TECHNISCHE UNIVERSITÄT WIEN

AUTOMATION & CONTROL INSTITUTE
INSTITUT FÜR AUTOMATISIERUNGS- & REGELUNGSTECHNIK

Robotic Drawing on a 3D Object

DIPLOMA THESIS

Conducted in partial fulfillment of the requirements for the degree of a

Diplom-Ingenieur (Dipl.-Ing.)

supervised by

Univ.-Prof. Dr. techn. A. Kugi
Dipl.-Ing. T. Weingartshofer

submitted at the

TU Wien
Faculty of Electrical Engineering and Information Technology
Automation and Control Institute

by
Amin Haddadi
Matriculation number 01127433

Vienna, November 2020

Complex Dynamical Systems Group
A-1040 Wien, Gußhausstr. 27–29, Internet: https://www.acin.tuwien.ac.at

Preamble

I would like to express my deep gratitude to my supervisor Dipl.-Ing. Thomas Weingartshofer, who supported me tirelessly throughout this master thesis. Without his persistent guidance, this thesis would not have been accomplished.

I would like to thank Univ.-Prof. Dr. techn. Andreas Kugi for giving me the opportunity to conduct this thesis and also for his very interesting lectures, which aroused my interest in robotics and control engineering in general.

I am also very grateful to Dr. techn. Christian Hartl-Nesic for his invaluable theoretical and practical help and especially for providing all the necessary resources for the robotic setup.

Last but certainly not least, I want to thank my family for their continuous support and encouragement during my years of study.

Vienna, November 2020

Abstract

Industrial robots are widely used in manufacturing tasks such as assembling, grinding and polishing to achieve fully automated production with high quality and efficiency. Robotic drawing treated in this work is a suitable scenario to demonstrate the effectiveness of robotics in such industrial applications.

The goal of this thesis is the robotic drawing of an arbitrary pattern on the surface of a known 3D object. The desired 2D pattern is drawn by the user via mouse or touch screen. Before the pattern can be projected onto the surface of the object, the 3D surface of the object model is flattened using least-squares conformal mapping. The 2D pattern points are then mapped onto the 3D object surface by linear interpolation using barycentric coordinates. For the projected pattern points, a trajectory for both position and orientation of the pen is constructed, which must be followed by the robot end-effector. This can be achieved by a trajectory tracking controller. For this purpose, first the dynamic model of the robot is discussed. The control concept consists of two parts. On the one hand, the position and orientation of the end-effector are controlled. On the other hand, since the pen is in contact with the object surface, the contact force is regulated. The combination of the two proposed controllers leads to a hybrid force/motion control scheme.

In order to improve the absolute accuracy of the robot during the drawing task, an optical measurement system is used for obtaining the position and orientation of the end-effector. Since the 3D object can be positioned arbitrarily in the robot workspace, the measurement system also has to provide the pose information of the 3D object.

The proposed concept is validated in experiments using a 7-axis robot arm. The robot system is able to draw on both planar and curved surfaces. It is also shown that by using the optical measuring system the drawing accuracy can be increased.

Kurzzusammenfassung

Industrieroboter werden häufig bei Fertigungsaufgaben wie Montage, Schleifen und Polieren eingesetzt, um eine vollautomatische Produktion mit hoher Qualität und Effizienz zu erreichen. Das robotische Zeichnen ist ein geeignetes Beispiel, um die Effektivität vom Einsatz der Robotik in den genannten industriellen Anwendungen zu demonstrieren.

Ziel dieser Arbeit ist es, mittels eines Roboters ein beliebiges Muster auf der Oberfläche eines bekannten 3D-Objekts zu zeichnen. Das 2D-Muster wird vom Benutzer mit einer Maus oder einem Touchscreen ausgeführt. Bevor das 2D-Muster auf die Oberfläche des Objekts projiziert werden kann, muss die 3D-Oberfläche mithilfe einer konformen Abbildung basierend auf der Methode der kleinsten Quadrate abgeflacht werden. Die 2D-Musterpunkte werden durch lineare Interpolation unter Verwendung von baryzentrischen Koordinaten auf die 3D-Objektoberfläche abgebildet. Dann wird für die projizierten Musterpunkte eine Trajektorie sowohl für die Position als auch für die Orientierung des Stifts konstruiert, welche mit einer Trajektorienfolgeregelung gefolgt werden muss. Zu diesem Zweck wird zunächst das dynamische Modell des Roboters hergeleitet. Basierend darauf wird ein Regler entworfen, um der gewünschten Trajektorie auf der Objektoberfläche zu folgen. Das Regelungskonzept besteht aus zwei Teilen. Zum einen wird die Position und Orientierung des Endeffektors geregelt. Zum anderen, da der Stift in Kontakt mit der Objektoberfläche ist, wird die Kontaktkraft geregelt. Die Kombination der beiden vorgeschlagenen Regelungskonzepte führt zum sogenannten hybriden Kraft-/Positionsregler.

Um die Absolutgenauigkeit des Roboters während des Zeichnens zu verbessern, wird ein optisches Messsystem verwendet, welches die Position und Orientierung des Endeffektors ermittelt. Da das 3D-Objekt beliebig im Arbeitsbereich des Roboters positioniert werden kann, muss das Messsystem auch die Poseinformation des 3D-Objekts liefern.

Das präsentierte Konzept wird an einem experimentellen Aufbau mit einem 7-Achs-Roboterarm validiert. Das Robotersystem kann sowohl auf planaren als auch auf gekrümmten Oberflächen zeichnen. Es wird außerdem gezeigt, dass unter Verwendung von optischen Messsystemen die Genauigkeit vom Roboter erhöht werden kann.

Contents

List of Figures

1 Introduction

The impact of robotics in manufacturing has been increased in recent decades. In production systems, CAD-based offline robot programming is used to program a robot with information from a 3D CAD environment. For working on a 3D object, "CAD to Production" systems are utilized, so that the robot does not need to be reprogrammed by the operator each time the work process changes.

This thesis demonstrates the robotic drawing of an arbitrary 2D pattern on the surface of a 3D object with known CAD model. The 2D pattern is made by the user via mouse, touch screen or is directly generated in the computer. In order to project the 2D pattern points onto the surface of the 3D object, as a pre-processing step, the surface of the object model is decomposed into a set of segments. Then the surface segments are flattened using least-squares conformal mapping, which minimizes angle deformations. The pattern points are projected onto the 3D object surface by a linear mapping using barycentric coordinates. Finally, a trajectory is generated out of the position and orientation information of each projected pattern point.

The robot used in this work is a KUKA LWR iiwa 14 R820, which has 7 degrees-of-freedom (DOF). For drawing, the robot is equipped with a pen as the end-effector. Figure 1.1 shows the robot with a 3D-printed rabbit as object to be drawn on. To enable the robot end-effector follow the desired trajectory, a trajectory tracking controller is designed in operational space. Since during the drawing task the pen is in contact with the object surface, both the motion of the end-effector and the contact force normal to the surface are controlled. This is achieved by using a hybrid force/motion control scheme. For this scheme, the contact force on the surface has to be determined in real time. In this work, a model-based estimation of the contact force is implemented.

In order to improve the absolute accuracy of the robot during the drawing task, an optical measuring system is used, which provides the position and orientation of the end-effector with sub-millimeter accuracy. Since the 3D object is placed arbitrarily in the robot workspace, the pose of the object is also measured by the optical tracking system.

1.1 Literature Review

In recent years, there have been numerous works dealing with robotic drawing applications. An early work on this topic is [1], where a humanoid robot is used to draw human portraits on a paper. Canny edge detection is applied to a picture to extract important face features for trajectory planning. Based on the resulting operational space trajectory, desired joint space trajectories are computed using inverse kinematics. A PID control law is used for trajectory tracking. A similar work is given in [2], where a two-armed humanoid robot draws portraits. A classical joint space PI controller is implemented to control the robot motors during the drawing process.

Figure 1.1: KUKA iiwa during drawing on a 3D-printed rabbit.

In [3], a portrait drawing robot is presented, which is capable of drawing on a paper or on a whiteboard. First a picture is captured from a person sitting in front of a camera. Then the image processing method based on the canny edge detection is used. In [4], Tresset *et al.* introduce Paul, a planar robotic arm constrained to 4 DOF, which draws portraits in an artistic style trying to mimick the human drawing technique.

Focusing on automatic feature extraction and optimal path planning methods, in [5] an industrial robot is used, which performs drawing with a mark pen. An elastic pen holder equipped with a spring ensures a smooth contact with the paper during the drawing. In [6], the same method is applied, but more detailed information of facial features are taken into account. In [7], a 6-axis robot manipulator that draws on a paper is presented. First, the canny algorithm is applied to get contours of the face and then, as a post processing step, cubic splines are constructed.

All robotic drawing systems mentioned so far can only draw on flat surfaces. To be able to draw on an arbitrary non-calibrated 3D surface the contact force has to be taken into account in the control concept. In [8], Jain *et al.* propose an approach in which the robot can draw on a non-calibrated non-planar surface. They use a 6-axis force/torque sensor, which is mounted on the robot wrist. The drawing pen is always oriented normal to the surface and is moved tangentially using a contour tracing method based on the measured contact force. However, this method has the drawback that depending on the surface curvature, the mapping of the drawing pattern onto the 3D surface could cause significant distortions.

In [9], a semi-autonomous robotic drawing system is demonstrated that can draw on a non-flat surface with varying thickness of pen strokes. An industrial 7-DOF manipulator is employed which is position and impedance controlled. The unknown object surface is adaptively estimated by incremental sample points during the drawing process. A limitation of this approach is that the drawing can be only performed on 3D surfaces,

which are monotonic along the height.

The robotic drawing system presented in this thesis is able to draw on a 3D object surface, whose CAD model is known. Hence, the curvatures of the 3D surface can be arbitrary, as long as the surface is smooth and continuously differentiable, i.e. there are no sharp edges. Least-squares conformal mapping is applied for projection of the 2D pattern on the 3D surface. This mapping method has the advantage that the angular distortions are minimized. Furthermore, in this work no force/torque sensor attached to the robot wrist is needed. Instead, a model-based estimation of the contact force is utilized, which is cost effective. Based on the CAD model of the object, a 3D drawing trajectory is planned for the position and orientation of the end-effector. Then a hybrid force/motion control is applied for trajectory tracking along the object surface and simultaneously controlling the contact force to a desired value. Compared to the impedance control used in [9], the hybrid force/motion control scheme has the advantage that a decoupled control of the end-effector motion and the contact force can be achieved.

1.2 Overview of this Thesis

This thesis is structured as follows: The mathematical model of the robot is derived in Chapter 2, which will be considered for the controller design. In Chapter 3, first a segmentation method of the object model surface is described. Second, by applying least-squares conformal mapping the object surface is flattened, which is shown in detail. The 2D pattern points are then projected onto the object surface. The hybrid force/motion control scheme is presented in Chapter 4. To ensure the stability of the null space of the redundant manipulator, a null-space controller is designed. In Chapter 5, the proposed approach is applied to an experimental setup and the results are discussed. Finally in Chapter 6, conclusions of the work are given.

2 Mathematical Model

In this chapter, the mathematical model of the robot is introduced. The robot used in this work is a KUKA LWR iiwa 14 R820. This 7-axis robot is modeled in form of rigid bodies which are connected with elastic joints.

First, in Section 2.1 the kinematics of the robot is described. Then, in Section 2.2.1 the dynamic model of a rigid-body robot is discussed. Finally, the complete and the reduced dynamic model of a robot with flexible joints are derived in Section 2.2.2.

2.1 Kinematics

Kinematics describes the motion of bodies without considering forces and torques that cause the motion. In this section, the direct and inverse kinematics and the manipulator Jacobian of the KUKA LWR iiwa are derived.

2.1.1 Direct Kinematics

The operational space of the robot is the six-dimensional space consisting of the Cartesian position $\mathbf{p} \in \mathbb{R}^3$ and the orientation expressed with Euler angles $\mathbf{\Phi} \in \mathbb{R}^3$. The end-effector pose $\mathbf{x}_e$ describes the position and the orientation of the end-effector. The direct kinematics determines the pose of the end-effector as a function of the joint angles $\mathbf{q} = [q_1 \ \cdots \ q_7]^{\mathrm{T}}$, i.e. $\mathbf{x}_e = \mathbf{f}(\mathbf{q})$. For this purpose, the homogeneous transformation is employed.

Let $\mathbf{R}_i^j \in \mathrm{SO}(3)$ and $\mathbf{d}_i^j \in \mathbb{R}^3$ be the rotation matrix and translation vector of the coordinate frame i with respect to the coordinate frame j. Then, the homogeneous transformation describing the position and orientation of frame j with respect to the frame i is given by

$$\mathbf{T}_i^j = \begin{bmatrix} \mathbf{R}_i^j & \mathbf{d}_i^j \\ \mathbf{0} & 1 \end{bmatrix} . \tag{2.1}$$

To take a systematic approach in the choice of each rigid-body frame the Denavit-Hartenberg convention is used, see [10]. First, to each link of the robot a frame is attached and then the homogeneous transformations between the frames are constructed. Thereby, the homogeneous transformation between two consecutive joints is computed with

$$\begin{aligned}\mathbf{T}_{i-1}^{i} &= \mathrm{Rot}_z(\vartheta_i)\mathrm{Trans}_z(d_i)\mathrm{Trans}_x(a_i)\mathrm{Rot}_x(\alpha_i) \\ &= \begin{bmatrix} \cos(\vartheta_i) & -\sin(\vartheta_i) & 0 & 0 \\ \sin(\vartheta_i) & \cos(\vartheta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} .\end{aligned} \tag{2.2}$$

Here, $\mathrm{Rot}_z(\vartheta_i)$ and $\mathrm{Rot}_x(\alpha_i)$ denote rotations around the z-axis and x-axis, respectively. The matrices $\mathrm{Trans}_z(d_i)$ and $\mathrm{Trans}_x(a_i)$ are transformation matrices for translations along the z-axis and x-axis, with the Denavit-Hartenberg parameters denoted as $(\vartheta_i, d_i, a_i, \alpha_i)$ and given in Table 2.1. The numeric values of d_i are listed in Appendix A. Figure 2.1 shows the robot with its coordinate frames which are set according to the Denavit-Hartenberg convention. A pen gripper as end-effector is also depicted. An additional coordinate frame is set at the pen tip as the end-effector frame. The direct kinematics is obtained as

$$\mathbf{T}_0^e(\mathbf{q}, d_e) = \mathbf{T}_0^1(q_1)\mathbf{T}_1^2(q_2)\dots\mathbf{T}_6^7(q_7)\mathbf{T}_7^e(d_e) \ . \tag{2.3}$$

i	ϑ_i	d_i	a_i	α_i
1	q_1	d_1	0	$\pi/2$
2	q_2	0	0	$-\pi/2$
3	q_3	d_3	0	$-\pi/2$
4	q_4	0	0	$\pi/2$
5	q_5	d_5	0	$\pi/2$
6	q_6	0	0	$-\pi/2$
7	q_7	0	0	0
e	0	d_e	0	0

Table 2.1: Denavit-Hartenberg parameters of the KUKA LWR iiwa, with the numeric values in Appendix A.

