MULTI-SENSOR CONTROLLED ASSEMBLY AND APPLICATION WITH MANIPULATORS

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul INGINERIE ELECTRONICĂ
ŞI TELECOMUNICAŢII
de către

Ing. Philipp Roebrock

Conducător ştiinţific:	prof.univ.dr.ing Marius Oteşteanu
Referenţi ştiinţifici:	prof.univ.dr.ing. Virgil Tiponuţ
	prof.univ.dr.ing. Ivan Bogdanov
	prof.univ.dr.ing. Adrian Graur
	prof.dr.ing. Michael Schnell

Ziua susţinerii tezei: 06.02.2009

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Acknowledgements

This work would not have been possible without the help of many people. At this point, I would like to take the opportunity to say: "Thank you!"

First of all, I would like to thank my doctoral supervisor, Prof. Dr. Ing. Virgil Tiponuţ. His great help and motivation were essential. Then, many thanks to all professors at the Faculty of Electronics and Telecommunications Engineering at the "Politehnica" University of Timisoara, especially to the Dean, Prof. Dr. Ing. Marius Oteşteanu, and the head of the Department of Applied Electronics, Prof. Dr. Ing. Ivan Bogdanov, for supporting me and believing in me as a doctoral student. I would like to thank Prof. Dr. Ing. Adrian Graur and Prof. Dr. Ing. Michael Schnell for being members of the referee board and their precious comments.

Further persons that made this thesis actually possible are Dr. Frank Grünewald, Dr. Werner Neddermeyer and Prof. Dr. Wolfgang Winkler. They gave me the unique chance to combine my research activities with working for a provider of industrial automation solutions. The connection of scientific research and practical oriented work was a fruitful combination. Thank you for your continuous and generous support.

I want to thank Klaus Lehmann for our valuable discussions and his very precious tips. They were an indispensable source of insight and inspiration. Thank you for your time and your support. Further thanks go to my fellow doctorate students Kay Böhnke and Michael Kleinkes. It was nice sharing this experience with you. Thanks to Andreas Goebels and Mark Jones for proofreading my thesis and for their substantial comments.

Last but not least, I want to thank my parents for all their love and support.

Roebrock, Philipp

Multi-Sensor Controlled Assembly and Application with Manipulators

Teze de doctorat ale UPT, Seria 7, Nr. 11, Editura Politehnica, 2009, 96 pagini, 47 figuri, 6 tabele.

ISSN: 1842-7014

ISBN: 978-973-625-823-7

Cuvinte cheie:
Visual Servoing, Learning Approach, Jacobian, Stability, Hysteresis, Sampling Delays

Rezumat:
This thesis addresses the field of visual servoing and deals with solution methods for the non-adaptive estimation of the Jacobian in learning approaches for visual servoing. Sensors are taken as abstract sources of information that are used to identify robot movements by linearizing the coherence between sensor data deviations and robot position deviations in a Taylor expansion with a Jacobian. We compare different methods to determine the Jacobian from learning data such as the inversion of the Feature Jacobian or the direct solution of an over-determined system, regarding the properties of practical relevance for visual servoing applications like the flexibility of the calculation and the quality and the stability of the result. In the conclusion, we are able to suggest a procedure to extract the Jacobian out of learning data. The analyzes are illustrated by multiple examples from real world experiments.

Another part of this work deals with an effect that occurs when collecting learning data; a non-linearity of hysteresis-type in the samples of robot positions and the sensor data. This effect can cause problems when identifying the system with some of the previously mentioned methods. We explain the origin of this effect (overlay of sampling delays and mechanical robot effects) and discuss methods for the determination and compensation of the effect in the learning data.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

This chapter offers an introduction to the subject of this thesis as well as an overview of the related work of previous publications and a description of the structure of this document.

1.1. Overview

Modern industry is marked by an increasing level of automation. The main reasons behind this progress are either economical arguments or higher quality standards for the products. In the classical production environment, the most influential components on production accuracy are the manipulator system itself (consisting of the manipulator, the tool and possible external axes), the conveyor system and the amount of production tolerances in the work object. To be able to manage the occurring deviations, suitable sensor systems are required to be used to acquire error deviations and smart controller systems need to transform this information into corrections for the manipulator. So, if we want to be able to find automation solutions for tasks with growing complexity at high accuracy demands, this is only possible with continued research and development in the field of sensor-controlled manipulators and industrial robots. This particular field of engineering and computer science that deals with sensor-controlled robots, is called *visual servoing*.

The original approach in visual servoing was limited to the usage of camera sensors (as the word "visual" suggests). However, with the availability of newer, industrial grade sensor technologies such as laser stripe sensors, this approach has been generalized. Because every sensor technology has its own unique properties, it can be favorable to combine the advantages of different sensor technologies by using multiple sensors simultaneously. An example would be a robot position control relative to a metal sheet with a stamped hole in it. The position relative to the hole could be detected in two DOFs[1] by a hand-mounted camera, and the distance to the sheet (one DOF) and the tipping (two DOFs) can be determined by additional laser distance sensors. With this setup, we combine the easy detection of the hole with the camera with the simple distance measurement of the laser sensors to control the robot in five DOFs. Such a *multi-sensor* approach raises new questions and problems such as the necessary abstraction of the sensor data, the optimal rating of different information sources that provide similar information but with different reliability, or the problem of creating a controller capable of making decisions based on partially available sensor data. This touches the field of *sensor data fusion*, which is quite a popular subject in mobile robotics and military applications.

In this thesis, we pursue the *learning approach* of visual servoing, the identification of robot movements and changes in the feature space with a learning step (in our case by using supervised learning, which means it is based on a list of inputs (sensor information) and desired outputs (robot positions)). For system identification, we use analytical methods and limit the analysis to non-adaptive solutions so that we can identify the system once during setup in a calibration procedure and use a fixed Jacobian during the control process. For the discussion of adaptive solutions we refer to the papers [23] and [61]. Classifying the systems after [80] and [95], we limit our view to *image-based systems* that follow the *dynamic look and move* approach. The work of [58] (with follow-up in [59]) discusses in the introduction the difference between 3D and 2D visual servoing. In 3D visual servoing, we have a model of the target object and we are able to directly

[1] DOF=Degree Of Freedom

calculate the robot deviation using the feature deviations (for example, by using cameras and calculating a bundle adjustment), while in 2D visual servoing we transform the feature deviations using a Jacobian into the robot coordinate space. The said papers offer an in-between solution - referred to as the *hybrid approach* - by separately estimating the rotation and the scaled translation to the object, which results in an extremely stable approach due to this decoupling. The approach of this thesis is similar to that of 2D servoing, but instead of limiting the view to a hand-mounted single camera we consider a number of abstract control signals to any special kind of sensors, determine the deviation in the control signal space, and use a Jacobian to transform the result into robot coordinates. We make no assumptions of the object's geometry, so our approach is *model-free* too. Furthermore, our learning approach employs a complete system identification, so we do not have to have previous knowledge about any sensor TCPs.

The method described above - using a Jacobian matrix for the transformation of feature deviations into robot corrections - shows in practice a great robustness against changes of the Jacobian. Accordingly, if determining the Jacobian the result is not unambiguous. This leaves us with the problem of finding an optimal Jacobian to a given configuration and - prior to that - defining the term "optimal". The literature provides us with a number of solution methods and discusses their different mathematical advantages, but a discussion regarding the practical properties of the single methods (taking the most recent approaches into account) is not available. This thesis tries to fill that gap by defining a number of suitable, quality criteria to judge the quality of a solution method for a given learning data set (trace data set), and by comparing the collection of different methods by using these criteria. Additionally, an effect occurs in connection with the sampling of features and robot positions when collecting data for the learning step. This effect is a hysteresis-like effect that causes problems when determining the Jacobian with certain solution methods or when realizing robot controllers. In this thesis, the origin of this effect is analyzed and its compensation is discussed. All important results in this work are illustrated with examples and results from real world experiments to show the practical relevance.

1.2. Related Work

This thesis is basically a work in the field of visual servoing. The term "visual servoing" was introduced in [33] to differentiate a new method from previous approaches that treated the processes of taking pictures and moving the robot as separate tasks. To get a thorough overview of the results of the first fifteen years of research in the area of visual servoing with an extensive bibliography and multiple application examples, take a look at [18] and [19]. An introductory tutorial in the world of visual servoing can be found in [39]. A more recent introduction to visual servoing for manipulation applications can be found in [44]. Some of the latest tutorials on visual servoing have been published in [13] and [14]. Besides an introduction, they present the latest contributions in the field of visual servoing such as hybrid approaches or the usage of trajectory planning.

A paper that is closely related to this thesis is [52], in which the author discusses two methods for the estimation of the Jacobian, the direct calculation and the inversion of the Feature Jacobian. The author shows the advantage of the direct calculation of the Jacobian over the more common calculation via the pseudo-inverse of the Feature Jacobian, but points out stability issues of the direct calculation that still have to be addressed. This thesis extends the stability analysis of the directly calculated Jacobian and makes suggestions on how to improve the direct calculation in terms of stability by using regularization techniques. An example of a system making use of the estimation of the Jacobian using the Feature Jacobian can be found in [93]. It describes a robot path correction system for the reduction of dynamic robot errors, which uses local Jacobian matrices for each robot path point. Another paper that compares different learning methods can be found under [60], though this is done for systems using task sequencing, the separation of control tasks into multiple sub-tasks for enhanced performance, and stability.

With the idea of using different kinds of sensors in visual servoing for optimal information gathering, the field of data fusion is getting more and more important in visual servoing, and deals with the fusion of different information sources with individual accuracy and reliability. A historic overview over the field of data fusion, along with the basic concepts and tutorials, can be found in the works of [54], [55] and [56]. Another feature level approach to multi sensor data fusion with links to robotics can be found in [10] and [32].

There is a long list of example applications in the field of visual servoing that use different sensor technologies and different methods for system identification. A class of applications where visual servoing is commonly used is seam tracking for welding. The paper of [27] as part of the thesis [28] shows a recent overview of the history of robot seam tracking applications and describes the realization of a simulation environment consisting of a commercial robot simulation tool connected with a numerical computing environment for the design of a 3D seam tracking system for the welding of ship hulls. The work of [72] contains an in-depth analysis of components for real time seam tracking applications as well as some basic approaches for adaptive welding using triangulation sensors. In [57], a sensor-based seam tracking system for pipe welding is presented, similar to [64] and [65]. A classical visual servoing application can be found in [34] with the usage of a hand-mounted stereo vision camera and an affine stereo algorithm for control. In [66], the authors describe an offline supervised learning approach for the correction of robot programs in which the robot executes an offline taught program along a work object while a calibrated sensor measures the distances to the desired robot path. The differences are fed into a multi-layered neuronal network. Another approach using neuronal networks can be found under [17]. It describes the development of an image-based visual servoing controller with a feed-forward neuronal network that used feature information (in this case camera data) and robot states as its learning data. The presented approach is an adaptive approach. Similar is the work of [69], in which the authors use a position sensitive device (PSD) as sensor to control a robot using a back-propagation network approach. Works that focus mainly on the architecture of visual servoing applications can be found under [97], in which strategies for fast online path correction systems are discussed, or in [50] and [51] who suggest an architecture where the sensor control is divided into a robot positional control and a path planning part. This decoupling of the fast robot control and a possibly slower sensor-based path planning allows for very efficient path correction solutions. In [68], a laser-sensor based system for autonomous surface-following is developed, which uses no previous knowledge about the actual surface (no learning approach). Because of its importance for industrial applications, visual servoing is a patent subject matter too. One patent in that area is [35], who describes the path correction by a triangulation sensor. This uses a learning approach and utilises "conversion" matrices to determine the final robot corrections. Similar approaches for real time seam tracking with triangulation sensors can be found under [38] or [62].

Some visual servoing applications require a TCP[2] calibrated sensor, for example the online path planning systems with a laser triangulation sensor described in [91] or in [66]. An early work on the TCP calibration of hand-mounted triangulation sensors can be found under [1]. In [79], the authors use a Bayesian iterative model to solve for the TCP calibration. In [67], a least squares solution for the sensor calibration of triangulation sensors is determined so that the calibrated sensor can be used for the 3D measurement of free formed parts. The work of [73] describes an inverted Feature Jacobian approach for TCP calibration.

A part of this thesis deals with the analysis of the hysteresis effects of robots. Most works that have been published in connection with hysteresis effects are about the hysteresis effects

[2]TCP=Tool Center Point

of magnetic materials. A paper that describes the effect of valve stiction in the area of control of chemical plants can be found under [16], and provides good definitions of hysteresis-related terminology. The paper by [82] studies the compensation of actuator non-linearities of hysteresis-type by using neuronal networks. In [43], the author describes the identification of non-linear systems with saturation and hysteresis and the usage of a recurrent neural network and non-linear optimization techniques for their compensation.

1.3. Structure

After this initial chapter (Chapter 1), in which we give an overview of the subject of this thesis and list the related publications in that field, we offer in the next two chapters an introduction to two important components for visual servoing systems: Robots (Chapter 2) and sensors (Chapter 3). These chapters present the current, state-of-the-art technologies that have been used to realize a test system and focus on technical details that become important in the further course of the thesis. Furthermore, the central problem in visual servoing and the central point in this work is the identification of robot movements by observing sensor data changes. We treat this subject extensively in a chapter about system identification (Chapter 4). During this chapter, two more elaborate aspects arise that have been moved into separate chapters. The first one is the composition of the weights in the Jacobian matrix using different solution methods (Chapter 5), and the other is the appearance of non-linearities of hysteresis-type in the trace data (Chapter 6). In the next chapter, we develop a robot position control system by using the results of the previous chapters to test the usability and performance of the stated system in practice (Chapter 7). A closing chapter follows, in which we summarize the results, draw some conclusions, point out special contributions that have been made and mention ideas for a continuation of the work (Chapter 8).

2. INDUSTRIAL ROBOTS

Visual servoing involves using sensor information to control a manipulator or industrial robot. In this chapter, we take a closer look at the object of our control attempts by defining the term "robot" and by learning about its kinematics. Furthermore, we introduce a method for robot position interpolation and the effect of hysteresis before we see what kind of communication interfaces a modern industrial robot provides to accept external position corrections.

2.1. Definition

The international standard ISO 8373 (see [40]) defines a robot as:

> An automatically controlled, reprogrammable, multi-purpose manipulator, programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications.

A more geometrically oriented definition describes the robot as a chain of $l + 1$ rigid objects, called *links*, connected to each other with l *joints* with $l \in \mathbb{N}$. The joints have a translational or rotatory degree of freedom. The status of a single joint (length or angle) can be described by a value $\theta_i \in \mathbb{R}$, the vector

$$\theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_l \end{pmatrix} \in \mathbb{R}^l \tag{2.1}$$

represents the configuration of all l joints of the robot. The first link of the robot is usually fixed[1] and is therefore called the robot's *base* with the attached base coordinate system. The tool is connected to the last link, which is called an *end-effector* with the adherent tool coordinate system and its point of origin, the *tool center point* (*TCP*, see Figure 2.1).

The position and orientation of the end-effector in respect of the base coordinate system can be expressed with a transformation. Depending on the robot's geometry, this transformation contains $m \in \mathbb{N}, m \leq 6$ degrees of freedom up to a maximum of six

$$\mathbf{r} = \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} \in \mathbb{R}^m \tag{2.2}$$

where each single element contains either a translational degree of freedom (*DOF*) (X, Y, Z) or a rotatory one (Roll, Pitch, Yaw[2]).

[1] Exception: Robot applications in space

[2] Possible alternatives are Euler angles

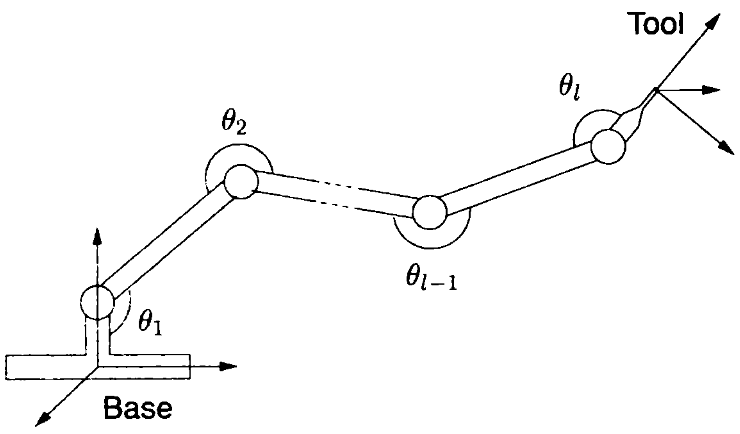


Figure 2.1: Geometrical Robot Model

2.2. Robot Kinematics

Because the robot is actuated in the joint space and the end-effector is moved in the Cartesian space of our physical world, it is necessary to be able to map a joint configuration to the end-effector position and vice versa. Thus, in practice there are two problematic areas of interest when controlling the robot's (see Figure 2.2):

1. What is the transformation $\mathbf{r}$ from the base coordinate system to the tool coordinate system when the joint configuration θ is given?
2. Which joint configuration θ has to be chosen to realize a certain transformation $\mathbf{r}$ from base to tool?

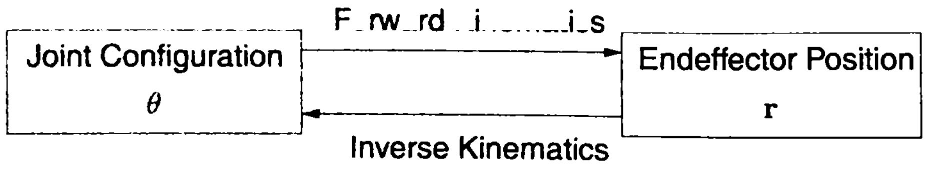


Figure 2.2: Robot Kinematics

The first question relates to *forward kinematics*. This can be expressed mathematically by the search for the function

$$f : \theta \longrightarrow \mathbf{r}. \tag{2.3}$$

Because every joint configuration θ realizes exactly one transformation $\mathbf{r}$, f is a mathematical function. With a known geometry of the links and joints, it is quite easy to find a closed solution for f by determining the chain of transformations from link to link. A classic method to do this in a standardized procedure is the *Denavit-Hartenberg* method, which can be found in basic robotics literature like [2] or [20]. A typical representation of f is a homogeneous matrix whose elements are dependent on the l joint configurations.

The second question addresses the *inverse kinematics* of the robot. Its solution requires the reversibility of f or the existence of

$$f^{-1} : \mathbf{r} \longrightarrow \theta. \tag{2.4}$$

The formal mathematical requirement for the invertibility of f is bijectivity. If the number of joints l is greater than the number of controlled DOF m, there might be the possibility of multiple joint configurations leading to the same transformation. Figure 2.3 (a) shows two different joint configurations realizing the same transformation $\mathbf{r}$. So, f does not necessarily have to be injective. On the other hand, f is not surjective as the transformation cannot be realized with any joint configuration because of the limited working area of the robot (see Figure 2.3 (b) for an example). Therefore, the existence of f^{-1} is bound to certain constraints that make its computation difficult. Other practical constraints are the physical dimensions of the links required to check if the robot is colliding with itself, or the user definition of work spaces the robot is not allowed to leave. It is quite common to solve inverse kinematics using numerical methods.

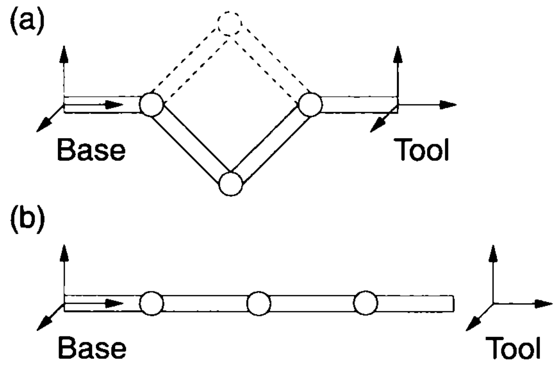


Figure 2.3: Ambiguous and Unsolvable Joint Configurations

This problem is not limited to the field of robotics. It occurs in computer graphics when modelling rigid bodies, in medicine when examining the forces and torques occurring in muscles moving limbs, or in molecular biology when calculating the possible structures of proteins (*loop closure problems*). For this reason, extensive research has been carried out. A good introduction to the inverse kinematics in robotics is provided by [11] and [12]. A special solution for inverse kinematics based on a matrix transposition is discussed in [22]. The book [86] provides a wide theoretical overview over the inverse problem theory and [24] describes a supervised learning approach to the inverse kinematics problem. A basic work on the application of inverse kinematics in computer graphics can be found under [96], which is based on the paper [94] and a more recent work based on a learning approach under [31]. For examples on the works related to neurobiology and medicine, see [41] and [83].

2.3. Robot Position Interpolation

The subject of robot position interpolation solves the problem of having two robot positions $\mathbf{r}_1$ and $\mathbf{r}_2$ - usually these are the position the robot is currently in and a target position - and wanting to find intermediate robot positions between these positions. This is important when establishing a path planning system or only realizing a simple proportional controller. The solution that comes first to mind - the component-wise interpolation of the position vectors $\mathbf{r}_1$ and $\mathbf{r}_2$

$$\mathbf{r}(u) = (1-u)\,\mathbf{r}_1 + u\,\mathbf{r}_2 \tag{2.5}$$

with a $u \in \mathbb{R}, 0 \leq u \leq 1$ - leads in practice to problems such as non-natural and non-smooth movements. While translations can be interpolated component-wise as shown above, rotations expressed in RPY angles or Euler angles should not be interpolated in that way.

A solution for the interpolation of rotations offers the so-called *SLERPs*[3] (see [85] as the original source or for further information [21]). They use a description of the rotations in the form of *quaternions*. Quaternions were originally invented as the skew field $\mathbb{H} = \mathbb{C}^2 = \mathbb{R}^4$ in the field of abstract algebra. Later, unit-quaternions were discovered as a way to express rotations. For a good introduction to the field of quaternion mathematics, see [45]. SLERPs ensure a natural and smooth interpolation between two positions that differ in a rotation.

When using SLERPS, we have to convert the rotatory components of $\mathbf{r}_1$ and $\mathbf{r}_2$ into the two quaternions $\mathbf{q}_1$ and $\mathbf{q}_2$. The interpolated quaternion for an interpolation factor $u \in \mathbb{R}, 0 \leq u \leq 1$ is then given by

$$\mathbf{q}(u) = \frac{\sin(1-u)\varphi}{\sin\varphi}\mathbf{q}_1 + \frac{\sin u\varphi}{\sin\varphi}\mathbf{q}_2 \tag{2.6}$$

with $\cos(\varphi) = \mathbf{q}_1 \cdot \mathbf{q}_2$. For $u = 0$ we get $\mathbf{q} = \mathbf{q}_1$ and for $u = 1$ the result is $\mathbf{q} = \mathbf{q}_2$. The resulting $\mathbf{q}(u)$ has to be converted back to the previous representation and merged with the translational component that has been interpolated using (2.5) with the same interpolation factor u.

2.4. Robot Hysteresis

So far, we have treated the robot as a system of ideal geometrical objects arranged for a flexible positioning of the end-effector. However, in reality there are multiple physical effects, mostly of non-linearity types, that affect the mechanical properties of the robot and therefore reduce its overall accuracy. Here is a list of the most important influences:

- Production tolerances of the mechanical parts
- Link and joint resilience
- Gear and bearing slackness
- Frictional effects
- Variations in temperature

It is fairly difficult to take care of these effects. After production, industrial robots are subject to a complex testing and measuring procedure to determine the actual robot parameters and to compensate the production tolerances by optimizing the actual robot parameters as model parameters in a model-based compensation approach. Some effects are much more difficult to address and might require robot hardware modifications that usually lead to higher production costs of the robot.

Now, we will take a closer look at an effect that is of some importance in the further course of this thesis. This is mainly caused by gear slackness and causes hysteresis-like non-linearities. For a small experiment, we suppose that we are able to measure the position of the robot with some external measurement system of high accuracy and high reliability. We move the robot in a single DOF, forward and backward in a closed loop. When comparing the robot position obtained from the robot controller with the robot position determined by the measurement system, we see a divergence of hysteresis-type. This is illustrated using simulated data by Figure 2.4.

[3]SLERP=Spherical Linear Interpolation

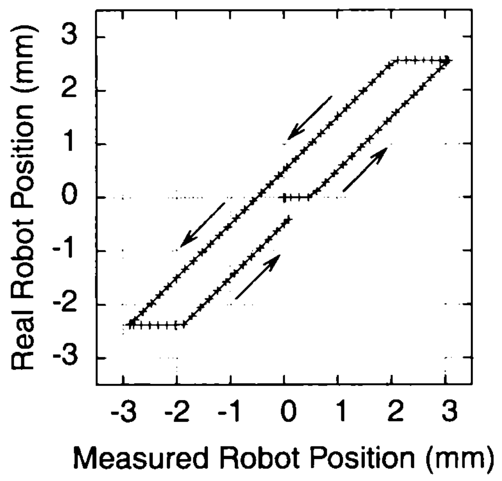


Figure 2.4: Mechanical Robot Hysteresis

This effect is made worse by a certain practice that is especially common for industrial robots. As you can see in Figure 2.5, in industrial robots the incremental resolver - responsible for the joint position determination - is usually mounted on the motor-side of the gears as shown in a) and not on the joint-side of the gears as in (b). This gives the advantage of having a higher resolution for the robot position because the gear ratio increases the resolution of the incremental resolver, and it is possible to use cheaper incremental resolvers with lower resolution. The downside, however, is that this leads to enhanced hysteresis effects because the joint position is measured indirectly by including certain gear effects.

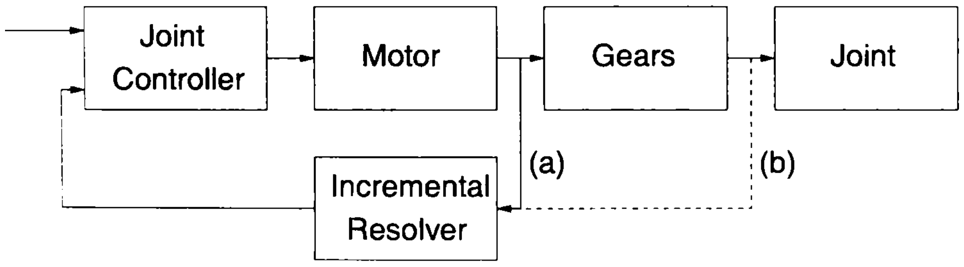


Figure 2.5: Incremental Resolver in Industrial Robots

In the following, we see the results of a real world experiment for estimating the dimension of the hysteresis effect. We define a target position for the robot and move the robot to this target position; once from the positive and once from the negative direction of a certain DOF. Every time we reach the robot's stop position, we measure the distance of the target position with an external measurement system. With this procedure, we get two measurements for one single cycle. Figure 2.6 shows the results after 1000 cycles. The absolute positioning error reflects the repeating accuracy of the robot and is here below one tenth of a millimeter. The distance between the average of the measurements from the positive and negative directions is over five hundredths of a millimeter and represents our mechanical robot hysteresis.

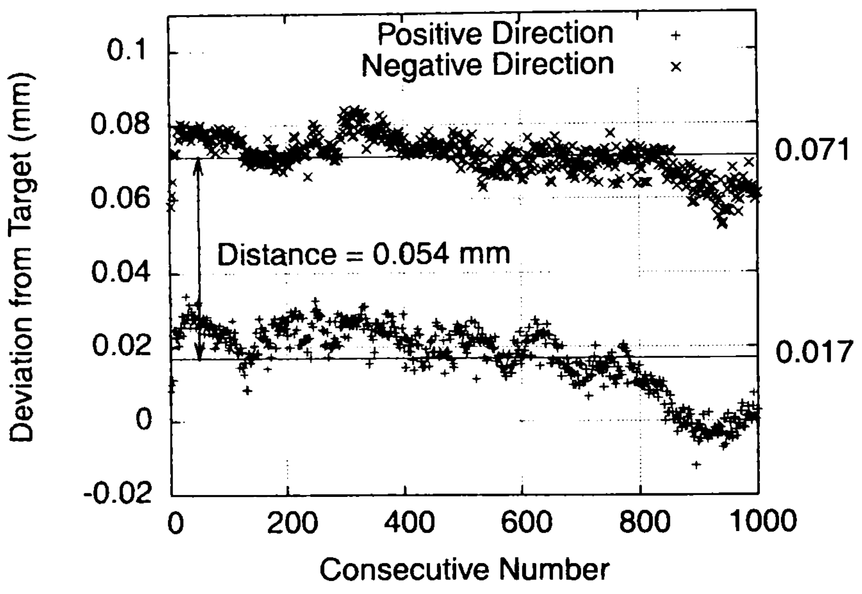


Figure 2.6: Reaching Target from Different Directions

The results of this experiment are bound to a certain robot and depend heavily on the absolute target position. Therefore, although they are not generally valid, they should give us an idea about the dimension of the hysteresis effect that we might expect.

2.5. Robot Control

Robot controllers are usually embedded into an industrial environment with the requirement to communicate with a variety of other components such as programmable logic controllers that are responsible for the central operation control, or with external measurement systems. As a consequence, robot controllers have to support a variety of different communication interfaces and communication protocols such as common industrial bus systems (Interbus, Profibus, Profinet). We distinguish the communication interface by its ability to directly control the robot in real time or by providing control values to the robot's commando interpreter and controlling the robot indirectly.

2.5.1. Real Time Control

Real time robot control requires low level access to the robot controller. The robot controller of industrial robots contains a path planning module that performs the planning of the robot path between two path points over the time. Part of this is the interpolator that calculates the intermediate points of the robot's path, and this is where the real time control takes place by accepting external robot target positions or by overlaying the robot's movement with external corrections. The cycle time of the interpolator determines the maximum frequency at which external corrections can be processed. The cycle time of the interpolator of today's KUKA robot controllers, for example, is 12 ms.

Today, only a few robot controllers for industrial robots support low level access to the control core and interpolator. Examples of this are the KRC controllers for KUKA robots. The interface is known as the *Robot Sensor Interface* (*RSI*, see [48]) and is a real time interface for external sensors. It is an object oriented extension of the *KUKA Robot Language* (*KRL*) and consists of a toolbox for accessing sensors that are connected with the I/O components of the robot controller and realizes basic controllers within the KRL. A further extension of the RSI, developed by the company Amatec Corporation (see [46]), allows for access to the robot interface via external computer systems' Ethernet, and is the way in which the robot can be controlled from an outside computer system: The robot program moves the robot into a starting position and passes control to the computer system - this returns control to the robot after finishing its task.

The RSI/Corob interface establishes a TCP/IP connection between the robot controller and computer system. Every $12\,\mathrm{ms}$, the robot sends a data packet in XML format containing information about the current Cartesian and joint positions, positions of the external axes, motor currents and the status of a set of I/O variables. The computer system replies with a packet containing Cartesian corrections, joint corrections and corrections for external axes as well as its own set of I/O variables. Since the Ethernet communication has no real time capabilities and there are certain wider time limits, the robot requires the acknowledgement of its packets by the reply of the receiver.

A technology review of the flexible sensor interface RSI for KUKA robots has been published in [9], which describes the development of the interface by analysing the industrial requirements for an open control system. A similar technology to the KUKA/RSI for robots from ABB is discussed in the article [7]. The authors describe the ABB S4CPlus controller as a fast and flexible real time sensor interface that tries to offer access to the robot's control core while taking security and performance issues into consideration.

2.5.2. Non-Real Time Control

If no real time control for the robot is available, it remains to control the robot by transmitting corrections via one of the available communication interfaces. These corrections are received by the robot's command line interpreter and can be processed with the robot's programming language. An example of the serial communication interface that is used in KUKA robots is the industrial serial communication protocol 3964/3964R. The KUKA robot language KRL (see [47]) is extended by commands for opening or closing a connection (`COPEN`/`CCLOSE`) and for receiving and sending data (`CREAD`/`CWRITE`) (see [49]). For the development of a complete robot control system, it is necessary to define certain message datagrams for sequential control, correction transmission and error signalling. The non-real time control of the robot via standard communications and the usage of the robot's programming language is the most versatile solution. One drawback is the slower speed of the control process because it requires a stricter synchronization of measurement and the robot's movements (synchronous system, see Chapter 7.1), as no intermediate robot positions are available while the robot is moving due to the last transmitted correction.

3. SENSORS

The sources of information used to control the robot in visual servoing are sensors. This chapter introduces some of today's important sensor technologies for robot control - Laser distance sensors, laser stripe sensors and cameras. Since most sensors provide information of greater complexity and each type of sensor technology provides different kinds of information, we learn in the following sections about a concept that explains how to use sensors as abstract sources of information to make it easier to fuse different sensor technologies in a single application and to make different sensor technologies more interchangeable.

