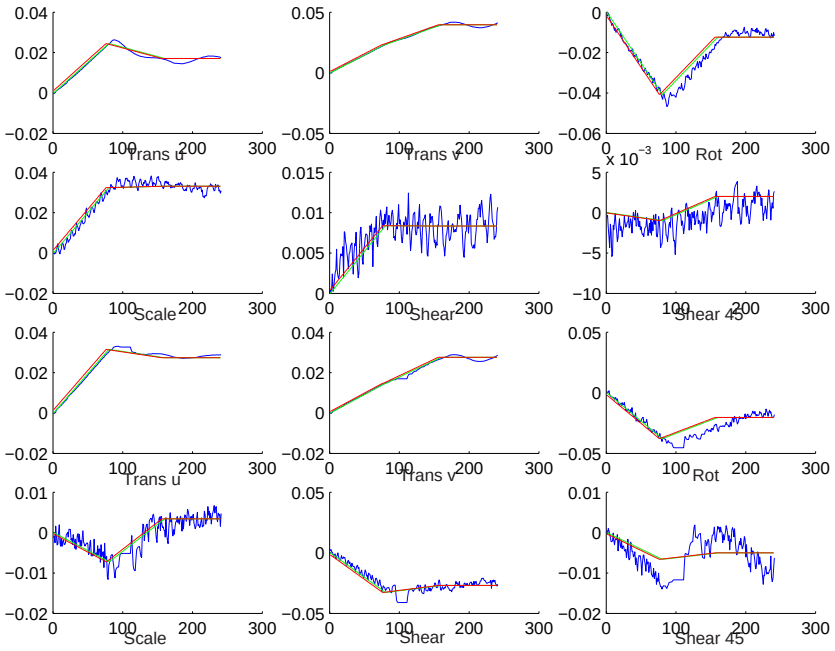


Figure 4.15: Response for a complex trajectory. The endeffector has first to take a position right above the dice and then to move straight down to grip it. Plotted is the achieved and the reference trajectory.

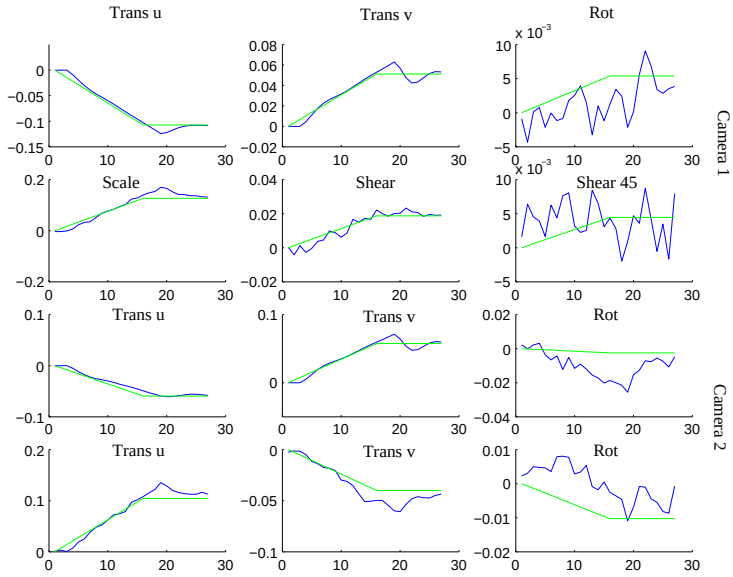


Figure 4.16: With increasing distance from the calibration Point of the Jacobian, the transient behavior gets worse. The controller was running at 10 Hz with a proportional gain of $P = 1$. Y-Axis: 1/1000 image pixel, X-Axis: Samples.

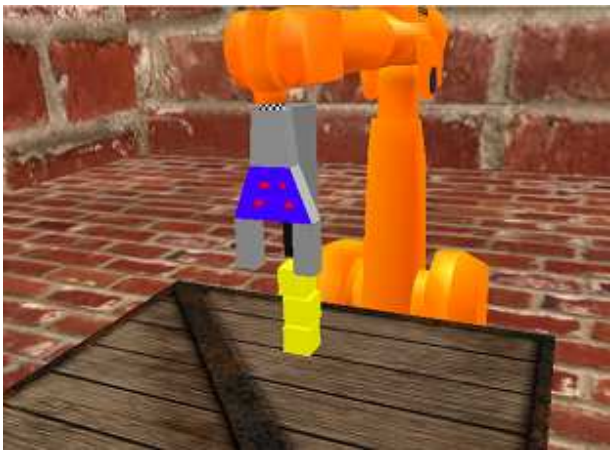


Figure 4.17: The robot has built a tower. The task was defined by trajectories in affine space.