2.1.2 Inverse Kinematics

Inverse kinematics is the problem of finding the joint angles for corresponding end-effector poses. It is more complex than the direct kinematics problem because the inverse function of (2.3) has to be derived and there might be no solution. Even if a solution exists, it may not be unique. For the calculation of the inverse kinematics, first from a given end-effector pose $\mathbf{T}_0^e$ the corresponding pose of the 7th frame is found. Using the inverse transformation $(\mathbf{T}_7^e)^{-1}$, the pose of the tip of the last robot link (flange) with respect to the base frame is determined as

$$\mathbf{T}_0^7(\mathbf{q}) = \mathbf{T}_0^e\,(\mathbf{T}_7^e)^{-1} \ . \tag{2.4}$$

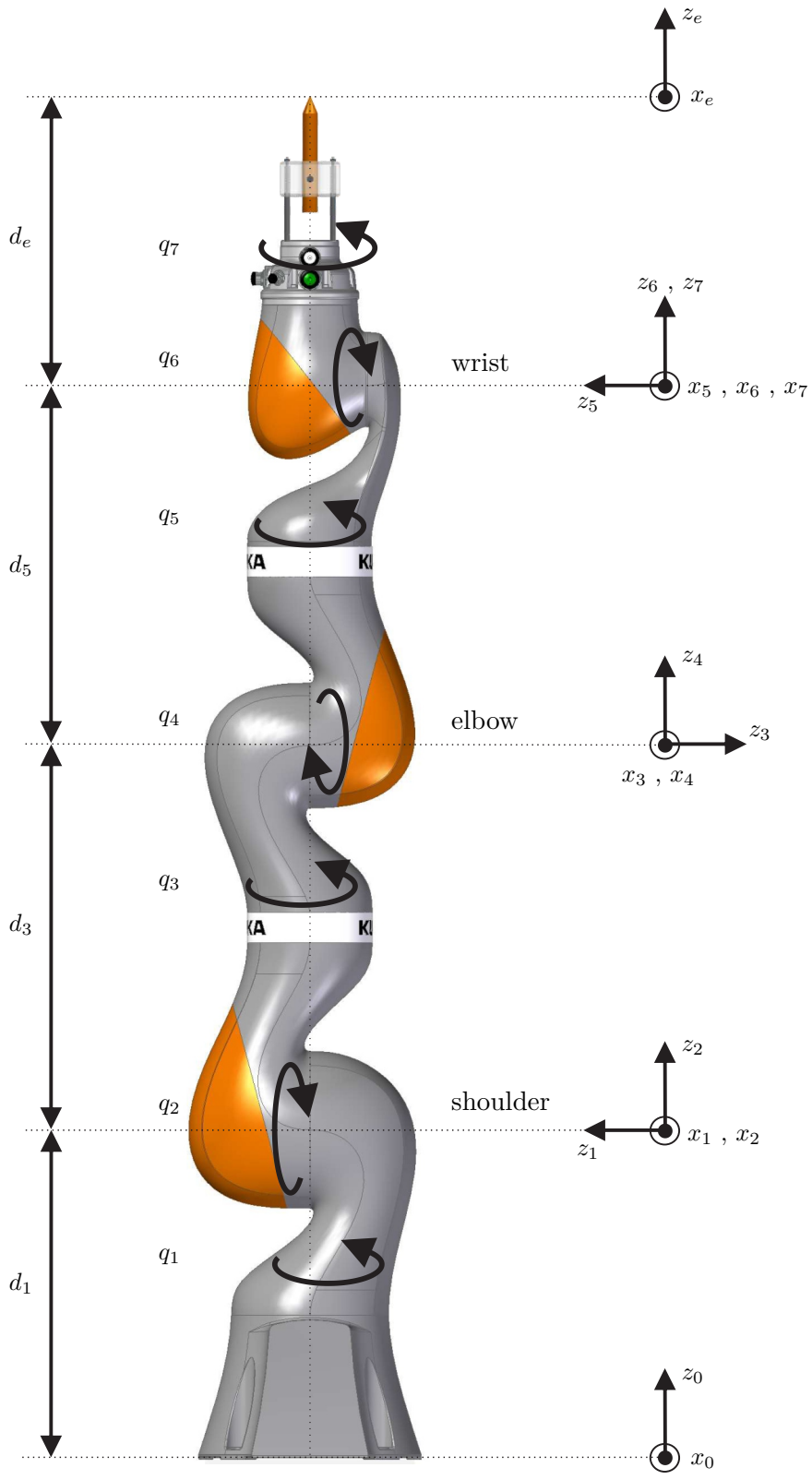


Figure 2.1: Robot KUKA LWR iiwa and the coordinate frames according to the Denavit-Hartenberg convention.

A manipulator having more degrees of freedom than the dimension of the operational space is called kinematically redundant manipulator. In this section, based on [11], an analytical approach is applied to solve the inverse kinematics problem for a 7-DOF redundant manipulator while satisfying the joint limits. The robot KUKA LWR iiwa can be described as a Spherical-Roll-Spherical (S-R-S) manipulator. The first three joints $(1, 2, 3)$ of the manipulator are considered as shoulder and the last three joints $(5, 6, 7)$ build up the wrist. The middle joint (4) is regarded as elbow.

In the following, the arm angle parameterization method proposed by Kreutz-Delgado [12] is used to represent the redundancy of the S-R-S manipulator. The arm angle ψ is defined as the angle between the arm plane, which is spanned by the shoulder (S), elbow (E), and wrist (W), and the reference plane, see Figure 2.1 and Figure 2.2.

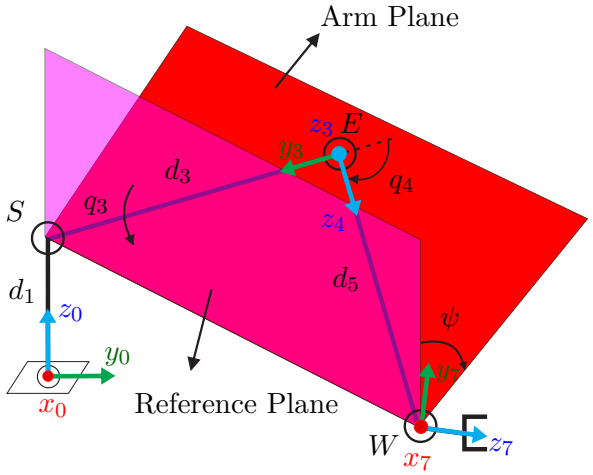


Figure 2.2: Definition of the arm plane and arm angle ψ.

Using the constant vectors

$$\mathbf{d}_0^S = \begin{bmatrix} 0 & 0 & d_1 \end{bmatrix}^{\mathrm{T}}, \tag{2.5a}$$

$$\mathbf{d}_{S,3}^E = \begin{bmatrix} 0 & -d_3 & 0 \end{bmatrix}^{\mathrm{T}}, \tag{2.5b}$$

$$\mathbf{d}_{E,4}^W = \begin{bmatrix} 0 & 0 & d_5 \end{bmatrix}^{\mathrm{T}}, \tag{2.5c}$$

the position of the tip of the robot link 7 with respect to the base coordinate frame is computed as

$$\mathbf{p}_0^7 = \mathbf{R}_0^3 \left(\mathbf{d}_{S,3}^E + \mathbf{R}_3^4 \, \mathbf{d}_{E,4}^W \right) + \mathbf{d}_0^S \, . \tag{2.6}$$

In (2.5), the vectors $\mathbf{d}_0^S$, $\mathbf{d}_{S,3}^E$ and $\mathbf{d}_{E,4}^W$ denote the translation vectors from base to shoulder, shoulder to elbow and elbow to wrist, respectively, and are expressed in the corresponding coordinate frame, see Figure 2.2. The rotation matrix describing the orientation of the end-effector is given by

$$\mathbf{R}_0^7 = \mathbf{R}_0^3 \, \mathbf{R}_3^4 \, \mathbf{R}_4^7 \, . \tag{2.7}$$

The vector $\mathbf{p}_{S,0}^{W}$ from the shoulder to the wrist is calculated as

$$\mathbf{p}_{S,0}^{W} = \mathbf{R}_{0}^{3}\left(\mathbf{d}_{S,3}^{E} + \mathbf{R}_{3}^{4}(q_4)\,\mathbf{d}_{E,4}^{W}\right) \ . \tag{2.8}$$

Using the axis-angle representation, the rotation of the wrist by the angle ψ around the shoulder-wrist axis is written as

$$\mathbf{R}_{0}^{W}(\psi) = \mathbf{I}_3 + \sin(\psi)\mathbf{S}(\mathbf{e}_{S,0}^{W}) + (1-\cos(\psi))\mathbf{S}^2(\mathbf{e}_{S,0}^{W}) \ . \tag{2.9}$$

Here, $\mathbf{I}_3 \in \mathbb{R}^{3\times3}$ is the identity matrix and $\mathbf{S}(\mathbf{e}_{S,0}^{W})$ denotes the skew-symmetric matrix of the unit vector $\mathbf{e}_{S,0}^{W}$. The wrist orientation expressed in the base coordinate frame is given by

$$\mathbf{R}_{0}^{4}(\psi) = \mathbf{R}_{0}^{W}(\psi)\,\mathbf{R}_{0}^{4,o} \ , \tag{2.10}$$

where $\mathbf{R}_{0}^{4,o}$ describes the orientation of the wrist when the arm plane coincides with the reference plane. The superscript o indicates the case for an arm angle of zero. For a given pose, the elbow joint angle q_4 does not depend on the arm angle ψ and is constant, i. e. $\mathbf{R}_{3}^{4}(\psi) = \mathbf{R}_{3}^{4,o}$. Therefore, using (2.10), the matrix $\mathbf{R}_{0}^{3}(\psi)$ is obtained as

$$\mathbf{R}_{0}^{3}(\psi) = \mathbf{R}_{0}^{W}(\psi)\,\mathbf{R}_{0}^{3,o} \ . \tag{2.11}$$

By substituting (2.11) into (2.7), the orientation of the end-effector is written as

$$\mathbf{R}_{0}^{7}(\psi) = \mathbf{R}_{0}^{W}(\psi)\,\mathbf{R}_{0}^{3,o}\,\mathbf{R}_{3}^{4,o}\,\mathbf{R}_{4}^{7}(\psi) \ . \tag{2.12}$$

According to Figure 2.2 the law of cosines is used to calculate the elbow joint angle q_4 with

$$\cos(q_4) = \frac{\left\|\mathbf{p}_{S,0}^{W}\right\|^2 - (d_3)^2 - (d_5)^2}{2d_3\,d_5} \ , \tag{2.13}$$

where $\left\|\mathbf{p}_{S,0}^{W}\right\|$ denotes the Euclidean norm of the vector $\mathbf{p}_{S,0}^{W}$. Since the shoulder joint angles q_1, q_2 and q_3 depend on the arm angle, first the reference joint angles are determined, when the arm angle is zero. In order to find the shoulder reference joint angles q_1^o and q_2^o, the joint angle q_3 in (2.8) is set to zero, i. e.

$$\mathbf{p}_{S,0}^{W} = \mathbf{R}_{0}^{1}(q_1^o)\mathbf{R}_{1}^{2}(q_2^o)\mathbf{R}_{2}^{3}(q_3)\Big|_{q_3=0}\left(\mathbf{d}_{S,3}^{E} + \mathbf{R}_{3}^{4}(q_4)\,\mathbf{d}_{E,4}^{W}\right) \ . \tag{2.14}$$

By inserting the constant vector $\mathbf{p}_{S,0}^{W}$, calculated from (2.8), into (2.14), the angles q_1^o and q_2^o are determined. When the arm angle ψ is given, the shoulder joint angles are obtained as follows. Combining (2.9) and (2.11) leads to

$$\mathbf{R}_{0}^{3}(\psi) = \mathbf{A}_s\sin(\psi) + \mathbf{B}_s\cos(\psi) + \mathbf{C}_s \ , \tag{2.15}$$

with the constant matrices

$$\mathbf{A}_s = \mathbf{S}(\mathbf{e}_{S,0}^W)\,\mathbf{R}_0^{3,o}\,, \tag{2.16a}$$
$$\mathbf{B}_s = -\mathbf{S}^2(\mathbf{e}_{S,0}^W)\,\mathbf{R}_0^{3,o}\,, \tag{2.16b}$$
$$\mathbf{C}_s = \left(\mathbf{e}_{S,0}^W\,(\mathbf{e}_{S,0}^W)^{\mathrm{T}}\right)\mathbf{R}_0^{3,o}\,. \tag{2.16c}$$

The rotation matrix $\mathbf{R}_0^3(\mathbf{q})$ can be calculated using the direct kinematics

$$\mathbf{R}_0^3(q_1,q_2,q_3) = \begin{bmatrix} * & \cos(q_1)\sin(q_2) & * \\ * & \sin(q_1)\sin(q_2) & * \\ \sin(q_2)\cos(q_3) & -\cos(q_2) & -\sin(q_2)\sin(q_3) \end{bmatrix}, \tag{2.17}$$

where the elements denoted by $*$ are omitted. Comparing (2.15) with (2.17) yields the shoulder joint angles as a function of the arm angle ψ

$$\tan(q_1) = \frac{\left(\mathbf{R}_0^3\right)_{22}}{\left(\mathbf{R}_0^3\right)_{12}} = \frac{(\mathbf{A}_s)_{22}\,\sin(\psi) + (\mathbf{B}_s)_{22}\,\cos(\psi) + (\mathbf{C}_s)_{22}}{(\mathbf{A}_s)_{12}\,\sin(\psi) + (\mathbf{B}_s)_{12}\,\cos(\psi) + (\mathbf{C}_s)_{12}}\,, \tag{2.18a}$$
$$\cos(q_2) = -\left(\mathbf{R}_0^3\right)_{32} = -(\mathbf{A}_s)_{32}\,\sin(\psi) - (\mathbf{B}_s)_{32}\,\cos(\psi) - (\mathbf{C}_s)_{32}\,, \tag{2.18b}$$
$$\tan(q_3) = \frac{-\left(\mathbf{R}_0^3\right)_{33}}{\left(\mathbf{R}_0^3\right)_{31}} = \frac{-(\mathbf{A}_s)_{33}\,\sin(\psi) - (\mathbf{B}_s)_{33}\,\cos(\psi) - (\mathbf{C}_s)_{33}}{(\mathbf{A}_s)_{31}\,\sin(\psi) + (\mathbf{B}_s)_{31}\,\cos(\psi) + (\mathbf{C}_s)_{31}}\,. \tag{2.18c}$$

Here and in the following, $(\mathbf{F})_{ij}$ denotes the (i,j)-th element of the matrix $\mathbf{F}$.