3.1. Sensor Hardware

3.1.1. Laser Distance Sensors

Optical distance sensors such as *laser distance sensors* for measuring small distances with high accuracy in robot control applications use the *triangulation* as a measurement principle. Figure 3.1 shows an overview of the sensor components and the measurement principle:

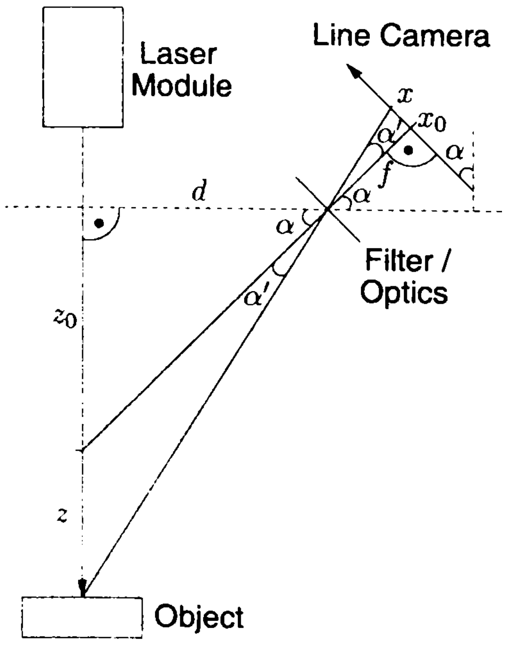


Figure 3.1: Laser Triangulation Principle

Triangulation sensors use a laser module to emit a laser beam, which is diffuse reflected back to the sensor. The returning laser light passes through a filter (to suppress light from other light sources that have different wavelengths to the laser light of the sensor) and one or more lenses, which finally project the light onto a line camera. The angle under which the laser light returns to the sensor depends on the distance between the object and the sensor. This angle is detected by determining the position where the laser light hits the line camera. The mathematical coherence between the distance z and the position of the maximum brightness on the camera chip can easily be derived. In the major triangle we get

$$\tan(\alpha+\alpha') = \frac{z_0+z}{d} = \frac{\tan\alpha+\tan\alpha'}{1-\tan\alpha\cdot\tan\alpha'} \tag{3.1}$$

and in the minor triangle we get

$$\tan\alpha' = \frac{x}{f}. \tag{3.2}$$

Combining both, we get the distance z depending on the deviation x on the line camera chip

$$z(x) = d\cdot\frac{f\cdot\tan\alpha+x}{f-x\cdot\tan\alpha} - z_0. \tag{3.3}$$

Because of the geometrical influences in the form of d and f, the adjustment of the sensor components should be kept fixed. Therefore, laser triangulation sensors are usually compact modules with all of the components encapsulated in a compact case, ready to be used in rough industrial environments. An example is shown in Figure 3.2: The sensor (Micro-Epsilon "optoMCDT 1401-10") for a measurement range of $20\ldots30\,\mathrm{mm}$ is not much bigger than a matchbox.

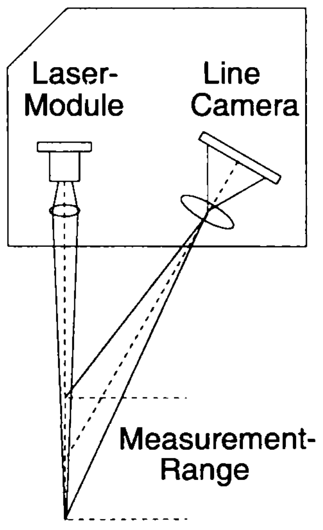


Figure 3.2: Laser Triangulation Sensor

The laser modules are usually semiconductor laser modules with a power of about $1\,\mathrm{mW}$ (eye-safe) and with a visible light that has a wavelength of about $670\,\mathrm{nm}$ (red color) for easy adjustment. The laser usually runs in pulsed mode. Most sensors have an automatic exposure time control (timing of laser pulses) to be able to cope with the different reflective behaviors and colors of the measurement objects. Internal electronics convert the deviation on the line camera into an analog signal whose value is linear to the measured distance or directly into a digital distance value provided to the user by standard interfaces (for example, serial interfaces). The measurement rate is usually around $1\ldots10\,\mathrm{kHz}$ but can go up to $500\,\mathrm{kHz}$. Table 3.1 shows the technical

Range	Frequency	Resolution	Linearity
2 mm (24 ... 26 mm)	2.5 kHz	$\pm 0.1\,\mu m$	$\pm 2\,\mu m$ ($\pm 0.1\%$)
20 mm (40 ... 60 mm)	2.5 kHz	$\pm 1.5\,\mu m$	$\pm 16\,\mu m$ ($\pm 0.08\%$)
200 mm (70 ... 270 mm)	2.5 kHz	$\pm 12\,\mu m$	$\pm 200\,\mu m$ ($\pm 0.1\%$)
750 mm (200 ... 950 mm)	2.5 kHz	$\pm 50\,\mu m$	$\pm 750\,\mu m$ ($\pm 0.1\%$)

Table 3.1: Common Laser Triangulation Sensors

specifications of typical laser distance sensors.

Even though laser triangulation sensors can cope with a wide variety of object surfaces, highly reflective materials can be problematic, as well as transitions of the sensor's laser light between surface regions with very different optical properties such as paint films. When measuring the distance to an object with a very jagged surface, some shadowing effects may occur - the laser point projected on the object cannot be detected by the sensor's camera line. The sensor should be aligned in a way that the laser beam stands normally on the object's surface. Tipping the sensor up to 15° from this position usually does not affect the quality of the measurement.

One basic problem with triangulation sensors is that with the growing measurement distance $z_0 + z$, the distance between the laser module and camera d has to be increased too. This has to be done to maintain a measurement with acceptable accuracy - if α gets nearer to 90° we get numerical problems with $\tan\alpha$ in (3.3), leading to a bad measurement accuracy. Increasing d leads to more space-consuming sensor solutions for higher measurement distances. To avoid this, sensors for the measurement of the distances above 1 m usually use the *time-of-flight* or *phase difference* measurement principles (see [6]), but due to the decreased measurement accuracy these sensors do not have many applications in connection with control systems for industrial robots.

3.1.2. Laser Stripe Sensors

The triangulation principle is also the measurement principle of *laser stripe sensors*. However, the laser projects no other point apart from a line onto the object and the line camera is replaced by a matrix camera. This way, the measurement dimension of the sensor module is increased by one and it is possible to simultaneously retrieve distance information for multiple points along the laser line. The measurement area is a trapezoid in the plane of the laser stripe projection, an example of which is given in Figure 3.3. The laser stripe is projected onto a step-like structure and the sensor provides information about a set of measurement points along the laser line; often called a *scan*. Each point has two assigned coordinates and an intensity value.

Although it is possible to set up a laser stripe sensor using single components - a laser module and a matrix camera - it is advisable to use ready-mounted systems due to the difficult alignment and calibration of the components and their sensitivity to external mechanical influences. Multiple manufacturers offer compact and ready-to-use solutions for industrial applications that contain a hardware-accelerated camera evaluation providing the user with linearized metrical measurement data via standard interfaces (Firewire, Ethernet). Figure 3.4 shows such a laser stripe sensor module (MEL "M2D"). Through the right window, the laser module emits the laser stripe, and the camera is located behind the left window. The total width of the sensor module is 140 mm. To reduce the size and weight of the sensor head, the data processing unit is located in a separate module.

Table 3.2 shows the properties of laser stripe sensors from two different manufacturers. The X coordinate describes the position of a measurement point along the laser line. Because

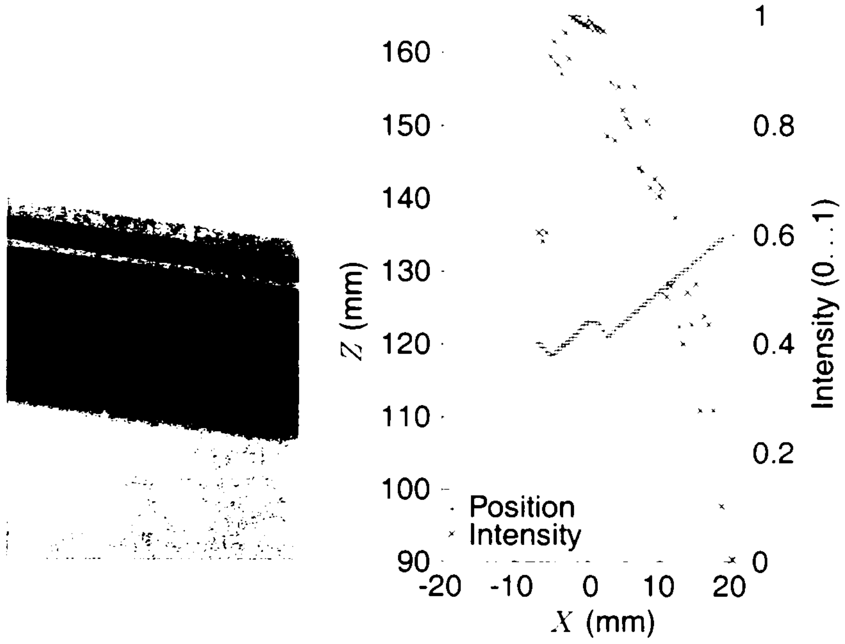


Figure 3.3: Laser Stripe Sensor Scan

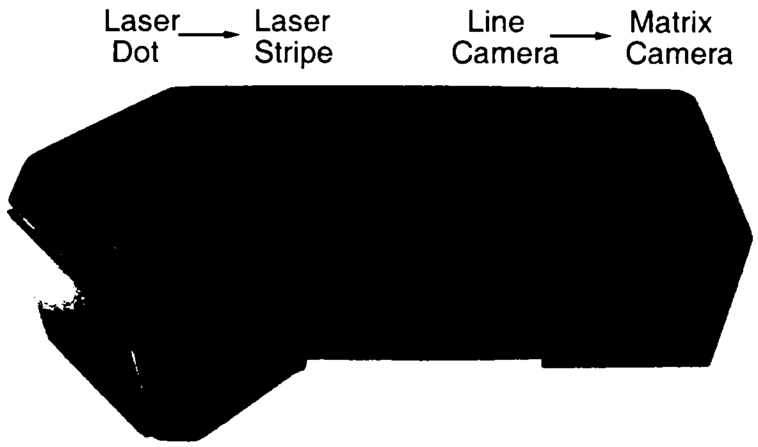


Figure 3.4: Laser Stripe Sensor

of the trapezoid measurement area of the sensor, the range for X is given twice in the table - the smaller value for the nearer side of the trapezoid and the greater value for the further one (dependent on the aperture angle of the emitted laser line). The sensors of both manufacturers have two operation modes with a different resolution in X. The Z coordinate represents the distance between a measurement point on the laser line and the sensor. Its measurement range starts at a basic distance from the sensor. Details about the linearity for the sensors are only known by Micro Epsilon. The frequency depends on the resolution mode chosen for the X coordinate. A doubled resolution in X halves the possible measurement frequency. All sensor modules work with visible laser light - in MEL sensors with $1\,\mathrm{mW}$ power, and in the Micro Epsilon sensor with $15\,\mathrm{mW}$ power. The higher the measurement frequency is, the shorter the camera exposure time has to be. A shorter exposure time demands an increased intensity of the laser line. All sensor modules have an automatic exposure time control to adapt the exposure time to different distances to the measurement object and different object surface properties.

	MEL M2D 7/6	MEL M2D 120/54	MEL M2D 1200/600	Micro Epsilon LLT2800-100
Range in X (mm)	$6/7$	$54/54$	$600/600$	$30/50$
X Resolution (points)	$283/566$	$283/566$	$283/566$	$256/512$
X Resolution (mm)	$0.015/$ 0.03	$0.25/0.5$	$1.00/2.00$	$0.16/0.08$
X Linearity (mm)	-	-	-	$\pm0.12 - \pm0.2$
Range in Z (mm)	7 $(20 - 27)$	120 $(110 - 230)$	1200 $(900 - 2100)$	100 $(145 - 245)$
Z Resolution (mm)	0.015	0.25	1.50	0.04
Z Linearity (mm)	-	-	-	±0.2
Frequency (Hz)	$50/110$	$50/110$	$50/110$	$500/1000$

Table 3.2: Common Laser Stripe Sensors

The limits of the laser stripe sensors are pretty much the same as for the laser triangulation sensors. To deal with different surface properties, most sensor modules contain an automatic exposure time control for the laser beam, but to be able to scan objects with different surface colors of high contrast, the exposure time can be set manually too. The problem of shadowing exists for laser stripe sensors too; when the camera is unable to see parts of the laser stripe because it is hidden behind some protruding parts of the object. This is the first problem when using laser stripe sensors in connection with an external axis in quality assurance applications (see for example [8]). To reduce this problem, special sensor models are available with one laser module and two cameras that observe the projected laser line on the measurement object from opposite directions.

3.1.3. Cameras

The model for the ideal camera is the *pinhole camera*. It uses an infinite small hole in an infinite thin material to project light onto a projection plane with a light-sensitive sensor element. Because of the low light sensitivity of the pinhole camera, real cameras use one or more lenses for projection. All the aberrations and internal properties of the camera system are concluded in the *internal camera parameters*, and the camera position is defined as *external camera parameters*. Some of these parameters are important when using the camera as a 3D measurement system. The literature shows a great variety of methods to estimate internal and external camera parameters - some of the most important being [4], [5], [53] and [92]. Which parameters are important depends on the particular application. For our application, we require a linear coherence between distances in the camera picture and physical distances in the real world. Knowledge of the internal or external camera parameters is not necessary. It is enough to have a scaling factor to convert pixel distances into millimeters for easier conception and the rest is done by the system's self-learning capability.

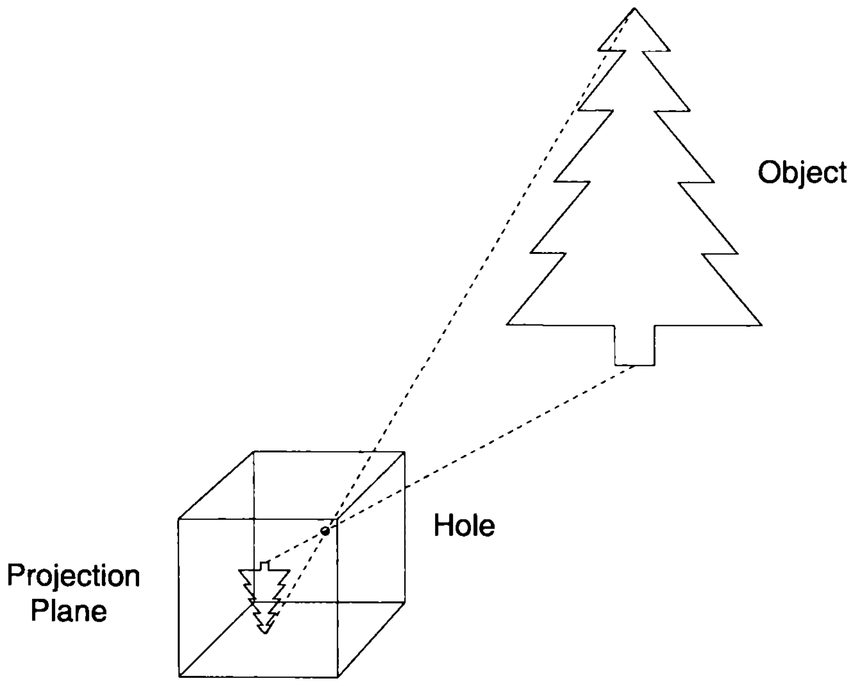


Figure 3.5: Pinhole Camera Schematic

The two standard digital camera sensor technologies today are the *CCD*[1] and *CMOS*[2]/*APS*[3] sensors. CCD chips provide the higher density of photo-sensitive elements on the chips' surface, but CMOS chips have the advantage of a more flexible evaluation (exposure and evaluation at the same time and pre-evaluation in parallel per sensor element) at much higher frame rates.

The classical, and at present most relevant, communication interface for cameras is still

[1]CCD=Charge-Coupled Device
[2]CMOS=Complementary Metal Oxide Semiconductor
[3]APS=Active Pixel Sensors

659. 976

the analogous transmission of the pixel data as a *video signal.* After transmission, the video signal is re-digitalized by special hardware called *frame grabber.* However, because of the decreasing analog signal quality due to electromagnetic interferences in rough industrial environments and more expensive hardware (framegrabber), modern cameras offer digital interfaces like Firewire (IEEE 1394) or the more important Gigabit-Ethernet (GigE) that are capable of handling the high data rates of high resolution cameras at high frame rates. An important current standard interface for Gigabit-Ethernet cameras is the *GigE Vision* standard.

A rather new type of camera is the so-called "intelligent camera". Besides the actual camera hardware, which consists of optics and a camera chip, these cameras contain custom hardware with FPGA or DSP capabilities to execute some basic picture processing algorithms. With cameras like this, it is, for example, easy to execute the task of finding the center point of a hole; visible as a black circle in the camera picture. The advantages of intelligent cameras are their evaluation speed, their compact design and their price compared to computer-based solutions. On the downside, the limited user interface prevents the application of complex picture processing tasks with extensive parameterization and more complicated setup and debugging requirements.

3.2. Sensor Abstraction

The measurements of a single laser distance sensor may be used directly to control the position of a robot. That said, most of the sensors introduced in the previous section provide more complex sensor raw data that cannot be used directly for robot control, like a monochrome camera. picture of a complex scene or a set of distance points from a laser stripe sensor. In these cases, we need to have a further evaluation of the sensor's raw data to extract information that is suitable for robot control. The output of the evaluation can be seen as some sort of sensor information itself. This raises the idea of having abstract information sources, called *sensor signals*, that provide numerical geometrical information about a certain feature independent from its source, a sensor hardware device or the result of an evaluation of more complex data. Figure 3.6 illustrates the concept of the abstraction of sensor data.

In essence, a sensor device is a piece of hardware. On the top we have a laser distance sensor, a laser stripe sensor and a camera. After executing a measurement, all these sensors provide sensor information in the form of raw data. For sensors with more complex raw data, we have to apply different evaluation methods to extract further information. In the case of the laser stripe sensor data, we determine the position of an edge in the sensor scan as well as the angle of the base line. For one of the camera pictures we determine the position of a screw nut in the picture. On the other camera picture we use a different evaluation to measure the distances of two edges of metal sheets using camera coordinates. All of this information describes simple geometrical information that is ready to use for robot control purposes. The concept of sensor signals allows for the handling of different kinds of sensors for robot control - the basis for multi-sensor fusion applications. Sensor signal values should be in the unit of millimeters or degrees to simplify conception when used in connection with robot movements, but this is not obligatory.

The mounting position of the sensors may vary. The only requirement is that the recorded sensor information leads to sensor signals, which change if the robot is moved, and that we are able to correlate the robot's positions with the control signal information. The sensors' mounting positions may be fixed, but it is quite applicable to mount the sensors as extensions of the robot tool. The tool extension together with the sensors is referred to as the *sensor tree.*

An important distinction of sensor signals is the question of whether the sensor signal values depend on the mounting position of the sensor that provides the raw data for the sensor

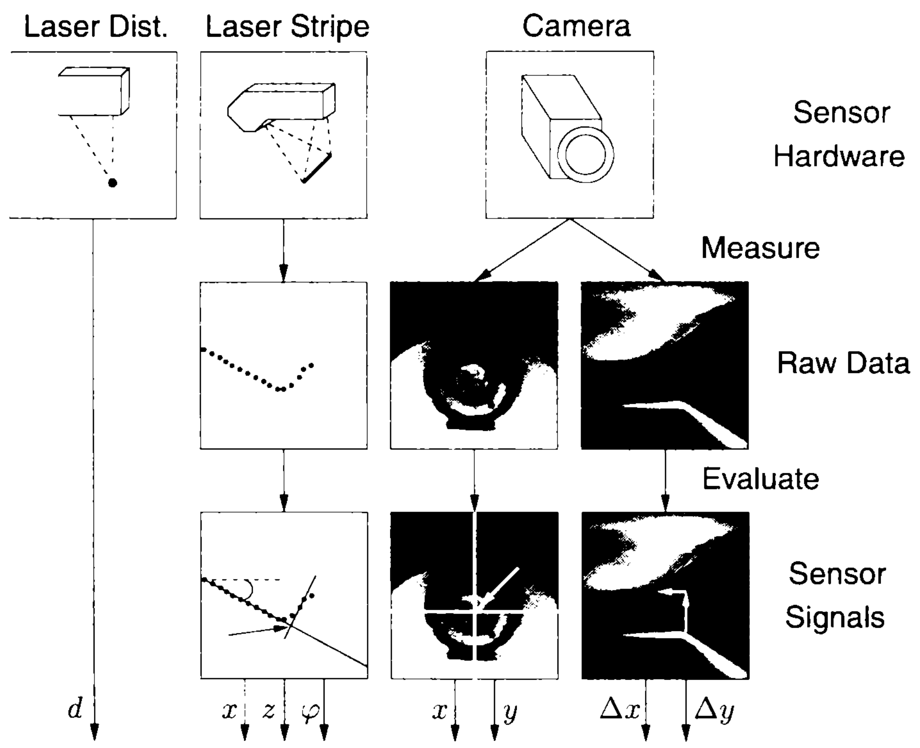


Figure 3.6: Sensor Abstraction

signal. An *absolute measuring* signal measures geometrical features relative to the sensor's coordinate system, such as the position of the screw nut in the camera picture or the relation of both edges to one another in the example above (the orientation of the coordinate system in which the distances are measured depends on the camera mounting position). A *relative measuring* signal measures geometrical features independent from any sensor's coordinate system. If we would define the relationship of both edges in a coordinate system that is fixed on one of the edges, camera relocation would not affect the measured distances. The question as to whether a signal is absolute or relative measuring is interesting when a sensor has to be replaced, for example, in the case of a sensor malfunction. After exchanging an absolute measuring sensor, the system has to be re-identified or, at least, the signal reference values have to be updated. This is not necessary for relative measuring signals.

In the literature, the abstraction of sensor information in connection with visual servoing is quite common. An example work that proposes a concept for a flexible multi-sensor fusion and integration can be found in [25]. Other works on sensor data fusion can be found under [10] and [32].

3.3. Control Signals

Instead of using the previously introduced sensor signals directly for robot control, we add another layer of abstraction by defining the *control signals.* These control signals allow us a more flexible usage of the sensor information for solving problems with additional requirements such as the conservation of symmetries. We consider having some sensors that provide us with a number of m sensor signals h_i with $i \in [1 \dots m]$. Using these sensor signals, we define a number of n control signals s_j with $j \in [1 \dots n]$ as a linear combination of a number of sensor signals. We do this by writing the sensor signals and control signals as vectors $\mathbf{h}$ and $\mathbf{s}$ and by using a matrix $\mathbf{C}$ of the size $n \times m$ to realize the desired transformation:

$$\mathbf{s} = \mathbf{C}\,\mathbf{h}. \tag{3.4}$$

Each row of $\mathbf{C}$ contains the weights for the j-th control signal. In the simplest case, each control signal is identical to one sensor signal without any further weighting. This means that $m = n$ and $\mathbf{C}$ are equal to an identity square matrix.

Nevertheless, there are applications that make the definition of more complex control signals interesting, for example, for the symmetric positioning of the robot. We take a look at the situation shown in Figure 3.7. A robot has grabbed some part and we want to control its position relative to the work object for a subsequent assembly. Six sensors provide information about the size of the gaps between the part and work object in the form of six control signal values. The work object should be positioned symmetrically in terms of

$$h_1 \approx h_6 \quad \text{and} \quad h_2 \approx h_5, \tag{3.5}$$

for h_3 and h_4 we have certain nominal values. Of course, we could set up a system with six control signal values like

$$s_1 = h_1,\ s_2 = h_2,\ \dots,\ s_6 = h_6, \tag{3.6}$$

but the following approach with four control signals shows better results

$$s_1 = (h_1 - h_6),\ s_2 = (h_2 - h_5),\ s_3 = h_3,\ s_4 = h_4. \tag{3.7}$$

The differences ensure a far better performance in terms of the symmetric positioning of the part when the part has some forms of deviation. This is because the differences represent directly a measure for the symmetry instead of taking the gap sizes as single values and leaving it to the system identification algorithm to distribute errors on those gaps, as done in the first approach. Therefore, defining the control signals in the second approach is a way to implement the previous knowledge about the desired symmetry into the system.

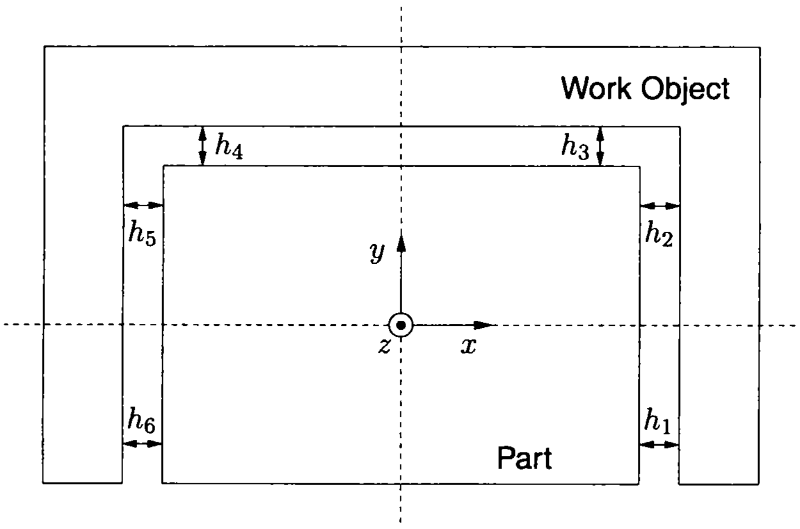


Figure 3.7: Symmetric Assembly using Control Signals

4. SYSTEM IDENTIFICATION

So far, we have learned about the basic components of a visual servoing system - Sensors as the providers of geometrical information and robots as objects for which position has to be controlled. This chapter introduces a method for the identification of robot movements using sensor information. The system identification is based on a learning approach and includes a linearization using a Taylor expansion with a Jacobian matrix as a local derivative. We learn about the basic idea of this system identification and how to collect learning data. After discussing different approaches to estimate the Jacobian, we compare these methods under the practical points of view (quality, flexibility, stability and geometrical considerations).

4.1. Problem Description

Suppose we want to control a robot in Cartesian space in a number of $m \in \mathbb{N},\ m \leq 6$ (translational or rotatory) DOFs. The status of the robot is expressed as a status vector

$$\mathbf{r} = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} \in \mathbb{R}^m \tag{4.1}$$

with the components of $\mathbf{r}$ containing up to three translational and up to three rotatory coordinates in Euler or RPY angles, such as in (2.2).

In special cases it might be in order to directly describe the status of the robot by its link status vector θ like in (2.1). This is first of all the case if the robot controller offers only basic robot control due to its lack of a build-in kinematic solver or in cases where the robot has multiple redundant degrees of freedom that should be taken into consideration by the visual servoing. However, the direct usage of robot links as status description in visual servoing can be problematic for robots with a more complicated kinematic coherence and an application with a tool-centered view: When moving the robot link-wise the visual servoing should be able to judge the resulting movement and speed of the robot's tool, which is not available without a known robot kinematic.

Let $n \in \mathbb{N}$ be the number of control signals. The control signals are abstract data sources that depend somehow on some geometrical features that change when the robot is moved. A snapshot of all the n control signal states can be written as a vector of control signal values

$$\mathbf{s} = \begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix} \in \mathbb{R}^n. \tag{4.2}$$

To control the robot using the control signal data, the knowledge of the vector-valued function

$$f : \begin{cases} \mathbb{R}^n & \longrightarrow & \mathbb{R}^m \\ \mathbf{s} & \longrightarrow & \mathbf{r} \end{cases} \tag{4.3}$$

or written as a set of scalar-valued functions

$$\mathbf{r} = \mathbf{f}(\mathbf{s}) = \begin{cases} r_1 = f_1(\mathbf{s}) \\ \vdots \\ r_m = f_m(\mathbf{s}) \end{cases} \tag{4.4}$$

would be convenient. However, since f is unknown and it is not necessary for our application to have global knowledge about the relationship between control signal values and robot positions, an approximate solution for f in a small area around a working point is sufficient. We denote the working point - or the *nominal robot position* - with $\mathbf{r}_0$ and the control signal values at this point (*nominal control signal values*) with $\mathbf{s}_0$. All absolute values are now expressed in relation to this working point:

$$\Delta\mathbf{r} = \mathbf{r} - \mathbf{r}_0 \tag{4.5}$$

$$\Delta\mathbf{s} = \mathbf{s} - \mathbf{s}_0. \tag{4.6}$$

For system identification, the literature offers a great variety of different identification methods such as analytical ones or approaches using neuronal networks as in [66]. In our approach, the approximation of f at the working point is made with a linearization. Key to this linearization is the Jacobian matrix of the size $n \times m$ containing the partial derivatives of each DOF with respect to each control signal:

$$\mathbf{J} = \begin{pmatrix} \frac{\partial r_1}{\partial s_1} & \frac{\partial r_m}{\partial s_1} \\ \vdots & \vdots \\ \frac{\partial r_1}{\partial s_n} & \frac{\partial r_m}{\partial s_n} \end{pmatrix}. \tag{4.7}$$

So, the i-th column vector of $\mathbf{J}$ with $i \in [1, \ldots, m]$ is the gradient of the scalar field f_i:

$$\operatorname{grad} f_i = \nabla f_i = \begin{pmatrix} \frac{\partial r_i}{\partial s_1} \\ \vdots \\ \frac{\partial r_i}{\partial s_n} \end{pmatrix}. \tag{4.8}$$

We use an analytical approach where we make the approximation with a simple Taylor expansion of s:

$$\mathbf{r} = f(\mathbf{s}) \tag{4.9}$$

$$\approx f(\mathbf{s}_0) + \mathbf{J}^{\mathsf{T}}(\mathbf{s}_0)(\mathbf{s} - \mathbf{s}_0) \tag{4.10}$$

$$= \mathbf{r}_0 + \mathbf{J}^{\mathsf{T}}(\mathbf{s}_0)(\mathbf{s} - \mathbf{s}_0) \tag{4.11}$$

which finally gives

$$\mathbf{r} - \mathbf{r}_0 = \mathbf{J}^{\mathsf{T}}(\mathbf{s}_0)(\mathbf{s} - \mathbf{s}_0) \tag{4.12}$$

$$\Delta\mathbf{r} = \mathbf{J}^{\mathsf{T}}\Delta\mathbf{s}. \tag{4.13}$$

The approximation is adequate if we limit the robot deviation $\Delta\mathbf{r}$ to a few millimeters for translational DOFs and to a few degrees for rotatory DOFs. The known fact that trigonometric functions like $\sin()$ and $\cos()$ can be approximated with linear functions for small angles ensures the required linear behavior, which can be substantiated with a look at real world data later on.

Since there is no unique solution for the Jacobian[1], the actual problem we have to solve is to find an appropriate Jacobian matrix that performs well in terms of our problem. This subject is addressed in the further course of this chapter.

[1] For an in-depth mathematical view on this fact, take a look at the *fixed point theorem* and the *inverse function theorem*.

4.2. Data Collection

To finally identify the correlation between robot movements and control signal changes, we choose the way of letting the system calibrate itself in an automated process - in visual servoing this is called the learning approach. During this self-calibration that we call *training*, we collect robot position samples and control signal samples while moving the robot.

Initially, the robot tool and the work object are both posed into their reference positions in a way that this configuration and setup will be reproduced with every work object during production. Since our approximation uses this working point as a reference, the first step in the training is to obtain a number of control signal samples without moving the robot. The mean values of these samples provide the nominal control signal values $\mathbf{s}_0$, and the statistical analysis of the samples gives an idea about the quality of the sensors, their configuration and the stability of a possibly underlying sensor data evaluation. Definition of the nominal robot position $\mathbf{r}_0$ is attained with the definition of a tool coordinate system which is the frame of reference for all robot movements. Usually, the origin of this tool coordinate system should be located somewhere near the barycenter of all positions where features are measured for the generation of the control signals to ensure an equally distributed change of the control signals upon angular displacement of the robot or of the work object and to avoid unnecessary attenuation of errors.

In the actual training, we move the robot to cause dislocations between the robot tool with adherent sensors and the work object in a controlled way. During production, the robot's start position (nominal robot position $\mathbf{r}_0$) is fixed and it is the work object that varies in position and geometry. The training is separated into m training steps - one for every DOF that should be controlled. For the i-th step with $i \in [1 \dots m]$, we execute a series of movements in the negative and positive direction of the i-th DOF within a certain training range; keeping all other DOFs constant and stopping at the nominal position again. During this movement, the changing robot positions and control signal values are continuously recorded. The robot movements for each DOF are orthogonal in the robot's position vector space to get enough linear independent information for the reconstruction of f in the area around the working points $\mathbf{r}_0$ and $\mathbf{s}_0$ respectively.