Chapter 5

Discussion

5.1 Simulation and development with the Virtual Robot

The virtual robot for simulation has been an invaluable tool for solving the given task. Besides that the evaluation cycles become much faster, some experiments, e.g. where topological changes of the cameras are desired, would not have been feasible in the real world, either due to lack of hardware or spatial limitations. Above all, some considered control strategies have lead to instability, which could have caused severe damages in the real world, if carried out on the real robot.
Another aspect is the repeatability of the virtual environment, especially on the vision part. Changing lighting conditions during the day, complicate the development of a robust vision mainly in the early stage of design.
In this thesis image processing and control was developed on the virtual environment and transferred in a fraction of time to the real robot. To handle changing light conditions, the fixed thresholds mentioned in section 3.7 were substituted by slider bars but the robot controller remained unchanged. As expected, the performance was slightly worse due to an increased noise level.

5.2 Camera placement

To maximize the effect of using more than one camera, the cameras should be placed with the angle between optical axes as large as possible. If the camera planes are parallel, then all 6 possible modes of affine transformations will almost be linear dependent of the affine transformation occurring in the other images. For example, a translation of the gripper will result in all camera images as an affine translation with different scale. This would give the same drawbacks as for a single-camera setup.
If the camera projection planes are perpendicular, a transformation in joint space leads to changes in different modes of affine space. For example, if the gripper is moving towards one camera, leading to a change in scale in affine space, the perpendicular camera will record a translation in image- and in affine space. Depending on the application it may be hard to realize perpendicularity between the projection planes as in this cases the gripper with the sampled feature points, may be difficult

to detect in different images at the same time.

5.3 Precision vs. Imageresolution

Assuming a linear projection of the light to the camera CCD, the intersection of two camera fields of view yields to a 3-D grid with different voxel size. The voxel size is increasing with growing distance to the cameras, yielding decreased resolution in the image. The voxel size in function of their position in space using the focal length, the resolution and the angle between the two cameras is a constraint for the achievable precision of visual servoing and has be taken into account for task specifications.

5.4 Conclusions and further work

It was demonstrated that visual servoing using the Lie algebra of affine transformation is a feasible approach to robust control a 6-DOF robot using two cameras. Hereby, the computer vision was the least robust aspect of visual servoing. As pointed out earlier, noise in detection of the visual features yields to incorrect Jacobians, outliers, resp. loose of tracking yield to unstable control. Noise reduction in image processing can be achieved by using of affine Snakes as shown in [HC94] and/or by an adaptive change of camera parameters to minimize the influence of changes in lighting.
Using a fixed Jacobian, we can only expect stability and accurate control in a neighborhood around the calibration point. Better results will arise if the Jacobian is not fix but changes dependent on location in task-space.
To increase the performance of the visual servo, dynamic effects and latencies in the system have to be taken into account. As the handling of latencies is well known [Bou01], a compensation for the dynamic effects can be found by a system-identification of the robot and vision system.

Bibliography

[BD85] T. Broecker and T. Dieck. *Representations of Compact Lie Groups*. Graduate Texts in Mathematics. Springer Verlag, 1985.

[Bou01] M. Bourmpos. Vision based robotic grasping tracking of a moving object. Master's thesis, Department of Automatic Control, Lund Institute of Technology, September 2001.

[CG96] P. Corke and M. Good. Dynamic effects in visual closed-loop systems. *IEEE Transactions on Robotics and Automation*, 12(5):671–683, Oct. 1996.

[CM00] F. Chaumette and E. Malis. 2 1/2 d visual servoing: a possible solution to improve image-based and position-based visual servoings. *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, pages 630–635, 2000.

[Cor01] Intel Corporation. *Open Source Computer Vision Library, Reference Manual*. Intel Corporation, 1999-2001.

[Cra89] J. Craig. *Introduction to Robotics: Mechanics and Control (2nd Edition)*. Addison-Wesley, 1989.

[CRW98] P. Corke, J. Roberts, and G. Winstanley. Vision-based control for mining automation. *IEEE Robotics & Automation Magazine*, 1998.

[DC00] T. Drummond and R. Cipolla. Application of lie algebras to visual servoing. *International Journal of Computer Vision*, 37(1), 2000.

[dM02] J. Luis de Menas. Virtual environment for development of visual servoing control algorithms. Master's thesis, Department of Automatic Control, Lund Institute of Technology, March 2002.

[ECR92] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3), 1992.

[FPEN95] G. Franklin, J. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, 1995.

[HBHN95] Thomas S. Huang, Alfred M. Bruckstein, Robert J. Holt, and Arun N. Netravali. Uniqueness of 3d pose under weak perspective: A geometrical proof. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:1220–1221, 1995.