Similarly the wrist joint angles q_5, q_6 and q_7 are obtained. On the one hand by using (2.9), the rotation matrix $\mathbf{R}_4^7(\psi)$ is given in the form

$$\mathbf{R}_4^7(\psi) = \mathbf{A}_w\sin(\psi) + \mathbf{B}_w\cos(\psi) + \mathbf{C}_w\,, \tag{2.19}$$

with the matrices

$$\mathbf{A}_w = \mathbf{R}_4^3\,\mathbf{A}_s^{\mathrm{T}}\,\mathbf{R}_0^7\,, \tag{2.20a}$$
$$\mathbf{B}_w = \mathbf{R}_4^3\,\mathbf{B}_s^{\mathrm{T}}\,\mathbf{R}_0^7\,, \tag{2.20b}$$
$$\mathbf{C}_w = \mathbf{R}_4^3\,\mathbf{C}_s^{\mathrm{T}}\,\mathbf{R}_0^7\,. \tag{2.20c}$$

On the other hand, the rotation matrix $\mathbf{R}_4^7(\mathbf{q})$ can be computed from the direct kinematics with

$$\mathbf{R}_4^7(q_5,q_6,q_7) = \begin{bmatrix} * & * & -\cos(q_5)\sin(q_6) \\ * & * & -\sin(q_5)\sin(q_6) \\ \sin(q_6)\cos(q_7) & -\sin(q_6)\sin(q_7) & \cos(q_6) \end{bmatrix}. \tag{2.21}$$

Comparing (2.21) with (2.19), the wrist joint angles are derived for a given arm angle ψ with

$$\tan(q_5) = \frac{-\left(\mathbf{R}_4^7\right)_{23}}{-\left(\mathbf{R}_4^7\right)_{13}} = \frac{-(\mathbf{A}_w)_{23}\,\sin(\psi) - (\mathbf{B}_w)_{23}\,\cos(\psi) - (\mathbf{C}_w)_{23}}{-(\mathbf{A}_w)_{13}\,\sin(\psi) - (\mathbf{B}_w)_{13}\,\cos(\psi) - (\mathbf{C}_w)_{13}}\ , \tag{2.22a}$$

$$\cos(q_6) = \left(\mathbf{R}_4^7\right)_{33} = (\mathbf{A}_w)_{33}\,\sin(\psi) + (\mathbf{B}_w)_{33}\,\cos(\psi) + (\mathbf{C}_w)_{33}\ , \tag{2.22b}$$

$$\tan(q_7) = \frac{-\left(\mathbf{R}_4^7\right)_{32}}{\left(\mathbf{R}_4^7\right)_{31}} = \frac{-(\mathbf{A}_w)_{32}\,\sin(\psi) - (\mathbf{B}_w)_{32}\,\cos(\psi) - (\mathbf{C}_w)_{32}}{(\mathbf{A}_w)_{31}\,\sin(\psi) + (\mathbf{B}_w)_{31}\,\cos(\psi) + (\mathbf{C}_w)_{31}}\ . \tag{2.22c}$$

When the movable ranges of the joints are limited, not all inverse kinematic solutions are feasible. To avoid joint limit violations, the resulting joint angles $\mathbf{q}$ are modified using the arm angle ψ. In a first step, for an arbitrary arm angle ψ the joint angles $\mathbf{q}$ are computed with (2.18) and (2.22). Then, by applying simple modifications, feasible solutions satisfying the joint limits are obtained. In this work, using the approach in [13], the modified joint angles are computed as follows

$$q_i' = q_i + p(q_i)\ , \qquad i = 1, 3, 5 \tag{2.23a}$$

$$q_{i+1}' = -q_{i+1}\ , \tag{2.23b}$$

$$q_{i+2}' = q_{i+2} + p(q_i)\ , \tag{2.23c}$$

with

$$p(q_i) = \begin{cases} \pi & \text{if } q_i < q_i^{min} \\ -\pi & \text{if } q_i > q_i^{max} \end{cases}\ . \tag{2.24}$$

Herein, variables with prime superscript denote modified joint angles. Equation (2.23) can be interpreted as flipping of the shoulder, elbow or wrist angles of the manipulator. If any of these joint angles exceed their limit, the whole calculation of the inverse kinematics (from equation (2.15) to (2.24)) is repeated using a different arm angle ψ. The joint angle limits of the KUKA iiwa are given in Appendix A.

2.1.3 Manipulator Jacobian

The geometric manipulator Jacobian $\mathbf{J}_0^i(\mathbf{q})$ describes the relation between joint velocities and linear and angular velocities of each link of the robot in the form

$$\begin{bmatrix} \mathbf{v}_0^i \\ \boldsymbol{\omega}_0^i \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{v,0}^i(\mathbf{q}) \\ \mathbf{J}_{\omega,0}^i(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}_0^i(\mathbf{q})\dot{\mathbf{q}}\ . \tag{2.25}$$

Herein, $\mathbf{v}_0^i(t)$ and $\boldsymbol{\omega}_0^i(t)$ describe the linear velocity and the angular velocity of the i-th link and $\dot{\mathbf{q}}$ denotes the time derivative of $\mathbf{q}$. The subscript 0 indicates that the velocities are expressed in the base coordinate frame.

By introducing the skew-symmetric matrix

$$\mathbf{S}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} , \tag{2.26}$$

which can be calculated with

$$\mathbf{S}(\boldsymbol{\omega}_0^i) = \dot{\mathbf{R}}_0^i(\mathbf{q})\Big(\mathbf{R}_0^i(\mathbf{q})\Big)^{\mathrm{T}} = \sum_{i=1}^{7}\left(\frac{\partial}{\partial q_i}\mathbf{R}_0^i(\mathbf{q})\right)\Big(\mathbf{R}_0^i(\mathbf{q})\Big)^{\mathrm{T}}\dot{q}_i \ , \tag{2.27}$$

the rotational Jacobian matrix $\mathbf{J}_{\omega,0}^i(\mathbf{q})$ can be obtained. In (2.27), $\mathbf{R}_0^i$ and $\dot{\mathbf{R}}_0^i$ are the rotation matrix and its time derivative.

The linear velocity of the i-th link can be written as

$$\mathbf{v}_0^i = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{p}_0^i(\mathbf{q}) = \sum_{i=1}^{7}\frac{\partial \mathbf{p}_0^i}{\partial q_i}\dot{q}_i = \mathbf{J}_{v,0}^i(\mathbf{q})\dot{\mathbf{q}} \ , \tag{2.28}$$

which leads directly to the translational Jacobian matrix $\mathbf{J}_{v,0}^i(\mathbf{q})$. Here and in the following, the symbols d and ∂ denote the total and partial derivatives, respectively. Further, $\mathbf{p}_0^i$ is the position vector to the center of mass of the i-th link of the robot expressed in the base coordinate frame. For the calculation of the translational Jacobian matrix of the end-effector $\mathbf{J}_{v,0}^e(\mathbf{q})$, the vector $\mathbf{p}_0^e$ is used, which is the position of the origin of the end-effector frame with respect to the base frame.

2.2 Dynamic Model

In this section, first the dynamic model of the robot as a rigid-body system is discussed. Then, the dynamic model is extended to obtain the robot model with flexible joints.

2.2.1 Robot Model with Rigid-Body Links

The equations of motion of the robot manipulator with n rigid links is derived using the Euler-Lagrange equations [14] as follows

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial}{\partial \dot{q}_i}L\right) - \frac{\partial}{\partial q_i}L = \tau_i \, , \qquad i = 1, \ldots, n \tag{2.29}$$

with the Lagrangian function of the system $L = T - V$, where T denotes the kinetic energy and V the potential energy. Here, τ_i is the generalized force associated with the generalized coordinate q_i. The KUKA iiwa is a robot with 7 links and therefore $n = 7$.

The kinetic energy consists of a translational and a rotational part and is given by

$$T = \frac{1}{2}\sum_{i=1}^{7}\left(m_i\Big(\mathbf{v}_0^i\Big)^{\mathrm{T}}\mathbf{v}_0^i + \Big(\boldsymbol{\omega}_0^i\Big)^{\mathrm{T}}\mathbf{I}_0^i\boldsymbol{\omega}_0^i\right) \ , \tag{2.30}$$

where m_i is the mass of the i-th link and $\mathbf{I}_0^i$ is the inertia tensor expressed in the base coordinate frame. Using the manipulator Jacobian from (2.25), equation (2.30) leads to

$$T = \frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}} \sum_{i=1}^{7} \left(m_i \left(\mathbf{J}_{v,0}^i(\mathbf{q}) \right)^{\mathrm{T}} \mathbf{J}_{v,0}^i(\mathbf{q}) + \left(\mathbf{J}_{\omega,0}^i(\mathbf{q}) \right)^{\mathrm{T}} \mathbf{I}_0^i \mathbf{J}_{\omega,0}^i(\mathbf{q}) \right) \dot{\mathbf{q}} \ . \tag{2.31}$$

By introducing the symmetric positive definite generalized mass matrix $\mathbf{M}(\mathbf{q})$ as

$$\mathbf{M}(\mathbf{q}) = \sum_{i=1}^{7} \left(m_i \left(\mathbf{J}_{v,0}^i(\mathbf{q}) \right)^{\mathrm{T}} \mathbf{J}_{v,0}^i(\mathbf{q}) + \left(\mathbf{J}_{\omega,0}^i(\mathbf{q}) \right)^{\mathrm{T}} \mathbf{I}_0^i \mathbf{J}_{\omega,0}^i(\mathbf{q}) \right) \ , \tag{2.32}$$

the kinetic energy T can simply be written as

$$T = \frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}} \mathbf{M}(\mathbf{q}) \dot{\mathbf{q}} \ . \tag{2.33}$$

The potential energy of the rigid bodies consists only of gravity potential. The potential energy of the i-th link is computed with the corresponding position of the center of mass $\mathbf{p}_0^i$. With the gravitational acceleration g acting in the positive z direction of the base frame, see Figure 2.1, for a ceiling-mounted robot, the potential energy is written as

$$V = \sum_{i=1}^{7} V_i = \sum_{i=1}^{7} -m_i \begin{bmatrix} 0 & 0 & g \end{bmatrix} \mathbf{p}_0^i(\mathbf{q}) \ . \tag{2.34}$$

Clearly, the potential energy is a function of the joint angles $\mathbf{q}$, but not of the joint velocities $\dot{\mathbf{q}}$.

Finally with the potential forces given by

$$\mathbf{g}(\mathbf{q}) = \frac{\partial V}{\partial \mathbf{q}} \ , \tag{2.35}$$

the Euler-Lagrange equations are written in matrix form as

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext} \ . \tag{2.36}$$

Herein, $\boldsymbol{\tau} = [\tau_1 \ \cdots \ \tau_7]^{\mathrm{T}}$ are the motor torques and $\boldsymbol{\tau}_{ext}$ denote the external torques, which also include the joint friction torques. Furthermore, the matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is called Coriolis matrix and the vector $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$ gives the Coriolis and centrifugal force terms. The elements of the matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ are given by

$$C_{kj}(\mathbf{q}) = \sum_{i=1}^{7} c_{ijk}(\mathbf{q})\dot{q}_i = \sum_{i=1}^{7} \frac{1}{2} \left\{ \frac{\partial}{\partial q_i} m_{kj}(\mathbf{q}) + \frac{\partial}{\partial q_j} m_{ki}(\mathbf{q}) - \frac{\partial}{\partial q_k} m_{ij}(\mathbf{q}) \right\} \dot{q}_i \ , \tag{2.37}$$

where the coefficients c_{ijk} are termed Christoffel symbols, and C_{kj} and m_{kj} denote the (k, j)-th elements of the matrices $\mathbf{C}$ and $\mathbf{M}$.

2.2.2 Robot Model with Flexible Joints

In this section, based on [15] the dynamic model of a robot with flexible joints is discussed. Under the assumption that the rotational part of the kinetic energy of the rotors is determined only by its relative movement with respect to the previous link, the reduced simplified model is derived. This assumption is justified if the gear ratios of the motors are high and therefore, the rotors will turn much faster than the links of the robot itself [15].

Joint flexibility can be modeled as linear springs located between the motors and the connected links. The extended dynamic model considering the flexible joints reads as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{K}(\boldsymbol{\vartheta} - \mathbf{q}) + \boldsymbol{\tau}_{ext} \ , \tag{2.38a}$$

$$\mathbf{B}\ddot{\boldsymbol{\vartheta}} + \mathbf{K}(\boldsymbol{\vartheta} - \mathbf{q}) = \boldsymbol{\tau}_m \ , \tag{2.38b}$$

where $\mathbf{B}$ denotes the positive definite mass matrix of the motors and $\mathbf{K}$ is a positive definite diagonal matrix whose elements represent the stiffness of the transmission gears. The vectors $\boldsymbol{\vartheta}$ and $\boldsymbol{\tau}_m$ are the angles of the motor shafts and the motor torques, respectively.

By applying the singular perturbation theory to the robot model with flexible joints (2.38), the reduced model is derived. The system is split up into a slow and a fast system. First, by substituting

$$\boldsymbol{\tau} = \mathbf{K}(\boldsymbol{\vartheta} - \mathbf{q}) \ , \qquad \ddot{\boldsymbol{\vartheta}} = \mathbf{K}^{-1}\ddot{\boldsymbol{\tau}} + \ddot{\mathbf{q}} \ , \tag{2.39}$$

in (2.38), one can obtain

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_{ext} \ , \tag{2.40a}$$

$$\mathbf{B}\mathbf{K}^{-1}\ddot{\boldsymbol{\tau}} + \boldsymbol{\tau} = \boldsymbol{\tau}_m - \mathbf{B}\ddot{\mathbf{q}} \ . \tag{2.40b}$$

Due to the high stiffness of the joints, the matrix $\mathbf{K}$ is replaced by $\mathbf{K} = \frac{\mathbf{K}_\varepsilon}{\varepsilon^2}$ with the small scalar parameter ε and the positive definite diagonal matrix $\mathbf{K}_\varepsilon$, see [15].

The control input $\bar{\boldsymbol{\tau}}_m$ is derived with

$$\bar{\boldsymbol{\tau}}_m = \bar{\boldsymbol{\tau}}_d - \mathbf{K}_\tau(\bar{\boldsymbol{\tau}} - \bar{\boldsymbol{\tau}}_d) - \varepsilon\mathbf{D}_\tau\dot{\bar{\boldsymbol{\tau}}} \ , \tag{2.41}$$

where $\mathbf{K}_\tau$ and $\mathbf{D}_\tau$ are positive definite controller gain matrices. Furthermore, $\bar{\boldsymbol{\tau}}_d$ is used as a new control input for the quasi steady-state system, which can be designed according to a control law for rigid robots. The overlines in (2.41) denote the quasi steady-state variables.