The recorded data, known as *trace data*, contains information about the change of the control signals under robot movement. It can be arranged in the form of two matrices

$$\mathbf{R}_i = \begin{pmatrix} \Delta\mathbf{r}_1^\mathrm{T} \\ \vdots \\ \Delta\mathbf{r}_{k_i}^\mathrm{T} \end{pmatrix} \quad \text{and} \quad \mathbf{S}_i = \begin{pmatrix} \Delta\mathbf{s}_1^\mathrm{T} \\ \vdots \\ \Delta\mathbf{s}_{k_i}^\mathrm{T} \end{pmatrix} \tag{4.14}$$

for the i-th training step with $k_i \in \mathbb{N}$ defining the number of collected samples in the i-th training step. $\mathbf{R}_i$ has the size $k_i \times m$ and contains all robot positions relative to the nominal values as row vectors. $\mathbf{S}_i$ has the size $k_i \times n$ and contains all related control signal values relative to the nominal control signal values as row vectors. The row ordering of $\mathbf{R}_i$ and $\mathbf{S}_i$ is free, but the robot position of the j-th row vector of $\mathbf{R}_i$ has to correspond with the control signal values in the j-th row vector of $\mathbf{S}_i$ for all $j \in [1 \dots k_i]$. If available, it might be interesting to save timestamps for every one of the k_i samples to analyze the movement of the robot over time, for example, in connection with hysteresis determination.

Figure 4.1 shows the result of a real example trace in a single translational DOF for three control signals plotted into a diagram. You can see that all signals change linearly when the robot moves in that DOF between $-5\,\mathrm{mm}$ and $+5\,\mathrm{mm}$. Signal 3 shows the most significant change whilst signal 2 remains nearly unchanged.

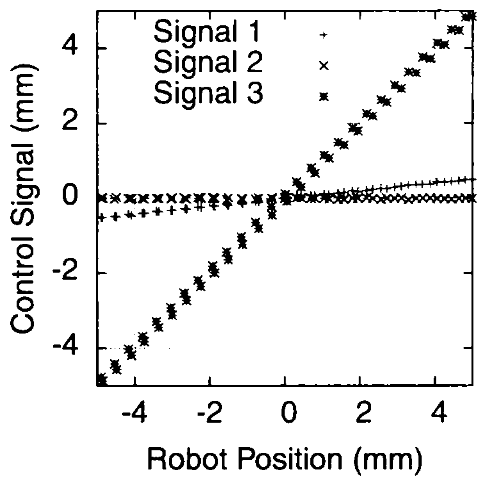


Figure 4.1: Trace Example

In practice, there is a number of things that can go wrong when recording trace data. When the system identification fails, we should take a closer look at the trace data. Figure 4.2 shows the effects of some common problems occurring in practice:

(a) The step-like structure might result from a non-plane surface structure near the measurement position of a distance sensor.

(b) A hole with multiple missing control signal values in the trace can be found if the measurement range of a sensor involved in the generation of the control signal has been reached. This is often the case at the outer borders of the trace but can happen inside the trace data as well.

(c) Intermittent missing control signal values, exceptional noise variations and outliers could be as a result of an improperly chosen or badly configured raw data evaluation.

(d) The saturation-type non-linearity occurs if some collision happens during the trace. Here, the robot movement was perfectly sensed by the control signal from a distance sensor measuring the distance between a fixed work object and a part moving with the robot, until the collision led to the simultaneous movement of both objects with constant distance.

(e) In rare cases, the robot system introduces effects, for example, a repeating saw-like structure as shown in th figure. We have seen effects like this when the robot is outstretched far to reach the nominal robot position and the trace leads to a movement mainly in the first joint. The behavior of the movement of this joint is amplified by the long lever arm of the whole outstretched robot and is reflected in the trace.

(f) A non-linearity of hysteresis-type (overstated in the figure) can be found in nearly every trace; characterized by a gap between the samples of the forward and the backward movement of the robot. Because of its frequent occurrence and that the consequences for the system identification and its stability are serious, the complete Chapter 6 attends to this subject.

The precondition we built our system on is a working sensor subsystem, providing stable and reliable control signals and near-linear coherence between control signals and robot relocations in an area around a working point. So, we must take action to ensure these preconditions

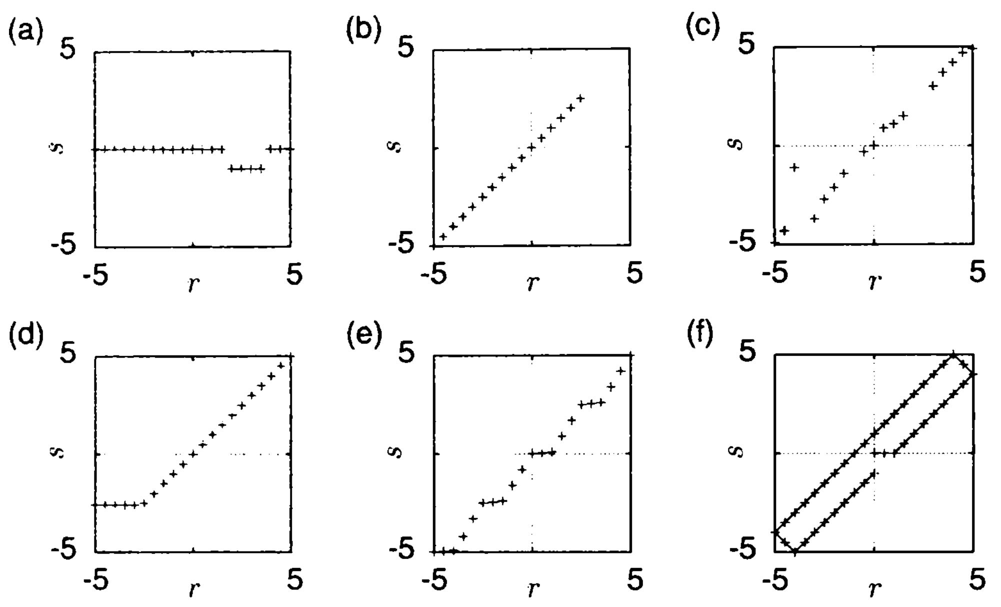


Figure 4.2: Irregularities in Trace Data

by fixing the sensor configuration and evaluation by reducing the training range or changing the nominal robot position. Another way to look at these effects is to state that every sensor has a limited measurement range and if we move the robot far enough away from its nominal position, it is clear that some sensors might leave its measurement range. Consequently, we can take this error effect as a way of automatically detecting the allowed range of the robot - we employ training for an extended training range and automatically determine the linear range in the trace diagram for each control signal with an algorithm. The expected linear range for a DOF is the range that is linearly detected by all control signals.

4.3. Solution Strategies

4.3.1. Direct Solution

For the direct calculation of the Jacobian $\mathbf{J}$, we combine all trace matrices $\mathbf{R}_i$ and $\mathbf{S}_i$ into two matrices

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_1 \\ \vdots \\ \mathbf{R}_m \end{pmatrix} \quad \text{and} \quad \mathbf{S} = \begin{pmatrix} \mathbf{S}_1 \\ \vdots \\ \mathbf{S}_m \end{pmatrix} \tag{4.15}$$

with $i \in [1 \dots m]$. The matrix $\mathbf{R}$ has the size $k \times m$, the matrix S the size $k \times n$ with $k \in \mathbb{N}$ the total number of samples

$$k = \sum_{i=1}^{m} k_i. \tag{4.16}$$

The direct solution now involves the solution of the linear system

$$\mathbf{S}\,\mathbf{J} = \mathbf{R}. \tag{4.17}$$

for $\mathbf{J}$. As, usually $m \leq n \ll k$, the linear system is over-determined and this means we have to try to find the matrix $\mathbf{J}$ that minimizes the sum of the squared distances between the estimated robot positions, which is calculated from the control signal values using the Jacobian and the real robot position

$$\min_{\mathbf{J}} ||\mathbf{S}\,\mathbf{J} - \mathbf{R}||_2^2. \tag{4.18}$$

In other words, we have to search for the *least-squares optimal* solution for $\mathbf{J}$. We can separate the problem into m independent sub-problems by solving m linear systems

$$\mathbf{S}\,\mathbf{j}_i = \mathbf{r}_i. \tag{4.19}$$

for $\mathbf{j}_i$, with $\mathbf{j}_i$ column vector of $\mathbf{J}$, $\mathbf{r}_i$ column vector of $\mathbf{R}$ and $i \in [1 \dots m]$. This leads to the optimization terms

$$\min_{\mathbf{j}_i} ||\mathbf{S}\,\mathbf{j}_i - \mathbf{r}_i||_2^2 \tag{4.20}$$

for each DOF to control.

For solving, we take the equation (4.17) and solve for $\mathbf{J}$. Because $\mathbf{S}$ is not a square matrix, we cannot directly invert it. So we calculate

$$\begin{aligned}
\mathbf{S}\,\mathbf{J} &= \mathbf{R} && (4.21)\\
\mathbf{S}^{\mathsf{T}}\mathbf{S}\,\mathbf{J} &= \mathbf{S}^{\mathsf{T}}\mathbf{R} && (4.22)\\
(\mathbf{S}^{\mathsf{T}}\mathbf{S})^{-1}(\mathbf{S}^{\mathsf{T}}\mathbf{S})\,\mathbf{J} &= (\mathbf{S}^{\mathsf{T}}\mathbf{S})^{-1}\mathbf{S}^{\mathsf{T}}\mathbf{R} && (4.23)\\
\mathbf{J} &= (\mathbf{S}^{\mathsf{T}}\mathbf{S})^{-1}\mathbf{S}^{\mathsf{T}}\mathbf{R} && (4.24)\\
\mathbf{J} &= \mathbf{S}^{+}\mathbf{R} && (4.25)
\end{aligned}$$

The matrix $\mathbf{S}^{+}$ denotes the *pseudo inverse* of $\mathbf{S}$, which is also called *Moore-Penrose Inverse* after its inventors (see [63] and [70] for the original reference).

If the column vectors of a matrix $\mathbf{A}$ are linearly independent, as in the case here, the pseudo-inverse of $\mathbf{A}$ is given by

$$\mathbf{A}^{+} = (\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}, \tag{4.26}$$

if the row vectors of $\mathbf{A}$ are linearly independent, the pseudo-inverse of $\mathbf{A}$ is given by

$$\mathbf{A}^{+} = \mathbf{A}^{\mathsf{T}}(\mathbf{A}\,\mathbf{A}^{\mathsf{T}})^{-1} \tag{4.27}$$

and if both the row and column vectors of $\mathbf{A}$ are linearly independent, the pseudo-inverse of $\mathbf{A}$ is equal to the inverse of $\mathbf{A}$

$$\mathbf{A}^{+} = \mathbf{A}^{-1}. \tag{4.28}$$

Besides this special case, the pseudo-inverse has the following properties as the weak inverse of the multiplicative semi-group of the matrices:

$$\begin{aligned}
\mathbf{A}\,\mathbf{A}^{+}\mathbf{A} &= \mathbf{A} && (4.29)\\
\mathbf{A}^{+}\mathbf{A}\,\mathbf{A}^{+} &= \mathbf{A}^{+}. && (4.30)
\end{aligned}$$

The best method for the numerical calculation of the pseudo-inverse of a matrix $\mathbf{A}$ is via the *singular value decomposition* (*SVD*) of $\mathbf{A}$. The theory of the SVD is an extension of the eigenvalue theory to non-square matrices. It states that for every matrix $\mathbf{A}$ of size $m \times n$ with range r, there exists a SVD

$$\mathbf{A} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^{\mathsf{T}} \tag{4.31}$$

with two square matrices $\mathbf{U}$ and $\mathbf{V}$ of sizes $m \times m$ respectively $n \times n$ and a diagonal matrix $\mathbf{\Sigma}$ of size $m \times n$. The non-zero elements σ_l of $\mathbf{\Sigma}$ with $l \in [1 \dots r]$ are called *singular values* of $\mathbf{A}$. The column vectors $\mathbf{u}_i$ of $\mathbf{U}$ with $i \in [1 \dots m]$ are known as *left-singular vectors*, and the column vectors $\mathbf{v}_j$ of $\mathbf{V}$ with $j \in [1 \dots n]$ are referred to as *right-singular vectors* of $\mathbf{A}$. For further information about the theory of the SVD, see [30]. For methods to calculate the SVD to a given matrix, take a look at [42] and [71].

To a given SVD of a matrix $\mathbf{A}$, the original matrix can be recalculated by

$$\mathbf{A} = \sum_{l=1}^{r} \sigma_l\, \mathbf{u}_l\, \mathbf{v}_l^{\mathsf{T}}, \tag{4.32}$$

the pseudo-inverse of $\mathbf{A}$ can now be determined by

$$\begin{aligned} \mathbf{A}^{+} &= \mathbf{V}\,\mathbf{\Sigma}^{+}\,\mathbf{U}^{\mathsf{T}} && (4.33)\\ &= \sum_{l=1}^{r} \frac{1}{\sigma_l}\, \mathbf{v}_l\, \mathbf{u}_l^{T} && (4.34) \end{aligned}$$

pretty easily.

The most important feature of the direct calculation is that its only optimization criterion is the minimization of the least-squares. Because this criterion is fulfilled - no matter what the cost - this might lead to stability problems. We will further state and investigate the problem in a separate section (see Chapter 4.6).

4.3.2. Inverse of Feature Jacobian

A different approach uses the fact that during the training we move the robot along the coordinate axes of the robot space. As a consequence, we are able to calculate the partial derivatives of each control signal with respect to every robot's DOF. If we want to know the derivative of the i-th control signal with respect to the j-th robot's DOF

$$\frac{\partial s_i}{\partial r_j} \tag{4.35}$$

for $i \in [1 \dots n]$ and $j \in [1 \dots m]$, we use the j-th column vector $\mathbf{r}_j$ of $\mathbf{R}_j$ and the i-th column vector $\mathbf{s}_i$ of $\mathbf{S}_j$. Figure 4.3 shows the result when plotting the k_i values in $\mathbf{r}_j$ against the k_i values in $\mathbf{s}_i$. The slope of the best-fit line represents the required partial derivative.

Furthermore, because we have a partial derivative for every control signal with respect to every DOF, we arrange all the partial derivatives in a matrix of size $m \times n$, known as the *Feature Jacobian*:

$$\mathbf{F} = \begin{pmatrix} \frac{\partial s_1}{\partial r_1} & \frac{\partial s_n}{\partial r_1} \\ \vdots & \vdots \\ \frac{\partial s_1}{\partial r_m} & \frac{\partial s_n}{\partial r_m} \end{pmatrix}. \tag{4.36}$$

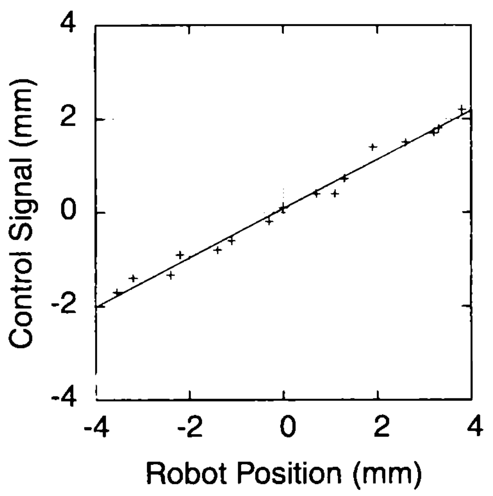


Figure 4.3: Determining the Local Derivative in Trace Data

Because each value in $\mathbf{F}$ represents the sensitivity of a control signal for a certain robot's DOF, it is sometimes called a *sensitivity matrix* or an *interaction matrix* (see [13]).

It is possible to calculate a full row of $\mathbf{F}$ at once - the matrix has as many rows as there are DOFs to control. To calculate the i-th row of $\mathbf{F}$ with $i \in [1 \dots m]$, we need the results from the i-th training step or, more precisely, the i-th column vector of $\mathbf{R}_i$ and the matrix $\mathbf{S}_i$. We get the result by solving the linear system

$$\begin{pmatrix} r_{1,i} \\ \vdots \\ r_{k_i,i} \end{pmatrix} \begin{pmatrix} f_{i,1} & & f_{i,n} \end{pmatrix} = \mathbf{S}_i \tag{4.37}$$

for the required column row of $\mathbf{F}$. Of course, it is possible to calculate $\mathbf{F}$ in one step similar to the Jacobian by solving a linear system after merging all the trace data matrices as in (4.15):

$$\mathbf{F}\,\mathbf{R} = \mathbf{S} \tag{4.38}$$

but this might lead to stability problems as in the direct calculation, which can be avoided.

For the calculation of the Jacobian from the Feature Jacobian, we invert the Feature Jacobian using the pseudo-inverse:

$$\mathbf{J} = \mathbf{F}^{+} = \mathbf{F}^{\mathsf{T}}(\mathbf{F}\,\mathbf{F}^{\mathsf{T}})^{-1}. \tag{4.39}$$

Table 4.1 now shows an overview of the properties of the Jacobian and the Feature Jacobian. The Jacobian is used for the local approximation of f to derive a robot correction to a given control signal deviation. The Feature Jacobian can be used to approximate f^{-1} and can be derived as a direct result from the trace data as the change of control signals caused by a certain robot's movement. The direct calculation determines $\mathbf{J}$ directly and the calculation via the Feature Jacobian inverts the Feature Jacobian.

	Jacobian	Feature Jacobian
Functional correlation	$\mathbf{r} = \mathbf{f}(\mathbf{s})$	$\mathbf{s} = \mathbf{f}^{-1}(\mathbf{r})$
Local solution	$\Delta\mathbf{r} = \mathbf{J}^{\mathsf{T}}\,\Delta\mathbf{s}$	$\Delta\mathbf{s} = \mathbf{F}^{\mathsf{T}}\,\Delta\mathbf{r}$
Relation to trace data	$\mathbf{S}\,\mathbf{J} = \mathbf{R}$ $\mathbf{J} = \mathbf{S}^{+}\,\mathbf{R}$	$\mathbf{R}\,\mathbf{F} = \mathbf{S}$ $\mathbf{F} = \mathbf{R}^{+}\,\mathbf{S}$

Table 4.1: Overview of Jacobian and Feature Jacobian

4.3.3. Inverse of Feature Jacobian with Weighting

It is possible to apply a certain weighting to a matrix before inverting it. We will check out the consequences of applying weighting to the Feature Jacobian before inverting it to get the Jacobian, as introduced in the previous section.

To do the weighting, we use two diagonal matrices as weight matrices. The first weight matrix $\mathbf{M}$ has the size $m \times m$ and is responsible for the row weighting of $\mathbf{F}$, and the second matrix $\mathbf{N}$ has the size $n \times n$ and does the column weighting of $\mathbf{F}$. Because in $\mathbf{F}$ we have one row per DOF, $\mathbf{M}$ is for the weighting of the DOFs. Likewise, $\mathbf{N}$ is for the control signal weighting. The partial derivatives in the Feature Jacobian have the control signal in the enumerator and the DOF in the denominator, as in the (4.35). With this in mind, we arrange $\mathbf{M}$ and $\mathbf{N}$ as:

$$\mathbf{M} = \begin{pmatrix} \mu_1 & 0 & & 0 \\ 0 & \mu_2 & & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & & 0 & \mu_m \end{pmatrix} \quad \mathbf{N} = \begin{pmatrix} \frac{1}{\nu_1} & 0 & & 0 \\ 0 & \frac{1}{\nu_2} & & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & & 0 & \frac{1}{\nu_n} \end{pmatrix}. \tag{4.40}$$

The Jacobian is finally calculated as

$$\mathbf{J} = \mathbf{N}\,(\mathbf{M}\,\mathbf{F}\,\mathbf{N})\,\mathbf{M}^{+}. \tag{4.41}$$

So far, the general theory of matrix weighting has been in connection with inversion. However, we are interested in a special case. When reconstructing the robot movement by analysing a number of control signals, we usually have at least as many control signals as robot DOFs to control - the more control signals, the better. This gives us $m \leq n$. In this case, the application of a column weighting is impossible because the term (4.41) becomes independent from $\mathbf{M}$ for $m \leq n$. Consequently, we can reduce our weighting to a single weight matrix

$$\mathbf{W} = \begin{pmatrix} \frac{1}{w_1} & 0 & & 0 \\ 0 & \frac{1}{w_2} & & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & & 0 & \frac{1}{w_n} \end{pmatrix}. \tag{4.42}$$

for control signal-based weighting, which simplifies the calculation of the Jacobian to

$$\mathbf{J} = \mathbf{W}\,(\mathbf{F}\,\mathbf{W})^{+}. \tag{4.43}$$

It is further interesting to note that for $m = n$, the term (4.43) becomes independent from the weight matrix $\mathbf{W}$ too - leaving us with the simple pseudo-inverse of $\mathbf{F}$. In this case, no redundant control signal information is available and therefore there is no need for weighting.

Weighting allows us to define a certain weight for every control signal in cases where there are more control signals than robot DOFs to control. This makes sense because this way we are able to affect the weighting of the control signals in the Jacobian and therefore influence the amount of information each single control signal that contributes to the reconstruction of the robot movement. This weighting can be determined manually by a user, for example, for completely suppressing a control signal by setting its weight to zero. Or - what is more interesting - the weighting can be done automatically, for example, by evaluation of the noise of the control signals and by suppressing noisy control signals. Chapter 5.2.2.3 makes a suggestion for noise-dependent signal weighting on an analytical basis.

4.4. Quality

A sufficient density of samples, together with the way the training movements are executed, is a precondition for obtaining enough information about the connection of robot movements and control value changes. Nevertheless, a well-conditioned solution requires a control signal configuration that is able to detect changes in every DOF that has to be controlled - if a movement in a certain DOF does not lead to significant control signal changes, this DOF cannot be controlled with this configuration. Assuming we have identified the system after the training by determining the Jacobian from the trace data with any of the solution methods presented previously, it would be beneficial to have some method to get an idea about the quality of the achieved solution or a way to compare two different solutions. So, the quality definition has two purposes:

1. Judgment of the current sensor and robot setup by determining the quality of the recorded trace data with respect to its usability for system identification
2. Evaluation of a certain solution method and comparison with other solution methods

This section suggests some useful tools for providing a quality estimation.

4.4.1. Residual Square Sums

For the direct solution, we have to solve an over-determined linear system. The optimization criterion used here is the least-squares criterion (see (4.18)). With this in mind, for a first quality conception we calculate the error value of least-squares optimization.

The trace data consists of robot deviations $\Delta\mathbf{r}_i$ out of $\mathbf{R}$ with the according control signal deviations $\Delta\mathbf{s}_i$ out of $\mathbf{S}$ with $i \in [1 \ldots k]$. For each pair we use the Jacobian to calculate the estimated robot deviation from the control signal deviation and calculate the difference to the real robot deviation. This can be done with a few basic matrix operations

$$\mathbf{E} = \mathbf{S}\,\mathbf{J} - \mathbf{R}. \tag{4.44}$$

The resulting *residual matrix* or *error matrix* has the size $k \times m$. To get a more compact quality statement, we separately sum up the squares of the residuals for each DOF to get a vector containing the *residual square sums*:

$$\mathbf{e} = \begin{pmatrix} \sum_{i=1}^{k} e_{i,1}^2 \\ \vdots \\ \sum_{i=1}^{k} e_{i,m}^2 \end{pmatrix}. \tag{4.45}$$

Let us take a look at an example. For an example configuration of four DOFs identified with six control signals, we get for the residual square sums

$$\mathbf{e}^{\mathsf{T}} = \begin{pmatrix} 0.689 & 0.360 & 0.022 & 0.152 \end{pmatrix} \tag{4.46}$$

after solving by the direct calculation and

$$\mathbf{e}^{\mathsf{T}} = \begin{pmatrix} 0.979 & 0.713 & 0.023 & 0.413 \end{pmatrix} \tag{4.47}$$

after solving by the Feature Jacobian with certain weighting. The direct calculation is the better one in terms of least-squares; just as expected. Nevertheless, the residual square sums do not allow for comparing the identification of the single DOFs with one another. DOF 4 has a lower residual square sum than DOF 2 but it would be wrong to say that DOF 4 is better recognized, because this DOF is a rotatory one and DOF 2 is a translational one. Another problem is that with a growing trace data set and, therefore, more information about the system to identify, this increases the residual square sum because of the growing sums in (4.45) and decreases the quality, which is hardly desired.

4.4.2. Coefficient of Determination

For our next quality definition we use a concept from the mathematical field of statistics. In the theory of linear regression, we have observed data, an independent variable x_i and a dependent variable y_i. Using a model, we predict values f_i for y_i on the basis of the independent variable x_i. The *coefficient of determination* (*COD*, see for example [78]) is defined by

$$R^2 = 1 - \frac{\sum(y_i - f_i)^2}{\sum(y_i - \bar{y})^2} \tag{4.48}$$

and contains the quotient of the variability of the prediction errors and the variability of the dependent variable. It measures how much of the variability of the dependent variable can be explained by the variability of the independent variable. The COD is a value between zero and one and is an indicator for the quality of the regression model with respect to the observed data. If the COD is near zero, the regression model is unsuited for the observed data and a COD of near one indicates a good adaption by the model.

In our trace, the independent variables are the control signals and the dependent variables the robot deviations. For each DOF, we get a COD by calculating

$$R_i^2 = 1 - \frac{\sum_{j=1}^{k} e_{j,i}^2}{\sum_{j=1}^{k} r_{j,i}^2 - \frac{1}{n}\left(\sum_{j=1}^{k} r_{j,i}\right)^2} \tag{4.49}$$

for $i \in [1 \ldots m]$. The nominator calculates square sums over the i-th column of the residual matrix $\mathbf{E}$ (residual square sum) and the denominator runs over the i-th column of the robot deviation matrix $\mathbf{R}$. Finally, we calculate the product of the CODs of all DOFs

$$R^2 = \prod_{i=1}^{m} R_i^2 \tag{4.50}$$

to get a single quality value.

Referring back to the previous example, we can see here quality in the form of CODs for each DOF and the total quality as a product of these values - here for the direct calculation

$$R^2 = 0.992 \cdot 0.996 \cdot 0.975 \cdot 0.843 = 0.812 \tag{4.51}$$

and here for the calculation via Feature Jacobian with weighting:

$$R^2 = 0.988 \cdot 0.992 \cdot 0.974 \cdot 0.572 = 0.546. \tag{4.52}$$

The CODs allow us to compare the quality of the reconstruction of the single DOFs - the fourth DOF was apparently not as well identified as the other DOFs. The product of the CODs is a good and quick indicator for the quality of a full trace data set. If its value is low, then something went wrong and checking the CODs of the single DOFs helps to reduce the problem to certain DOFs.

4.4.3. Jacobian Condition Number

To a given linear system in the form $\mathbf{A}\,\mathbf{x} = \mathbf{b}$, the *condition number* of the matrix $\mathbf{A}$ gives us an idea of how sensitive the solution $\mathbf{x}$ is against the changes in $\mathbf{A}$ and $\mathbf{b}$. The condition number κ of $\mathbf{A}$ is therefore a relative error attenuation factor and is defined as

$$\kappa(\mathbf{A}) = \frac{\max\limits_{\mathbf{x}} \dfrac{||\mathbf{A}\,\mathbf{x}||_2}{||\mathbf{x}||_2}}{\min\limits_{\mathbf{x}} \dfrac{||\mathbf{A}\,\mathbf{x}||_2}{||\mathbf{x}||_2}} = \frac{\sigma_{\max}}{\sigma_{\min}}. \tag{4.53}$$

The term $||\mathbf{x}||_2$ denotes the L2 norm - or Euclidean norm of vector $\mathbf{x}$, $||\mathbf{A}\,\mathbf{x}||_2$ - the norm of vectors $\mathbf{x}$ transformed with $\mathbf{A}$. The quotient is the relative change of $\mathbf{x}$ after transformation with $\mathbf{A}$. Because the enumerator is greater than or equal to the denominator, κ is always greater than or equal to one. Furthermore, as we use the Euclidean norm for the definition of the condition number, it can also be calculated by the quotient of the highest singular value $\sigma_{\max}$ and the lowest singular value $\sigma_{\min}$ too.

The condition number of a matrix describes the sensitivity of the result of a linear system with the matrix as a coefficient matrix on small changes to the matrix or the right-hand vector of the linear system. Consequently, it can be used to detect ill-conditioned linear systems. A high condition number occurs, for example, when the smallest singular value is very near to zero. Calculating the inverse of this matrix would lead to very high values in the inverse because in (4.33) the singular values occur in the denominator[2]. Compared to this, the concept of the determinant can be used to check if a matrix is invertible, but for the usage in our case it is limited to square matrices and is inferior compared to the condition number. If taking the example of a diagonal square matrix $\mathbf{A}$ of size 10×10 with all diagonal elements equal to 0.1, we get

$$\det(\mathbf{A}) = 1.0 \cdot 10^{-10} \qquad \kappa(\mathbf{A}) = 1 \tag{4.54}$$

The determinant warns of a near-singular matrix, while the condition number signals a well conditioned matrix. The latter is more realistic; after all, $\mathbf{A}\,\mathbf{x}$ just multiplies all elements in $\mathbf{x}$ with 0.1.

The condition number can be determined by using $\mathbf{J}$ or - in the case of calculating $\mathbf{J}$ as the pseudo-inverse of $\mathbf{F}$ and calculating the SVD of $\mathbf{F}$ for the pseudo-inverse anyway - by using $\mathbf{F}$ because

$$\kappa(\mathbf{J}) = \kappa(\mathbf{J}^+) = \kappa(\mathbf{F}). \tag{4.55}$$

[2]A singular value of exact zero is no problem because the index of the sum in (4.33) is limited to the range of the matrix.

If J has $\sigma_{\max}$ and $\sigma_{\min}$ as the highest and lowest singular values, then $\mathbf{J}^+$ has $\frac{1}{\sigma_{\min}}$ and $\frac{1}{\sigma_{\max}}$ as the highest and lowest singular values because of (4.33). In both cases, the condition number κ is the same.

It should be mentioned that there is one problem in connection with the condition number. If we take a look at the matrix

$$\mathbf{F} = \begin{pmatrix} 1 & 0 \\ 0 & 1000 \end{pmatrix} \tag{4.56}$$

and calculate the condition number, we get $\kappa(\mathbf{F}) = 1000$, which is quite high and would indicate an ill-conditioned system. However, the system is well-conditioned - just a little scaling away from the perfectly conditioned identity matrix. The second control signal might just provide values of a different scale (perhaps meter instead of millimeter). Especially when controlling rotatory DOFs and dealing with long lever arms, this problem of scaling may arise. So, the condition number is no absolute quality criterion but may be used to compare different solutions on the same trace data to discuss their stability.

To complete our previous example, we see here the results for the condition numbers for the direct solution (left) and the inverted Feature Jacobian with weighting (right):

$$\kappa = \frac{2.604}{0.144} = 18.110 \qquad \kappa = \frac{0.688}{0.145} = 4.732. \tag{4.57}$$

Both conditions indicate no significant instabilities, although the matrix resulting from the direct calculation is more sensitive to changes of outer conditions, so we consider it less stable.

4.5. Flexibility

This chapter addresses the point that a solution has to offer a certain flexibility in use to be applicable in practice. Imagine a situation of sensor failure during production that affects one or more of the control signals. A flexible solution should allow a system reconfiguration to offer continued functionality without the faulty sensor (assuming the remaining sensors provide enough information to do so) and without new training.

4.5.1. DOF Deactivation

Consider we have a trace data set for a number of DOFs and control signals and we have identified the system by calculating the Jacobian matrix. Now we want to completely deactivate the i-th DOF with $i \in [1 \ldots m]$. A first solution would be to remove all trace data that has been recorded for that particular DOF and carry out a recalculation of the Jacobian. That would require us to remove $\mathbf{R}_i$ and $\mathbf{S}_i$ and the i-th column in all $\mathbf{R}_j$ with $j \in [1 \ldots m]$ and $j \neq i$. Alternatively, we could just replace the i-th column of the Jacobian with zeros to prevent the calculation of a correction for that particular DOF. That would be more simple and independent from the calculation method. To see the difference between both approaches, we take a closer look at an example.