[HC94] N. Hollinghurst and R. Cipolla. Uncalibrated stereo hand-eye coordination. *Image and Vision Computing*, 12, 1994.

[HC00] S. Hutchinson and P. Corke. A new hybrid image-based visual servo control scheme. *Proceedings of the 39^{t}h IEEE Conference on Decision and Control Sydney, Australia*, pages 2521–2526, December 2000.

[HHC96] S. Hutchinson, D. HaGer, and P. Corke. A tutorial to visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5), 1996.

[Jae97] M. JaegersAnd. *On-line Estimation of Visual-Motor Models for Robot Control and Visual Simulation.* PhD thesis, University of Rochester, 1997.

[Ols01] T. Olsson. Vision guided force control in robotics. Master's thesis, Department of Automatic Control, Lund Institute of Technology, 2001.

[TV98] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[YZD02] M. Yim, Y. Zhang, and D. Duff. Modular robots. *IEEE Spectrum*, February 2002.

Appendix A

Analytical derivation of the image Jacobian

Assuming a point $\vec{P} = [x, y, z]^T$ is moving with translational velocity $\vec{T} = (T_x, T_y, T_z)$ and angular velocity $\vec{\Omega} = [\omega_x, \omega_y, \omega_z]$ in Euclidian space with respect to the camera frame.
We are interested in finding the Jacobian $\mathbf{J}$, that

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \mathbf{J} \begin{bmatrix} \vec{T} \\ \vec{\Omega} \end{bmatrix}. \tag{A.1}$$

Therefore, we begin to express the velocity of $\vec{P}$ by

$$\dot{\vec{P}} = \vec{\Omega} \times \vec{P} + \vec{T},$$

respectively

$$\begin{aligned} \dot{x} &= z\omega_y - y\omega_z + T_x \\ \dot{y} &= x\omega_z - z\omega_x + T_y \\ \dot{z} &= y\omega_x - x\omega_y + T_z \end{aligned} \tag{A.2}$$

Using a projective camera model as depicted in Figure 3.1, $\vec{P} = [x, y, z]^T$ will project onto the image plane with coordinates $\vec{p} = [u, v]^T$ given by

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{\lambda}{z} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{A.3}$$

Plugging equations A.3 into equations A.2 yields

$$\begin{aligned} \dot{x} &= z\omega_y - \frac{vz}{\lambda}\omega_z + T_x \\ \dot{y} &= \frac{uz}{\lambda}\omega_z - z\omega_x + T_y \\ \dot{z} &= \frac{z}{\lambda}(v\omega_x - u\omega_y) + T_z \end{aligned} \tag{A.4}$$

Using A.3, we calculate $\dot{u}$ and $\dot{v}$ using the quotient rule

$$\begin{aligned} \dot{u} &= \lambda \frac{z\dot{x} - x\dot{z}}{z^2} \\ \dot{v} &= \lambda \frac{z\dot{y} - y\dot{z}}{z^2}. \end{aligned} \tag{A.5}$$

Plugging A.4 into A.5 and using matrix notation according to A.1 yields:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{z} & 0 & -\frac{u}{z} & -\frac{uv}{\lambda} & \frac{\lambda^2+u^2}{\lambda} \\ 0 & \frac{\lambda}{z} & -\frac{v}{z} & -\frac{\lambda^2+v^2}{\lambda} & \frac{uv}{\lambda} \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{A.6}$$

This derivation can be fund in a number of references including [HHC96].

Appendix B

The epipolar constraint

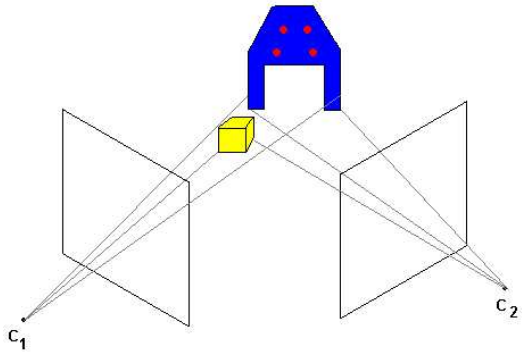


Figure B.1: Epipoles and Epipolar lines.

Looking at a point P in 3D-space, for example a point on the dice in Figure B.1, it spans a plane π with the centers of projection of the two cameras, $C1$ and $C2$. The plane π is called *epipolar plane* and the lines where π intersects with the image planes is called *conjugated epipolar lines*. The image in one camera of the projection center of the other is called *epipole*. The *epipolar constraint* is defined as follows:

> Corresponding points must lie on conjugated epipolar lines.

For a more comprehensive introduction to epipolar geometry, refer to [TV98].