By setting ε to zero and inserting (2.41) in (2.40), the closed-loop quasi steady-state system for the link dynamics is given by

$$\underbrace{\left[\mathbf{M}(\bar{\mathbf{q}}) + (\mathbf{I} + \mathbf{K}_\tau)^{-1}\mathbf{B}\right]}_{\tilde{\mathbf{M}}(\bar{\mathbf{q}})}\ddot{\bar{\mathbf{q}}} + \mathbf{C}\left(\bar{\mathbf{q}},\dot{\bar{\mathbf{q}}}\right)\dot{\bar{\mathbf{q}}} + \mathbf{g}(\bar{\mathbf{q}}) = \bar{\boldsymbol{\tau}}_d + \bar{\boldsymbol{\tau}}_{ext} \ , \tag{2.42}$$

with the identity matrix $\mathbf{I}$. It is seen that compared to the robot model with rigid bodies (2.36) only the mass matrix is modified to $\tilde{\mathbf{M}}(\bar{\mathbf{q}})$. For the sake of readability, all

overlines indicating the quasi steady-state variables are omitted in the following, and for the modified mass matrix $\tilde{\mathbf{M}}(\mathbf{q})$ the notation $\mathbf{M}(\mathbf{q})$ is used. Therefore (2.42) is written as

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}_d + \boldsymbol{\tau}_{ext} \ . \tag{2.43}$$

In order to compensate the friction of the robot joints, first the friction torques are estimated. Using the motor angles $\boldsymbol{\vartheta}$ and the measured joint torques $\boldsymbol{\tau}_s$, a friction observer is implemented in discrete time as

$$\hat{\boldsymbol{\tau}}_{f,k} = -\mathbf{K}_r\mathbf{B}\left(\dot{\boldsymbol{\vartheta}}_k - \dot{\hat{\boldsymbol{\vartheta}}}_k\right) , \tag{2.44a}$$

$$\dot{\hat{\boldsymbol{\vartheta}}}_{k+1} = \dot{\hat{\boldsymbol{\vartheta}}}_k + T_s\mathbf{B}^{-1}\left(\boldsymbol{\tau}_{m,k} - \boldsymbol{\tau}_{s,k} - \hat{\boldsymbol{\tau}}_{f,k}\right) , \tag{2.44b}$$

with the sampling time T_s, see [16, 17]. Herein, the subscript k denotes the signal value at the discrete time $t = kT_s$, $k = 1, 2, \ldots$, i.e. $\boldsymbol{\xi}_k = \boldsymbol{\xi}(kT_s)$. The vectors $\hat{\boldsymbol{\tau}}_{f,k}$ and $\dot{\hat{\boldsymbol{\vartheta}}}_k$ denote the estimated friction torque and the estimated motor velocity, respectively. The matrix $\mathbf{K}_r$ is a diagonal positive definite gain matrix of the observer. By adding $\hat{\boldsymbol{\tau}}_{f,k}$ from (2.44) to the discrete motor torques $\boldsymbol{\tau}_{m,k}$ from (2.41), the remaining joint friction torques are significantly reduced. In Chapter 4, the reduced robot model (2.43) will be considered for the controller design, and it will be assumed that due to the friction compensation, no joint friction is present.

3 Pattern Projection

As discussed in Chapter 1, the goal of this work is to enable the robot to draw on a known 3D object. For this purpose, first the user creates a drawing using a mouse or touch screen in 2D. In order to project those 2D drawing points onto the surface of the 3D object on which the robot should draw, the surface is first flattened using least-squares conformal mapping. This method is discussed in Section 3.1. In Section 3.2, it is described how the object model is decomposed into charts to facilitate the pattern mapping, by applying a segmentation algorithm. Mapping the pattern onto the object surface is the subject of Section 3.3. In Section 3.4, it is shown how a trajectory is constructed from the projected points, by using cubic spline interpolation. Finally, in Section 3.4.1 the measurement of the object pose with an OPTITRACK measurement system is discussed to be able to draw on an object whose location is not known beforehand.

3.1 Least-Squares Conformal Mapping

In this section, based on [18], a conformal mapping using a least-squares approximation of the Cauchy-Riemann equation is introduced. This method basically flattens the 3D object surface into a 2D planar domain.

A conformal map

$$\mathcal{X} : (u, v) \mapsto (x, y) \ , \tag{3.1}$$

is a function which preserves the angles of infinitesimal small figures but not necessarily their size, see [19]. In a conformal map, at each point the tangent vectors to the iso-u and to the iso-v curves are orthogonal to each other and have the same length [18], see Figure 3.1. Let $\mathbf{n}(u, v)$ be the unit normal vector to the surface. Then, the above statement can be written as

$$\mathbf{n}(u, v) \times \frac{\partial \mathcal{X}}{\partial u}(u, v) = \frac{\partial \mathcal{X}}{\partial v}(u, v) \ , \tag{3.2}$$

where $\times$ denotes the cross product of two vectors. In the following, by approximation, the conformality problem (3.2) is formulated as an unconstrained quadratic minimization problem, which can be solved efficiently.

As a first step, a triangulation method is performed to mesh the object surface. Then to each triangle a local orthonormal basis is assigned, with (x_1, y_1), (x_2, y_2) and (x_3, y_3) as the coordinates of its vertices in this local basis. The condition (3.2) can be expressed in the local basis of the triangle T as

$$\frac{\partial \mathcal{X}}{\partial u} - i \frac{\partial \mathcal{X}}{\partial v} = 0 \ , \tag{3.3}$$

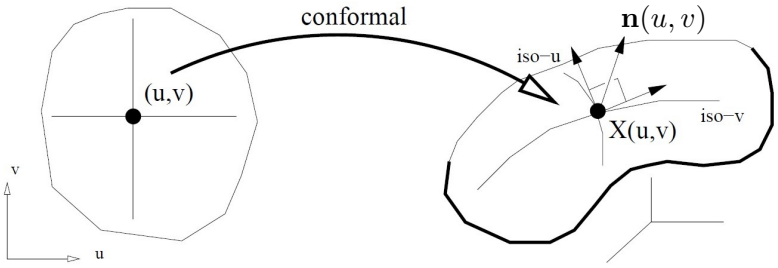


Figure 3.1: Tangent vectors in a conformal map, figure from [18] by Levy.

with $\mathcal{X}$ as a complex number, i.e. $\mathcal{X} = x + iy$, and the imaginary unit i. Using the inverse map

$$\mathcal{U} : (x, y) \mapsto (u, v) \tag{3.4}$$

and based on the Cauchy-Riemann equations, see, e. g., [20], (3.3) becomes

$$\frac{\partial \mathcal{U}}{\partial x} + i\frac{\partial \mathcal{U}}{\partial y} = 0 \ , \tag{3.5}$$

with $\mathcal{U} = u + iv$. This equation cannot be strictly satisfied on the whole surface. Therefore, a minimization of the violation of this condition is performed in the sense of least squares with the cost function

$$C(T) = \int_T \left| \frac{\partial \mathcal{U}}{\partial x} + i\frac{\partial \mathcal{U}}{\partial y} \right|^2 \mathrm{d}A \ . \tag{3.6}$$

The minimization problem for the whole triangulated surface $\mathcal{T}$ is given by

$$\min_{\mathcal{U}} \sum_{T \in \mathcal{T}} C(T) \ . \tag{3.7}$$

For each triangle the gradient is written as

$$\begin{bmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{bmatrix} = \frac{1}{d_T} \begin{bmatrix} y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \ , \tag{3.8}$$

with scalars u_1, u_2 and u_3 associated with the vertices of the triangle. In (3.8), d_T is twice the area of the triangle T and is given by

$$d_T = (x_1 y_2 - y_1 x_2) + (x_2 y_3 - y_2 x_3) + (x_3 y_1 - y_3 x_1) \ . \tag{3.9}$$

Using complex algebra, (3.8) is expressed as

$$\frac{\partial u}{\partial x} + i\frac{\partial u}{\partial y} = \frac{i}{d_T} \begin{bmatrix} W_1 & W_2 & W_3 \end{bmatrix} \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^{\mathrm{T}} \tag{3.10}$$

with

$$\begin{aligned} W_1 &= (x_3 - x_2) + i(y_3 - y_2) \ , \\ W_2 &= (x_1 - x_3) + i(y_1 - y_3) \ , \\ W_3 &= (x_2 - x_1) + i(y_2 - y_1) \ . \end{aligned} \tag{3.11}$$

Suppose the triangle T has the vertices j_1, j_2 and j_3. Introducing the complex numbers $U_j = u_j + iv_j$, the cost function $C(T)$ from (3.6) is written as

$$C(T) = \frac{1}{d_T} \left| \begin{bmatrix} W_{j1,T} & W_{j2,T} & W_{j3,T} \end{bmatrix} \begin{bmatrix} U_{j1} & U_{j2} & U_{j3} \end{bmatrix}^{\mathrm{T}} \right|^2 \ . \tag{3.12}$$

Accordingly, with the vector $\mathbf{U} = [U_1 \ \cdots \ U_n]^{\mathrm{T}}$ the cost function for the whole surface consisting of n vertices is given by

$$C(\mathbf{U}) = \sum_{T \in \mathcal{T}} C(T) \ . \tag{3.13}$$

Since $C(\mathbf{U})$ is quadratic in the complex numbers $U_1, \ldots, U_n$, it is written as

$$C(\mathbf{U}) = \mathbf{U}^* \mathbf{C} \mathbf{U} \ . \tag{3.14}$$

Herein, $\mathbf{C}$ is a Hermitian symmetric matrix and $\mathbf{U}^*$ denotes the Hermitian conjugate of the complex vector $\mathbf{U}$. Let $\mathbf{N} \in \mathbb{R}^{n' \times n}$ be a sparse matrix, where n' is number of triangles and n is number of vertices. Then, $\mathbf{C}$ is expressed as

$$\mathbf{C} = \mathbf{N}^* \mathbf{N} \ . \tag{3.15}$$

The matrix elements $(\mathbf{N})_{ij}$ are computed with

$$(\mathbf{N})_{ij} = \begin{cases} \frac{W_{j,T_i}}{\sqrt{d_{T_i}}} & \text{if vertex } j \text{ belongs to triangle } T_i \ , \\ 0 & \text{otherwise} \ . \end{cases} \tag{3.16}$$

In order for the optimization problem to have a unique and nontrivial solution, at least two vertices must be pre-fixed. The vector $\mathbf{U}$ can be decomposed as $[\mathbf{U}_f^{\mathrm{T}} \ \mathbf{U}_p^{\mathrm{T}}]^{\mathrm{T}}$, where $\mathbf{U}_f$ is the vector of $n - p$ free coordinates of $\mathbf{U}$ and $\mathbf{U}_p$ is the vector of p pre-fixed coordinates of $\mathbf{U}$. It should be noted that only $\mathbf{U}_f$ are the variables of the optimization problem. Accordingly, $\mathbf{N}$ can be decomposed in block matrices as

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}_f & \mathbf{N}_p \end{bmatrix} \ , \tag{3.17}$$

with matrices $\mathbf{N}_f \in \mathbb{R}^{n' \times (n-p)}$ and $\mathbf{N}_p \in \mathbb{R}^{n' \times p}$. Combining (3.14) and (3.15) yields

$$C(\mathbf{U}) = \|\mathbf{N}\mathbf{U}\|^2 = \left\| \mathbf{N}_f \mathbf{U}_f + \mathbf{N}_p \mathbf{U}_p \right\|^2 \ . \tag{3.18}$$

With the matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{N}_f^1 & -\mathbf{N}_f^2 \\ \mathbf{N}_f^2 & \mathbf{N}_f^1 \end{bmatrix}, \qquad \mathbf{b} = -\begin{bmatrix} \mathbf{N}_p^1 & -\mathbf{N}_p^2 \\ \mathbf{N}_p^2 & \mathbf{N}_p^1 \end{bmatrix} \begin{bmatrix} \mathbf{U}_p^1 \\ \mathbf{U}_p^2 \end{bmatrix} \tag{3.19}$$

the cost function (3.13) is rewritten as

$$C(\mathbf{z}) = \|\mathbf{A}\mathbf{z} - \mathbf{b}\|^2 . \tag{3.20}$$

The superscripts ^{1} and ^{2} in (3.19) denote the real and imaginary part and $\mathbf{z}^{\mathrm{T}} = [\mathbf{U}_f^1 \ \mathbf{U}_f^2]$ is the vector of unknowns. This least-squares minimization problem can be solved efficiently using numerical programs like MATLAB.

As suggested in [18], in order to achieve an optimal mapping, the best value for p is 2. In this work, the two vertices with maximum distance from each other are chosen to be pre-fixed.

3.2 Segmentation of the Object Surface

For complicated object models, flattening of the whole model with the parameterization method from Section 3.1 yields no reasonable results. In such cases, a significant angular distortion could occur. Moreover, some parts of the object surface might be flattened on top of each other, which makes the mapping of the 2D pattern much more difficult. To avoid this issue, partitioning of the model is needed. The segmentation method presented here is based on [18]. In order to decompose the object surface into a set of charts, first the chart boundaries must be found. Chart boundaries in flat regions lead to more artifacts in a subsequent pattern mapping. Consequently, boundaries in flat regions should be avoided and therefore large charts with boundaries in high curvature regions are obtained. In this way, the mapping artifacts are minimized. Figure 3.5 shows an example of segmentation applied to a rabbit model.

(a) (b)

Figure 3.2: An example of segmentation of the object surface: (a) Triangular mesh model of the rabbit. (b) Result of the segmentation algorithm.

As a first step, a feature detection algorithm is discussed, which finds feature curves in high curvature regions of the model. Then a chart generation algorithm is implemented, creating charts with boundaries at feature curves.

3.2.1 Feature Detection

For the feature detection phase, a sharpness criterion is applied to the mesh edges. The goal of this phase is to assign a weight value to every edge in the triangular mesh. This weight is proportional to the probability that the edge belongs to a feature. The information obtained in this phase is then used to construct the mesh features. For this purpose, there are several operators in the literature, see [21]. In this work, the second order difference (SOD) operator is used, which uses the angle between the normals of adjacent triangles and is computed with

$$w = \arccos\left(\frac{\mathbf{n}_i \cdot \mathbf{n}_j}{\|\mathbf{n}_i\|\|\mathbf{n}_j\|}\right) . \tag{3.21}$$

The vectors $\mathbf{n}_i$ and $\mathbf{n}_j$ denote the normals of the two triangles, which share the same edge. The operator w is computed efficiently for all edges of the input mesh. However, on highly detailed or noisy meshes, the SOD operator suffers from poor performance, resulting in a large number of small features [21]. Therefore, by choosing a threshold value for w, only the most important edges are selected and the rest of them are neglected. For each of the remaining edges, a feature curve is created by running a feature growing algorithm. A detailed explanation of the algorithm can be found in [18].

3.2.2 Chart Construction

After detection of the sharp features on the object surface, the charts are constructed using a greedy algorithm. Thereby, the charts are generated simultaneously from a set of seeds. This approach is similar to the s-source Dijkstra algorithm proposed in [22], where they use a heap data structure as a priority queue for seeds. In order for the chart boundaries to meet at the feature curves, as suggested in [18], the s-source algorithm is modified as described below.

The detected feature edges, determined in the last section, serve as initial part of the chart boundaries. The seeds of the s-source algorithm are selected in the following way. For each mesh triangle a distance-to-features function is calculated. This function expresses the geodesic distance of each triangle from the detected feature edges. The geodesic distance between two vertices is defined as the number of edges in the shortest path connecting them. The set of seeds is then initialized with the triangles whose distance-to-features values are local maxima. In case no feature is detected on a surface, the two triangles having the maximal distance from each other are set as the seed points.

In the next step, for each of these seeds a chart is constructed and the edges of the seed triangles are stored in the heap. Then by iterating through the elements of the heap, at each iteration the edge element with the largest distance-to-features value is dequeued from the heap and processed as follows. If no chart is assigned to the neighbor triangle of the seed triangle which shares the dequeued edge, the neighbor triangle is added to

the chart of the considering seed. In addition, those edges of the neighbor triangle which belong to the chart boundaries are stored in the heap. If a chart is already assigned to the neighbor triangle, then the distance between these two charts are computed and charts are merged if they meet at a small distance from their seeds. Furthermore, if the required position on which the pattern should be drawn is overlapping several charts, these charts are also merged together. The iteration of the heap is executed until the heap is empty. The charts are then individually flattened using the least-squares conformal mapping presented in Section 3.1.