With the X and the Y direction we want to control two DOFs and we have two control signals for that. We execute training and receive the following result for the trace matrices:

$$\mathbf{R}_1 = \begin{pmatrix} 1.0 & 0.0 \\ 2.0 & 0.0 \\ 3.0 & 0.0 \end{pmatrix} \qquad \mathbf{R}_2 = \begin{pmatrix} 0.0 & 1.0 \\ 0.0 & 2.0 \\ 0.0 & 3.0 \end{pmatrix} \tag{4.58}$$

$$\mathbf{S}_1 = \begin{pmatrix} 1.0 & 0.5 \\ 2.0 & 1.0 \\ 3.0 & 1.5 \end{pmatrix} \qquad \mathbf{S}_2 = \begin{pmatrix} 0.0 & 0.5 \\ 0.0 & 1.0 \\ 0.0 & 1.5 \end{pmatrix}. \tag{4.59}$$

It seems that signal 1 senses only the robot's movement in X with a sensitivity of 1 and signal 2 senses movements in both DOFs with a sensitivity of 0.5. We set up the Feature Jacobian and calculate the Jacobian matrix:

$$\mathbf{F} = \begin{pmatrix} 1.0 & 0.5 \\ 0.0 & 0.5 \end{pmatrix} \qquad \mathbf{J} = \mathbf{F}^+ = \begin{pmatrix} 1.0 & -1.0 \\ 0.0 & 2.0 \end{pmatrix}. \tag{4.60}$$

For the first approach we reduce the trace data to the first DOF only:

$$\mathbf{R}_1 = \begin{pmatrix} 1.0 \\ 2.0 \\ 3.0 \end{pmatrix} \qquad \mathbf{S}_1 = \begin{pmatrix} 1.0 & 0.5 \\ 2.0 & 1.0 \\ 3.0 & 1.5 \end{pmatrix} \tag{4.61}$$

The results for the Feature Jacobian and the Jacobian are the following:

$$\mathbf{F} = \begin{pmatrix} 1.0 & 0.5 \end{pmatrix} \qquad \mathbf{J} = \mathbf{F}^+ = \begin{pmatrix} 0.8 \\ 0.4 \end{pmatrix}. \tag{4.62}$$

The second approach takes the Jacobian from (4.60) and removes the second row:

$$\mathbf{J}' = \begin{pmatrix} 1.0 \\ 0.0 \end{pmatrix}. \tag{4.63}$$

After training, we relocate the work object by moving it two millimeters in the direction of the second DOF. This leads to a signal deviation of

$$\Delta \mathrm{s} = \begin{pmatrix} 0.0 \\ 0.1 \end{pmatrix} \tag{4.64}$$

as you can see from the second row of $\mathbf{S}_2$. If we estimate the robot deviation using the two Jacobian matrices $\mathbf{J}$ and $\mathbf{J}'$, we get the following results:

$$\begin{pmatrix} 0.8 & 0.4 \end{pmatrix} \begin{pmatrix} 0.0 \\ 0.1 \end{pmatrix} = 0.4 \qquad \begin{pmatrix} 1.0 & 0.0 \end{pmatrix} \begin{pmatrix} 0.0 \\ 0.1 \end{pmatrix} = 0.0 \tag{4.65}$$

Both Jacobian matrices are not identical and are therefore the approximations of the said deviation. We have to take a closer look to the consequences of these results. The system analyzed so far could have a setup similar to that shown in Figure 4.4. In (a) we see the reference setup and in (b) the situation after moving the work object in Y-direction. Using the Jacobian from the first approach would cause the robot to be moved a little in the negative X-direction, as shown in (c). If using the second Jacobian, it would not lead to any robot relocation - the situation would stay as it is in (b).

If, when identifying the system for just one DOF, there are deviations in other DOFs, the system tries to minimize the errors in the control signals by moving the robot in the directions that have been identified. This behavior was seen in (c). In the other case, we identify the system in both DOFs. As a consequence, the system "knows" that the current signal deviation is caused by a deviation in Y-direction, but by zeroing out that column in the Jacobian we prevent the system from moving the robot in that designated direction. This result is important when deciding about the DOFs to be controlled and the DOFs to be trained.

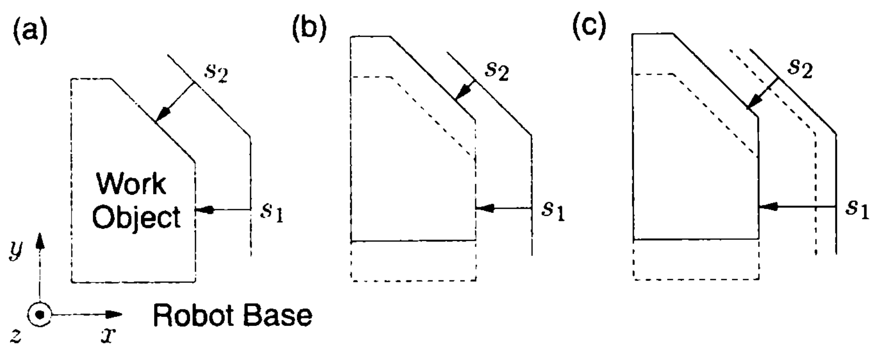


Figure 4.4: Deviations in Uncontrolled DOFs

4.5.2. Control Signal Deactivation

The complete deactivation of the i-th control signal with $i \in [1 \ldots n]$ can be done by zeroing out the i-th column of the Feature Jacobian before inverting it. This leads to a weighting of zero for this control signal in the i-th row of the Jacobian and a re-weighting of the other rows. In order to do the same for the direct solution, we have to operate on the trace data by removing the i-th column in $\mathbf{S}$ for $i \in [1 \ldots n]$ and then recalculating $\mathbf{J}$.

The deactivation of the j-th control signal only for the i-th DOF with $i \in [1 \ldots m]$ and $j \in [1 \ldots n]$ is more complicated. When solving the system directly, we are able to separate the solution with a single linear system, like in (4.17), into m steps by determining each column of $\mathbf{J}$ separately by

$$\mathbf{S}\,\mathbf{j}_i = \mathbf{r}_i. \tag{4.66}$$

with $\mathbf{j}_i$ being the i-th column vector of $\mathbf{J}$ and $\mathbf{r}_i$ the i-th column vector of $\mathbf{R}$. For deactivating the j-th control signal for the i-th DOF we modify the i-th linear system of (4.66) by replacing the j-th column of $\mathbf{S}$ with zeros. For a solution using the Feature Jacobian, the deactivation cannot be done. Setting $f_{j,i}$ to zero does not lead to a zero at the correlating element of $\mathbf{J}$ and the inversion of $\mathbf{F}$ cannot be separated into multiple steps.

4.5.3. General Flexibility

Solutions using the Feature Jacobian make use of the trace movements that are orthogonal in robot space to identify the local partial derivatives of the elements of the Feature Jacobian. This limitation is not present for the direct solution. If using trace data with any kind of robot position samples and control signal samples, the direct solution is still able to determine a solution for the Jacobian. This flexibility may be used, for example, to utilize trace data collected during a control process to re-identify the system after a sensor change.

4.6. Direct Solution Stability Issues

4.6.1. Problem

In the sections above, we have stated a certain stability problem in connection with the direct system identification. To understand the problem, we take a look at a small example. We consider that we want to control a single robot DOF using three control signals. Here, we see the result from a trace experiment with a robot position matrix of $\mathbf{R}$ and a resulting control signal matrix S:

$$\mathbf{R} = \begin{pmatrix} 1.0 \\ 2.0 \\ 3.0 \\ 4.0 \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} -0.0004 & 0.9600 & 0.0004 \\ 0.0003 & 2.0300 & -0.0003 \\ -0.0002 & 2.9800 & 0.0002 \\ 0.0001 & 4.0100 & -0.0001 \end{pmatrix}. \tag{4.67}$$

The DOF is only really reflected by the second control signal and this fact is confirmed by a look into the Feature Jacobian

$$\mathbf{F} = \begin{pmatrix} 0.0000 & 1.0000 & 0.0000 \end{pmatrix}. \tag{4.68}$$

To get the Jacobian, we invert the Feature Jacobian and get the following result

$$\mathbf{J} = \mathbf{F}^{+} = \begin{pmatrix} 0.0000 \\ 1.0000 \\ 0.0000 \end{pmatrix}. \tag{4.69}$$

Everything is fine - the result is as we expected - but if we solve the system directly, we get the following Jacobian:

$$\mathbf{J} = \begin{pmatrix} -50.0000 \\ 1.0000 \\ 50.0000 \end{pmatrix}. \tag{4.70}$$

The result is quite unacceptable. The matrix contains two very high attenuation factors for the control signals 1 and 3. Only the slightest of changes to the mounting position of a sensor involved in one of these control signals would lead to great robot corrections. Even if we cannot determine the instability of this matrix in terms of the matrix condition number (there is only one singular value which is the highest and the lowest at the same time, so the condition number is 1), the SVD shows a singular value of 70.72, which would lead to quite a high condition number if this matrix was part of a Jacobian with additional DOFs.

What has happened? All the control signals contained some additional noise of decreasing intensity. The noise is a hundred times greater on control signal 2 than on the other control signals. The reason for this noise could be vibrations in the sensor tree caused by a jerkily starting robot movement. To reduce the residual square sum, the direct solution uses a high attenuation for control signals 1 and 3 to cancel the noise in the second control signal. By doing this, it realizes a residual square sum of zero.

This might have been a maliciously constructed example, but it illustrates the problem of the direct solution that optimizes for the least squares; sometimes at the cost of stability. Unfortunately, this problem occurs in the real world too when solving trace data using the direct calculation. We take a look at the result of an experiment with the setup shown in Figure 4.5. A robot is positioned relative to a fixed work object using four distance sensors mounted at the robot's tool; providing us with four control signals. The robot moves in all three translational DOFs.

After training, we get the following Jacobian matrix by using the inversion of the Feature Jacobian without weighting:

$$\mathbf{J} = \begin{pmatrix} 0.498 & -0.029 & -0.001 \\ -0.024 & -0.030 & 1.007 \\ 0.006 & -1.012 & 0.001 \\ 0.489 & -0.017 & 0.001 \end{pmatrix}. \tag{4.71}$$

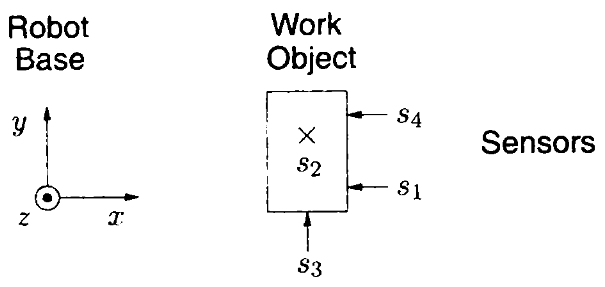


Figure 4.5: Experimental Setup

The result is just as expected. The X direction is reconstructed using control signals 1 and 4, the Y direction by control signal 3 and the Z direction by control signal 2. Now, we solve the linear system to directly estimate the Jacobian. The result is as follows:

$$\mathbf{J} = \begin{pmatrix} 0.720 & -3.290 & -0.284 \\ -0.023 & -0.034 & 1.004 \\ 0.008 & -0.964 & 0.003 \\ 0.256 & 3.295 & 0.289 \end{pmatrix}. \tag{4.72}$$

The reconstruction of X is still acceptable. Control signal 1 is favored over control signal 4, but that might be because of the lower noise ratio of signal 1 (this is not the case, but see below). The problem is the high attenuations of control signals 1 and 4 in the reconstruction of (most notably) Y and Z, just like in the constructed example.

4.6.2. Solutions

4.6.2.1. Hysteresis Compensation

The stability problem of the direct identification, based on the least-squares optimal solution of an over-determined linear system, is worsened by the hysteresis effect, which has been addressed briefly in the previous section about the trace data collection and which will be analyzed in more depth in Chapter 6. This effect superimposes certain information to all control signals, causing the direct solution to use redundant control signals with high attenuation to balance out this effect to accomplish a lower residual square sum. Later in Chapter 6.3.2, we will discuss a method for the compensation of the hysteresis in trace data. At this point, we will test the application of this method to achieve more stability of the Jacobian derived from our experimental trace data. When calculating the Jacobian with a direct solution after applying the hysteresis compensation we get:

$$\mathbf{J} = \begin{pmatrix} 0.226 & -1.437 & -0.017 \\ -0.021 & -0.030 & 1.003 \\ 0.006 & -0.991 & -0.001 \\ 0.764 & 1.418 & 0.019 \end{pmatrix}. \tag{4.73}$$

The stability problem is still there, though in a diminished state. Apparently, the hysteresis compensation can improve the situation but cannot solve the problem entirely. It is interesting to note that for the reconstruction of the X direction, the shares of signals 1 and 4 have changed compared to the situation without hysteresis compensation in (4.72). The reason is that the hysteresis effect prevents the direct calculation from using a reasonable weighting of the control signals due to the signal noises. To understand this effect, we take a look at Figure 4.6. On the left side we see a number of k samples (x_i, y_i) without any hysteresis, the regression line and the residuals. The residual square sum - the optimization criterion for the direct calculation - depends only on the noise of the samples σ_y^2 and can be estimated as:

$$\sum_{i=1}^{k} (y_i - f(x_i))^2 \approx k\,\sigma_y^2. \tag{4.74}$$

On the right side we have added a hysteresis h to the samples. The regression line is nearly the same, but the residuals have grown. Their square sum can be estimated now by:

$$\sum_{i=1}^{k} (y_i - f(x_i))^2 \approx k\,(h^2 + \sigma_y^2). \tag{4.75}$$

We can see that the hysteresis is misinterpreted as control signal noise and this way the direct calculation fails to implement a correct noise-based weighting. The hysteresis compensation solves at least this problem.

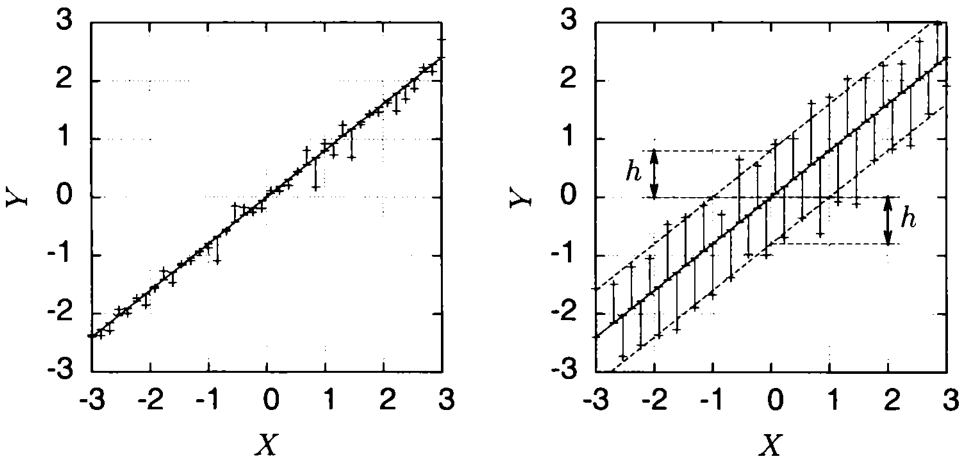


Figure 4.6: Hysteresis Influence on Residual Square Sum

4.6.2.2. Manual Deactivation

If a stability problem has been spotted, we can manually deactivate the affected control signals for certain DOFs, as already discussed in Chapter 4.5.2. In our case, we will deactivate control signals 1 and 4 for the DOFs Y and Z. The result is:

$$\mathbf{J} = \begin{pmatrix} 0.226 & 0.000 & 0.000 \\ -0.021 & -0.028 & 1.003 \\ 0.006 & -1.004 & -0.001 \\ 0.764 & 0.000 & 0.000 \end{pmatrix}. \tag{4.76}$$

The first column of J for the X direction is the same, just as it has to be (the hysteresis compensation is still used). In the other DOFs, the weighting for signals 1 and 4 is exactly zero. The result is very stable, but of course this method for solving stability problems has a significant drawback -

it does not work automatically. We have manually identified and solved the stability problem for a very clear case. In a more complex situation we cannot be sure to spot all stability problems and it requires an experienced user to analyze the situation and to take useful action.

4.6.2.3. L2 Regularization

For the direct solution, we receive a least-squares optimal solution that optimizes the term

$$\min_{\mathbf{j}_i} ||\mathbf{S}\,\mathbf{j}_i - \mathbf{r}_i||_2^2 . \tag{4.77}$$

for $i \in [1 \ldots m]$. The mathematical solution for our stability problems is the introduction of an additional optimization criterion, often called a *prior* or a *penalty term*. In the case of the *L2 regularization*, also called *Tikhonov regularization* or *ridge regression* (see [88]), we introduce a term that depends on the L2 norm (Euclidean distance)

$$||\mathbf{x}||_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{\frac{1}{2}} \tag{4.78}$$

of the Jacobian column vector

$$\min_{\mathbf{j}_i} \left(||\mathbf{S}\mathbf{j}_i - \mathbf{r}_i||_2^2 + \lambda \; ||\mathbf{j}_i||_2^2 \right) . \tag{4.79}$$

With the constant $\lambda \in \mathbb{R}$, $\lambda \geq 0$ we can control how important it is that the solution is least-squares• optimal and how important it is to get a small result vector $\mathbf{j}_i$. For $\lambda = 0$, the solution is the same as for the least-squares optimal solution - increasing λ, the L2 norm of the solution vector gets smaller, which makes the solution less accurate. For further information and a more general overview of the inverse problem theory, see, for example, [86].

For the direct calculation, we have used the pseudo-inverse of S

$$\mathbf{j}_i = \mathbf{S}^{+}\,\mathbf{r}_i = (\mathbf{S}^{\mathsf{T}}\,\mathbf{S})^{-1}\,\mathbf{S}^{\mathsf{T}}\,\mathbf{r}_i , \tag{4.80}$$

for the Tikhonov regularization we have to calculate

$$\mathbf{j}_i = (\mathbf{S}^{\mathsf{T}}\,\mathbf{S} + \lambda\,\mathbf{I})^{-1}\,\mathbf{S}^{\mathsf{T}}\,\mathbf{r}_i . \tag{4.81}$$

Here, we get an idea how this method enhances the stability - by adding a diagonal matrix we ensure the invertibility of the covariance matrix $\mathbf{S}^{\mathsf{T}}\,\mathbf{S}$. The Tikhonov regularization helps to achieve the stability of the solution, but because of the square norm it encourages the elements in $\mathbf{j}_i$ to become similar to one another with increasing λ, which is an unwanted effect. The reason for this is that the square function of the L2 norm in the prior initially punishes components of $\mathbf{j}_i$ with high values, including those important for the system identification.

We check the result of identifying our experimental trace data using the Tikhonov regularization. With $\lambda = 10.0$ we get

$$\mathbf{J} = \begin{pmatrix} 0.489 & -0.036 & 0.000 \\ -0.020 & -0.026 & 0.979 \\ 0.003 & -0.983 & -0.001 \\ 0.485 & -0.004 & 0.002 \end{pmatrix} \tag{4.82}$$

We realize that we have achieved a good stability but we see that the weights for signals 1 and 4 for the X direction are very similar. This is the effect described before - that for the achievement of a small square norm of the column vectors of $\mathbf{J}$, all elements are shrunk simultaneously. This cancels the noise-based weighting in the Jacobian too, which is not desirable.

4.6.2.4. L1 Regularization

The solution lies in the application of the *L1 Regularization*, which has been presented independently as *LASSO*[3] (see [87]) and as *Basis Pursuit Denoising* (see [15]). The unconstrained formulation of the L1 regularization is given by the formula

$$\min_{\mathbf{j}_i} \left(||\mathbf{S}\,\mathbf{j}_i - \mathbf{r}_i||_2^2 + \lambda\, ||\mathbf{j}_i||_1 \right). \tag{4.83}$$

Compared to the Tikhonov regularization, the norm in the prior has been replaced by an L1 norm (Manhattan norm)

$$||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|. \tag{4.84}$$

Again, the constant $\lambda \in \mathbb{R}$, $\lambda \geq 0$ determines the compromise between the accuracy and stability of the solution. Moreover, for $\lambda = 0$ the solution is identical to the least-squares optimal solution. There are constrained formulations of the problem too, which assure that each element of the solution vector is smaller than a certain boundary, but these formulations are less applicable for our application.

Since the optimization criterion introduces some numerical problems (the optimization term is not differentiable for any component of $\mathbf{j}_i$ equal to zero), there is no closed form available for the solution. Instead, there is a variety of different methods for determining the solution for a problem using the L1 regularization. The results below have been calculated using a recent Gauss-Seidel approach from Logistic Regression (see [84]). Another approach that has been well tested for its usage in the field of system identification is the so called "Shooting" approach, which is described in [29] and has its easy implementation as an advantage. For a comparative study of different methods and for implementations, see the work of [81]. These methods are currently under heavy development and are being employed for a variety of different problems like the linearization of problems depending on a high amount of redundant input variables.

The interesting properties of the L1 regularization are that the obtained solution is not only more stable, but the prior enforces a certain sparsity of the solution matrix easing the interpretation of the solution. If calculating the solution for our experimental data using the L1 regularization with $\lambda = 1$, we get the Jacobian

$$\mathbf{J} = \begin{pmatrix} 0.328 & -0.040 & 0.000 \\ -0.020 & -0.026 & 1.002 \\ 0.004 & -1.005 & 0.000 \\ 0.659 & 0.000 & 0.000 \end{pmatrix}. \tag{4.85}$$

This solution has no stability issues and keeps the weights of control signals 1 and 4 for direction X. If we increase λ to a value of 35 we get the following result:

$$\mathbf{J} = \begin{pmatrix} 0.936 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.960 \\ 0.000 & -0.962 & 0.000 \\ 0.000 & 0.000 & 0.000 \end{pmatrix}. \tag{4.86}$$

The solution matrix becomes increasingly more sparse. Smaller and less important weights of the matrix are replaced step-by-step with zeros.

[3]LASSO=Least Absolute Selection and Shrinkage Operator

The question which is still open is how to choose the parameter λ. First, it is important to note that we can choose a different λ for each DOF. Even if we had used a single λ for the examples above, the theory presented so far has calculated the columns of **J** separately, so we can choose individual λ_i with $i \in [1 \dots m]$. To make things clear, we study the effect of the choice of λ_i on the quality of the solution by using an example. This time we take a less academic example, whereby we use the data obtained from the configuration of an industrial application where five DOFs are controlled using eleven control signals.

First, we take a look at the dependency of the coefficient of determination on the size of the λ_i. Figure 4.7 shows that for all DOFs, the initially high COD decreases with increasing λ_i until the COD reaches zero - indicating that the solution is no longer a solution. A typical behavior that can be found for any experimental data is that the quality of the rotatory DOFs decrease faster than for translational DOFs. This is because of the usually smaller entries in the column vectors of **J** that is responsible for the rotatory DOFs. This demonstrates the need for individual choices for λ for each DOF.

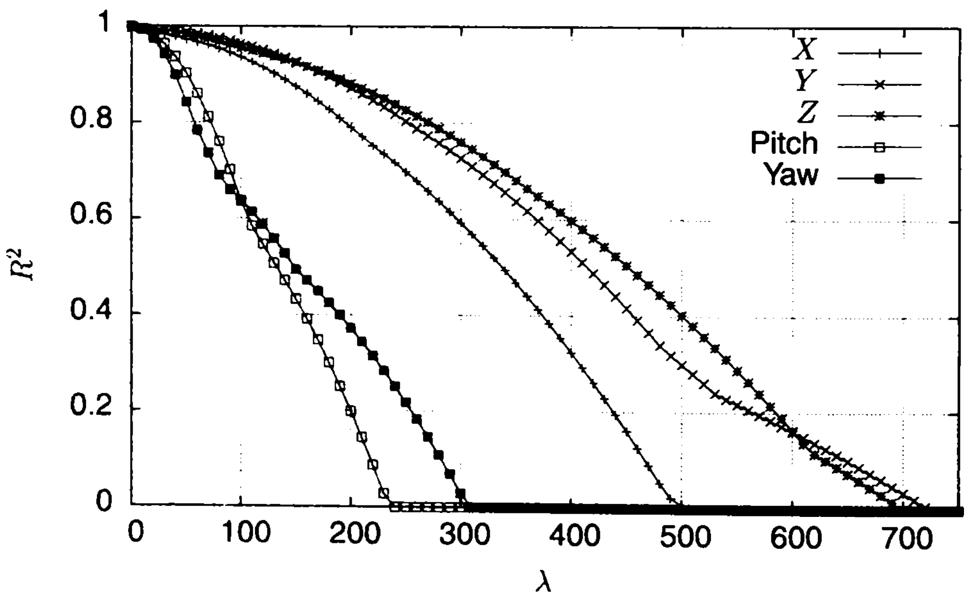


Figure 4.7: Coefficient of Determination and λ

If we analyze how the sparsity of the Jacobian increases with increasing λ, we see a picture as in Figure 4.8. Here, we see how many non-zero components of the first column vector of the Jacobian (responsible for the reconstruction of X) are left. The number decreases from eleven (all eleven control signals have a weight different from zero) to five within an increase of λ up to a value of 2.0. For a value of λ above 500, all components of the first column vector of the Jacobian are zero and this is the time when the COD for the X direction drops to zero too (compare Figure 4.7). It is further interesting to note that sometimes (for example, visible here in the zoomed-in interval) the number of non-zero elements increases when increasing λ for a short time. However, because we have two optimization criteria, a small residual square sum and a small L1 norm of the solution, we cannot expect that the number of non-zero elements is in any way monotone over λ.

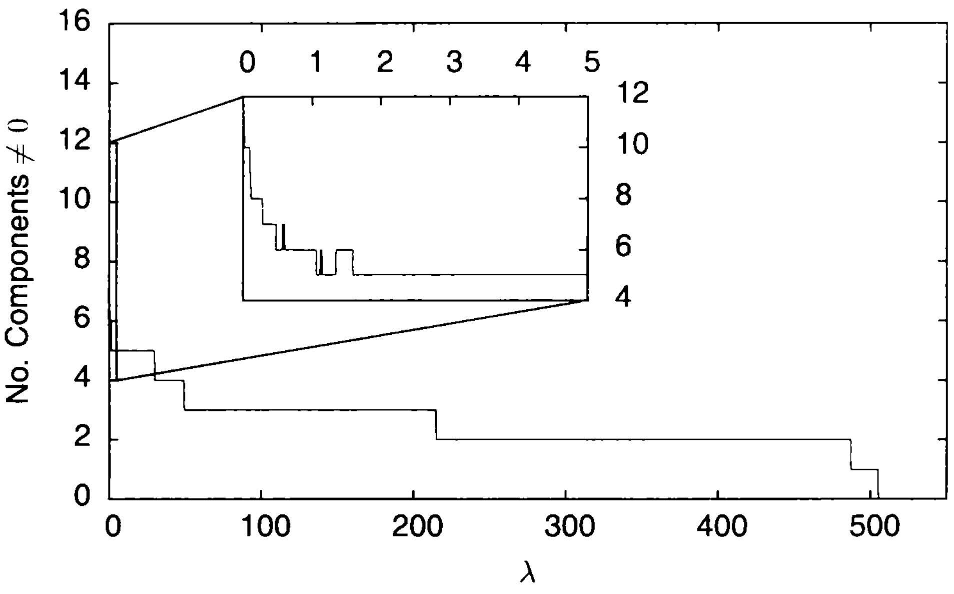


Figure 4.8: Number of Non-Zero Components and λ

Next, we check the consequences for the stability of the Jacobian on the choice of λ (see Figure 4.9 for this). The Jacobian condition number κ is plotted against the size of λ (the same λ for all DOFs to keep it manageable). First, the κ decreases (see zoomed-in view); indicating a more stable solution for $\mathbf{J}$. After a while, κ increases more and more until λ reaches a value of about 234.5. This time, the fourth column vector of $\mathbf{J}$ (pitch angle) is a zero vector (see Figure 4.7) and κ is infinity. There is something important to note here. While the previous figures were qualitatively similar for all trace data occurring in practice, this figure might look quite different depending on the trace data. Sometimes, there is no decrease of the Jacobian condition when increasing λ and the shape of the curve might be completely different. The only thing that all curves have in common is that for a high λ, the value of κ increases until it reaches infinity.

Consequently, the L1 regularization offers a more stable solution as well as a sparse solution matrix. To see the advantage of a sparse solution matrix, we take a look at the Jacobian matrix of our current example; determined by inverting the Feature Jacobian with additional weighting:

$$\mathbf{J} = \begin{pmatrix} 0.067 & 0.476 & -0.084 & -0.025 & -0.060 \\ -0.012 & 0.267 & 0.204 & 0.010 & -0.032 \\ -0.458 & 0.018 & -0.355 & 0.001 & 0.020 \\ 0.059 & 0.101 & 0.142 & -0.008 & 0.004 \\ -0.232 & -0.007 & 0.078 & -0.025 & -0.012 \\ 0.020 & -0.038 & 0.073 & -0.023 & 0.027 \\ -0.127 & 0.023 & 0.161 & -0.027 & -0.017 \\ 0.006 & 0.054 & 0.011 & 0.003 & 0.028 \\ -0.462 & -0.018 & 0.090 & 0.031 & 0.006 \\ 0.059 & 0.371 & -0.041 & 0.028 & -0.005 \\ -0.037 & 0.135 & 0.273 & 0.021 & -0.015 \end{pmatrix}. \qquad (4.87)$$

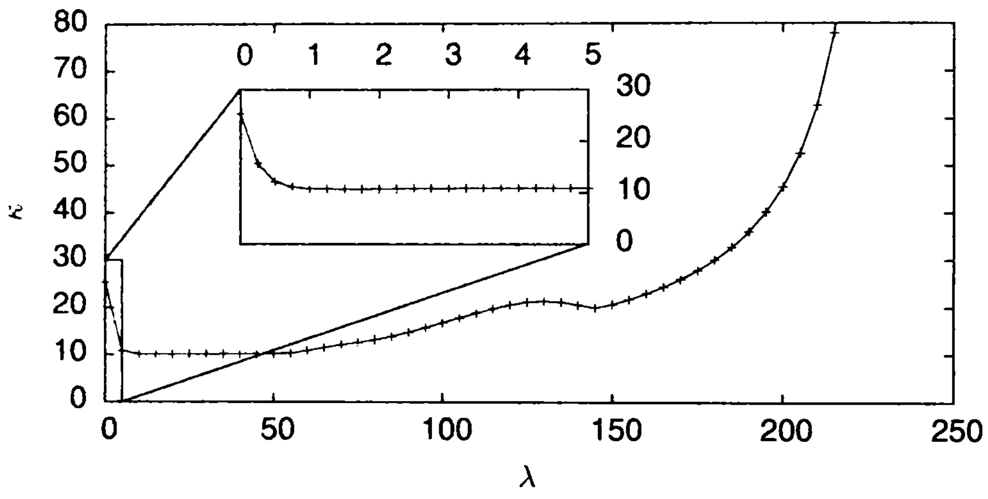


Figure 4.9: Jacobian Condition and λ

The overall quality in terms of COD is given by:

$$R^2 = 0.991 \cdot 0.996 \cdot 0.996 \cdot 0.999 \cdot 0.999 = 0.980. \tag{4.88}$$

We now sacrifice some of the really good quality of the system identification by using a direct solution with L1 regularization by adapting the λ_i in a way that we have a COD of $R_i^2 = 0.980$ for all $i \in [1 \ldots m]$:

$$R^2 = 0.980 \cdot 0.980 \cdot 0.980 \cdot 0.980 \cdot 0.980 = 0.904. \tag{4.89}$$

We can find the necessary λ_i by obtaining the intersection points of the curves in Figure 4.7 with the horizontal line at 0.980. This can be done manually or by numerical root-finding. In our case, we choose

$$\lambda_1 = 36.70,\ \lambda_2 = 70.57,\ \lambda_3 = 60.51,\ \lambda_4 = 21.31,\ \lambda_5 = 17.21. \tag{4.90}$$

The resultant matrix is given by

$$\mathbf{J} = \begin{pmatrix} 0.000 & \mathbf{0.341} & 0.000 & 0.000 & \mathbf{-0.041} \\ 0.000 & \mathbf{0.325} & 0.000 & 0.000 & \mathbf{-0.034} \\ \mathbf{-0.153} & 0.000 & \mathbf{-0.064} & 0.000 & 0.000 \\ \mathbf{0.009} & 0.000 & \mathbf{0.098} & 0.000 & 0.000 \\ \mathbf{-0.347} & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & \mathbf{0.068} & 0.000 & \mathbf{-0.010} & 0.000 \\ 0.000 & 0.000 & \mathbf{0.262} & \mathbf{-0.058} & \mathbf{-0.012} \\ 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.064} \\ \mathbf{-0.514} & 0.000 & 0.000 & \mathbf{0.018} & 0.000 \\ 0.000 & \mathbf{0.358} & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & \mathbf{0.558} & \mathbf{0.044} & 0.000 \end{pmatrix} \tag{4.91}$$

Due to its sparsity (only 20 out of the 55 matrix elements are non-zero), the result is much easier to interpret. We can identify the sensors most important for the reconstruction of certain degrees much easier than was the case in the original matrix and this sparsity, together with an enhanced stability, was bought for only a small loss of quality.

When using the L1 regularization in practice, a preferable method for finding adequate values for the λ_i with $i \in [1 \ldots m]$ is to define the ratio of COD we are willing to sacrifice for the sake of stability and sparsity. We call this share $p \in \mathbb{R}$ with $p \in [0 \ldots 1]$. If the direct solution for the i-th DOF gave a COD of R_i^2, then this quality is what the L1 regularization method would return for $\lambda_i = 0$ and the highest COD that any solution may achieve. If we want to sacrifice, let's say $5\,\%$ of that maximal achievable COD ($p = 0.05$), we find the value of λ_i, where the COD of the L1 regularized solution is still $95\,\%$ of the COD of the direct solution. Because of the benign coherence between λ_i and R_i^2 (see Figure 4.7), this problem can be solved by numerically finding the root of

$$g_i(\lambda_i) = R_i(\lambda_i) - (1-p)\,R_i(0) \tag{4.92}$$

for λ_i. This may take a few runs of the L1 regularization algorithm, but because we calculate the Jacobian once during system setup, time is not an important issue here. The stated property of $R_i(\lambda_i)$ to be well behaved should be confirmed, but this may be difficult because there is no closed form solution of the L1 regularization available.

4.7. Geometrical Considerations

In this section, we will analyze how the weighting of the control signals in the Jacobian will affect the geometrical result of the approximation of the robot deviation. We do this on the basis of a small example. Considering the situation shown in Figure 4.10, we want to position a part fixed in the robot's hand symmetrically between the gap inside a work object. There is only one DOF to control and we have two control signals determined by the gaps between the work object and the part at two different positions.

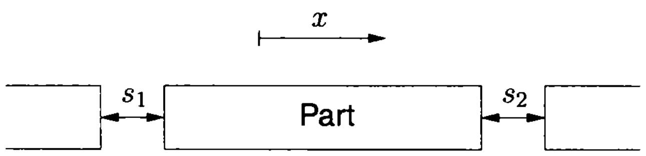


Figure 4.10: Symmetry Example Setup

The reference situation should be symmetrical and we have a gap of $1.0\,\mathrm{mm}$ to the left and right of the part. The Feature Jacobian is quite trivial too:

$$\mathbf{s}_0 = \begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} 1.0 & -1.0 \end{pmatrix}. \tag{4.93}$$

The according Jacobian depends on the solution method. If just inverting the Feature Jacobian, we get an equal weighting of both control signals, as in $\mathbf{J}_1$, but if we use any noise-weighting solution and the control signal two contains less noise, we might get a weighting as in $\mathbf{J}_2$:

$$\mathbf{J}_1 = \begin{pmatrix} 0.5 \\ -0.5 \end{pmatrix} \qquad \mathbf{J}_2 = \begin{pmatrix} 0.2 \\ -0.8 \end{pmatrix}. \tag{4.94}$$

In the production, we might find a situation where the part is moved $0.5\,\mathrm{mm}$ to the right:

$$\mathbf{s} = \begin{pmatrix} 1.5 \\ 0.5 \end{pmatrix} \qquad \Delta\mathbf{s} = \mathbf{s} - \mathbf{s}_0 = \begin{pmatrix} 0.5 \\ -0.5 \end{pmatrix}. \tag{4.95}$$

The approximation using each of the Jacobian matrices gives the following result:

$$\Delta x_1 = \mathbf{J}_1^{\mathsf{T}}\,\Delta\mathbf{s} = 0.5 \qquad \Delta x_2 = \mathbf{J}_2^{\mathsf{T}}\,\Delta\mathbf{s} = 0.5 \tag{4.96}$$

where both results reflect the same correct replacement. Now, we consider that we have a situation where the part is subject to a form deviation - it is actually $1.0\,\mathrm{mm}$ too short:

$$\mathbf{s} = \begin{pmatrix} 1.0 \\ 2.0 \end{pmatrix} \qquad \Delta\mathbf{s} = \begin{pmatrix} 0.0 \\ 1.0 \end{pmatrix}. \tag{4.97}$$

We now get the approximations using the Jacobian matrices

$$\Delta x_1 = \mathbf{J}_1^{\mathsf{T}}\,\Delta\mathbf{s} = -0.5 \qquad \Delta x_2 = \mathbf{J}_2^{\mathsf{T}}\,\Delta\mathbf{s} = -0.8 \tag{4.98}$$

and receive different results. If we use the results to move the robot to balance out the deviations, we receive the new control signals and the new control signal deviations. In the first case, we get a situation where the part is positioned symmetrically just as demanded; the residual errors are distributed evenly over both control signals:

$$\mathbf{s}_1 = \begin{pmatrix} 1.5 \\ 1.5 \end{pmatrix} \qquad \Delta\mathbf{s}_1 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}. \tag{4.99}$$

In the second case, the situation is different; the part is not positioned in a symmetrical way:

$$\mathbf{s}_1 = \begin{pmatrix} 1.8 \\ 1.2 \end{pmatrix} \qquad \Delta\mathbf{s}_1 = \begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix}. \tag{4.100}$$

Apparently the residuals, caused by the form deviation of the part, are distributed over the measured features in a way that is determined by the weighting of the control signals in the Jacobian. An equal weighting of the control signals causes equally distributed residuals. If we apply another certain weighting to the control signals, this affects the distribution of possible residuals. So, if we have redundant control signals for the reconstruction of a DOF and if we expect form deviations of significant size, we should be aware that the weightings of control signals influence the distribution of the residuals on the control signals. To deal with this, one solution can be the dynamic adaption of the nominal values $\mathbf{s}_0$ due to the current work object dimensions for example based on a preceding measurement or the choice of a solution method less susceptible to.

If dealing with form deviations of the work object, it might be interesting to estimate the form deviations and to monitor them during the control process. Our estimated robot correction is determined as described previously by calculating

$$\Delta\mathbf{r} = \mathbf{J}^{\mathsf{T}}\,\Delta\mathbf{s}. \tag{4.101}$$

We take this robot correction $\Delta\mathbf{r}$ to estimate the control signal situation after applying the robot correction. This is done by transforming $\Delta\mathbf{r}$ back to sensor space by using the Feature Jacobian and subtracting the control signal deviation:

$$\begin{aligned} \mathbf{s}_e &= \mathbf{F}^{\mathsf{T}}\,\Delta\mathbf{r} - \Delta\mathbf{s} & (4.102)\\ &= \mathbf{F}^{\mathsf{T}}\,\mathbf{J}^{\mathsf{T}}\,\Delta\mathbf{s} - \Delta\mathbf{s} & (4.103)\\ &= (\mathbf{J}\,\mathbf{F})^{\mathsf{T}}\,\Delta\mathbf{s} - \Delta\mathbf{s}. & (4.104) \end{aligned}$$

The vector s_e is now the expected form deviation for each control signal and can be calculated at every control step. A check of s_e against the maximum allowed values can be used to prevent the control process to continue for work objects with extreme form deviations. In the preferable situation of s_e equal to the zero vector, we get

$$(\mathbf{J}\,\mathbf{F})^{\top}\,\Delta s = \Delta s, \tag{4.105}$$

which is the case of the square matrix $\mathbf{J}\,\mathbf{F}$ of size $n \times n$ being the identity matrix. If $m \neq n$ and the Jacobian is calculated as the pseudo-inverse of the Feature Jacobian, it is not guaranteed that $\mathbf{J}\,\mathbf{F}$ is an identity matrix, because this is not necessarily a property of the pseudo-inverse (see (4.29) and (4.30)). Only if $m = n$ and all the row and column vectors of $\mathbf{F}$ are linearly independent, we get an identity matrix for $\mathbf{J}\,\mathbf{F}$ (see (4.28)). If we have a highly redundant configuration with much more control signals than DOFs to control, meaning $m \ll n$, we have the problem, that the for the calculation of the form deviation, we first transform the control signal vector Δs from $\mathbb{R}^n$ into $\mathbb{R}^m$. This involves an information reduction which leads to problems with the subsequent transformation back to $\mathbb{R}^n$. In these cases the estimation of the form deviation is quite bad.

4.8. Conclusion

In this chapter, we discussed several solution methods for the learning approach in visual servoing to identify the system using linearization and Jacobian matrices - the direct solution of an over-determined linear system and the classical method of inverting the Feature Jacobian with and without additional weighting. After that, we analyzed different methods for the estimation of the quality and stability of system identification. We looked at additional aspects such as flexibility and some geometrical considerations. To address the stability problems of the direct solution, we checked the usability of different enhancement techniques for the direct calculation, such as L2 and L1 regularization. As we saw, the hysteresis effect has malign effects on a proper noise-based weighting of the control signals. A compensation of this effect is therefore advisable. Before we can decide which method is the best for our field of application, we have to take a further look at the hysteresis effects that could be found in the trace data. Another point to address is how the weighting of the control signals in the Jacobian depends on the properties of those control signals. Finally, we have not yet found an adequate weighting to use in connection with the inversion of the Feature Jacobian. These points are addressed in the next chapters.

5. MODEL-BASED SIGNAL-WEIGHTING ANALYSIS

When calculating the current robot deviation for a single DOF, we determine the dot product of the vector containing the current control signals values with the appropriate column vector of the Jacobian. This way, each element of this column vector states how much each control signal takes part in the resulting reconstruction of the DOF. Nonetheless, the calculation of the Jacobian does not necessarily reveal details about how the elements of the Jacobian depend on the properties of the control signals or the robot. If directly solving a linear system using the trace data, the result is not easy to interpret. If a control signal is nearly zeroed out in the Jacobian it is difficult to find out why this happened.

In this chapter, we specify a model consisting of different data sources for control signals and robot positions with individual sensitivities for the robot movement and individual noise ratios as model parameters (see Figure 5.1). Using this model, we are able to generate trace data and determine the Jacobian with different solving methods. By an additional calculation of the solutions directly from the model parameters, we are able to analyze the influence of the single model parameters on the solution and compare different solution methods.

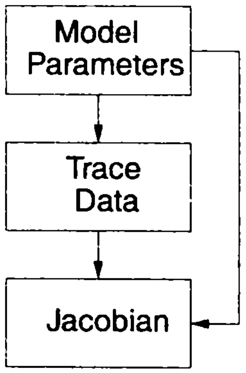


Figure 5.1: Modelling Overview

The contents of this chapter have been presented at a conference, and the corresponding paper can be found under [75].

5.1. Basic Model

5.1.1. Without Noise

First, we assume a simple model with only two signals and without any noise. Consider a function

$$f : \begin{cases} \mathbb{R} & \longrightarrow & \mathbb{R} \\ x_i & \longrightarrow & y_i = f(x_i) = g\,x_i \end{cases} \tag{5.1}$$

that describes the linear dependence of two signal values x_i and y_i with the gain $g \in \mathbb{R}$ and with $i \in [1 \ldots k]$, $k \in \mathbb{N}$ and $k > 1$ the number of samples. The values of x_i are equidistantly distributed over an interval $[a, b]$ with $a, b \in \mathbb{R}$ and $x_1 = a < b = x_k$ (see Figure 5.2).

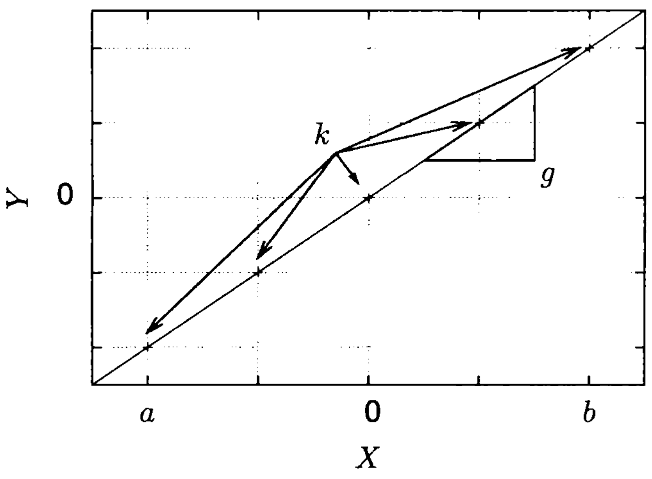


Figure 5.2: Simple Model

To calculate the best-fit linear regression from the sample data to reconstruct the gain as the model parameter, one would calculate

$$g' = \frac{\mathbf{x}^{\mathsf{T}}\mathbf{y}}{\mathbf{x}^{\mathsf{T}}\mathbf{x}} = \frac{\sum_{i=1}^{k} x_i y_i}{\sum_{i=1}^{k} x_i x_i} \tag{5.2}$$

where the result can be expressed in terms of the sample data, either with the scalar products of sample data vectors or with sum terms. We will now try to calculate the sums in the nominator and denominator directly from the model parameters:

$$\sum_{i=1}^{k} x_i^2 \tag{5.3}$$

$$= \sum_{i=1}^{k}((a+(b-a)\frac{i-1}{k-1}))^2 \tag{5.4}$$

$$= \sum_{i=1}^{k} a^2 + 2a(b-a)\frac{i-1}{k-1} + (b-a)^2\frac{(i-1)^2}{(k-1)^2} \tag{5.5}$$

$$= a^2k + 2a(b-a)\frac{1}{k-1}\sum_{i=1}^{k}(i-1) +$$

$$(b-a)^2\frac{1}{(k-1)^2}\sum_{i=1}^{k}(i-1)^2 \tag{5.6}$$

$$= a^2k + 2a(b-a)\frac{1}{k-1}\frac{k(k-1)}{2} +$$

$$(b-a)^2\frac{1}{(k-1)^2}\frac{k(k-1)(2k-1)}{6} \tag{5.7}$$

$$= a^2k + a(b-a)k + (b-a)^2 k\frac{2k-1}{6k-6} \tag{5.8}$$

$$= \left(ab + (b-a)^2\frac{2k-1}{6k-6}\right) k \tag{5.9}$$

$$= \tau k \qquad \text{with } \tau := ab + (b-a)^2\frac{2k-1}{6k-6} \tag{5.10}$$

For the regression we now get

$$g' = \frac{\sum_{i=1}^{k} x_i y_i}{\sum_{i=1}^{k} x_i x_i} = \frac{g\sum_{i=1}^{k} x_i x_i}{\sum_{i=1}^{k} x_i x_i} = \frac{g\tau k}{\tau k} = g \tag{5.11}$$

So far, so good. This is what we expected, but calculated in a very complicated way. That is true, but we need some of the results later on.

A property of τ we need later on is the fact that τ is always greater than zero for all $k > 1$ and $a < b$. If we call

$$\gamma := \frac{2k-1}{6k-6} \tag{5.12}$$

we get

$$\tau > 0 \tag{5.13}$$

$$\Leftrightarrow \quad ab + \gamma(b-a)^2 > 0 \tag{5.14}$$

$$\Leftrightarrow \quad \gamma a^2 + (1-2\gamma)ab + \gamma b^2 > 0 \tag{5.15}$$

$$\Leftrightarrow \quad \gamma a^2 + \gamma b^2 > (2\gamma - 1)\,ab \tag{5.16}$$

$$\Leftrightarrow \quad a^2 + b^2 > (2 - \frac{1}{\gamma})\,ab. \tag{5.17}$$

Because of $\frac{1}{2} \geq \gamma > \frac{1}{3}$ we know that

$$0 \geq (2 - \frac{1}{\gamma}) > -1. \tag{5.18}$$

So this estimation

$$0 < (a-b)^2 \Leftrightarrow a^2 + b^2 > 2ab > (2 - \frac{1}{\gamma})\,ab \tag{5.19}$$

proves that τ is always greater than zero.

5.1.2. With Noise

As a next step, we add noise to the samples and check for the consequences. We assume that x_i and y_i both contain the additional normal distributed noise of zero mean and standard deviation σ_x and σ_y, respectively. When recalculating the regression, the sum in the denominator becomes noise dependent:

$$g' = \frac{\sum_{i=1}^{k} x_i\, y_i}{\sum_{i=1}^{k} x_i\, x_i} = \frac{g\,\tau\,k}{(\tau + \sigma_x^2)k} \tag{5.20}$$

It is interesting to note that the result of the regression is independent from the noise in the ordinate σ_y. This is because the x_i and the y_i in the nominator are uncorrelated (which is cancelling the noise) while in the denominator the noise in x_i is accumulated in the sum. Additionally we get $g' \leq g$ for all noises $\sigma_x^2 \geq 0$.

5.2. Enhanced Model

5.2.1. Description and Model Parameters

Now back to our problem with the identification of robot movements by observing a number of control signals. Here we develop a model of describing the deviation of the robot in a single DOF $\Delta r \in \mathbb{R}$ and reconstructing this deviation using a number of $n \in \mathbb{N}$ control signal changes with a status vector of $\Delta \mathrm{s} \in \mathbb{R}^n$.

We start by considering that there is a deviation between the work object and the robot tool. During system setup, this deviation is caused by a robot movement when recording a trace. In production, the work object is replaced every production cycle with a new one at varying positions; causing this deviation. In the following, we call this real physical deviation the *true deviation*.

When the robot is moved, the robot controller determines the robot's joint angles with incremental encoders. From this data the robot controller calculates the robot position using the forward transformation. Of course, this process contains various sources of errors. For our model we simply assume that the robot deviation we get from the robot controller is the true deviation sensed with a gain of $g_r \in \mathbb{R}$ and overlaid with zero-mean noise with a standard deviation of σ_r. The gain g_r should, of course, be close to one.

The n control signals measure the true deviation depending on the geometrical feature they are measuring and the influence of the deviation in a certain DOF on this feature. For our model, we presume an individual gain and noise for each control signal. The control signal gain is a vector

$$\mathbf{g} = \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix} \in \mathbb{R}^n \tag{5.21}$$

as well as the standard deviations of the additional zero-mean noise

$$\sigma = \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix} \in \mathbb{R}^n \tag{5.22}$$

For the description of the sampling process, we use the same as used above. A trace is executed in the robot DOF r between two limits $r_{\min}$ and $r_{\max}$, creating a number of $k \in \mathbb{N}$ samples with $k > 1$. To make things clear, here is a short Matlab code snippet for generating the trace matrices $\mathbf{R}$ and S out of the model parameters:

```
true_dev = [ r_min : (r_max-r_min)/(k-1) : r_max ]';
R = g_r * true_dev + sigma_r * randn(k,1);
S = zeros(k, n);
for (i = 1:n)
    S(:,i) = g(i) * true_dev + sigma(i) * randn(k,1);
end
```

Finally, Table 5.1 gives an overview of the model parameters.

Parameter	Meaning
n	Number of control signals
k	Number of samples in trace
$[r_{\min}, r_{\max}]$	Robot trace interval
g_r	Robot gain
σ_r	Robot noise standard deviation
$g_1, \ldots, g_n$	Control signal gains
$\sigma_1, \ldots, \sigma_n$	Control signal noise standard deviations

Table 5.1: Enhanced Model Parameters

And for the value τ introduced above we get now:

$$\tau := r_{\min} r_{\max} + (r_{\max} - r_{\min})^2 \frac{2k-1}{6k-6} \tag{5.23}$$

5.2.2. Solving the System

>From a given trace we are able to calculate the Jacobian using the different methods that were mentioned previously. In the case of one robot DOF to control, the Jacobian matrix

consists of a single column

$$\mathbf{J} = \begin{pmatrix} \frac{\partial r}{\partial s_1} \\ \vdots \\ \frac{\partial r}{\partial s_n} \end{pmatrix}. \tag{5.24}$$

5.2.2.1. Direct Calculation

In (5.2), we have seen the best-fit solution for a regression with a function $y_i = f(x_i) = g\, x_i$ depending on a single variable. For higher degrees, one has to solve terms that minimize the residual square sum, shown here for a robot movement depending on two control signals:

$$\sum_{i=1}^{k}(r_i - f(\mathbf{s}_i))^2 = \sum_{i=1}^{k}(r_i - j_1\, s_{i,1} - j_2\, s_{i,2})^2 = \min \tag{5.25}$$

Besides the calculation via the pseudo inverse of S, the solution can be made by setting the partial derivatives of the sum to the single function parameters to zero and solving the resulting linear system. For the above minimization, for example, the result is:

$$j_{1,1} = \frac{\sum_{i=1}^{k} s_{i,1}\, s_{i,2} \sum_{i=1}^{k} r_i\, s_{i,2} - \sum_{i=1}^{k} s_{i,2}^2 \sum_{i=1}^{k} r_i\, s_{i,1}}{\left(\sum_{i=1}^{k} s_{i,1}\, s_{i,2}\right)^2 - \sum_{i=1}^{k} s_{i,1}^2 \sum_{i=1}^{k} s_{i,2}^2} \tag{5.26}$$

$$j_{1,2} = \frac{\sum_{i=1}^{k} s_{i,1}\, s_{i,2} \sum_{i=1}^{k} r_i\, s_{i,1} - \sum_{i=1}^{k} s_{i,1}^2 \sum_{i=1}^{k} r_i\, s_{i,2}}{\left(\sum_{i=1}^{k} s_{i,1}\, s_{i,2}\right)^2 - \sum_{i=1}^{k} s_{i,1}^2 \sum_{i=1}^{k} s_{i,2}^2}. \tag{5.27}$$

To obtain the result of the Jacobian of the direct solution, we must use the result from the optimization like (5.26) and (5.27) and replace all sum terms with appropriate terms, basing on the model parameters. We must take into account that when a sum contains factors of two different signals out of r_i, $s_{i,1}$ or $s_{i,2}$, the individual noises on both signals are statistically balanced out in the sum of the product with a high number of samples[1]. For our model, these sums can be replaced by the product of both gains, the number of samples and τ, for example

$$\sum_{i=1}^{k} s_{i,1}\, s_{i,2} = g_1\, g_2\, \tau\, k. \tag{5.28}$$

In the case that the sum contains the square of a single signal, the noise is not canceled. We get another sum containing the summarized noise:

$$\sum_{i=1}^{k} s_{i,1}^2 = g_1^2\, \tau\, k + \sigma_1^2\, k. \tag{5.29}$$

After replacing all sums in the optimization result and simplifying the term, we finally get the result we were looking for - the Jacobian depending on nothing but the model parameters:

$$J = \frac{g_r}{\frac{1}{\tau} + \sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}} \begin{pmatrix} \frac{g_1}{\sigma_1^2} \\ \vdots \\ \frac{g_n}{\sigma_n^2} \end{pmatrix} \tag{5.30}$$

[1] This is because the correlation of two uncorrelated signals is zero.

The Jacobian determined by direct calculation has some interesting properties:

- The size of a single element of J is how much a control signal contributes to the reconstruction of the DOF. This size increases linearly with the gain of the control signal and is reciprocally proportional to the square of the standard deviation of the noise of the control signal. If considering two control signals with the same information content - one with the double amount of noise - the noisy signal would have a quarter of the weight of the less noisy one. This weighting is very convenient because it suppresses noisy control signals.
- If the interval $[r_{\min}, r_{\max}]$ is very small, τ is very small and therefore $\frac{1}{\tau}$ becomes very big. This leads to numerical problems for tiny intervals.

5.2.2.2. Inverse of Feature Jacobian

The solution via the Feature Jacobian starts with the determination of the Feature Jacobian $\mathbf{F}$, which contains the partial derivatives of each control signal with respect to the single DOF:

$$\mathbf{F} = \begin{pmatrix} \frac{\partial s_1}{\partial r} & \frac{\partial s_n}{\partial r} \end{pmatrix}. \tag{5.31}$$

The elements of the matrix can be calculated by getting the slope of a regression line through the samples of robot positions and control signal values from the trace data. We have seen in (5.2) how to do this:

$$\mathbf{F} = \frac{1}{\sum_{i=1}^{k} r_i^2} \begin{pmatrix} \sum_{i=1}^{k} r_i \, s_{i,1} & \sum_{i=1}^{k} r_i \, s_{i,n} \end{pmatrix} \tag{5.32}$$

Following the rules from (5.28) and (5.29), we replace the sums with terms depending on the model parameters:

$$\mathbf{F} = \begin{pmatrix} \frac{g_1 \, g_r \, \tau \, k}{g_r^2 \, \tau \, k + \sigma_r^2 \, k} & \frac{g_n \, g_r \, \tau \, k}{g_r^2 \, \tau \, k + \sigma_r^2 \, k} \end{pmatrix} \tag{5.33}$$

The last step is the inversion of F to get the Jacobian. The special case of the pseudo inverse of a matrix containing only a single row

$$\mathbf{F} = \begin{pmatrix} f_1 & f_n \end{pmatrix} \tag{5.34}$$

has the solution

$$\mathbf{F}^+ = \frac{1}{f_1^2 + \ldots + f_n^2} \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}. \tag{5.35}$$

After applying this to (5.33) and summarizing it we get the final result

$$\mathbf{J} = \mathbf{F}^+ = \frac{\rho}{g_1^2 + \ldots + g_n^2} \begin{pmatrix} g_1 \\ \vdots \\ g_n \end{pmatrix} \tag{5.36}$$

with

$$\rho := \frac{g_r^2\,\tau + \sigma_r^2}{g_r\,\tau} = g_r + \frac{\sigma_r^2}{g_r\,\tau}. \tag{5.37}$$

The factor $\rho \approx g_r$ depends only on properties of the robot and reflects the reconstructed robot gain, which is a consequence of the sampling of the data points for calculating the Feature Jacobian.

The first thing that is apparent is the absence of any control noise terms in the solution for $\mathbf{F}$. This means that without a noise dependent weighting, two sensors with the same information content but different noise levels are equally weighted in contrast to the direct solution. It is important to mention here, that for the direct solution it is possible to break down a single problem of $m \in \mathbb{N}$ DOFs to identify with $n \in \mathbb{N}$ control signals into m problems of identifying one DOF with n control signals. This is not possible for the solution via the Feature Jacobian $\mathbf{F}$, because the inversion of $\mathbf{F}$ cannot be separated into the inversion of single rows of $\mathbf{F}$ like it was done in the special case of (5.35). Likewise it is not possible to extend the solution shown in (5.36) to multiple DOFs that easy.

5.2.2.3. Inverse of Feature Jacobian with Weighting

It is possible to apply some sort of weighting when inverting the Feature Jacobian. We take a look at a per control signal weighting of the form

$$\mathbf{J} = \mathbf{W}\,(\mathbf{F}\,\mathbf{W})^{+} \tag{5.38}$$

with $\mathbf{W}$ a diagonal weight matrix of the form

$$\mathbf{W} = \begin{pmatrix} \frac{1}{w_1} & 0 & & 0 \\ 0 & \frac{1}{w_2} & & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & & 0 & \frac{1}{w_n} \end{pmatrix}. \tag{5.39}$$

After setting up $\mathbf{F}$, as in (5.33), and solving for $\mathbf{J}$, we get:

$$J = \frac{\rho}{\sum\limits_{i=1}^{n} \frac{g_i^2}{w_i^2}} \begin{pmatrix} \frac{g_1}{w_1^2} \\ \vdots \\ \frac{g_n}{w_n^2} \end{pmatrix} \tag{5.40}$$

An eye-catching fact is the similarity between the direct solution (5.30) and the current one, if setting the weight in $\mathbf{N}$ to the control signal standard deviations

$$w_1 = \sigma_1, \ldots, w_n = \sigma_n \tag{5.41}$$

to get a per control signal weighting based on the particular noise.

5.2.3. Noise of the Reconstructed Signal

A good possibility for comparing the Jacobian matrices obtained by the three different solving methods is to take a look at the noise of the reconstructed robot movement using the control signals. We have the noise of the single control signals as model parameters and the

Jacobian contains the rate at which each control signal participates in the result. So, the resulting noise can be calculated by

$$\sigma^2 = \sum_{i=1}^{n} j_{1,i}^2 \, \sigma_i^2. \tag{5.42}$$

For the direct calculation of the Jacobian, we get the term

$$\sigma_{\text{direct}}^2 = g_r^2 \, \frac{\sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}}{\left(\frac{1}{\tau} + \sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}\right)^2}. \tag{5.43}$$

If using the inverted Feature Jacobian the noise is

$$\sigma_{\text{feature}}^2 = \rho^2 \, \frac{\sum_{i=1}^{n} g_i^2 \sigma_i^2}{\left(\sum_{i=1}^{n} g_i^2\right)^2} \tag{5.44}$$

and if applying a control signal based weighting of $w_i = \sigma_i$ for all $i \in [1 \ldots n]$, the noise can be written as

$$\sigma_{\text{weighted}}^2 = \rho^2 \, \frac{1}{\sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}}. \tag{5.45}$$

If comparing the single noises, a very clear result can be deduced - the direct solution leads to the smallest noise in the reconstructed signal; inverting the Feature Jacobian leads to the noisiest signal. If applying the mentioned weights, the results lie right between:

$$\sigma_{\text{direct}}^2 < \sigma_{\text{weighted}}^2 < \sigma_{\text{feature}}^2. \tag{5.46}$$

The first inequality is easy to prove. If assuming $\rho \approx g_r$ (see (5.37)), an easy transformation leads to

$$\left(\sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}\right)^2 < \left(\frac{1}{\tau} + \sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}\right)^2. \tag{5.47}$$

The second inequality is more difficult to prove, but it leads to the term

$$\left(\sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}\right)\left(\sum_{i=1}^{n} g_i^2 \sigma_i^2\right) - \left(\sum_{i=1}^{n} g_i^2\right)^2 > 0 \tag{5.48}$$

which can be transformed into a sum of terms that are greater than zero and

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} g_i^2 \, g_j^2 \, \frac{(\sigma_i^2 - \sigma_j^2)^2}{\sigma_i^2 \, \sigma_j^2} > 0 \tag{5.49}$$

prove the postulated presumption.

Another interesting fact that can be derived from the terms for calculating the noise of the reconstructed signal is that adding another signal might enforce no higher resulting noise in the reconstructed signal. The easiest way to show this is to add another $(n+1)$-th control signal to a configuration with n control signals and to solve via the inverted Feature Jacobian with weighting:

$$\rho^2 \frac{1}{\sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}} \geq \rho^2 \frac{1}{\sum_{i=1}^{n+1} \frac{g_i^2}{\sigma_i^2}} \tag{5.50}$$

$$\sum_{i=1}^{n+1} \frac{g_i^2}{\sigma_i^2} \geq \sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2} \tag{5.51}$$

$$\frac{g_{n+1}^2}{\sigma_{n+1}^2} \geq 0. \tag{5.52}$$

We see from our model that it is guaranteed that the noise of the reconstructed signal does not increase if another control signal is added, no matter how noisy this new control signal may be. A similar behavior can be shown for the direct calculation, but this is more complicated so we just make an estimation. If replacing in (5.43) the term $\frac{1}{\tau}$ with zero (this is acceptable for a robot trace interval of sufficient size), we get a similar term to that of the inverted Feature Jacobian with weighting, with the same result. For the inverted Feature Jacobian without weighting, the desired property cannot be proven. When adding a new, very noisy control signal to an existing configuration, the solution using the inversion of the Feature Jacobian might lead to an increasing of the noise of the reconstructed signal, which is a serious drawback.

5.2.4. Coefficient of Determination

In Chapter 4.4.2 we have introduced the coefficient of determination as a quality criterion. We will now try to calculate the COD directly from the model parameters. The COD depends on the residuals. The error matrix has been defined in (4.44) as:

$$\mathbf{E} = \mathbf{S}\,\mathbf{J} - \mathbf{R}. \tag{5.53}$$

In our case, the error matrix has just one single column. We determine the effective gain and noise of the residuals as

$$g_e = \left(\sum_{i=1}^{n} g_i\, j_{i,1}\right) - g_r \tag{5.54}$$

$$\sigma_e^2 = \left(\sum_{i=1}^{n} \frac{j_{i,1}^2}{\sigma_i^2}\right) + \sigma_r^2. \tag{5.55}$$

Using the definition of the COD in (4.49) and the rule for calculating the square sum of a signal with certain gain and noise in (5.29), we get for the COD

$$R^2 = \frac{g_e^2\,\tau\,k + \sigma_e^2\,k}{g_r^2\,\tau\,k + \sigma_r^2\,k} = \frac{g_e^2\,\tau + \sigma_e^2}{g_r^2\,\tau + \sigma_r^2}. \tag{5.56}$$

If calculating the COD for the direct calculation depending on the model parameters, we get

$$R^2 = \frac{g_r}{g_r + \frac{\sigma_r^2}{g_r \tau}} \frac{\sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}}{\frac{1}{\tau} + \sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}} = \frac{g_r}{\rho} \frac{\sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}}{\frac{1}{\tau} + \sum_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}}. \tag{5.57}$$

The robot dependent first part of the term is near one if σ_r is very small, and it tends more and more to zero if σ_r is growing. The signal dependent part is near zero for small sum terms and near one for greater sum terms. The sum term increases with increasing signal gains and decreasing signal noises. The COD is near one if both parts are near one, and the COD is near zero if one of the two parts is near zero.

For the solution via the Feature Jacobian (unweighted and weighted), the COD is less compact to display and less easy to interpret. For reasons of completeness, here is the result for the inverted Feature Jacobian

$$R^2 = \frac{g_r^2\,\tau - \sigma_r^2}{g_r^2\,\tau} - \frac{g_r^2\,\tau + \sigma_r^2}{g_r^2\,\tau^2}\,\frac{\sum\limits_{i=1}^{n} g_i^2\,\sigma_i^2}{\left(\sum\limits_{i=1}^{n} g_i^2\right)^2} \tag{5.58}$$

and for the inverted Feature Jacobian with a control signal based weighting of $w_i = \sigma_i$ for all $i \in [1 \ldots n]$:

$$R^2 = \frac{g_r^2\,\tau - \sigma_r^2}{g_r^2\,\tau} - \frac{g_r^2\,\tau + \sigma_r^2}{g_r^2\,\tau^2}\,\frac{1}{\sum\limits_{i=1}^{n} \frac{g_i^2}{\sigma_i^2}}. \tag{5.59}$$

•

5.3. Conclusion

In this chapter we examined how different control signal properties influence the weighting of those control signals in the Jacobian matrix for different solution approaches. All these analyses were model-based. An important result was that for a weighting with a weight matrix containing the (estimated) reciprocal standard deviations of the control signals, we received a good noise-weighting. The quality of this solution was between the inversion of the Feature Jacobian and the direct calculation (in terms of the noise of the reconstructed signal). Some of the experiences made in practice could be confirmed using the model, such as the possibility to decrease the quality of the system identification by adding a noisy control signal to an existing setup and using the inverted Feature Jacobian as the Jacobian.