3.3 Inverse Mapping of the Pattern onto the Object Surface

In this section, based on [23], the inverse mapping of the pattern points is explained in detail. As a pre-processing step, the 2D pattern needs to be scaled, rotated, and translated to a desired position on the flattened surface. In the next step, the pattern is inversely mapped onto the object surface to generate a trajectory for the end-effector.

In order to map the pattern on the flattened surface within the box $[x_{s,min}, x_{s,max}]$ and $[y_{s,min}, y_{s,max}]$, first the center of the flattened surface is determined as

$$\begin{aligned} x_{s,cent} &= \frac{1}{2}\Big(x_{s,max} + x_{s,min}\Big) \ , \\ y_{s,cent} &= \frac{1}{2}\Big(y_{s,max} + y_{s,min}\Big) \ . \end{aligned} \tag{3.22}$$

Similarly, the center of the 2D pattern is computed as

$$\begin{aligned} x_{p,cent} &= \frac{1}{2}\Big(x_{p,max} + x_{p,min}\Big) \ , \\ y_{p,cent} &= \frac{1}{2}\Big(y_{p,max} + y_{p,min}\Big) \ , \end{aligned} \tag{3.23}$$

where $[x_{p,min}, x_{p,max}]$ and $[y_{p,min}, y_{p,max}]$ describe the bounding box of the pattern. Hence, the translation vector from the 2D pattern to the planar surface is given by

$$\begin{aligned} \Delta x &= x_{s,cent} - x_{p,cent} \ , \\ \Delta y &= y_{s,cent} - y_{p,cent} \ . \end{aligned} \tag{3.24}$$

In case the size of the 2D pattern is beyond the range of the flattened surface, the pattern must be scaled to a suitable size using a scaling factor given by

$$s_{scale} = \gamma \min\left(\frac{x_{s,max} - x_{s,min}}{x_{p,max} - x_{p,min}}, \frac{y_{s,max} - y_{s,min}}{y_{p,max} - y_{p,min}}\right) . \tag{3.25}$$

Herein, γ is a coefficient for adjusting the scaling factor. In case an adjustment of the orientation of the pattern is desired, the pattern can be rotated around the z-axis, which is normal to the flattened surface, using the rotation matrix

$$\mathbf{R}_z(\vartheta) = \begin{bmatrix} \cos(\vartheta) & -\sin(\vartheta) \\ \sin(\vartheta) & \cos(\vartheta) \end{bmatrix} , \tag{3.26}$$

with ϑ as the rotation angle around the z-axis. Finally, the k-th pattern point $(x_{p,k}, y_{p,k})$ is transformed to the desired position on the flattened surface with

$$\mathbf{h}_k = s_{scale}\, \mathbf{R}_z(\vartheta) \begin{bmatrix} x_{p,k} \\ y_{p,k} \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} , \tag{3.27}$$

with the transformed pattern point $\mathbf{h}_k$.

3.3.1 Determining the Triangle Containing the Pattern Point

Before inversely mapping the pattern point to the object surface, the triangle which contains the pattern point must be found. A simple way to implement that is to check all triangles if they contain the point. However, it is a very costly approach for large numbers of triangles and pattern points. A more efficient approach can be applied by finding the closest vertex to the pattern point. In this case, only the one-ring neighbor triangles of the vertex need to be searched, not all of them.

To quickly locate the closest vertex to the pattern point, an efficient data structure is needed. In this work, a simple uniform grid structure is used, like in [23]. For this purpose, each vertex is placed in a bin according to its (x, y) coordinates. The size of the grid L_s is chosen depending on the number of vertices and also the size of the flattened surface. After a bin structure is laid over all vertices, the bin containing the pattern point $\mathbf{h}_k(x_k, y_k)$ is easily found according to its row and column indices (i, j) calculated by

$$i = \text{int}\left(\frac{x_k - x_{s,min}}{L_s}\right) , \qquad j = \text{int}\left(\frac{y_k - y_{s,min}}{L_s}\right) . \tag{3.28}$$

Here, $\text{int}(x)$ denotes the integer of the number x. Now, from the vertices stored in $\text{bin}[i][j]$, the closest vertex to $\mathbf{h}_k$ is determined. Finally, only the one-ring neighbor triangles of this vertex is searched to find the triangle enclosing the pattern point $\mathbf{h}_k$.

3.3.2 Linear Mapping Using Barycentric Coordinates

After the triangle containing the pattern point $\mathbf{h}_k$ is determined, this point is inversely mapped onto the 3D object surface by a simple linear interpolation. Let $\mathbf{h}_k \in \mathbb{R}^2$ be a point located within the triangle $\Delta(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)$ on the flattened plane. Then, its corresponding point $\mathbf{r}_k \in \mathbb{R}^3$ lying within the triangle $\Delta(\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3)$ of the 3D object surface can be obtained employing barycentric coordinates, see Figure 3.3. Note that the vertices of the flattened surface are $\mathbf{g}_i \in \mathbb{R}^2$, whereas $\mathbf{w}_i \in \mathbb{R}^3$.

Barycentric coordinates can be used to express the position of any point located within a triangle with respect to the vertices of the triangle, see, e. g., [24]. These coordinates, which are proportional to the area of the three sub-triangles defined by $\mathbf{h}_k$, are computed as

$$A_1 = \frac{\text{area}(\mathbf{h}_k, \mathbf{g}_2, \mathbf{g}_3)}{\text{area}(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)} . \tag{3.29}$$

Here, the function area$(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3)$ calculates the area of the triangle with

$$\text{area}(\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3) = \frac{1}{2} \left\| (\mathbf{g}_2 - \mathbf{g}_1) \times (\mathbf{g}_3 - \mathbf{g}_1) \right\| . \tag{3.30}$$

A_2 and A_3 are computed similarly such that $A_1 + A_2 + A_3 = 1$, i. e. barycentric coordinates are normalized.

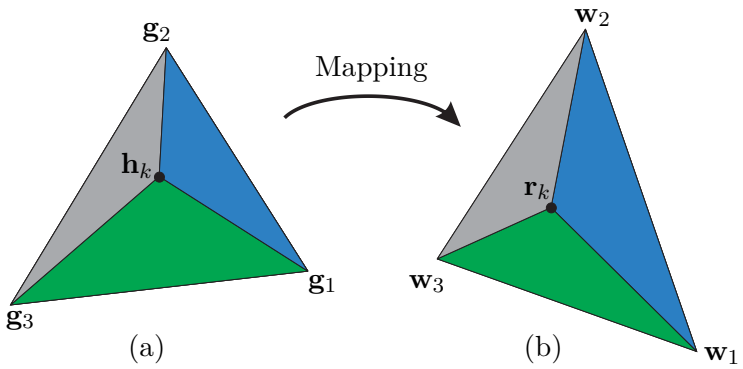


Figure 3.3: Mapping from the flattened surface to the object surface using linear interpolation. (a) Triangle on the flattened surface. (b) Triangle on the object surface.

The mapped point on the object surface is given by

$$\mathbf{r}_k = A_1\mathbf{w}_1 + A_2\mathbf{w}_2 + A_3\mathbf{w}_3 . \tag{3.31}$$

Applying the above method to all points, the pattern to be drawn is mapped onto the object surface. Furthermore, the normal vectors of the triangles containing the pattern points are calculated, which are then used for the desired orientation of the end-effector during the drawing task.

3.4 Trajectory Planning

To enable the end-effector following a desired trajectory, position and orientation information is needed for each projected pattern point. The orientation of a vector can be parametrized using e. g. Euler angles, roll-pitch-yaw angles or unit quaternions. The former two representations are geometrically more intuitive to interpret. However, they have the problem of representation singularities, such as the so-called gimbal lock. This drawback can be overcome using unit quaternions, which are more efficient in numerical computation due to the linearity of quaternion algebra. A quaternion consists of a scalar part η and a vector part $\boldsymbol{\epsilon}$ written as

$$Q = \{\eta \quad \boldsymbol{\epsilon}\} = \{\eta \quad \epsilon_1 \quad \epsilon_2 \quad \epsilon_3\} . \tag{3.32}$$

A rotation can be expressed by a unit quaternion, which satisfies the condition

$$\|Q\|_2^2 = \eta^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 = 1 . \tag{3.33}$$

For a given rotation matrix $\mathbf{R}$, the corresponding quaternion is computed as

$$\eta = \frac{\sqrt{(\mathbf{R})_{11} + (\mathbf{R})_{22} + (\mathbf{R})_{33} + 1}}{2} \tag{3.34a}$$

$$\boldsymbol{\epsilon} = \frac{1}{2}\begin{bmatrix} \operatorname{sign}((\mathbf{R})_{32} - (\mathbf{R})_{23})\sqrt{(\mathbf{R})_{11} - (\mathbf{R})_{22} - (\mathbf{R})_{33} + 1} \\ \operatorname{sign}((\mathbf{R})_{13} - (\mathbf{R})_{31})\sqrt{(\mathbf{R})_{22} - (\mathbf{R})_{33} - (\mathbf{R})_{11} + 1} \\ \operatorname{sign}((\mathbf{R})_{21} - (\mathbf{R})_{12})\sqrt{(\mathbf{R})_{33} - (\mathbf{R})_{11} - (\mathbf{R})_{22} + 1} \end{bmatrix}, \tag{3.34b}$$

see, e. g., [10]. Here, the sign function is defined as

$$\operatorname{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}. \tag{3.35}$$

Now, to obtain a trajectory for the end-effector, first a parametric cubic spline curve is constructed for both position and orientation, which passes through the projected points $\mathbf{r}_k$ from (3.31) with the corresponding triangle orientation in quaternion $Q_k = \{\eta_k \quad \boldsymbol{\epsilon}_k\}$ from (3.34). Then, the spline curve is time parameterized with a twice continuously differentiable curve according to Figure 3.4. This leads to an acceleration of the end-effector at the initial stage of drawing, starting from velocity zero. At the end, a deceleration is performed, which stops the end-effector.

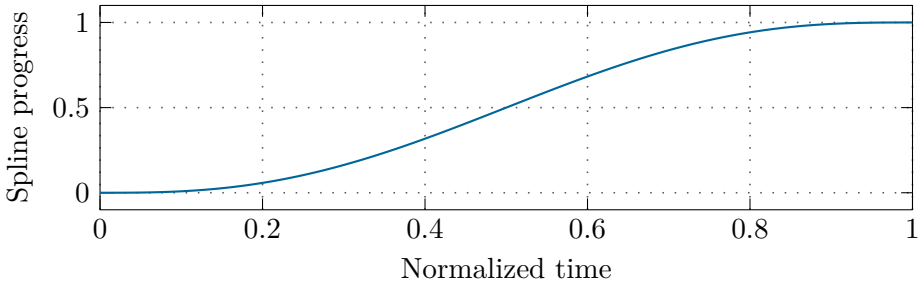


Figure 3.4: Time evolution of the spline curve progress.

3.4.1 Object Pose Measurement

The 3D object on which the robot draws is positioned in the robot workspace. In order to obtain the position and orientation of the object, the OPTITRACK measurement system [25] is used. For this purpose, six Prime 17W infrared cameras, see Figure 3.5, are used to capture the environment, in which the object can be placed arbitrarily. A rigid body, fitted with passive retro-reflective markers, is attached to the tracked object.

The OPTITRACK system consists also of the software MOTIVE, which is a platform designed to control the motion capture system for various tracking applications. MOTIVE computes the position and orientation of the rigid body frame r with respect to its reference coordinate frame m. The orientation information is given by quaternions, which

Figure 3.5: OPTITRACK Prime 17W camera, from [25].

are then converted to the corresponding rotation matrices. The measured pose of the rigid body is expressed as a homogeneous transformation $\mathbf{T}_m^r$. To draw on the 3D object with the robot, the measured object pose has to be transformed to the robot base frame. The transformation of this pose is performed in two steps, see Figure 3.6. First, the object coordinate frame j is transformed to the frame of the rigid body r using the transformation matrix

$$\mathbf{T}_r^j = \begin{bmatrix} \mathbf{R}_x(-90°) & \mathbf{d}_r^j \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_z(-90°) & 0 \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.36}$$

where $\mathbf{R}_x$ and $\mathbf{R}_z$ are rotation matrices around the x-axis and z-axis, respectively. In the second step, the pose is transformed from the OPTITRACK reference frame m to the robot base frame with the matrix

$$\mathbf{T}_0^m = \begin{bmatrix} \mathbf{R}_z(-90°) & \mathbf{d}_0^m \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_x(-90°) & 0 \\ \mathbf{0} & 1 \end{bmatrix}. \tag{3.37}$$

Now, by multiplying all three matrices, the object pose expressed in the robot base frame is computed with

$$\mathbf{T}_0^j = \mathbf{T}_0^m \mathbf{T}_m^r \mathbf{T}_r^j . \tag{3.38}$$

The vectors $\mathbf{d}_r^j$ and $\mathbf{d}_0^m$ denote the translation vectors between the coordinate frames and the corresponding numerical values are given in Appendix A.

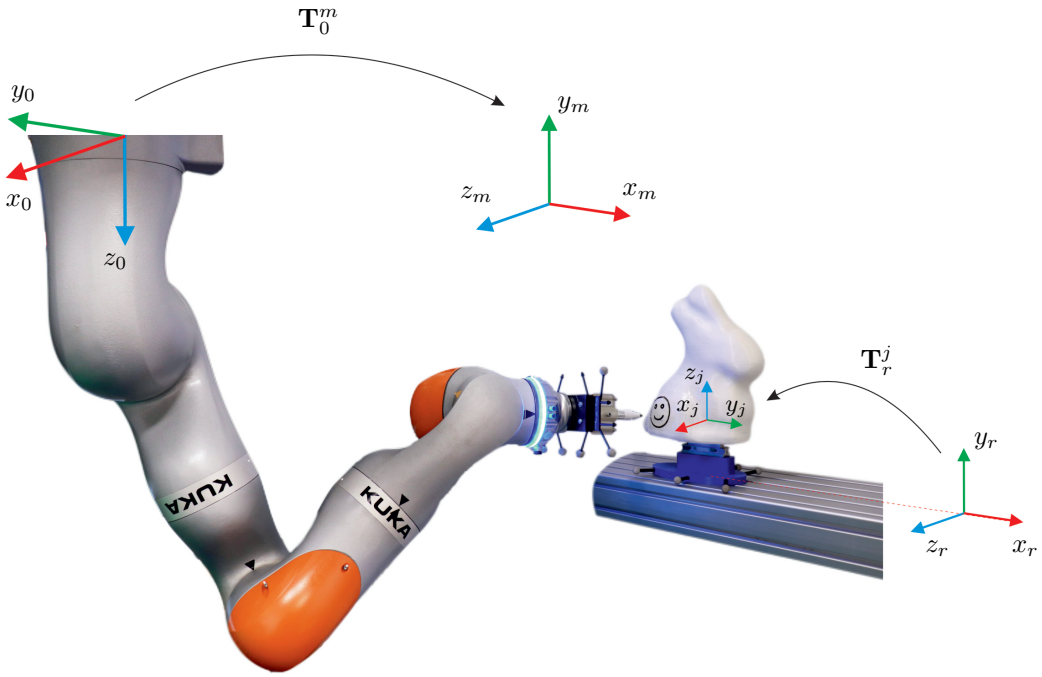


Figure 3.6: Coordinate frames for the transformation of the object pose to the robot base frame.