The equations developed in this chapter are good for clarifying the properties of the solution methods. However, as some experiments confirmed, their usage in practice is limited. If you take trace data from a real world experiment (see Figure 5.3) and try to deduct the model parameters from the data, you get different Jacobian matrices if calculating it from the model parameters ($\mathbf{J}_1$) or from the trace data directly ($\mathbf{J}_2$). This is because reality is more complex than our limited model. The equality $\mathbf{J}_1 = \mathbf{J}_2$ is only valid in the case that the trace data has been generated using the same model (compare with Figure 5.1).

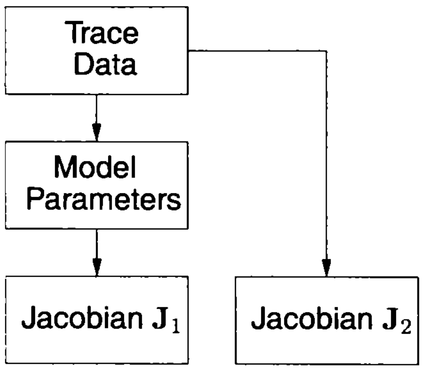


Figure 5.3: Practical Usage of Model

6. HYSTERESIS

In this chapter, we make an in-depth analysis of the hysteresis-effect that has been detected in the trace data in the chapter before. First, we check the official definition of hysteresis before we start the analysis of the origin of this effect - mechanical robot effects and delays in the sampling process. After that, we check methods to detect and compensate the hysteresis in trace data and discuss the effect of the hysteresis on the control process.

6.1. Introduction and Definition

When recording a trace, the robot is moved forward and backward in a single DOF while the control signals are continuously recorded. If plotting the samples of a single control signal against the robot position, we would expect the resulting trace data points to lie around a line through the point of origin according to occurring noises. The slope of the line depends on the sensitivity of the geometrical features observed by the control signal for the robot DOF, as shown in Figure 6.1 on the left-hand side. The results, though, obtained from practical experiments show a very different result, as shown in Figure 6.1 on the right-hand side where we see a non-linearity behavior of hysteresis-type.

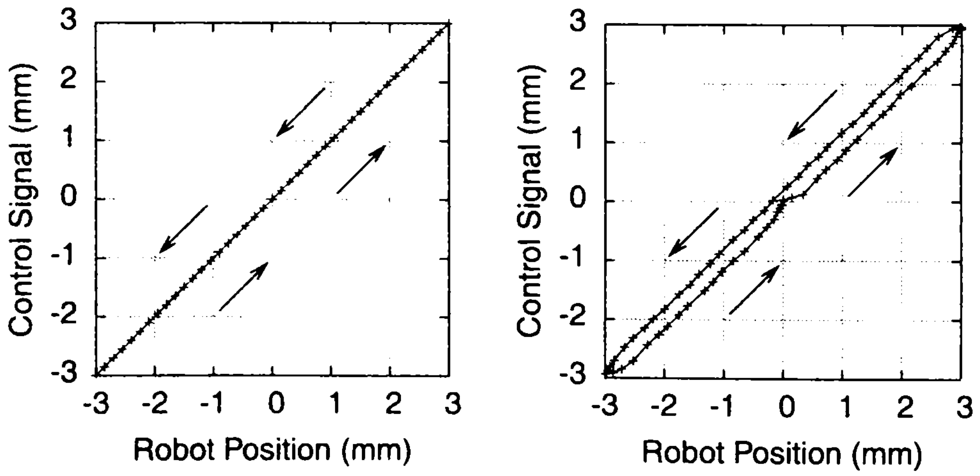


Figure 6.1: Ideal Trace and Experimental Result

The analysis of this behavior is the subject of this chapter. A shorter version of this research has been published at a conference, and the paper can be found under [76].

To clarify the term *hysteresis*, we take a look at an official definition given by a norm from the American National Standards Institute (ANSI, see [3]):

Hysteresis is that property of the element evidenced by the dependence of the value of the output, for a given excursion of the input, upon the history of prior excursions and the direction of the current traverse. (...) It is usually determined by subtracting the value of dead-band from the maximum measured separation between upscale-going and downscale-going indications of the measured variable (during a full-range traverse, unless otherwise specified) after transients have decayed.

As with the norm, we define a hysteresis as the difference of the measured values - in our case the control signal - between the forward and the backward movement, as shown in Figure 6.2. The figure shows a zoomed-in view around the point of origin of the trace shown in Figure 6.1 on the right side. The hysteresis size h is defined as half of the difference of the control values - the control signal value from the forward movement minus the one from the backward movement. This way, the hysteresis is a signed value and, in the case of this example, h is negative. It is useful to define hysteresis size in the control signal space because without the system identification, this hysteresis is difficult to interpret in terms of degrees of freedom and it is first of all a value bound to a certain control signal.

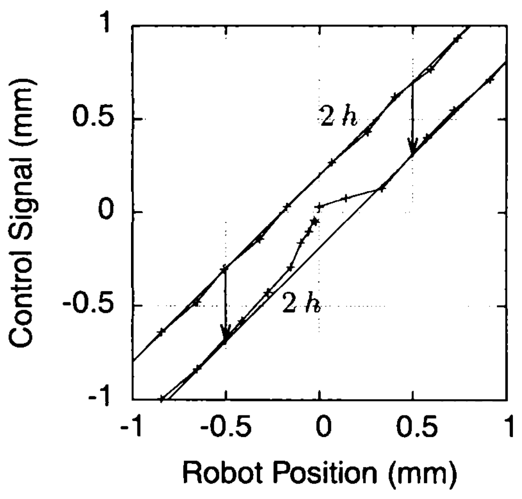


Figure 6.2: Definition of Hysteresis Size h

Because there is a hysteresis size h for every DOF and every control signal, we collect all the values in a single matrix $\mathbf{H}_s$ of size $m \times n$ (the same size as the Feature Jacobian) The subscript denotes that the hysteresis is measured in signal space. Of course, we want to see the hysteresis effects in the robot coordinate system, so we calculate

$$\mathbf{H}_r = \mathbf{H}_s \mathbf{J} \tag{6.1}$$

The matrix $\mathbf{H}_r$ has the size $m \times m$. Its rows contain vectors that describe the size of the hysteresis for the traces in each DOF and the columns the size of the hysteresis in each DOF.

Taking a look at the result of a real world experiment with eight control signals controlling a standard six axis industrial robot in three DOFs, we get the following result for the hysteresis in the control signal space:

$$\mathbf{H}_s = \begin{pmatrix} 0.041 & 0.061 & -0.102 & 0.072 & -0.193 & -0.046 & -0.219 & -0.002 \\ 0.087 & -0.238 & 0.036 & -0.242 & 0.021 & 0.212 & -0.044 & 0.167 \\ 0.136 & 0.056 & 0.108 & 0.017 & -0.103 & 0.000 & -0.059 & -0.005 \end{pmatrix}. \tag{6.2}$$

The data is difficult to interpret without further information, so we have included the Feature Jacobian:

$$\mathbf{F} = \begin{pmatrix} -0.046 & 0.001 & 0.567 & 0.087 & 0.641 & 0.069 & 1.000 & 0.102 \\ -0.550 & 0.833 & -0.435 & 0.835 & 0.106 & -0.988 & 0.055 & -0.987 \\ -0.804 & -0.555 & -0.679 & -0.505 & 0.674 & 0.120 & 0.272 & -0.107 \end{pmatrix}. \tag{6.3}$$

One coherence that is easily spotted is that if the sensitivity of a signal for a certain DOF is high according to the Feature Jacobian, then the absolute value of the hysteresis is high too. By repeating the same experiment multiple times, we get a stable result for $\mathbf{H}_s$, which indicates that we are observing a certain non-volatile effect. For further analysis, we transform $\mathbf{H}_s$ into the robot coordinate space:

$$\mathbf{H}_r = \begin{pmatrix} -0.212 & 0.017 & -0.069 \\ -0.041 & -0.208 & 0.070 \\ -0.015 & -0.011 & -0.138 \end{pmatrix}. \tag{6.4}$$

The values have an absolute size up to tenths of a millimeter, which is a little bit greater than the expected range because the repeating accuracy of the used industrial robot is a little smaller (in this case $0.15\,\mathrm{mm}$ for a "KUKA KR60 HA") and the hysteresis is not the only effect responsible for the limited absolute accuracy of the robot. See Section 2.4 for an estimation of the dimensions of robot hysteresis. All values are different from zero, which is acceptable because if assuming mechanical robot properties like gear slackness as a reason for the hysteresis effect, the hysteresis takes place in the joints of the robot and therefore usually influences the movement in more than one DOF in each of the Cartesian coordinate systems of the robot. This is a direct consequence of the robot kinematics introduced in Section 2.2. It is notable that the values in the diagonal of $\mathbf{H}_r$ are significantly greater than those outside the diagonal. This is suspicious, but later on we will find an explanation for this.

6.2. Analysis of Hysteresis Effects

Considering the experimental results reviewed so far, we could take the mechanical properties of the robot as the only reason for the hysteresis effects. Nevertheless, repeating the previous experiment with exactly the same configuration and with different robot speeds during the trace, we experience a surprise - the size of the hysteresis effect changes with the changing robot trace speed v:

$$\mathbf{H}_s = \mathbf{H}_s(v) \quad \text{and} \quad \mathbf{H}_r = \mathbf{H}_r(v). \tag{6.5}$$

We analyze this behavior by means of an example. Consider we run a trace in all three translational degrees of freedom of an industrial robot. We have four control signals; each observing a different geometrical feature with different sensitivity. The Feature Jacobian, which is speed independent, is determined as

$$\mathbf{F} = \begin{pmatrix} -0.023 & -0.006 & 0.542 & -0.033 \\ -0.547 & -0.796 & -0.392 & -0.867 \\ -0.791 & 0.491 & -0.741 & 0.457 \end{pmatrix}. \tag{6.6}$$

For a complete trace cycle, the robot is moved in a DOF in positive and negative directions. This leads to phases where the robot accelerates or retards. When talking about the robot speed, we refer to the speed of the robot in phases of linear movement - as marked with arrows in Figure 6.3 which shows the robot moving along a full trace cycle. The acceleration in the figure has been

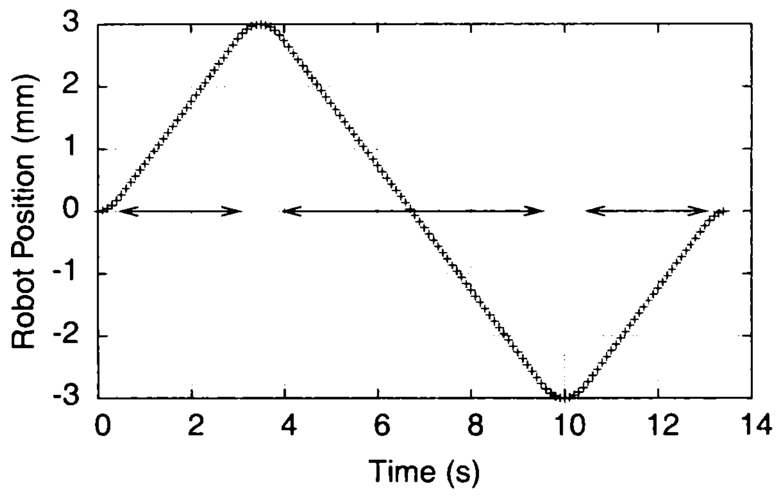


Figure 6.3: Robot Movement Speed

chosen with $2\,\frac{\text{mm}}{\text{s}^2}$ lower than typical to better illustrate the robot's behavior. We vary the robot speed between $0.2\,\frac{\text{mm}}{\text{s}}$ and $2.4\,\frac{\text{mm}}{\text{s}}$ with a step-size of $0.2\,\frac{\text{mm}}{\text{s}}$, which produces 12 samples in total.

First, we take a look at the occurring hysteresis in the control signal space and therefore at the contents of the matrix $\mathbf{H}_s$ at different robot speeds. In Figure 6.4, on the left side we see the hysteresis in control signal space for the trace in X direction (first row of $\mathbf{H}_s$) and on the right side for the trace in Y direction (second row of $\mathbf{H}_s$) over the robot speed.

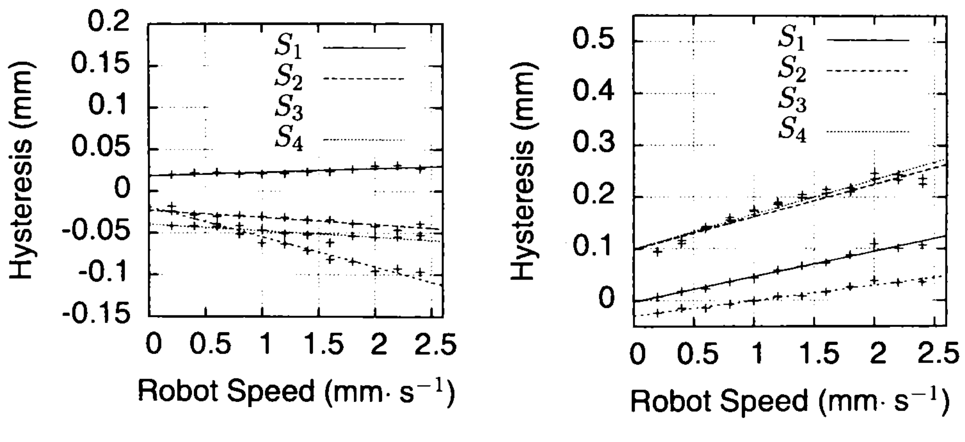


Figure 6.4: Hysteresis in Control Signals for X and Y

As the next step, we plot the contents of $\mathbf{H}_r$; representing the hysteresis in robot space. The result can be seen in Figure 6.5. On the left side we see all diagonal elements of $\mathbf{H}_r$ over the speed approximated with a solid line and all elements outside the diagonal approximated with a dashed line. The diagonal elements, which denote the hysteresis in the main trace directions, are also shown on the right side along with the key identifying the single DOFs.

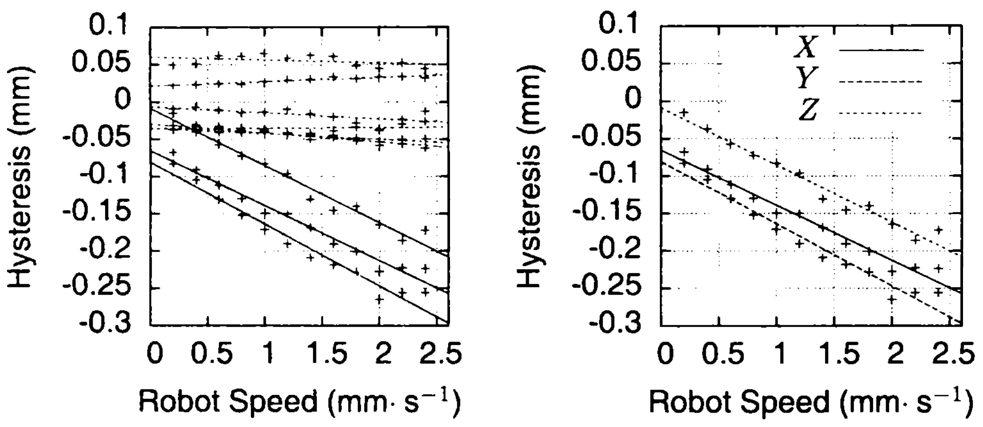


Figure 6.5: Hysteresis in Robot Coordinates

The assumption of a linear coherence between robot speed and hysteresis size is clearly visible. It can be strengthened by more examples with different configurations and over wider speed ranges. This linear behavior can be found for all elements of $\mathbf{H}_s$ and because of the linearized coherence in (6.1) for $\mathbf{H}_r$ too. Therefore, we assume a linear model with

$$\mathbf{H}_s(v) = \Delta\mathbf{T}_s\, v + \mathbf{H}_{s0} \tag{6.7}$$

$$\mathbf{H}_r(v) = \Delta\mathbf{T}_r\, v + \mathbf{H}_{r0}. \tag{6.8}$$

In the linear model $\mathbf{H}_{s0}$ and $\mathbf{H}_{r0}$ are some hysteresis offset that is added to the speed dependent hysteresis component described by $\Delta\mathbf{T}_s$ and $\Delta\mathbf{T}_r$. The elements in $\mathbf{H}_{s0}$ and $\mathbf{H}_{r0}$ have the unit millimeters, the elements in $\Delta\mathbf{T}_s$ and $\Delta\mathbf{T}_r$ the unit seconds. This way the products of $\Delta\mathbf{T}_s\, v$ and $\Delta\mathbf{T}_r\, v$ have the unit millimeters, too.

In the next two sections, we will take a look at the model parameters determined from the experimental data of our example configuration. On the basis of these results, we analyze the physical meaning of the model parameters and their practical relevance.

6.2.1. Timing Issues

For the model parameters $\Delta\mathbf{T}_s$ and $\Delta\mathbf{T}_r$, we get from our experimental data the following results:

$$\Delta\mathbf{T}_s = \begin{pmatrix} 0.004 & -0.008 & -0.036 & -0.008 \\ 0.049 & 0.064 & 0.029 & 0.067 \\ 0.060 & -0.040 & 0.052 & -0.033 \end{pmatrix} \tag{6.9}$$

$$\mathbf{\Delta T_r} = \begin{pmatrix} -0.074 & 0.006 & -0.008 \\ -0.012 & -0.083 & -0.004 \\ -0.008 & 0.001 & -0.077 \end{pmatrix} \tag{6.10}$$

In our model, the matrices $\mathbf{\Delta T_s}$ and $\mathbf{\Delta T_r}$ represent the dynamic component of the hysteresis and have the physical quantity of time. This time is the delay or lagging between the sampling of the robot position and the sampling of the raw sensor data that is evaluated to finally obtain the control signals. To test this assumption, we set up a simulation where we move a robot with an attached sensor in a way as if executing a trace and collect robot positions and sensor samples. The sensor has a sensitivity for the robot movement of one. After artificially retarding the sampling of the sensor values for $500\,\mathrm{ms}$, we get the result shown in Figure 6.6. The delay, together with the robot trace speed of $1\,\frac{\mathrm{mm}}{\mathrm{s}}$, leads to a hysteresis of $0.5\,\mathrm{mm}$. Increasing the robot speed and constant sample delay, the hysteresis increases linearly - just as observed in the real world experiment.

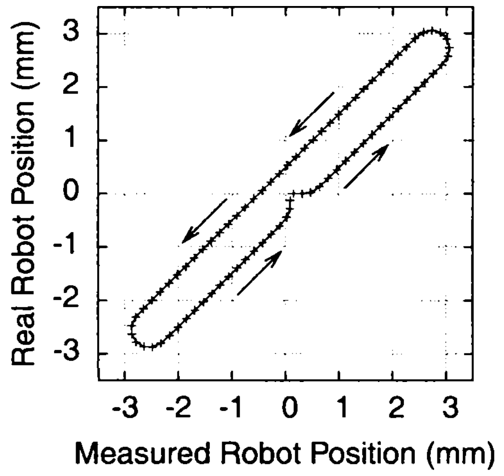


Figure 6.6: Hysteresis caused by Sample Delays

If the sample delay is positive, then the sampling of the robot position takes place before the sampling of the sensor raw data. If the delay is negative, the sensor data sampling happens first, as shown in Figure 6.7. So, all sample delays are relative to the robot position sampling, which makes sense because the sampling of the raw data for the considered control signal may happen at different times and this way we take the robot position sampling time as a reference for all sensor sampling times.

The matrix $\mathbf{\Delta T_r}$ describes the speed-dependent part of the hysteresis in the robot space. The speeds occurring during the trace are vectors pointing in the directions of the basis vectors of the robot coordinate system. It is, consequently, not surprising that the elements outside the diagonal of $\mathbf{\Delta T_r}$ are approximately zero - the first line of the matrix describes the speed-dependent component of the first trace which was done in the first DOF with a speed vector, with only the first element non-zero. Assuming a software system executing the trace with a single constant mean sample delay Δt for the complete time over the whole trace process as in our experiment, we are able to simplify our linear model by writing

$$\mathbf{\Delta T_r} = \mathbf{I}\,\Delta t \tag{6.11}$$

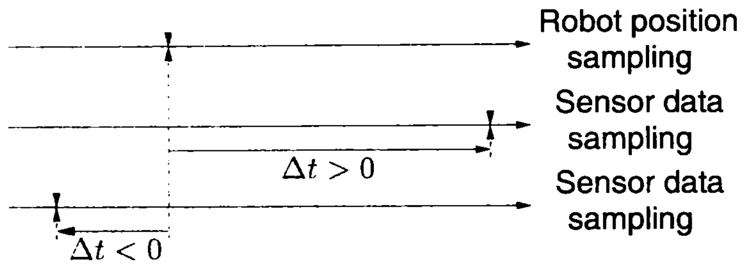


Figure 6.7: Sign of Sample Delay

and

$$\mathbf{H_r}(v) = \mathbf{I}\, v\, \Delta t + \mathbf{H_{r0}}, \tag{6.12}$$

where $\mathbf{I}$ is the identity matrix. We now know that the speed dependent part affects only the diagonal of $\mathbf{H_r}$, which explains its distinct diagonal mentioned previously when referring to the greater • diagonal elements suspicious. For our experiment, we get a sampling delay Δt of $-78\,\mathrm{ms}$.

If assuming different sample delays per sensor signal and per trace, it would be interesting to calculate those values. The basis for this calculation is the matrix $\mathbf{\Delta T_s}$. Comparing these values with the values of the Feature Jacobian $\mathbf{F}$, we see that if the sensitivity of a signal for a certain DOF is high, the value in $\mathbf{\Delta T_s}$ is high too and vice versa. The reason for this is clear - the speed vector points in the direction of the movement and the sensor signal senses it only at the rate that is defined by the according element of $\mathbf{F}$. To compensate for this we can divide each element of $\mathbf{\Delta T_s}$ by the according element of $\mathbf{F}$. The result for our example is:

$$\begin{pmatrix} -0.190 & 1.323 & -0.066 & 0.232 \\ -0.089 & -0.080 & -0.075 & -0.078 \\ -0.076 & -0.082 & -0.070 & -0.072 \end{pmatrix}. \tag{6.13}$$

When interpreting the results, some caution is advised. Most elements show a sample delay that is similar to the one in the diagonal of $\mathbf{\Delta T_r}$, but some values diverge. These are values where the sensitivity of the signal for the DOF is too small. In this case the value in $\mathbf{F}$ is nearly zero and so a division by zero is pending. Obviously, we cannot determine the sample delay for a sensor signal at a certain DOF if the sensor is not sensitive for that DOF, which is indicated by a near-zero value in $\mathbf{F}$.

6.2.2. Mechanical Causes

The component of the hysteresis that is independent from the robot speed can be found in the matrices $\mathbf{H_{s0}}$ and $\mathbf{H_{r0}}$. The matrices' values for the experimental data are:

$$\mathbf{H_{s0}} = \begin{pmatrix} 0.018 & -0.023 & -0.019 & -0.040 \\ -0.003 & 0.098 & -0.030 & 0.099 \\ 0.027 & 0.021 & 0.017 & 0.031 \end{pmatrix}. \tag{6.14}$$

$$\mathbf{H_{r0}} = \begin{pmatrix} -0.065 & 0.021 & -0.035 \\ -0.029 & -0.081 & 0.060 \\ -0.006 & -0.036 & -0.009 \end{pmatrix}. \tag{6.15}$$

BUPT

These matrices contain the hysteresis that would occur if we were able to execute a zero speed at trace, so that the sample delay would not play any role. The causes, finally, for this hysteresis are the mechanical effects taking place in the robot gears, as introduced previously in Section 2.4. Creating a similar simulation like before with a robot moving a hand-mounted sensor, but this time with a gear dead-zone of $0.5\,\mathrm{mm}$, we receive the result shown in Figure 6.8. The hysteresis size is independent from the robot speed.

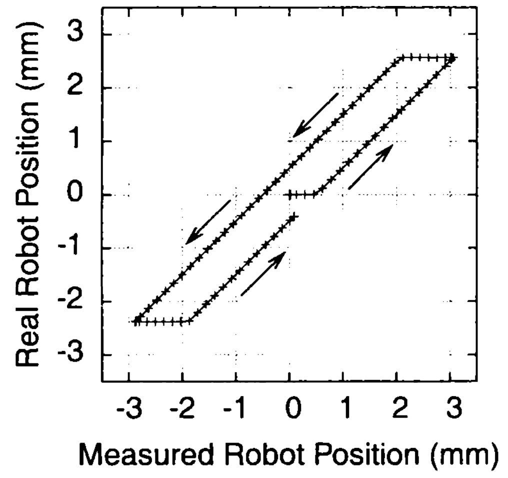


Figure 6.8: Hysteresis Caused by Mechanical Effects

Note that the shape of the curve is different than that caused by the sample delays in Figure 6.6. If the robot could be stopped during its trace for a short time, the curve would be the same if only a mechanical hysteresis were present. In the case of an existing sample delay this is not the case.

In practice, the matrix $\mathbf{H}_{\mathrm{ro}}$ has a higher relevance because it describes the mechanical hysteresis in terms of robot coordinates is easier to interpret. As mentioned in the introduction, the reason for the values outside the diagonal are that the mechanical hysteresis takes place in the robot's joints and influences more than one DOF than just the DOF the trace is executed in. To demonstrate this effect we take a look at another experiment. The setup is as shown in Figure 6.9. We have four distance sensors at the robot tool, measuring the distances to a simple box as a work object. Each sensor has a sensitivity of nearly one for one of the translational DOFs and nearly zero for all other DOFs.

We execute a trace in the X direction and observe the sensor values for sensors 2 and 3. The result is shown in Figure 6.10, with sensor 2 on the left and sensor 3 on the right. Because the trace is executed in X, the speed vector shows in that direction too. So, independent from the speed of the robot and the size of the sample delay the hysteresis caused by a sample delay is zero. With a sensitivity of about one for sensors 2 and 3 (the mechanical mounting of sensors 2 and 3 should be quite accurate), we observe here directly the movement of the robot in Y and Z while actually executing a move in X. The hysteresis sizes of about $0.15\,\mathrm{mm}$ in Y and $0.02\,\mathrm{mm}$ in Z are quite remarkable.

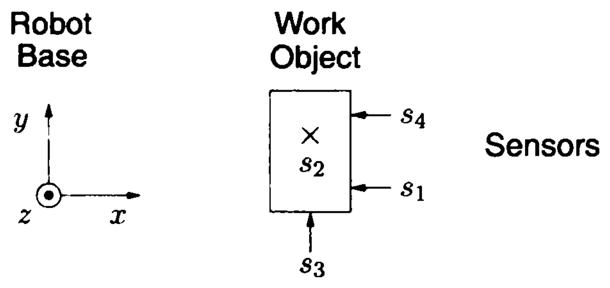


Figure 6.9: Experimental Setup

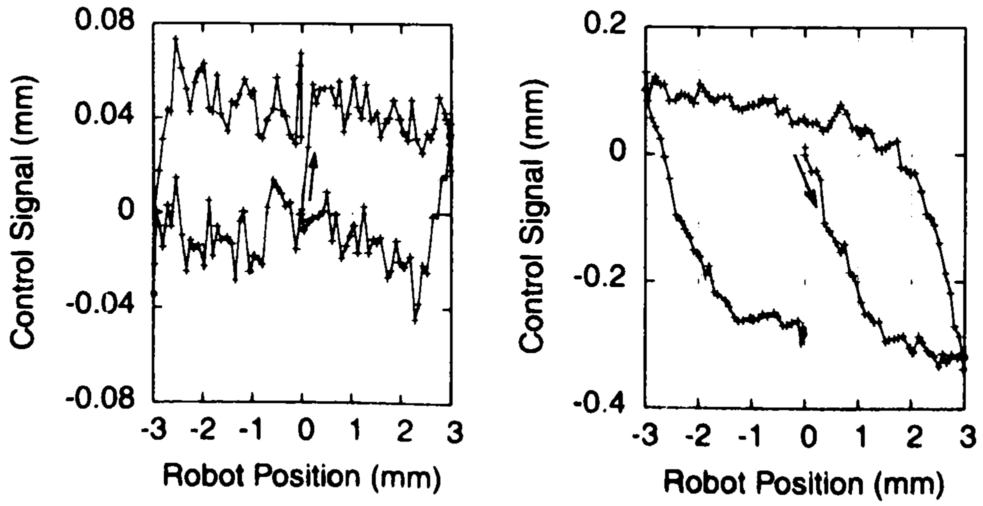


Figure 6.10: Hysteresis for Trace in X

6.2.3. Hysteresis Stability

In this short section, we check the stability of the hysteresis in terms of the signal configuration. What we expect is that the size of the hysteresis is independent from the signal configuration as long as the signals carry enough information to reconstruct the robot movement. We use the same experimental setup as used in the previous sections to control a robot in three translational DOFs. However, this time we vary the number of control signals that we use for the system identification and the hysteresis determination from between three and seven signals. Each of these control signals adds new information about a new geometrical feature. The results are shown in Table 6.1.

Number of	Mean Sample	Robot Hysteresis (mm)		
Signals	Delay (s)	X	Y	Z
3	-0.078	-0.064	-0.082	-0.011
4	**-0.078**	**-0.065**	**-0.081**	**-0.009**
5	-0.079	-0.064	-0.079	-0.005
6	-0.078	-0.064	-0.074	-0.006
7	-0.078	-0.055	-0.075	-0.007

Table 6.1: Stability of Hysteresis

The first column shows the number of control signals used, the second column the average sampling delay as the average of the diagonal of the matrix $\Delta \mathbf{T}_\mathbf{r}$, and the last three columns contain the diagonal of $\mathbf{H}_{\mathbf{r}0}$. The second row is marked because it contains the results of a signal configuration with four signals, as used before. The sample delay remains very stable at $\sim 78\,\mathrm{ms}$. The hysteresis in the robot's coordinates is less stable, but still at a sufficient level.

6.3. Hysteresis Handling

6.3.1. Hysteresis Determination

So far, we have seen the definition of the hysteresis. Now we develop a method for determining the hysteresis in trace data. We concentrate on the trace data of a single control signal and a single DOF to extract h. For a more extensive trace, this method has to be applied for each control signal and each DOF to obtain the complete result of $\mathbf{H}_\mathbf{s}$. The procedure of the algorithm can be comprehended using Figure 6.11.

1. In the first step, we remove all trace data points from the trace that results from a non-linear movement of the robot. This is the case at the beginning and the end of the trace and at the turning points of the robot path. The size of the intervals in which the robot moves non-linearly can be determined by observing the robot speed if logging timestamps with the trace data.

2. We calculate an approximate line through the trace data with slope g and intersection c.

3. The trace data is then separated into two sets - samples that were sampled with the robot moving in a forward direction and those that result from a backward movement. Through each data set we place an approximate line with the previously calculated slope g and with intersection points h_f and h_b for the forward and backward movement, respectively.

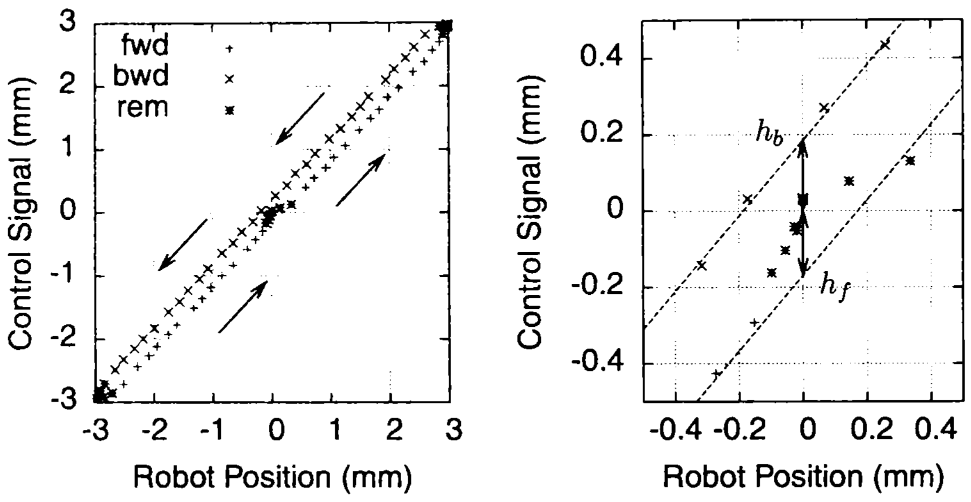


Figure 6.11: Hysteresis Determination from Trace Data

4. The resulting hysteresis h can now be determined as

$$h = \frac{h_f - h_b}{2} \tag{6.16}$$

This is the basic approach for hysteresis determination and experience shows that it performs very well for a great variety of real world testing data. A probable source of small inaccuracies might arise from trace data irregularities in the form of outliers, as shown, for example, in Figure 4.2 (c), and we use least squares regression techniques to calculate the approximate lines in the approach described above. This is because the result of the least squares regression is quite sensitive to outliers. Although it is advisable to find the source of the problem causing the outliers and to repair it, it is still interesting to harden the hysteresis determination for cases like this by replacing the least squares regression with a robust regression method.

The most common method for robust regression is the *M-estimation*. The original idea was published in [36] and its application to regression in [37]. A regression takes a number of $n \in \mathbb{N}$ observed data tuples of an independent variable x_i and of a dependent variable y_i and uses a model to predict values f_i for y_i on the basis of the independent variable x_i. The errors or residuals for each sample tuple are given by $e_i = y_i - f_i$. The M-estimation applies a function known as the *objective function* $\rho(e_i)$ to each of the residuals and minimizes its sum over all residuals

$$\sum_{i=1}^{n} \rho(e_i). \tag{6.17}$$

For $\rho(e_i) = e_i^2$, the sum is identical to the least squares approach. Nevertheless, while the least squares approach uses an equal weight for the summed-up residuals, the M-estimation defines a *weight function* $w(e_i)$ to apply an individual weight to each sample. Because of a circular dependency of weights, residuals and estimated model parameters, the result cannot be determined directly. This is done in an iterative algorithm known as *iteratively re-weighted least squares* (*IRLS*). There is a set of available objective functions and weight functions of favorable behavior, such as the Huber or the Tukey bi-square estimators.

In Figure 6.12, we see a comparison between the performance of a least squares regression and an M-estimation with a Huber weight function using a concrete example. The latter shows much less influence on the group of outliers. For a very good introduction to the theory of robust estimation, take a look at the book [26].

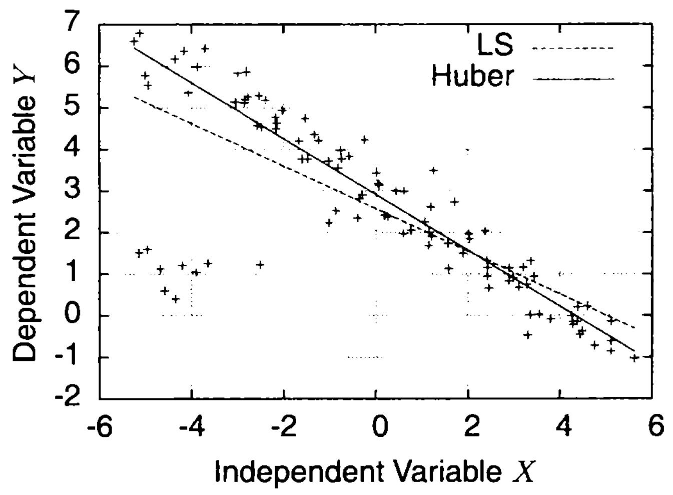
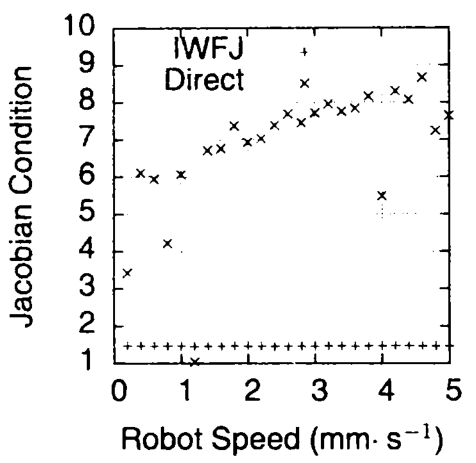


Figure 6.12: Robust Regression with M-estimation

6.3.2. Hysteresis Compensation

With an increasing robot trace speed, the quality of the trace data decreases because of the increasing hysteresis size caused by the sample delay. This might lead to problems when identifying the system on the basis of the trace data with certain methods. Figure 6.13 illustrates this problem by showing results of a real world experiment. Both plots show the condition of the Jacobian as result of a system identification on the trace data over the trace speed of the robot. The condition of the Jacobian is the ratio of the largest and the lowest singular value and gives us an idea about the stability of the identified system. The different solution methods used are derived via the inverted Feature Jacobian with weighting and the direct solution. In the left plot we see the result using the raw trace data, and on the right side using trace data with applied hysteresis compensation. Independent from the hysteresis compensation, the solution via the inverted weighted Feature Jacobian is indifferent against changes to the robot speed. However, the direct solution, which has . tability issues as discussed before, produces solutions of growing instability with increasing robot speed. A hysteresis compensation can lower these effects and make the stability of the solution more predictable due to the lower variability of the condition.

This compensation is quite easy to apply if the hysteresis size has been determined as described in the previous section. After separating the trace data into the set containing the trace data sampled when moving forward and the trace data sampled when moving backward, we subtract the h_f from the control signal values of the first set and h_b from the control signal values of the second set. The new trace data contains the union of both sets without the previously removed samples. Of course, this must be done - as with the hysteresis determination - for each control signal and each DOF separately. This way, the hysteresis is artificially set to zero.



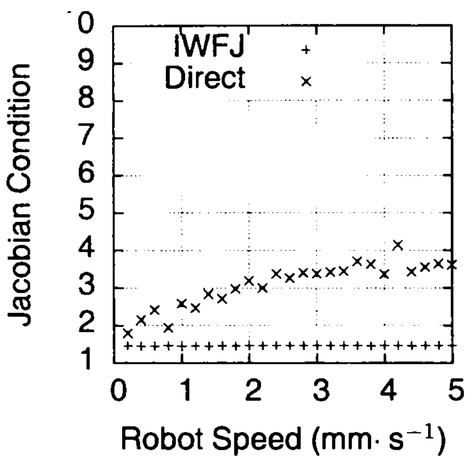


Figure 6.13: Robot Speed vs. Jacobian Condition

Finally, we check the behavior of the residuals that occur with or without hysteresis compensation for the direct solution and the solution via the Feature Jacobian. The example configuration contains eleven control signals to control five DOFs. First, we see the results for the residuals for the Y direction based on a trace of $k = 688$ samples over all DOFs without hysteresis compensation in Figure 6.14. In the solution via $\mathbf{F}$, we have a residual square sum of $17.732\,\mathrm{mm}^2$ (upper figure) and for the direct solution of $13.581\,\mathrm{mm}^2$ (lower figure). This is all right - the direct solution is least-squares optimal - but looking at the figure, the direct solution looks like the noisier one. Because of the hysteresis, the direct solution tries to reduce the high residuals by choosing a weighting in $\mathbf{J}$ that minimizes the squared residuals and not their noise. The situation with hysteresis-compensated trace data ($k = 647$ samples left) is shown in Figure 6.15. The residual square sum for the solution via $\mathbf{F}$ (upper figure) is $0.615\,\mathrm{mm}^2$ and for the direct solution (lower figure) $0.494\,\mathrm{mm}^2$. Both residual graphs look quite similar, but now it is the solution via $\mathbf{F}$ that has the noisier residuals.

6.3.3. Dual Speed Trace

We have discussed before how to determine the hysteresis in trace data and how to distinguish the effects from mechanical robot properties and delays in the sensor subsystem. It requires the execution of a series of traces at different speeds and to fit the data to a linear system. In practice, this is quite a time consuming task. A simple idea how to simplify things and to do this in a single trace is to vary the robot speed during the trace.

To have a trace data set of sufficient size, we choose to use two different robot speeds during the trace. One way to do this is to use one speed for the forward and one speed for the backward movement. This would lead to two approximate lines through the data from the forward and backward movement, whose offsets have to be considered relative to zero. That said, since we use data relative to nominal values, it is no certainty that zero is the exact value the offsets should be referred to. To avoid those problems we use different speeds in the positive and the negative robot movement range of the trace, just as shown in Figure 6.16. The speed in the first half of the trace was $0.1\,\frac{\mathrm{mm}}{\mathrm{s}}$ and in the second half with $1.0\,\frac{\mathrm{mm}}{\mathrm{s}}$ it was ten times as high. You can clearly see that the hysteresis in the first half is smaller than in the second half and that the density

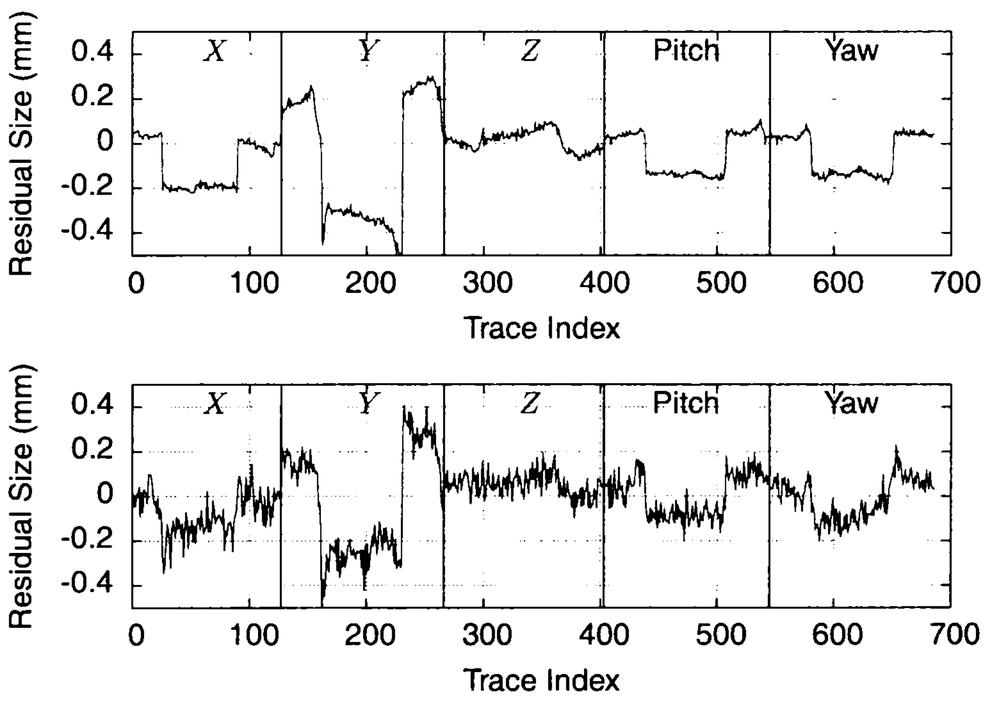


Figure 6.14: Residuals without Hysteresis Compensation (via **F**/Direct)

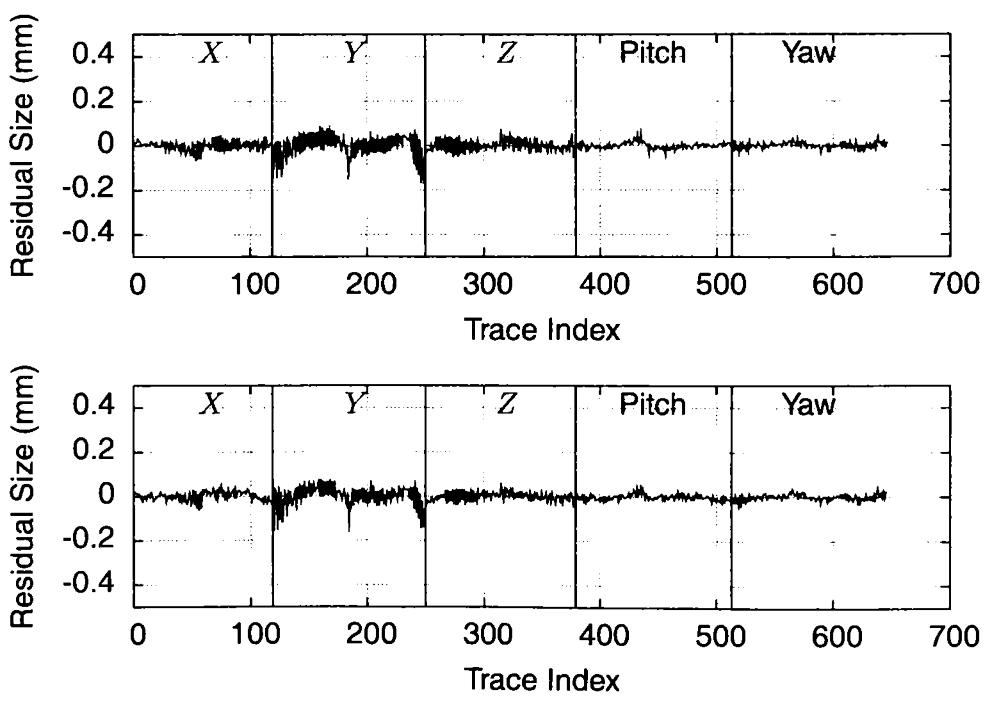


Figure 6.15: Residuals with Hysteresis Compensation (via **F**/Direct)

of the sample points is lower when the robot moves faster.

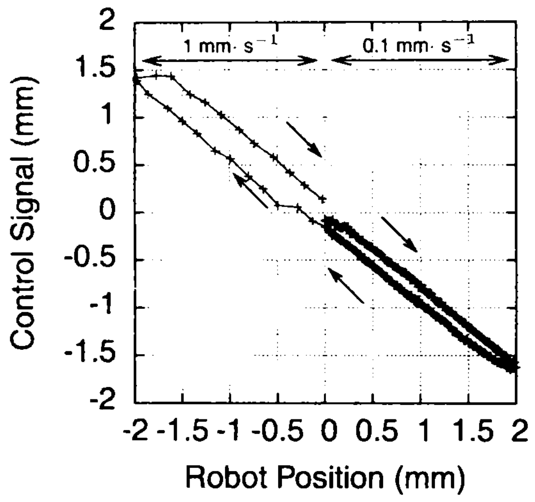


Figure 6.16: Dual Speed Trace

We apply this dual speed strategy to the experimental setup used before to obtain new data. By a separate analysis of the data of the first and the second half of the trace, we get the two matrices $\mathbf{H_s}(v_1)$ and $\mathbf{H_s}(v_2)$ with $v_1 = 0.1\,\frac{\mathrm{mm}}{\mathrm{s}}$ and $v_2 = 1.0\,\frac{\mathrm{mm}}{\mathrm{s}}$. This gives us the minimum amount of data to solve for the linear model, which means determining the approximate line by only two points. The results are as follows:

$$\mathbf{\Delta T_s} = \begin{pmatrix} 0.008 & -0.015 & -0.045 & -0.032 \\ 0.054 & 0.103 & 0.037 & 0.094 \\ 0.074 & -0.043 & 0.062 & -0.027 \end{pmatrix} \tag{6.18}$$

$$\mathbf{\Delta T_r} = \begin{pmatrix} -0.098 & 0.021 & -0.022 \\ 0.001 & -0.111 & 0.009 \\ -0.014 & -0.009 & -0.089 \end{pmatrix} \tag{6.19}$$

$$\mathbf{H_{s0}} = \begin{pmatrix} 0.017 & -0.016 & -0.012 & -0.030 \\ -0.006 & 0.072 & -0.035 & 0.078 \\ 0.018 & 0.017 & 0.008 & 0.027 \end{pmatrix}. \tag{6.20}$$

$$\mathbf{H_{r0}} = \begin{pmatrix} -0.052 & 0.014 & -0.029 \\ -0.039 & -0.060 & 0.051 \\ -0.011 & -0.028 & -0.003 \end{pmatrix}. \tag{6.21}$$

Comparing these results with (6.9) and (6.10) and (6.14) and (6.15), respectively, which are based on a complete sample set of twelve different speeds, we see that we get a rough estimation on the basis of our dual speed trace data. However, since a linear approximation with only two samples does not allow an error estimation, the execution of two dual-traces would lead to a sufficient amount of experimental data for a hysteresis analysis at the cost of an additional trace run.

6.4. Hysteresis and Control

Despite of all our efforts to cancel the hysteresis effects in the trace data, we should not forget that the hysteresis effects influence the control process as well. If using a simple proportional controller (see Figure 7.3) for robot control without any precautions to deal with hysteresis effects, the controller might be able to move the robot near to the target position. Then again, as soon as the dimension of the hysteresis effects is reached, we experience the expected oscillation effects. We can find this behavior in the results of a little experiment with the said controller and an example setup. The actual control process is over in less than ten seconds, but we do not stop the control process until two minutes have elapsed. The actuation for the X coordinate for the last twenty seconds of the control process is shown in Figure 6.17. We see an oscillation with a period length of about $3.5\,\mathrm{s}$ and an amplitude of about $0.01\,\mathrm{mm}$. With growing control gain, the amplitude and the frequency of the oscillation increase. For the modeling and cancellation of hysteresis effects in control systems, take a look at [43].

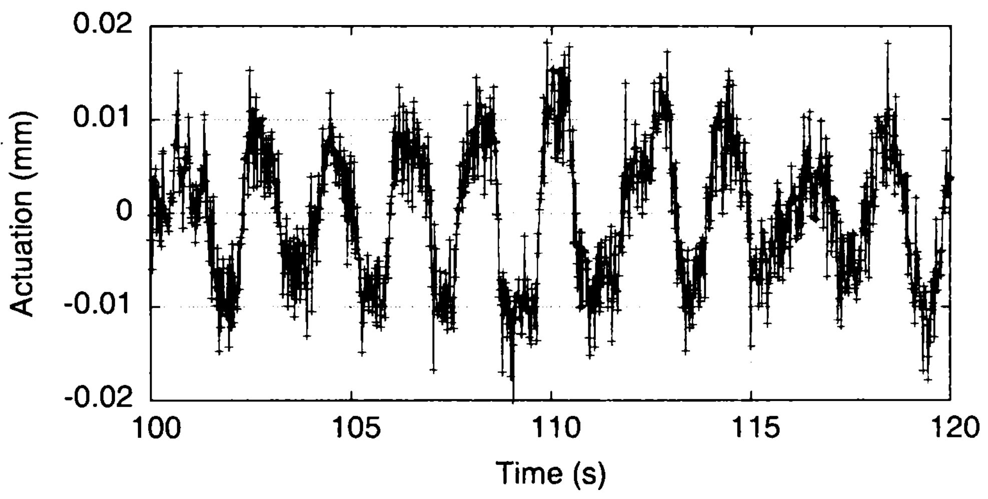


Figure 6.17: Remaining Oscillation in X Coordinate

6.5. Conclusion

In this chapter, we analyzed the non-linearities of hysteresis-type that occurred when recording trace data. After defining the hysteresis size, we saw the connections of the hysteresis in robot and signal space. The hysteresis turned out to be linearly dependent from the robot's trace speed. The dynamic part of the hysteresis had the delays between the robot position sampling and the control signals sampling as a reason, and the static part had the mechanical effects of the robot as a cause. We learned that the robot-induced hysteresis effect takes place in the robot's joints, which leads to hysteresis effects; even in DOF when the robot is not actually moved. Later on, we learned how to distinguish both hysteresis effects in the trace data and how to compensate for the hysteresis in the trace data. The hysteresis compensation helped when using solution methods with stability issues such as the direct solution. Additionally, it was a condition for a proper noise-based weighting of the control signals in the Jacobian. This chapter is only about hysteresis detection and its compensation in trace data. To take care of hysteresis effects during control, other methods have to be used, though the hysteresis determination may help to provide data for a model-based hysteresis compensation approach.

7. ROBOT CONTROL SYSTEMS

First, we introduce a general classification of visual servoing systems to distinguish between the often occurring terms used to describe robot control systems such as, for example, "position control", "online" or "multi-step" and we take a look at official classification attempts. In the next part of this chapter, we build an example visual servoing system to check the usability and performance of the previously introduced methods.

7.1. System Classification

Some properties of robot control systems allow a classification into different categories to make evaluations and comparisons of systems easier.

Robot control systems in common can be divided into *robot position correction systems* and *robot path correction systems.* The former ones determine the position change of the work object or a robot-grabbed part relative to a fixed setup position. The sensors can be fixed sensors such as multiple cameras; each observing a certain work object feature. Then, the global position change can be calculated using bundle adjustment techniques (see, for example, [89] or [90]). The sensors can also be hand-mounted. In this case, the robot moves the measurement system subsequently to the individual measurement positions. The main application fields for position correction systems are the compensation of tolerances in the conveyor system for the work objects or the positioning of work object parts before a subsequent assembly. Robot path correction systems, on the other hand, provide local correction of single robot path points using hand-mounted sensors. These can be used to balance out tolerances in the work object.

Figure 7.1 illustrates the differences between both systems. During system setup, the work object is posed into the nominal position in the robot cell and the application program for the robot is taught (a). The nodes along the robot's path are the path points taught in the robot program and refer to the work object coordinate system. During normal system operation, the conveyor places a new work object with certain production tolerances in the robot cell at a slightly different than the nominal position (b). With a position correction, the work object coordinate system can be corrected to have the robot path run along the work object again (c). To compensate for the production tolerances of the work object, a path correction of the single path points is advisable (d).

In *online* systems, the corrections are directly applied to the robot path, while *offline* systems separate the process into a measurement run and an application run. During the measurement run, the system acquires the measurement data and after determining and transmitting the corrections the robot executes its application. The offline system is slower because of the extended robot movement for separate measurement and application, but it still has advantages such as no mutual interference of the measurement and application. For a more in-depth comparison, see [74].

Robot control systems usually manage the complex global coherence between sensor information and robot position by using a simplified model for a local working point. An online *one-step* solution uses a single set of sensor information to determine a robot correction, and an online *multi-step* solution continues the measurement after the robot has been moved towards the target

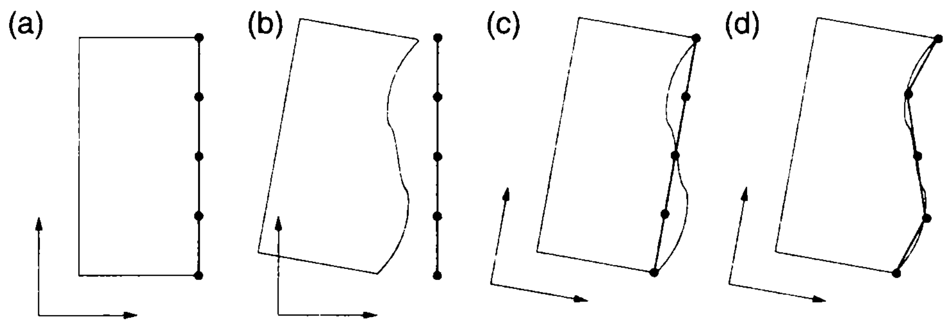


Figure 7.1: Position Correction vs. Path Correction

position and generates successive, new corrections. This enhances the accuracy of the process at the cost of additional time.

On the input side, robot control systems acquire sensor data, and on the output side corrections are delivered to the robot. For online multi-step robot control systems, both events may be coupled or not. In the case of *synchronous* systems, the next measurement is not executed before the robot has applied the last correction. In *asynchronous* systems, the next measurement is executed while the robot is still executing the last correction. This allows for a much faster correction process because the current correction is continuously updated. Nevertheless, for asynchronous systems we need a real time communication interface to the robot to provide it continuously with corrections.

Another common classification method or taxonomy, especially for visual servoing systems, can be found originally in [80] and [95], with further explanations in [39]. They distinguish between systems with a hierarchical control structure where the vision system provides set points to the robot's joint controller (*dynamic look-and-move systems*) and systems that directly control the joints of the robot (*direct visual servo systems*). The second characteristic for distinction is the definition of the error signal in terms of the 3D coordinates of the robot (*position-based systems*) or in image coordinates (*image-based system*). A further question is whether the system observes only the target object (*endpoint open-loop systems (EOL)*) or both the target object and the robot end-effector (*endpoint closed-loop systems (ECL)*).

7.2. Robot Controller

In this chapter, we will put together the previously introduced technologies and approaches into a fully functional robot position correction system (see Chapter 7.1). The functionality of the system is illustrated in Figure 7.2. We want to control the position of a standard industrial robot (see Chapter 2.1) using a number of control signals. After choosing one or more adequate sensor technologies (see Chapter 3.1), we will define - based on the sensor signals (see Chapter 3.2) - a number of control signals (see Chapter 3.3) depending on our special application. In the next step, we set up the robot and the work object in their nominal positions, as in (a), and execute a training run to record the nominal positions as well as the trace data (see Chapter 4.2). Then we use one of the solution strategies (see Chapter 4.3) to determine the Jacobian matrix and, therefore, identify the system. During production, we have a situation as in (b), where we have a work object positioned at a different position to that of the nominal position and the work object may contain certain form deviations. The objective the position correction system has to achieve is to position the robot in a way relative to the work object that the control signal deviations are minimized, as in (c).

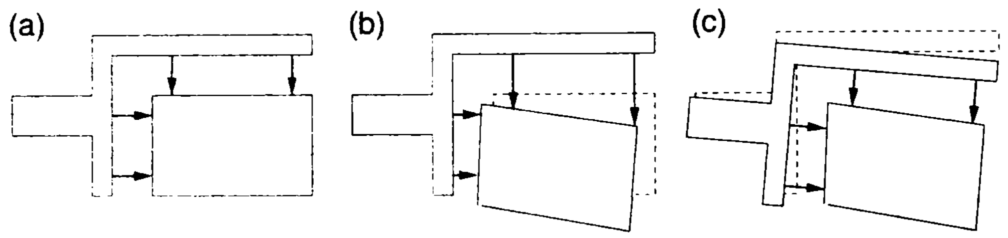


Figure 7.2: Robot Position Correction System

We use a simple proportional controller to control the robot's position, as shown in Figure 7.3. The feedback from the sensors in the form of the control signals $\mathbf{s}$ is converted into control signal values relative to the nominal control signals $\mathbf{s}_0$. These control signal deviations $\Delta\mathbf{s}$ are transformed into robot deviations $\Delta\mathbf{r}$ using the Jacobian matrix $\mathbf{J}$, by calculating

$$\Delta\mathbf{r} = \mathbf{J}^{\mathsf{T}}\Delta\mathbf{s} \tag{7.1}$$

as described in Chapter 4.1. To realize the controller gain K_p of the proportional controller, we use SLERPs (see Chapter 2.3), where K_p plays the role of the interpolation factor u such that $0 \leq K_p \leq 1$. The result is transmitted to the robot and the movement of its tool leads to a change in the current control signal values.

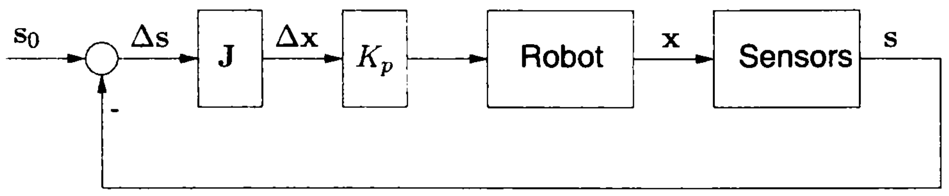


Figure 7.3: Robot Controller Diagram

The stop criterion halts the control process when the current correction $\Delta\mathbf{r}$ is small enough over a certain period of time (to avoid precipitate control stops on zero-crossings of an oscillating $\Delta\mathbf{r}$ during control).

How the calculated robot corrections are transmitted to the robot depends on its communication and control interfaces (see Chapter 2.5). In the case of unavailable real time control, we can realize a synchronous communication where the calculated corrections are transmitted to the robot using standard communications. The robot applies the correction and signals its readiness for a new correction. The controller triggers the next measurement and calculates the requested correction. This synchronous communication is the most portable one. If real time control is available, as it is with RSI/Corob systems, the robot controller is able to process a new position correction every $12\,\mathrm{ms}$. Here, we can realize an asynchronous communication by decoupling the measurement and the correction calculation and delivery. The current correction is buffered and transmitted to the robot as long as no new correction is available. Therefore, the sensor subsystem can provide control signals at lower speed, which is not a big problem because the robot may be able to accept new corrections at that high speed, but the mechanical parts of the robot are much slower.

It is fairly advisable to implement the control system in a way that it can cope with the failure of one or more sensors that lead to the unavailability of one or more control signals. In optimal cases, this should even be possible during the control process. In case of a sensor failure, we use the method described in Chapter 4.5.2 for a complete deactivation of the control signal to reconfigure the system and to continue the control process at a lower accuracy. Such a feature allows the usage of redundant sensors for a more reliable system.

The parameter K_p can be used to control the speed of the robot. For lower values of K_p, the total control time depends almost linearly on K_p, where low values of K_p lead to long total control times. Increasing K_p, the total control time decreases until a point is reached where the high gain leads to harder robot movements that are accompanied by an increase of the total control time. For very high values of K_p, it may be impossible to fulfill the requested accuracy requirements due to the coarse robot movements.

Now we take a look at an example control process. The system controlled the directions X, Y and Z with three control signals; Figure 7.4 shows on the left side the current actuation values $\Delta\mathrm{x}$ sent to the robot over the control time. The accuracy chosen as the stop criterion was $0.01\,\mathrm{mm}$ for all three DOFs over a time of $120\,\mathrm{ms}$, the control gain was $K_p = 0.2$, and the resulting total control time was around $3.2\,\mathrm{s}$. Figure 7.4 displays on the right side the sensor deviations from the nominal position $\Delta\mathrm{s}$ over the time. The data in both figures was thinned out to make the single signals distinguishable.

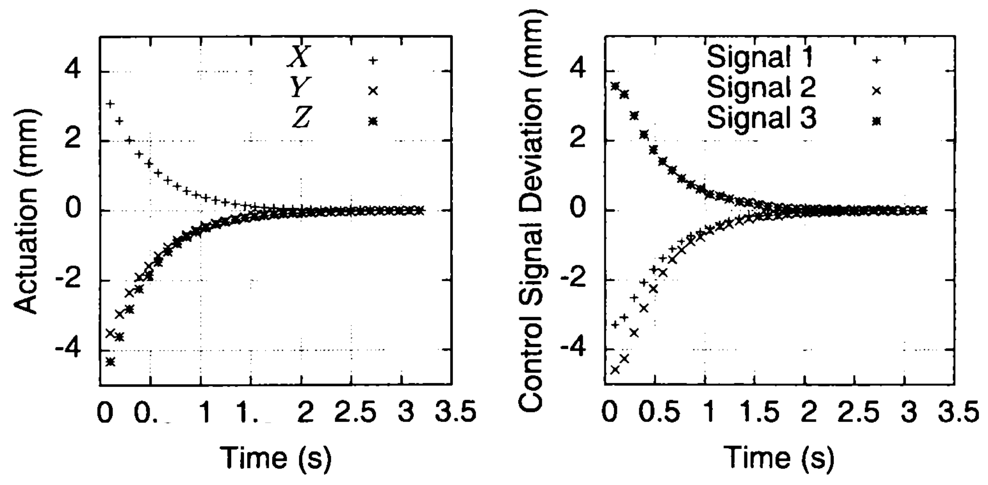


Figure 7.4: Robot Control Cycle in Robot and Signal Space

The result of the control process was quite sufficient. The relative positioning to the work object was better even than the repeating accuracy of the robot. The robot in the experiment had a repeating accuracy of $\pm 0.15\,\mathrm{mm}$ but it was possible to position it relative to the work object with a quite remarkable accuracy of $\pm 0.01\,\mathrm{mm}$, and in a stable way. This makes the usage of the system interesting in connection with applications with high accuracy positioning demands.

8. CONCLUSION AND FUTURE WORK

This chapter provides a conclusion of the results of the previous chapters, along with an overview of the analyzed methods and a suggestion for a successful system identification based on these insights. Furthermore, it contains a list of personal contributions that have been made in this thesis and possible starting points for a continuation of the work of this thesis.

8.1. Conclusion

This work contains the in-depth practical comparison of a number of different methods for the estimation of the Jacobian for the learning approach in visual servoing. The most commonly used method to calculate the Jacobian as the inverse of the Feature Jacobian has its weaknesses compared to other solution approaches such as the direct solution, as previous works like [52] have also shown. We have checked the given solution methods against a list of properties that are important when applying the theory to real world scenarios, like stability, behavior in the presence of redundant signals with different noise ratios and flexibility. Based on the results, we have varied existing methods such as applying weighting before inverting the Feature Jacobian, or using existing mathematical solutions to improve the approaches, such as the usage of regularization techniques to gain stability for the direct solution. The methods have been implemented and tested using simulated and real world robot trace data. The Table 8.1 shows an overview of the results by naming the advantages and disadvantages of the single system identification approaches. References to the adherent chapters are given to make it easier to find the discussion of that particular problem.

Though the inversion of the Feature Jacobian is most commonly used, its result is easily tested for stability and it is invariant against problems connected with the hysteresis effect; it has a number of serious drawbacks. Some of them can be compensated for by using certain weighting, but the results are still not very convincing. The best solution in terms of least-squares is provided by the direct solution, but it has serious problems connected with instabilities. The standard method of enhancing the stability by taking the L2 norm of the result into consideration helps, but it applies unwanted changes to the signal weighting. The usage of the L1 norm for the system identification and implementation of appropriate calculation methods has been discussed in recent mathematical publications, but it performs well in the field of visual servoing and in experiments with real world trace data. The L1 regularization enhances the stability and adds with the better interpretability - because of the sparsity of the solution - another nice feature. The more complicated calculation of the solution is a less important disadvantage as we determine the Jacobian once during system setup and because of this there are no tight time-constraints.