4 Control Concepts

For drawing a pattern on the object, the pen is in contact with the object surface. Therefore, both the motion of the end-effector and the force normal to the surface are controlled.

In this chapter, the motion control is introduced, where first a short discussion of the orientation error in operational space is given. Then in Section 4.1.1, the joint space computed torque control law is described. For drawing on the object, the motion control of the end-effector is performed in operational space. This controller is described in Section 4.1.2. The control laws considering the contact force are introduced in Section 4.2. A model-based estimation of the contact force using the joint angle and joint torque measurements is addressed in Section 4.2.1. The combination of the force/motion control is derived in Section 4.2.2. Finally, since the robot is a redundant manipulator, a null-space control law is required, which is presented in Section 4.3.

4.1 Motion Control

For the trajectory following control in the operational space, the orientation error should be defined. Using the unit quaternions $Q = \{\eta \quad \boldsymbol{\epsilon}\}$, the orientation error can be obtained as follows, see, e. g., [26]. Let $\mathbf{R}_d$ and $\mathbf{R}_e$ denote the desired and the current rotation matrix of the end-effector, respectively. The orientation error is given by

$$\Delta\mathbf{R} = \mathbf{R}_d\mathbf{R}_e^{\mathrm{T}} \ . \tag{4.1}$$

Considering the following relations from the quaternion algebra

$$Q^{-1} = \{\eta \quad -\boldsymbol{\epsilon}\} \tag{4.2a}$$

$$Q_1 \otimes Q_2 = \{\eta_1\eta_2 - \boldsymbol{\epsilon}_1^{\mathrm{T}}\boldsymbol{\epsilon}_2, \quad \eta_1\boldsymbol{\epsilon}_2 + \eta_2\boldsymbol{\epsilon}_1 + \boldsymbol{\epsilon}_1 \times \boldsymbol{\epsilon}_2\} \ , \tag{4.2b}$$

the orientation error can be expressed as

$$\Delta Q = Q_d \otimes Q_e^{-1} \ , \tag{4.3}$$

where Q_d and Q_e are the corresponding quaternions to $\mathbf{R}_d$ and $\mathbf{R}_e$ and $\Delta Q = \{\Delta\eta, \Delta\boldsymbol{\epsilon}\}$. Note that if the desired and the current orientations are aligned, the error is calculated to be $\Delta Q = \{1, \mathbf{0}\}$. Therefore, the orientation error of the end-effector $\mathbf{e}_o$ is defined as the vector part of ΔQ

$$\mathbf{e}_o = \Delta\boldsymbol{\epsilon} = \eta_e\boldsymbol{\epsilon}_d - \eta_d\boldsymbol{\epsilon}_e - \mathbf{S}(\boldsymbol{\epsilon}_d)\boldsymbol{\epsilon}_e \ . \tag{4.4}$$

Herein, $\mathbf{S}(\cdot)$ denotes the skew-symmetric operator, see (2.26).

4.1.1 Joint Space Computed Torque Control

In the first step of the drawing process, the robot end-effector must be brought from its current pose to the drawing start pose. For this purpose, a computed torque control in joint space is constructed as follows.

The control law consists of two parts. First, the nonlinear terms are canceled using exact input-output linearization. Then, by means of a linear feedback control law the trajectory error is stabilized, see, e. g., [27]. Consider the reduced model, which is derived in Chapter 2 to

$$\mathbf{M}(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}_d + \boldsymbol{\tau}_{ext}\ . \tag{4.5}$$

It is assumed that during the approach of the pen to the object the robot moves in free space and that the joint friction torques are compensated. Therefore, no external forces $\boldsymbol{\tau}_{ext}$ are present. The exact input-output linearization of the system dynamics with the control input

$$\boldsymbol{\tau}_d = \mathbf{M}(\mathbf{q})\mathbf{v} + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}})\ , \tag{4.6}$$

with

$$\mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})\ , \tag{4.7}$$

is used. The new control input

$$\mathbf{v} = \ddot{\mathbf{q}}_d + \mathbf{K}_d\,\dot{\tilde{\mathbf{q}}} + \mathbf{K}_p\,\tilde{\mathbf{q}} + \mathbf{K}_i \int \tilde{\mathbf{q}}\ \mathrm{d}t\ , \tag{4.8}$$

with the positive definite control gains $\mathbf{K}_d$, $\mathbf{K}_p$ and $\mathbf{K}_i$, yields the asymptotically stable error dynamics

$$\ddot{\tilde{\mathbf{q}}} + \mathbf{K}_d\,\dot{\tilde{\mathbf{q}}} + \mathbf{K}_p\,\tilde{\mathbf{q}} + \mathbf{K}_i \int \tilde{\mathbf{q}}\ \mathrm{d}t = \mathbf{0}\ . \tag{4.9}$$

Here, $\tilde{\mathbf{q}} = \mathbf{q}_d - \mathbf{q}_e$ is the trajectory tracking error in the joint space. The complete computed torque control law is written as

$$\boldsymbol{\tau}_d = \mathbf{M}(\mathbf{q})\left(\ddot{\mathbf{q}}_d + \mathbf{K}_d\,\dot{\tilde{\mathbf{q}}} + \mathbf{K}_p\,\tilde{\mathbf{q}} + \mathbf{K}_i \int \tilde{\mathbf{q}}\ \mathrm{d}t\right) + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}})\ . \tag{4.10}$$

It can be seen that the implementation of the computed torque control law (4.10) requires the on-line computation of the mass matrix $\mathbf{M}(\mathbf{q})$, the Coriolis matrix $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})$, and the vector of gravitational forces $\mathbf{g}(\mathbf{q})$. The integral term of the control law is used to eliminate steady-state errors.

4.1.2 Operational Space Computed Torque Control

If the desired trajectory for the end-effector is specified in the operational space, which is done with the trajectory planning from Chapter 3, it is reasonable to use operational space control schemes. Applying the feedback linearization

$$\boldsymbol{\tau}_d = \mathbf{M}(\mathbf{q})\mathbf{v}_m + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) \tag{4.11}$$

to (4.5) without external forces $\boldsymbol{\tau}_{ext}$ and $\mathbf{n}$ from (4.7) leads to

$$\ddot{\mathbf{q}} = \mathbf{v}_m \ . \tag{4.12}$$

Now, consider the velocity of the end-effector $\dot{\mathbf{x}}_e$ with respect to the base frame given by

$$\dot{\mathbf{x}}_e = \begin{bmatrix} \dot{\mathbf{p}}_0^e \\ \boldsymbol{\omega}_0^e \end{bmatrix} = \mathbf{J}_0^e(\mathbf{q})\,\dot{\mathbf{q}} \ . \tag{4.13}$$

With the time derivative of (4.13)

$$\ddot{\mathbf{x}}_e = \mathbf{J}_0^e(\mathbf{q})\,\ddot{\mathbf{q}} + \dot{\mathbf{J}}_0^e(\mathbf{q})\,\dot{\mathbf{q}} \ , \tag{4.14}$$

one can obtain

$$\ddot{\mathbf{q}} = (\mathbf{J}_0^e(\mathbf{q}))^\dagger \left(\ddot{\mathbf{x}}_e - \dot{\mathbf{J}}_0^e(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} \right) \ . \tag{4.15}$$

Here, $(\mathbf{J}_0^e(\mathbf{q}))^\dagger$ is the right pseudo-inverse of the geometric Jacobian matrix $\mathbf{J}_0^e$, which is computed as

$$(\mathbf{J}_0^e(\mathbf{q}))^\dagger = (\mathbf{J}_0^e(\mathbf{q}))^{\mathrm{T}} \left(\mathbf{J}_0^e(\mathbf{q}) (\mathbf{J}_0^e(\mathbf{q}))^{\mathrm{T}} \right)^{-1} \ . \tag{4.16}$$

Comparing (4.12) with (4.15), the control input $\mathbf{v}_m$ is derived as

$$\mathbf{v}_m = (\mathbf{J}_0^e(\mathbf{q}))^\dagger \left(\mathbf{v}_c - \dot{\mathbf{J}}_0^e(\mathbf{q}, \dot{\mathbf{q}})\,\dot{\mathbf{q}} \right) \ , \tag{4.17}$$

which yields

$$\ddot{\mathbf{x}}_e = \mathbf{v}_c \ . \tag{4.18}$$

In order to enable the tracking of the desired trajectory, the new control input $\mathbf{v}_c$ is chosen as

$$\mathbf{v}_c = \begin{bmatrix} \mathbf{v}_{c,p} \\ \mathbf{v}_{c,o} \end{bmatrix} = \begin{bmatrix} \ddot{\mathbf{p}}_d + \mathbf{K}_D\,\dot{\tilde{\mathbf{p}}} + \mathbf{K}_P\,\tilde{\mathbf{p}} + \mathbf{K}_I \int \tilde{\mathbf{p}}\ \mathrm{d}t \\ \dot{\boldsymbol{\omega}}_d + \mathbf{K}_\omega\,(\boldsymbol{\omega}_d - \boldsymbol{\omega}_0^e) + \mathbf{K}_o\,\mathbf{e}_o \end{bmatrix} \ , \tag{4.19}$$

where the position and the orientation of the end-effector are controlled separately, see [10]. Herein, $\tilde{\mathbf{p}} = \mathbf{p}_d - \mathbf{p}_0^e$ denotes the error between the desired end-effector position $\mathbf{p}_d$ expressed in the robot base frame and its current position $\mathbf{p}_0^e$. The orientation error $\mathbf{e}_o$ is the quaternion error computed from (4.4). Furthermore, $\boldsymbol{\omega}_d$ and $\boldsymbol{\omega}_0^e$ are the desired and current angular velocity of the end-effector with respect to the base frame, respectively. With the positive definite matrices $\mathbf{K}_D$, $\mathbf{K}_P$, $\mathbf{K}_I$, $\mathbf{K}_\omega$ and $\mathbf{K}_o$, (4.19) leads to an asymptotically stable closed-loop system, see [10, 26].

For applying this control law, operational space variables of the end-effector, e. g. $\mathbf{p}_0^e$, $\dot{\mathbf{p}}_0^e$, $\boldsymbol{\omega}_0^e$, ... are required. They can be obtained by the evaluation of the direct kinematics (2.3) and differential kinematics (2.25) using the measured joint angles $\mathbf{q}$ and their time derivatives $\dot{\mathbf{q}}$. Another approach is to directly measure the end-effector pose by means of

e. g. the OPTITRACK system and performing a numerical differentiation and filtering to compute the linear and angular velocity of the end-effector.

For the implementation of the control input (4.17), the computation of the geometric Jacobian matrix and its inverse is needed. Since singularities and redundancies of the robot influence the Jacobian matrix, operational space controllers are generally more complex than the joint space controllers, see [10]. For an operational space control law, the self-motion of the robot has to be handled inside the feedback loop, which is explained in Section 4.3.

4.2 Force Control

Motion control is mostly used if the end-effector is following a trajectory in the free space. However, when a contact is made between the end-effector and the object, pure motion control is not always sufficient. Pure motion control would be suitable if both the robot model and environment including the object model are known accurately, which is not guaranteed precisely in practice. Better control performance is achieved by not only controlling the position and orientation of the end-effector, but also regulating the contact force additionally. Furthermore, to achieve different line thicknesses while drawing on the 3D object, the contact force has to be controlled. To ensure that the contact force occurs normal to the object surface, the orientation of the end-effector is continually adjusted by the motion controller.

The force control methods can be classified into indirect and direct force control, see, e. g., [27]. In the former method, the interaction force is indirectly regulated by controlling the end-effector pose. In this work, a direct force control is utilized, which enables the regulation of the contact force to a desired value by explicitly closing a force feedback loop. Hence, the direct force control has the advantage that the contact force can be controlled separately from the motion control. In the following, the vector $\mathbf{h} \in \mathbb{R}^6$ denotes the vector of forces $\mathbf{f} \in \mathbb{R}^3$ and moments $\boldsymbol{\mu} \in \mathbb{R}^3$, i.e. $\mathbf{h}^{\mathrm{T}} = [\mathbf{f}^{\mathrm{T}} \ \boldsymbol{\mu}^{\mathrm{T}}]$.

Let $\dot{\mathbf{p}}_f$ be the velocity of the contact point in the normal direction to the surface. Then, the control of the contact force $\mathbf{f}_c$ to a desired value $\mathbf{f}_d$ is performed by using the PID control law

$$\mathbf{v}_F = \mathbf{K}_{Pf}(\mathbf{f}_d - \mathbf{f}_c) + \mathbf{K}_{If} \int (\mathbf{f}_d - \mathbf{f}_c)\, \mathrm{d}t - \mathbf{K}_{Df}\dot{\mathbf{p}}_f \ , \tag{4.20}$$

with the positive definite gain matrices $\mathbf{K}_{Pf}$, $\mathbf{K}_{If}$ and $\mathbf{K}_{Df}$. The term $-\mathbf{K}_{Df}\dot{\mathbf{p}}_f$ damps the velocity in the normal direction, see [28, 29]. Adding the integral term in the control law results in more robustness to constant disturbance forces. It should be mentioned that the values of the gain matrices are chosen depending on the compliance of the environment. The control input $\mathbf{v}_F$ will be used in Section 4.2.2 for the design of a hybrid force/motion controller.

4.2.1 Contact Force Estimation

During the drawing process, the contact force between the pen and the object surface must be determined in real time. Since in this work no force/torque sensor is mounted on

the end-effector, the contact force must be estimated. In this section, based on [30], a model-based estimation of the contact force is given.

Assuming that the contact force and moment $\mathbf{h}_c$ is applied to the end-effector at the point $\mathbf{p}_0^e$, the resulting external torques are given by

$$\boldsymbol{\tau}_{ext} = \left(\mathbf{J}_0^e(\mathbf{q})\right)^{\mathrm{T}} \mathbf{h}_c \ . \tag{4.21}$$

Furthermore, the generalized momentum of the robot $\mathbf{i}$ is computed as

$$\mathbf{i} = \mathbf{M}(\mathbf{q})\,\dot{\mathbf{q}} \ . \tag{4.22}$$

From the dynamic model of the robot, the residual vector $\mathbf{r}$ is expressed as

$$\mathbf{r}(t) = \mathbf{K}_c \left(\mathbf{i} - \int_0^t \left(\boldsymbol{\tau} + \mathbf{C}^{\mathrm{T}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) + \mathbf{r} \right) \mathrm{d}s \right) \ , \tag{4.23}$$

where $\mathbf{K}_c$ is a diagonal positive definite gain matrix, see [30]. Since the matrix $\dot{\mathbf{M}}(\mathbf{q}) - 2\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is skew-symmetric, see, e. g., [31], $\dot{\mathbf{M}}(\mathbf{q})$ reads as

$$\dot{\mathbf{M}}(\mathbf{q}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{C}^{\mathrm{T}}(\mathbf{q}, \dot{\mathbf{q}}) \ . \tag{4.24}$$

Therefore, the stable dynamic evolution of the residual $\mathbf{r}$ is given by

$$\dot{\mathbf{r}} = \mathbf{K}_c \left(\boldsymbol{\tau}_{ext} - \mathbf{r} \right) \ , \tag{4.25}$$

which for a sufficiently large gain $\mathbf{K}_c$ leads to

$$\mathbf{r} \approx \boldsymbol{\tau}_{ext} = \left(\mathbf{J}_0^e(\mathbf{q})\right)^{\mathrm{T}} \mathbf{h}_c \ . \tag{4.26}$$

Notice that measurement noise and model uncertainties could cause accuracy problems. For that reason, it is reasonable to define a suitable threshold r_t for $\mathbf{r}$. Contact detection can be achieved if the condition $\|\mathbf{r}\| > r_t$ holds.