Therefore, if implementing a system for the identification of the Jacobian on the basis of trace data, we can make the following suggestions based on the insights of the previous chapters. After recording the trace data, we should apply a hysteresis compensation after Chapter 6.3.2 to minimize the stability problems introduced by the hysteresis effect and allowing proper noise-based weighting. The next step would be the selection of the control signals we want to use, to identify the particular DOFs and the deactivation of unnecessary combinations, based on the concept of the sensor setup (see Chapter 4.5.2). The solution based on this pre-processed trace data should be calculated using the L1 regularization - a method which showed the best result in the tests. For enhanced stability and sparsity, a ratio of quality drop should be defined (see Chapter 4.6.2.4)

Method	Advantages	Disadvantages
Inverse of the Feature Jacobian	• Transparent • Easy stability analysis • Invariant against Hysteresis Effects	• Inflexible (Chapter 4.5) • No noise-dependent weighting (Chapter 5.2.2.2) • Quality drops upon addition of noisy sensors (Chapter 5.2.3)
Inverse of the Feature Jacobian with Weighting	• Transparent • Easy stability analysis	• Inflexible (Chapter 4.5)
Direct Solution	• Least Squares Optimal	• Unstable (Chapter 4.6) • Hysteresis effects aggravate instabilities (Chapter 6.3.2)
L2 Regularization	• Enhanced Stability • Closed-form solution available	• Encourages unfavorable similarity of weightings (Chapter 4.6.2.3)
L1 Regularization	• Enhanced stability • Sparse solution	• No closed-form solution available (Chapter 4.6.2.4)

Table 8.1: Solution Method Overview

to be used in connection with the L1 regularization. We must take special care when we expect higher work object deviations and use combinations of redundant sensors to assemble parts in a certain way. In these cases, the weighting in the Jacobian influences the final control position (see Chapter 4.7). This problem may be solved by defining proper control signals; redundant control signals should be avoided by combining respective sensor signals to a single control signal (see Chapter 3.3).

Another significant part of this work regards the analysis of the hysteresis effect, its cause and its consequences for the solution methods. The hysteresis effect, appearing in the trace data, has two different reasons. One is the mechanical properties of the robot system, the other delays between the sampling of the robot position and of the raw data of sensors. The first part is independent from the robot speed and affects not only the trained DOF because the effect takes place in the robot's joints. The second part depends linearly on the robot's trace speed and affects only the trained DOF. Besides the determination of the hysteresis from trace data and its separation into hardware-dependent part and sampling delay, we have discussed methods for the compensation of the hysteresis effect in trace data. This is interesting because the hysteresis effect is unfavorable when using the direct solution method or any solution with noise-based weighting. In these cases, it leads to instabilities or misinterpretation of the hysteresis for sensor noise. Although the results may help to determine model parameters for a model-based cancellation of the hysteresis effect, this work does not deal with this subject.

In the end, we have put together all the previously developed and evaluated methods together with the introduced components to a fully functional robot position correction system. The working system shows the usability of the theories from this thesis in practice and, besides that, archives a remarkable accuracy.

8.2. Contributions

The most important subject of this thesis is the comparison of different solution methods for the Jacobian of the learning approach in visual servoing from a practical point of view, and the development of a procedure for a successful system identification. The analyzed methods include well known approaches, but some of those such as the additional noise-dependent weighting of the inverted Feature Jacobian have been improved . To get a comprehensive comparison, new approaches like the L1 regularization techniques - currently more a matter of theoretical mathematical publications - have been examined for their usability in connection with visual servoing. The analysis has been done with a number of tools to estimate the quality of a solution method in terms of stability and linearity in connection with a given data set. Some of these tools are standard tools like the stability estimation using the condition number. However, other tools such as the COD[1], which is a common tool in statistics, have been successfully adapted to our problem. Practical problems like the flexibility of the solutions regarding the ability to work with different kinds of trace data or the deactivation of features for certain DOFs have been taken into consideration. To get more information on the way, the weighting of the features in the Jacobian matrix depend on the properties of the feature information and the chosen solution method, we have realized a model-based analysis of the weighting providing detailed insight into the subject.

Another contribution of this work is connected with an effect detected in the recorded trace data that leads to problems in connection with a system identification using certain solution methods. It prevents a proper noise-based weighting of the sensor information and introduces stability problems. The effect is a non-linear behavior between the robot position and the feature information of hysteresis-type. In this thesis, we have found two important reasons for this effect - the mechanical robot hysteresis and the delays in the sampling process. We have developed a method to distinguish both effects from one or more trace data sets and to compensate the effect for a proper system identification.

To assign the author's contributions to single chapters of this thesis, we can say that the introductory chapters about robots and sensors contain almost common knowledge, even if the concept of sensor signals and control signals for specifying different control tasks has not been described in other publications yet. The chapter about system identification introduces well known system identification methods and quality estimation approaches, but further to these, all system identification related and previously described topics are discussed. The analysis of signal weighting by using a model in the next chapter gives a substantial insight into the origin of the signal weights on the basis of signal properties. In the chapter on hysteresis analysis, the distinction between two known effects is presented. The methods used here are unique and of practical relevance, e.g. for hysteresis compensation. The description of a robot control system and its practical results in the following chapter is not new, although it does prove the usability of the specified complete identification and control concept. In the concluding chapter, a new method for system identification is given.

The results of this thesis are interesting for any kind of applications using the supervised learning approach in connection with linearization in visual servoing to control the path or position

[1]COD=Coefficient Of Determination

of a robot. These applications use a learning data set (trace data) and a combination of input data (sensor data) and desired output data (robot deviations) to determine the elements of a Jacobian matrix. For this task, a number of solution methods come to mind which have different advantages and disadvantages. This work helps to understand the mechanisms of different approaches and helps to choose the right approach for the required application. Furthermore, it offers an insight into the causes of the hysteresis-type non-linearities that occur every time we sample robot and sensor data, and it makes suggestions on how to detect and cancel the hysteresis effect in the learning data.

The author of this thesis had the opportunity to test different theories in practice using a laboratory setup of an industrial robot with a great variety of different hand-mounted sensor systems, including laser distance sensors, laser stripe sensors and camera systems. This experimental environment allowed us to test different methods for the execution of training runs and for the recording of trace data, as well as to carry out control experiments with different work object configurations. Besides these testing facilities in the laboratory, the author is part of a development team that realizes a robot positioning system of industrial grade with a number of installations of greater scale in the factories of an automobile manufacturer. Consequently, there was a great amount of trace data of different control configurations available to gain new experiences and enhance given theories.

For the analysis of the trace data, we have implemented a number of software tools in the Matlab environment:

- Simple simulation of robot training runs using different control signal gains and different control signal noises
- Physical simulation of a robot with hand-mounted sensors to simulate training runs with acceleration/deceleration phases and the robot moving at constant speed, robot hysteresis and delays in the sensor subsystem, and a geometrical definition of the work object and distance sensors
- Detection of robot linearity ranges for a given trace data set: Robot movement range in which the signal response of the sensors is nearly linearly dependent on the robot position.
- Hysteresis detection and hysteresis compensation
- System identification using various solution approaches
- Quality and stability analyzes for a given solution
- Comparative analysis of the results of different solution approaches

Additionally, the results from this work have been used to improve the previously mentioned industrial robot position control system. The experiences from this real world operation were the basis for further enhancements of the theories.

During the work on this thesis, the author has published a number of papers with ideas and interim results. The paper [73] rose from the initial work on solving non-linear systems by a Taylor expansion using Jacobian matrices for identifying the TCP of hand-mounted laser stripe sensors. A further publication [8] in cooperation with K. Böhnke was the result of examining the optical properties of laser stripe sensors and their suitability of use as hand-mounted sensors on robots. In the paper [74], we discuss the design of an offline path correction system and its advantages and disadvantages over online path correction systems. The publication [75] compares different methods for the estimation of Jacobian matrices regarding noise and stability. The static and dynamic components of hysteresis occurring on trace recording for visual servoing applications has been analyzed in [76]. Finally, the conception of a robot positioning system and the requirements for its operation in industrial environments is presented in [77].

Here we give a summarized list of the most important theoretical contributions of this thesis as a quick overview:

- Definition of quality measures for a system identification (trace data and resulting Jacobian): linearity, stability, flexibility, geometrical correctness and interpretability
- Comparison of existing methods for system identification, including ones not used in visual servoing before regarding to previously defined quality measures
- Enhancement of existing methods: Additional noise-based weighting for inverted Feature Jacobian approach
- Model-based analysis of weights in Jacobian providing enhanced insight in the results of different approaches
- Analysis of causes for the hysteresis effect occurring in learning data: Mechanical robot properties and sampling delays of robot position and sensor data
- Approach of automatic distinction of causes for the hysteresis effect and its compensation in trace data which leads to increased stability and correct weighting of noisy redundant sensor information
- Development of a successful procedure for system identification

The practical contributions include the development of software tools for the simulation of visual servoing setups with robot and sensors to generate learning data and tools for the analysis and system identification of simulated and real world learning data. The results of the theoretical work were used to develop and improve an industrial robot positioning system application which shows the practical applicability of this work.

8.3. Future Work

When writing a thesis, one has to establish priorities - it is not possible to work on every detail with the same effort and to follow every interesting thought. With this in mind, this is a collection of loose threads that one may wish to pick up when continuing the work that was started with this thesis:

- A subject of high practical interest would be the integration of new control signals (or the re-integration of slightly modified control signals due to the replacement or adjustment of sensors) without a complete new training run, which costs precious production time. The author has experimented with an approach to use the samples of robot positions and control signals that can be collected during one or more control processes to re-identify the system. Nevertheless, because the control process is based on the known sensors, all methods that have been tested so far favoured the known sensors over the new sensors. It would be nice to have a method to slowly adapt the additional knowledge of the new control signals to the Jacobian, but because this a completely new subject on its own, it has been excluded from this thesis. Possible solution methods would be adaptive Jacobian techniques (see [23] or [61]) or neuronal network approaches (self-organizing maps, also known as Kohonen maps, see [17] and [69]).

- In this thesis, we have developed a method for the estimation of the hysteresis effect and we have presented a method for the compensation of this effect in trace data. It would be nice if this numerical identification could be used to compensate for the hysteresis effect in the control process too. This could be done, for example, by developing an inverse system model of robots and sensors including the hysteresis, which could be fed with the numerical results from the hysteresis determination from this thesis. The model could then be used to realize a controller that takes the hysteresis into consideration and achieves higher resulting accuracies.

- The situation of system identification under the circumstances of expectant high work object form deviations has been described in this thesis. However, the solution of deactivating or joining redundant control signals to avoid unequal residual distributions is not very satisfactory. An approach to solve this problem would be a separate weighting of the control signals for the reconstruction of a robot DOF (for example noise-based weighting) and for the distribution of the residuals after finishing the control process (for example, equally distributed residuals).

- It would be interesting to find out how different Jacobian matrices influence the performance of the control process, like the overall control time or the maximum achievable accuracy, depending on the different possible controller architectures. Accuracy and control time are the most interesting system properties from a user's point of view.

A. GLOSSARY

COD "Coefficient Of Determination", statistical indicator for the quality of a regression model with respect to observed data

Control Signal A source of information that is used to control the robot, which is calculated by a linear combination of sensor signals

DOF "Degree Of Freedom"

Euler Angles Method to describe any 3D rotation R by a combination of three rotations around the basis vectors of the coordinate system:

$$R_{\text{Euler}}(\alpha,\beta,\gamma) = R(Z,\gamma)\cdot\ R(Y,\beta)\cdot\ R(Z,\alpha) \tag{A.1}$$

Feature Jacobian Matrix with as many rows as DOFs to control and as many columns as control signals available, contains partial derivatives of each control signal with respect to each DOF, often called "sensitivity matrix" or "interaction matrix"

IRLS "Iteratively Re-Weighted Least Squares", iterative algorithm to determine regressions that are robust against outliers in the regression data

Jacobian Matrix with as many rows as control signals available and as many columns as DOFs to control, contains partial derivatives of each DOF with respect to each control signal

KUKA, KRC, KRL KUKA Robotics Corporation is a German robot manufacturer, the robots are controlled with a KUKA Robot Controller (KRC), the robot's programming language is the KUKA Robot Language (KRL)

L1 Norm or "Manhattan Norm", sum of the absolutes of the elements of a vector $\mathbf{x}$ of length $n \in \mathbb{N}$:

$$||\mathbf{x}||_1 = \sum_{i=1}^{n} |x_i|. \tag{A.2}$$

L2 Norm or "Euclidean Norm", root of the sum of the squares of the elements of a vector $\mathbf{x}$ of length $n \in \mathbb{N}$:

$$||\mathbf{x}||_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{\frac{1}{2}} \tag{A.3}$$

Learning Approach A class of approaches in the field of visual servoing, that tries to adapt the knowledge of the coherence of robot movements and feature changes in a supervised (based on the list of inputs and desired outputs) or unsupervised learning (no such knowledge) procedure

RPY Angles "Roll-Pitch-Yaw Angles", method to describe any 3D rotation R by a combination of three rotations around basis vectors of the coordinate system:

$$R_{\text{RPY}}(\alpha,\beta,\gamma) = R(Z,\gamma)\cdot\ R(Y,\beta)\cdot\ R(X,\alpha) \tag{A.4}$$

RSI/Corob Ethernet-based low-level, real time control protocol for robots from KUKA.

Sensor Signal Single source of information about a geometrical feature that changes if the robot is moved, often a result of the evaluation of more complex sensor raw data

SLERP "Spherical Linear Interpolation", method to interpolate between two transformations, each including position and rotation, in a smooth way

TCP "Tool Center Point", point of origin of the robot's tool coordinate system; a "TCP calibrated sensor" is a sensor mounted at the robot's hand, which tool transformation is known

Visual Servoing The area in the field of engineering and computer science that is about using (originally) visual sensor information for the control of a robot

BIBLIOGRAPHY

[1] G. J. Agin: *Calibration and Use of a Light Stripe Range Sensor Mounted to the Hand of a Robot*, Tech. Report CMU-RI-TR-85-20, Robotics Institute, Carnegie Mellon University, Nov. 1985

[2] J. Angeles: *Fundamentals of Robotic Mechanical Systems*, Springer Verlag, 1997

[3] American National Standard Institution: *Process Instrumentation Terminology*, ANSI/ISA S51.1-1979 (R1993)

[4] H. Bacakoglu, M. S. Kamel: *A three-step camera calibration method*, IEEE Transactions on Instrumentation and Measurement, Volume 46, Issue 5, Pages 1165-1172, Oct. 1997

[5] J. Batista, H. Araujo, A. T. de Almeida: *Iterative multistep explicit camera calibration*, IEEE Transactions on Robotics and Automation, Volume 15, Issue 5, Pages 897-917, Oct. 1999

[6] F. Blais: *Review of 20 Years of Range Sensor Development*, Journal of Electronical Imaging, Volume 13, Issue 1, Pages 231-240, Jan. 2004

[7] A. Blomdell, G. Bolmsjö, T. Brogårdh, P. Cederberg, M. Isaksson, R. Johansson, M. Haage, K. Nilsson, M. Olsson, T. Olsson, A. Robertsson, J. Wang: *Extending an Industrial Robot Controller-Implementation and Applications of a Fast Open Sensor Interface*, IEEE Robotics & Automation Magazine, Volume 12, Issue 3, Pages 85-94, Sep. 2005

[8] K. Böhnke, P. Roebrock: *Industrielle optische 3D-Messung von Airbaggehäusen mit Laserlichtschnittsensoren*, 6. Oldenburger 3D-Tage, Oldenburg Germany, Jan. 2007

[9] H. Born, J. Bunsendal: *Programmable multi sensor interface for industrial applications*, Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems 2001, Pages 189-194, 2001

[10] S. Bruder, M. Farooq, M. M. Bayoumi: *A feature-level approach to multi-sensor integration: emphasizing robotic applications*, IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems 1994, Pages 477-484, Feb. 1994

[11] S. R. Buss: *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods*, University of California San Diego, Apr. 2004

[12] S. R. Buss, J.-S. Kim: *Selectively Damped Least Squares for Inverse Kinematics*, Journal of Graphics Tools, vol. 10, no. 3 (2005) 37-49, Oct. 2004

[13] F. Chaumette, S. Hutchinson: *Visual Servo Control Part I: Basic Approaches*, IEEE Robotics & Automation Magazine, Volume 13, Issue 4, Pages 82-90, Dec. 2006

[14] F. Chaumette, S. Hutchinson: *Visual Servo Control Part II: Advanced Approaches*, IEEE Robotics & Automation Magazine, Volume 14, Issue 1, Pages 109-118, Mar. 2007

[15] S. S. Chen, D. L. Donoho, M. A. Saunders: *Atomic decomposition by basis pursuit*, SIAM Journal on Scientific Computing, Volume 20, Number 1, Pages 33-61, 1999

[16] M. A. A. Shoukat Choudhury, N. F. Thornhill, S. L. Shah: *Modelling valve stiction*, Control Engineering Practice, Elsevier Ltd. Volume 13, Issue 5, Pages 641-658, May 2004

[17] M. A. P. Cisneros: *Intelligent Model Structures In Visual Servoing*, Ph. D. Thesis, University of Manchester, Institute of Science and Technology, Sep. 2004

[18] P. I. Corke: *Visual control of robot manipulators - A review*, Visual Servoing K, Hashimoto Ed. Singapore: World Scientific, Pages 1-31, 1993

[19] P. I. Corke: *High-Performance Visual Closed-Loop Robot Control*, Ph. D. Thesis, University of Melbourne, Dept. Mechanical and Manufacturing Engineering, 1994

[20] J. J. Craig: *Introduction to Robotics. Mechanics and Control*, 3rd Edition, Prentice Hall, Aug. 2004

[21] E. Dam, M. Koch, M. Lillholm: *Quaternions, Interpolation and Animation* DIKU-TR-98/5, Department of Computer Science, University of Copenhagen, Jul. 1998

[22] H. Das, J.-E. Slotine, T. B. Sheridan: *Inverse kinematic algorithms for redundant systems*, Proceedings of the IEEE International Conference on Robotics and Automation 1988, Volume 1, Pages 43-48, Apr. 1988

[23] D. Dimogianopoulos, R. Lozano: *Adaptive control for linear slowly time-varying systems using direct least-squares estimation*, Automatica, Volume 37, Number 2, Pages 251-256, Feb. 2001

[24] A. D'Souza, S. Vijayakumar, S. Schaal: *Learning inverse kinematics*, Proceedings of the IEEE International Conference on Intelligent Robots and Systems 2001, Volume 1, Pages 298-303, Oct. 2001

[25] K. Feldmann, U. Schonherr, J. Zeller: *Multisensor integration for sensor guided robots*, Proceedings of the IEEE International Conference on Intelligent Robots and Systems 1994, Volume 3, Pages 1730-1735, Sep. 1994

[26] J. Fox: *Applied Regression Analysis, Linear Models, and Related Methods*, SAGE Publications Inc., Apr. 1997

[27] M. Fridenfalk, G. Bolmsjö: *Design and validation of a sensor guided robot control system for welding in shipbuilding*, International Journal for the Joining of Materials, Volume 14, Number 3/4, Pages 44-55, 2002

[28] M. Fridenfalk: *Development of intelligent robot system based on sensor control*, Ph. D. Thesis, Department of Mechanical Engineering, Lund University, Mar. 2003

[29] W. Fu: *Penalized regressions: The bridge versus the LASSO*, Journal of Computational Graphical Statistics, Volume 7, Number 3, Pages 397-416, 1998

[30] G. H. Golub, C. F. Van Loan: *Matrix Computations*, Chapter 8.6: Computing the SVD, The Johns Hopkins University Press, 1996

[31] K. Grochow, S. L. Martin, A. Hertzmann, Z. Popović: *Style-based Inverse Kinematics*, ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004), 2004

[32] D. L. Hall, J. Llinas: *An introduction to multisensor data fusion*, Proceedings of the IEEE, Volume 85, Issue 1, Pages 6-23, Jan. 1997

[33] J. Hill, W. T. Park: *Real time control of a robot with a mobile camera*, Proceedings of the 9th International Symposium on Industrial Robotics '79 (ISIR 1979), Pages 233-246, Mar. 1979

[34] N. J. Hollinghurst, R. Cipolla: *Uncalibrated Stereo Hand Eye Coordination*, Proceedings of the 4th British Machine Vision Conference, Pages 389-398, Apr. 1994

[35] B. Huang, V. Milenkovic: *Method and system for correcting a robot path*, U.S. Patent Number 4945493, applied for 1988-09-26, received 1990-07-31

[36] P. J. Huber: *Robust Estimation of a Location Parameter*, Annals of Mathematical Statistics 35:73 101, 1964

[37] P. J. Huber: *Robust regression: Asymptotics, conjectures, and Monte Carlo*, Annals of Statistics, 1, 799-821, 1973

[38] J. P. Huissoon, D. L. Strauss: *System and method for tracking a feature on an object using a redundant axis*, U.S. Patent Number 5465037, applied for 1994-01-13, received 1995-11-07

[39] S. A. Hutchinson, G. D. Hager, P. I. Corke: *A tutorial on visual servo control*, IEEE Transactions on Robotics and Automation, Volume 12, Number 5, Pages 651-670, Oct. 1996

[40] International Organization for Standardization: *Manipulating industrial robots - Vocabulary*, CEN EN ISO 8373: 1996, Jan. 1994

[41] L. E. Kavraki: *Protein Inverse Kinematics and the Loop Closure Problem*, The Connexions Project, May 2006

[42] V. Klema, A. Laub: *The singular value decomposition: Its computation and some applications*, IEEE Transactions on Automatic Control, Volume 25, Issue 2, Pages 164-176, Apr. 1980

[43] Y. Kobayashi, T. Okita: *Identification of nonlinear systems with hysteresis characteristics*, Proceedings of the 41st SICE Annual Conference SICE 2002, Volume 3, Pages 1577-1581, Aug. 2002

[44] D. Kragic, H. I. Christensen: *Survey on Visual Servoing for Manipulation*, Center For Autonomous Systems, Numerical Analysis and Computer Science, KTH - Royal Institute of Technology, Stockholm, Jan. 2002

[45] J. B. Kuipers: *Quaternions and Rotation Sequences*, Princeton University Press, Aug. 2002

[46] KUKA Robotics Corporation / Amatec Robotics Corporation: *RSI-Ethernet Interface CoRob*, Version 3.6.0

[47] KUKA Robotics Corporation: *KRC2/KRC3 Expert Programming*, Release 5.2, Sep. 2003

[48] KUKA Robotics Corporation: *Robot Sensor Interface (RSI)*, Release 2.0

[49] KUKA Robotics Corporation: *Communication CREAD/CWRITE*, Software Version 5.4

[50] F. Lange, G. Hirzinger: *A universal sensor control architecture considering robot dynamics*, Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems 2001, Pages 277-282, 2001

[51] F. Lange, G. Hirzinger: *Stability Preserving Sensor-Based Control for Robots with Positional Interface*, Proceedings of the IEEE International Conference on Robotics and Automation 2005, Pages 1700-1705, Apr. 2005

[52] J. T. Lapreste, F. Jurie, M. Dhome, F. Chaumette: *An Efficient Method to Compute the Inverse Jacobian Matrix in Visual Servoing*, Proceedings of the IEEE International Conference on Robotics and Automation 2004, Volume 1, Pages 727-732, Apr. 2004

[53] R. Lenz: *Linsenfehlerkorrigierte Eichung von Halbleiterkameras mit Standarobjektiven für hochgenaue 3D-Messungen in Echtzeit*, Informatik-Fachberichte, Vol. 149, Mustererkennung 1987, 9. DAGM-Symposium, Springer-Verlag, 1987

[54] R. C. Luo, M.-H. Lin, R. S. Scherp: *Dynamic multi-sensor data fusion system for intelligent robots*, IEEE Journal of Robotics and Automation, Volume 4, Issue 4, Pages 386-396, Aug. 1980

[55] R. C. Luo, M. G. Kay: *A tutorial on multisensor integration and fusion*, 16th Annual Conference of IEEE Industrial Electronics Society, IECON '90, Pages 707-722, Nov. 1990

[56] R. C. Luo, Chih-Chen Yih, Kuo Lan Su: *Multisensor fusion and integration: approaches, applications, and future research directions*, IEEE Sensors Journal, Volume 2, Issue 2, Pages 107-119, Apr. 2002

[57] O. Madsen, C. B. Sørensen, R. Larsen, L. Overgaard, N. J. Jacobsen: *A system for complex robotic welding*, Industrial Robot: An International Journal, Volume 29, Number 2, Pages 127-131, 2002

[58] E. Malis, F. Chaumette, S. Boudet: *2-1/2-D visual servoing*, IEEE Transactions on Robotics and Automation, Volume 15, Issue 2, Pages 238-250, Apr. 1999

[59] E. Malis, F. Chaumette: *Theoretical Improvements in the Stability Analysis of a New Class of Model-Free Visual Servoing Methods*, IEEE Transactions on Robotics and Automation, Volume 18, Issue 2, Pages 176-186, Apr. 2002

[60] N. Mansard, M. Lopes, J. Santos-Victor, F. Chaumette: *Jacobian Learning Methods for Tasks Sequencing in Visual Servoing*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Pages 4284-4290, Oct. 2006

[61] M. de Mathelin, R. Lozano: *Robust adaptive identification of slowly time-varying parameters with bounded disturbances*, Automatica, Volume 35, Number 7, Pages 1291-1305, Jul. 1999

[62] N. Miyake, M. Sumita, S. Sarugaku: *Method and apparatus for controlling tracking path of working point of industrial robot*, U.S. Patent Number 4954762, applied for 1990-01-30, received 1990-09-04

[63] E. H. Moore: *On the reciprocal of the general algebraic matrix*, Bulletin of the American Mathematical Society 26, Pages 394-395, 1920

[64] N. Nayak, A. Ray: *An integrated system for intelligent seam tracking in roboticwelding: I. Conceptual and analytical development*, Proceedings of the IEEE International Conference on Robotics and Automation 1990, Volume 3, Pages 1892-1897, May 1990

[65] *An integrated system for intelligent seam tracking in roboticwelding: II. Design and implementation*, Proceedings of the IEEE international Conference on Robotics and Automation 1990, Volume 3, Pages 1898-1903, May 1990

[66] W. Neubauer, M. Moller, S. Bocionek, W. Rencken: *Learning Systems Behavior For The Automatic Correction And Optimization Of Off-line Robot Programs*, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Volume 2, Pages 1355-1362, Jul. 1992

[67] A. Niel, P. Sommer, S. H. Kolpl, Y. Lypetskyy: *High precision measurement of free formed parts using industrial robots* IEEE International Workshop on Robot Sensing, Pages 79-84, May 2004

[68] U. Nunes, P. Faia, A. T. de Almeida: *Sensor-based 3-D autonomous surface-following control*, Mathematics and Computers in Simulation, Volume 41, Issue 5-6, Pages 429-444, Aug. 1996

[69] H.-G. Park, S.-Y. Oh: *A neural network based real-time robot tracking controller using position sensitive detectors*, IEEE World Congress on Computational Intelligence, IEEE International Conference on Neural Networks, Volume 5, Pages 2754-2758, Jun. 1994

[70] R. Penrose: *A generalized inverse for matrices*, Proceedings of the Cambridge Philosophical Society 51, Pages 406-413, 1955

[71] W. H. Press, A. S. Teukolsky, W. T. Vetterling: *Numerical Recipes in C*, Chapter 2.6: Singular value decomposition, Cambridge University Press, Jan. 1993

[72] A. Reek: *Strategien zur Fokuspositionierung beim Laserstrahlschweißen*, Doctoral Thesis, University of München, Oct. 1999

[73] P. Roebrock: *TCP Identification of Contactless Measurement Systems*, 7th Symposium of Electronics and Telecommunications, "Politehnica" University of Timişoara Romania, Sep. 2006

[74] P. Roebrock, K. Böhnke: *Offline Path Correction System for Industrial Robots*, 9th WSEAS International Conference on Automatic Control, Modelling & Simulation (ACMOS'07), Istanbul Turkey, May 2007

[75] P. Roebrock: *Comparing Methods for Estimating the Jacobian in Visual Servoing*, 7th WSEAS International Conference on Signal Processing, Robotics & Automation (ISPRA'08), Cambridge UK, Feb. 2008

[76] P. Roebrock: *Analysis of Non-Linearities of Hysteresis Type occurring in Learning Approach of Visual Servoing*, 12th WSEAS International Conference on Systems (ICS'08), Heraklion/Crete Greek, Jul. 2008

[77] P. Roebrock: *A flexible multi-sensor positioning system for industrial robots*, WSEAS Transactions on Electronics, Issue 3, Volume 5, Pages 93-100, 2008

[78] S. M. Ross: *Introduction to Probability and Statistics for Engineers and Scientists*, Academic Press, 3rd Edition, Jul. 2004

[79] M. Sallinen, T. Heikkilä: *A simple Hand-Eye Calibration Method for a 3D Laser Range Sensor*, 4th IFIP/IEEE Internatinal Conference on Information Technology for Balanced Automation Systems in Manufacture and Transportation, Berlin Germany, Sep. 2000

[80] A. C. Sanderson, L. E. Weiss: *Image-based visual servo control using relational graph error signals*, Proceedings of the IEEE International Conference on Robotics and Automation 1980, Pages 1074-1077, 1980

[81] M. Schmidt, G. Fung, R. Rosales: *Fast Optimization Methods for L1 Regularization: A Comparative Study and Two New Approaches*, The Eighteenth European Conference on Machine Learning (ECML), Pages 286-297, 2007

[82] R. R. Selmic, V. V. Phoha, F. L. Lewis: *Intelligent compensation of actuator nonlinearities*, Proceedings of the 42nd IEEE Conference on Decision and Control, Volume 4, Pages 4327-4332, Dec. 2003

[83] R. Shadmehr, S. P. Wise: *The Computational Neurobiology of Reaching and Pointing: A Foundation for Motor Learning (Computational Neuroscience)*, The MIT Press, Chapter 7.5: Forces and Torques, Jan. 2005

[84] S. K. Shevade, S. S. Keerthi: *A simple and efficient algorithm for gene selection using sparse logistic regression*, Bioinformatics, Volume 19, Number 17, Pages 2246-2253, 2003

[85] K. Shoemake: *Animating rotation with quaternion curves*, ACM SIGGRAPH Computer Graphics 1985, Volume 19, Number 3, Pages 245-254, Jul. 1985

[86] A. Tarantola: *Inverse Problem Theory and Methods for Model Parameter Estimation*, Society for Industrial and Applied Mathematics, SIAM, 2005

[87] R. Tibshirani: *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society B, Volume 58, Number 1, Pages 267-288, 1996

[88] A. N. Tikhonov, V. Y. Arsenin: *Solutions of ill-posed problem*, John Wiley & Sons Inc. New York, 1977

[89] B. Triggs, Ph. F. McLauchlan, R. I. Hartley, A. W. Fitzgibbon: *Bundle Adjustment - A Modern Synthesis*, Proceedings of the International Workshop on Vision Algorithms (ICCV) Pages 298-372, Springer-Verlag 1999

[90] B. Triggs, Ph. F. McLauchlan, R. I. Hartley, A. W. Fitzgibbon: *Bundle Adjustment - A Modern Synthesis*, Springer Berlin/Heidelberg, 2000

[91] W. Trunzer: *Strategien zur On-Line Bahnplanung bei Robotern mit 3D-Konturfolgesensoren*, Doctoral Thesis, University of München, Dec. 1995

[92] R. Y. Tsai: *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses*, IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, Pages 323-344, Aug. 1987

[93] Y. Veryba, L. Kourtch, B. Verigo, V. Murashko: *Method and system for robot end effector path correction using 3-D Ultrasound sensors*, Proceedings of the 3rd World Congress on Intelligent Control and Automation, Volume 2, Pages 1240-1243, 2000

[94] L.-C. T. Wang, C. C. Chen: *A combined optimization method for solving the inverse kinematics problems of mechanical manipulators*, IEEE Transactions on Robotics and Automation, Volume 7, Issue 4, Pages 489-499, Aug. 1991

[95] L. E. Weiss: *Dynamic Visual Servo Control of Robots: An Adaptive Image-Based Approach*, Doctoral Dissertation, Tech. Report CMU-RI-TR-84-16, Robotics Institute, Carnegie Mellon University, Apr. 1984

[96] C. Welman: *Inverse Kinematics and Geometric Constraints For Articulated Figure Manipulation*, Master Thesis, Simon Fraser University, Sep. 1993

[97] J. Zeller: *Sensorplanung und schnelle Sensorregelung für Industrieroboter*, Hanser Fachbuchverlag, Jan. 1998

•

ERATĂ

La pagina 2 a tezei elaborată de ing. Philipp Roebroch, în poziţia de „Conducător ştiinţific" ar fi trebuit să apară prof.univ.dr.ing. Virgil Tiponuţ. Precizăm că prof.univ.dr.ing. Marius Oteşteanu a îndeplinit funcţia de preşedinte al Comisiei de doctorat.

Editura Politehnica îşi cere scuze pentru această eroare.