The assumption that during the drawing with the pen only point-wise forces are applied, yields for the vector of moments $\hat{\boldsymbol{\mu}}_c = \mathbf{0}$. Therefore, the vector of estimated contact force and moment $\hat{\mathbf{h}}_c$ expressed in the robot base frame is obtained as

$$\hat{\mathbf{h}}_c = \begin{bmatrix} \hat{\mathbf{f}}_c \\ \mathbf{0} \end{bmatrix} \ , \tag{4.27}$$

with

$$\hat{\mathbf{f}}_c = \left(\left(\mathbf{J}_{v,0}^e(\mathbf{q})\right)^{\mathrm{T}} \right)^{\dagger} \mathbf{r} \ . \tag{4.28}$$

Here, $\mathbf{J}_{v,0}^e(\mathbf{q})$ is the translational geometric Jacobian matrix of the end-effector. Note that the estimated contact force $\hat{\mathbf{f}}_c$ only detects components that are not in the null space of $\mathbf{J}_{v,0}^e(\mathbf{q})$, see [28].

4.2.2 Hybrid Force/Motion Control

In this section using the approach in [28], a hybrid force/motion control is presented. During the drawing task, the contact force along the normal of the 3D object surface and simultaneously the pose of the end-effector along the trajectory of the pattern must be controlled. The basic idea behind the hybrid control is to achieve motion and force control to desired values at the same time while respecting the constraints due to environment geometry.

The constraints for motion control are along the normals of the object surface. On the other hand, with the assumption that no friction is present during contact, the force control constraints appear along the tangent of the surface, see [10]. Since the object surface on which the robot draws is in general non-planar, the constraint directions are time-varying during the task execution.

The formulation of the control task must be performed consistently with the constraints. Therefore, it is necessary to determine the active motion and force constraints at any time, which can be difficult in practical situations. One approach is to select small controller gains for both motion and force control, resulting in a softer control action, see [29]. However, this causes performance degradation due to a lower accuracy during the drawing task. A more efficient approach is to use the real-time contact force estimation described in Section 4.2.1 to identify the constraint directions. In order to do so, the description of the contact task frame is necessary. The contact frame, which is located at the contact point, is defined using a rotation matrix $\mathbf{R}_0^c$. The z-axis of the contact frame is aligned with the estimated contact force $\hat{\mathbf{f}}_c$, i. e.

$$\hat{\mathbf{f}}_c^c = (\mathbf{R}_0^c)^{\mathrm{T}}\, \hat{\mathbf{f}}_c = \begin{bmatrix} 0 \\ 0 \\ \left\| \hat{\mathbf{f}}_c \right\| \end{bmatrix}, \tag{4.29}$$

where $\hat{\mathbf{f}}_c^c$ denotes the contact force expressed with respect to the contact frame. Since it is assumed that during the drawing task the contact force is applied at the point $\mathbf{p}_0^e$ and in the z-axis of the end-effector frame, in this work, for the rotation matrix of the contact frame $\mathbf{R}_0^c = \mathbf{R}_0^e$ is used. For a detailed explanation of how to determine $\mathbf{R}_0^c$ in a general case, refer to [28].

Due to the fact that in orthogonal directions the motion and force control loop are decoupled, their controller gains can be chosen independently, see [27]. The control laws (4.19) and (4.20) must then be projected on the corresponding motion and force controlled subspaces. For this purpose, first consider the so-called selection matrices $\mathbf{Y}_f^c$ and $\mathbf{Y}_v^c$, which are used for the task specification of the contact force and motion, respectively. During the pattern drawing task the contact force along the z-axis must be regulated while the other DOF of the operational space are subject to the motion control. Hence,

the selection matrices are given by

$$\mathbf{Y}_f^c = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{Y}_v^c = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.30}$$

Obviously, the DOF of force and motion controlled subspaces are complementary, which is seen with

$$(\mathbf{Y}_f^c)^{\mathrm{T}}\, \mathbf{Y}_v^c = \mathbf{0}\ . \tag{4.31}$$

Using the velocity transformation matrix

$$\mathbf{T}_{0,v}^c = \begin{bmatrix} \mathbf{R}_0^c & \mathbf{S}\left(\mathbf{p}_0^e\right)\mathbf{R}_0^c \\ \mathbf{0} & \mathbf{R}_0^c \end{bmatrix}, \tag{4.32}$$

the end-effector velocity is transformed from the contact frame to the base frame [13], i.e.

$$\dot{\mathbf{x}}_e = \mathbf{T}_{0,v}^c \dot{\mathbf{x}}_c^e\ . \tag{4.33}$$

Since the contact frame is time-varying, the time derivative of (4.33) is given by

$$\ddot{\mathbf{x}}_e = \mathbf{T}_{0,v}^c \ddot{\mathbf{x}}_c^e + \dot{\mathbf{T}}_{0,v}^c \dot{\mathbf{x}}_c^e\ , \tag{4.34}$$

where $\dot{\mathbf{T}}_{0,v}^c$ denotes the time derivative of the transformation matrix $\mathbf{T}_{0,v}^c$. By substituting (4.34) into (4.15), the joint accelerations are written as

$$\ddot{\mathbf{q}} = (\mathbf{J}_{0,M}^e(\mathbf{q}))^{\dagger}\left(\mathbf{T}_{0,v}^c \ddot{\mathbf{x}}_c^e + \dot{\mathbf{T}}_{0,v}^c \dot{\mathbf{x}}_c^e - \dot{\mathbf{J}}_0^e(\mathbf{q}, \dot{\mathbf{q}})\, \dot{\mathbf{q}}\right)\ . \tag{4.35}$$

Here, $(\mathbf{J}_{0,M}^e(\mathbf{q}))^{\dagger}$ denotes the inertia weighted right pseudo-inverse of the geometric Jacobian $\mathbf{J}_0^e(\mathbf{q})$ and is given by

$$\left(\mathbf{J}_{0,M}^e(\mathbf{q})\right)^{\dagger} = (\mathbf{M}(\mathbf{q}))^{-1}(\mathbf{J}_0^e(\mathbf{q}))^{\mathrm{T}}\left(\mathbf{J}_0^e(\mathbf{q})(\mathbf{M}(\mathbf{q}))^{-1}(\mathbf{J}_0^e(\mathbf{q}))^{\mathrm{T}}\right)^{-1}\ . \tag{4.36}$$

For the hybrid control law, all the control variables have to be expressed with respect to the motion and the force controlled subspaces, respectively. In order to do so, first consider the matrices $\mathbf{Y}_f$ and $\mathbf{Y}_v$ defined as

$$\mathbf{Y}_f = (\mathbf{T}_{0,v}^c)^{\mathrm{T}}\mathbf{Y}_f^c \qquad \text{and} \qquad \mathbf{Y}_v = \mathbf{T}_{0,v}^c \mathbf{Y}_v^c\ , \tag{4.37}$$

which enable the transformation from the force and motion subspaces to the robot base frame, see [13]. Since the constraints are expressed with respect to the contact frame, the

matrices $\mathbf{Y}_f$ and $\mathbf{Y}_v$ are time-varying. The vector of contact force and moment expressed in the force controlled subspace $\mathbf{h}_{c,f}^c$ is obtained as

$$\mathbf{h}_{c,f}^c = \mathbf{Y}_f{}^\dagger \, \mathbf{h}_c \ , \tag{4.38}$$

see [10]. Note that for the drawing task, the contact force is along the z-axis of the contact frame, and therefore $\mathbf{h}_{c,f}^c$ is a scalar. The velocity of the end-effector in the motion controlled subspace $\dot{\mathbf{x}}_{c,v}^e$ is given by

$$\dot{\mathbf{x}}_{c,v}^e = \mathbf{Y}_v{}^\dagger \, \dot{\mathbf{x}}_e \ . \tag{4.39}$$

The matrices $\mathbf{Y}_f{}^\dagger$ and $\mathbf{Y}_v{}^\dagger$ denote the left pseudo-inverses of $\mathbf{Y}_f$ and $\mathbf{Y}_v$. Similarly, the desired force and velocity values are transformed to the corresponding subspaces. The transformed controlled variables $\dot{\mathbf{x}}_{c,v}^e$ and $\mathbf{h}_{c,f}^c$ are used in the control laws (4.19) and (4.20), respectively.

The current end-effector position $\mathbf{p}_0^e$ and its orientation denoted by the rotation matrix $\mathbf{R}_0^e$ are first transformed from the robot base frame to the contact frame, by using the homogeneous transformation

$$\mathbf{T}_c^0 = \begin{bmatrix} (\mathbf{R}_0^c)^{\mathrm{T}} & -(\mathbf{R}_0^c)^{\mathrm{T}} \mathbf{p}_0^e \\ \mathbf{0} & 1 \end{bmatrix} . \tag{4.40}$$

Then the end-effector pose is expressed in the velocity controlled subspace by multiplying with the selection matrix $(\mathbf{Y}_v^c)^{\mathrm{T}}$ from (4.30).

Now the control inputs $\mathbf{v}_c$ from (4.19) and $\mathbf{v}_F$ from (4.20) are combined to design the hybrid controller. Applying exact linearization to the robot dynamics (4.5), the control law

$$\boldsymbol{\tau}_d = \mathbf{M}(\mathbf{q})\, \mathbf{a} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) - (\mathbf{J}_0^e(\mathbf{q}))^{\mathrm{T}} \, \hat{\mathbf{h}}_c \ , \tag{4.41}$$

with the new control input $\mathbf{a}$ is derived. Note that the term $-(\mathbf{J}_0^e(\mathbf{q}))^{\mathrm{T}} \, \hat{\mathbf{h}}_c$ in the control law exactly compensates the estimated contact forces, resulting in infinite stiffness of the robot with respect to the external forces, see [10]. Introducing the natural inertia of the robot at the contact point as

$$\mathbf{M}_d = \left(\mathbf{J}_0^e(\mathbf{q}) \mathbf{M}^{-1} (\mathbf{J}_0^e(\mathbf{q}))^{\mathrm{T}} \right)^{-1} \ , \tag{4.42}$$

using the control laws $\mathbf{v}_c$ from (4.19) and $\mathbf{v}_F$ from (4.20) with the control variables expressed in the corresponding subspaces and considering (4.35), the new input variable $\mathbf{a}$ is chosen as

$$\mathbf{a} = (\mathbf{J}_{0,M}^e(\mathbf{q}))^\dagger \, \mathbf{M}_d^{-1} \left(\underbrace{\mathbf{Y}_f \mathbf{v}_F + \mathbf{Y}_v \mathbf{v}_c}_{\mathbf{a}_c} + \mathbf{M}_d \left(\dot{\mathbf{T}}_{0,v}^c \, \dot{\mathbf{x}}_c^e - \dot{\mathbf{J}}_0^e(\mathbf{q})\, \dot{\mathbf{q}} \right) \right) \ , \tag{4.43}$$

see [28]. Assuming that during the drawing process the contact frame changes only very slowly, the term $\mathbf{M}_d \dot{\mathbf{T}}^c_{0,v} \dot{\mathbf{x}}^e_c$ in (4.43) is neglected. The final hybrid control law becomes

$$\boldsymbol{\tau}_d = \mathbf{M}(\mathbf{q}) \left(\mathbf{J}^e_{0,M}(\mathbf{q})\right)^{\dagger} \mathbf{M}_d^{-1} \Big(\mathbf{Y}_f \mathbf{v}_F + \mathbf{Y}_v \mathbf{v}_c - \mathbf{M}_d \dot{\mathbf{J}}^e_0(\mathbf{q}) \dot{\mathbf{q}} \Big) + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) - \left(\mathbf{J}^e_0(\mathbf{q})\right)^{\mathrm{T}} \hat{\mathbf{h}}_c \ . \tag{4.44}$$

The block diagram representing the hybrid force/motion control is shown in Figure 4.1.

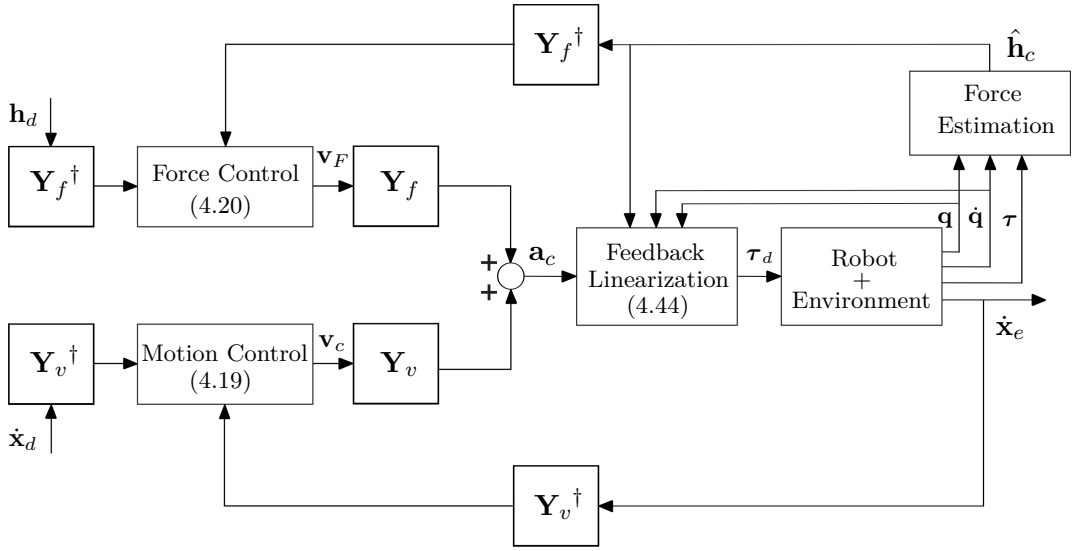


Figure 4.1: Block diagram of hybrid force/motion control.

4.3 Null-Space Control

As mentioned in Chapter 2, the KUKA LWR iiwa has 7 DOF and the operational space consists of 6 DOF. Therefore, the control of the remaining redundant DOF is discussed in this section. For a redundant manipulator, the geometric Jacobian matrix $\mathbf{J}^e_0$ has more columns than rows and hence, a null space exists. Null-space motions of the manipulator do not affect the pose of the end-effector. In the literature, there are several possibilities to control the null-space dynamics. In this work, a simple projection based approach is used, see, e. g., [15, 32]. Consider the control law

$$\mathbf{v}_n = -\mathbf{K}_{dn}\dot{\mathbf{q}} - \mathbf{K}_{pn}\mathbf{q} \ . \tag{4.45}$$

The first term with $\mathbf{K}_{dn} > 0$ damps the joint velocities of the manipulator. The second term with $\mathbf{K}_{pn} > 0$ represents a virtual spring, which keeps the joint angles as close as possible to zero, which are the angles of center of the mechanical joint limits. In this way, joint limit violations are avoided. Using a suitable projection matrix $\mathbf{P}$, the vector $\mathbf{v}_n$ is projected on the null space. A convenient way to construct the projection matrix is

$$\mathbf{P} = \mathbf{I}_7 - (\mathbf{J}^e_0)^{\dagger} \mathbf{J}^e_0 \ , \tag{4.46}$$

where $\mathbf{I}_7 \in \mathbb{R}^{7\times7}$ is the identity matrix. The null-space controller is then computed as

$$\boldsymbol{\tau}_n = \mathbf{M}(\mathbf{q})\,\mathbf{P}\,\mathbf{v}_n \ . \tag{4.47}$$

This term $\boldsymbol{\tau}_n$ is added to the control torque from (4.44). Asymptotic stability of the closed-loop system is shown using Lyapunov's method, see [33].

Note that by using the above mentioned null-space controller, the kinematic singularities are not avoided. At singularities, which occur at certain joint configurations, the end-effector is unable to move in some directions of the operational space. At these configurations, the geometric Jacobian matrix $\mathbf{J}_0^e$ is rank-deficient, see [10].

5 Experiments

In this chapter, the implementation of the proposed drawing system is outlined and experimental results are discussed. First, the experimental setup with the robot KUKA LWR iiwa 14 R820 is introduced. As an example for a planar surface, the drawing method is applied to a whiteboard. To illustrate the ability of the system to draw on non-flat arbitrary surfaces, the drawing task is performed on a 3D-printed plastic rabbit with known CAD model. Finally, the drawing results of pure motion control and the hybrid force/motion control are compared.

5.1 Experimental Setup

The experimental setup, see Figure 5.1, includes the robot KUKA iiwa equipped with a pen gripper holding a pen as end-effector, OPTITRACK cameras as optical measuring system, a PC, a whiteboard and a 3D-printed rabbit.

The KUKA iiwa is a collaborative 7-DOF robot, which enables a safe and intelligent human-machine interaction in production systems. The KUKA iiwa' s high repeatability of ± 0.1mm, see [34], is optimal for performing precise assembly tasks. The 3D-printed pen gripper is spring-loaded, such that a soft contact between the pen tip and the object surface is ensured. The user can also easily replace the pen.

A PC with an Intel i7-8700K processor and 16 GB RAM is used for all computations from trajectory planning to control law evaluation for the robot. The controller designed in Chapter 4 is implemented as a MATLAB/SIMULINK model. From this model, a C++ code is generated, which is executed on the *TwinCAT* runtime from *Beckhoff Automation* [35]. The sampling time of the whole model is set to $T_s = 125\,\mu$s. *EtherCAT* is used for communication between *TwinCAT* and the KUKA robot. The measured sensor signals such as joint torques and angles are available in *TwinCAT*. The joint velocities are computed numerically with a filter using the measured joint angles.

To obtain the pose information of the 3D object in real time, six OPTITRACK cameras are used, which are connected to the PC via a PoE switch using Ethernet cables. These cameras track the retro-reflective markers attached to the object with sub-millimeter precision. In order to improve the absolute accuracy of the robot, the optical measuring system provides also the position and orientation of the end-effector. After a calibration of the cameras with the software MOTIVE, the measured pose signals are available in the *TwinCAT* runtime.

MATLAB running on a 64bit Windows 10 operating system is also used for the implementation of the pattern projection and trajectory planning. Surface parameterization of the triangular mesh using conformal mapping and robot trajectory computing require about one minute.

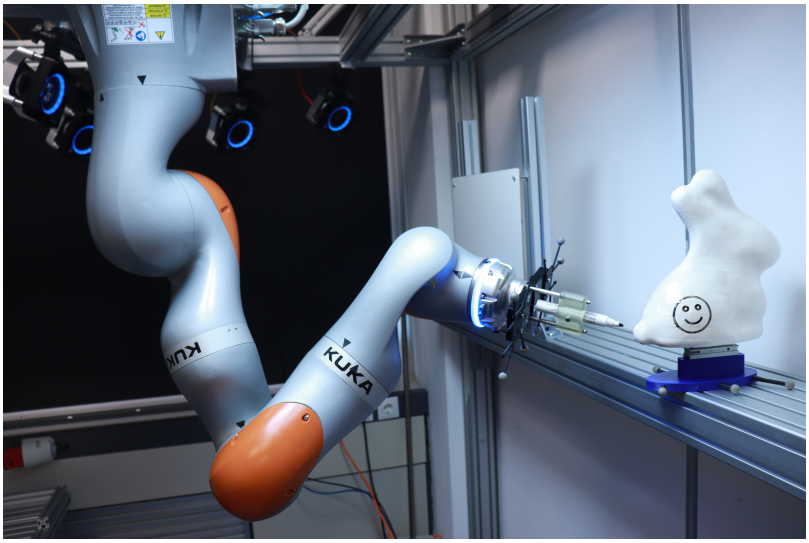


Figure 5.1: KUKA iiwa after performing a drawing on the 3D-printed rabbit.

5.2 Experimental Results

In this section, first the main steps for the trajectory planning are briefly discussed. Then the results of drawing on the whiteboard and the 3D-printed rabbit are presented and analysed.

5.2.1 Drawing Trajectory Generation

- **Projection**

  In order to achieve smooth trajectories for a smooth motion of the end-effector during the drawing task, as a pre-processing step, the input 2D pattern is interpolated by cubic splines. For pattern projection, first the object surface is decomposed into a set of charts. Then these surface charts have to be parametrized using least-squares conformal mapping described in Section 3.1. After rescaling and rotating the pattern to fit the flattened object surface, pattern points are projected onto the 3D object surface using linear mapping with barycentric coordinates.

- **Transition**

  In cases, where the drawing pattern contains multiple segments and the pen has to be lifted from the surface, intermediate points are added for transition phases. The transition phase is constructed in such a way that at the end point of one segment the pen moves away normal to the surface and approaches the starting point of the next segment orthogonal to the surface. Since for the subsequent trajectory generation, the spatial information of the pattern points is used as time parameterization, the number of intermediate points determine the duration of the transition phase. Therefore, the number of added transition points is chosen proportional to the distance between the two segment end points.

- **Trajectory Generation**

  From the projected pattern points together with the intermediate points a cubic spline is generated. For trajectory planning, the spline is then time-parameterized such that for each pattern segment an acceleration at the beginning and a deceleration at the end is achieved, as shown in Section 3.4. Note that at this step the trajectory is expressed with respect to the object coordinate frame.

- **Pose Measurement**

  The pose of the rigid body attached to the 3D object is measured with the OPTITRACK system. Using this pose information, the trajectory is transformed from the object coordinate frame to the robot base frame.

The block diagram of the processing steps is shown in Figure 5.2. To bring the pen into the drawing start pose on the object surface, the joint space computed torque controller as in Section 4.1.1 is used. For this purpose, a twice continuously differentiable trajectory of the 7-DOF robot joint angles is planned, which starts at the current robot configuration and ends at the initial pose of the drawing trajectory. Then, for performing the drawing task, the controller is switched to the operational space computed torque or hybrid force/motion controller, respectively.

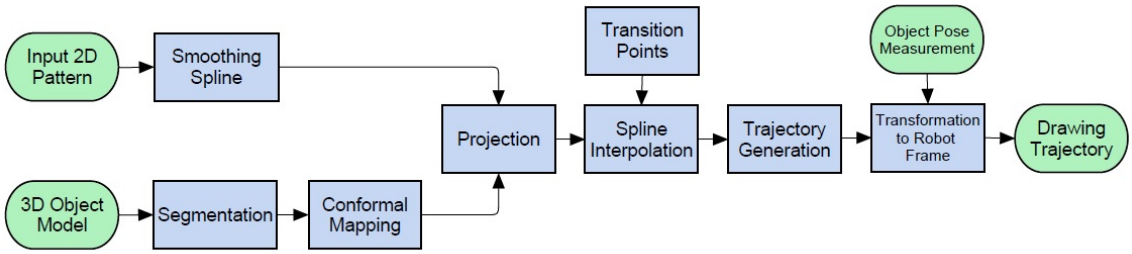


Figure 5.2: Block diagram of the process steps of drawing trajectory generation.

5.2.2 Filtering the Measured End-Effector Pose

The OPTITRACK system measures the pose of the end-effector. Since the measured pose is too noisy as control variable, a state filter is applied for noise reduction. Further, the time derivative is derived with this filter. The used filter is in the form

$$\dot{\boldsymbol{\kappa}} = \mathbf{A}\boldsymbol{\kappa} + \mathbf{b}\mathbf{u} \,, \tag{5.1}$$

with the filter gains

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ -b_0 & -b_1 \end{bmatrix} \,, \qquad \mathbf{b} = \begin{bmatrix} 0 \\ b_0 \end{bmatrix} \,. \tag{5.2}$$

Herein, for filtering the position, $\mathbf{u}$ is the measured position $(\mathbf{p}_0^e)^{\mathrm{T}}$ in Cartesian space, and the vector $\boldsymbol{\kappa}^{\mathrm{T}} = [\hat{\mathbf{p}}_0^e \quad \dot{\hat{\mathbf{p}}}_0^e]$ denotes the filtered position of the end-effector and its time derivative.

For filtering the measured orientation given as quaternion, $\mathbf{u} = [\eta_e \quad \boldsymbol{\epsilon}_e^{\mathrm{T}}]$ is used as the filter input. The filter output $\boldsymbol{\kappa}^{\mathrm{T}} = \left[\left[\hat{\eta}_e \quad \hat{\boldsymbol{\epsilon}}_e^{\mathrm{T}}\right]^{\mathrm{T}} \quad \left[\dot{\hat{\eta}}_e \quad \dot{\hat{\boldsymbol{\epsilon}}}_e^{\mathrm{T}}\right]^{\mathrm{T}}\right]$ describes the filtered quaternion and its time derivative. The corresponding angular velocity of the end-effector $\hat{\boldsymbol{\omega}}_0^e$ is obtained as

$$\hat{\boldsymbol{\omega}}_0^e = 2\left(\hat{\eta}_e\dot{\hat{\boldsymbol{\epsilon}}}_e - \dot{\hat{\eta}}_e\hat{\boldsymbol{\epsilon}}_e - \dot{\hat{\boldsymbol{\epsilon}}}_e \times \hat{\boldsymbol{\epsilon}}_e\right), \tag{5.3}$$

see, e. g., [10].

The values used for the filter gains are given in Appendix A. The continuous-time filter is then discretized using the zero-order hold method with the sampling time of $T_s = 125\,\mu\mathrm{s}$.

In the following, the results of multiple scenarios are given and discussed. The controller parameters used in the experiments are listed in Appendix A.

5.2.3 Drawing on the Whiteboard

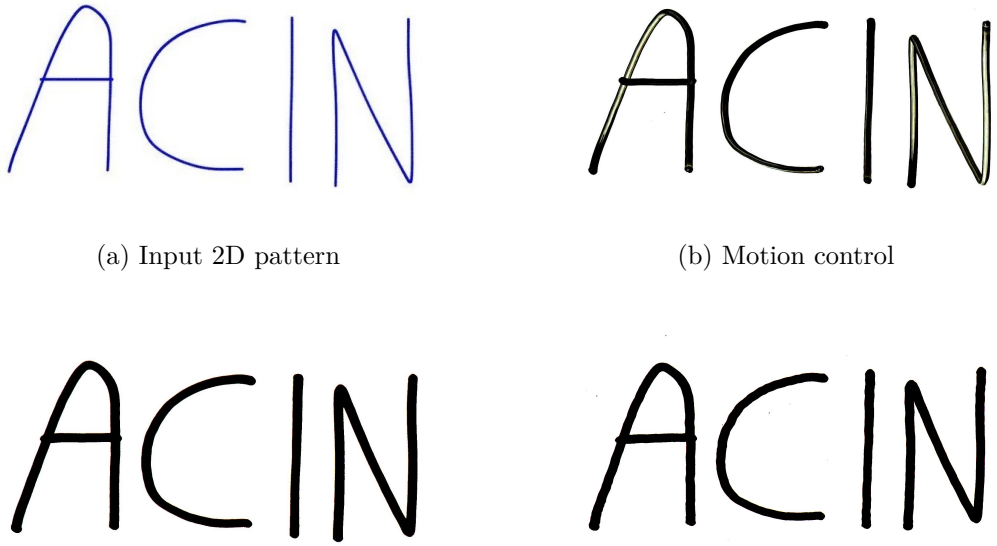
In this section, the results of drawing on the whiteboard are presented. The input 2D pattern was drawn using a computer mouse, which can be seen in Figure 5.3. The pattern is rescaled to fit the whiteboard size. Figure 5.3 shows also the drawing results of the experiments. As can be observed, the drawing with hybrid force/motion control is performed more uniformly compared to the case with pure motion control because during the drawing the contact force is additionally controlled.

Drawing on the Whiteboard with Motion Control

In this experiment, the results of drawing on the whiteboard are discussed, while no force control is applied. Therefore, the computed torque control law (4.11) is used.

The 3D drawing trajectory $\mathbf{p}_e$ including the transition phases between pattern segments is shown in Figure 5.4. As mentioned before, the transition between pattern segments are performed in a way that always a retraction from and an approach to the whiteboard surface in normal direction is executed. In Figure 5.4, the normal vectors of the desired and actual end-effector orientation are also depicted. It is observed that during the drawing the pen is always normal to the whiteboard.

Since the drawing pattern consists of multiple line segments, the force tracking is also divided into multiple segments. During transition phases between the segments, in all the experiments, the pure motion control is applied. Hence, in experiments where force control is used for drawing each segment, the controller has to be switched from pure motion control to hybrid force/motion control and vice versa. To indicate the switching time, a contact mode trajectory is designed, see Figure 5.5. A value of one for the contact mode means that a contact between pen and whiteboard exists and zero denotes that there is no contact, which occurs in transition phases. At the initial stage of each controller switching, the desired contact mode is increased from zero to one, and at the end stage decreased to zero as a ramp. At these stages, the desired force value is multiplied with



(a) Input 2D pattern

(b) Motion control

(c) Hybrid control without Optitrack

(d) Hybrid control with Optitrack

Figure 5.3: The input 2D pattern and the drawing results on the whiteboard.

the contact mode trajectory to enable a soft switching between the controllers, such that no jerky end-effector movements occur.

The estimated contact force $\hat{f}_c$ between the pen tip and the whiteboard is seen in Figure 5.5. Since in this experiment no force control is applied, the resulting contact force is not influenced directly. This leads to a fluctuation of the contact force, which is observed especially at around $t = 60\,\mathrm{s}$, and also between $t = 37\,\mathrm{s}$ and $t = 46\,\mathrm{s}$. Furthermore, the estimated contact force is noisy. This is explained by the fact that the force estimation is based on the approach discussed in Section 4.2.1, where the noisy measured joint torques $\boldsymbol{\tau}$ are used, see Figure 5.6.

The joint torque signals $\boldsymbol{\tau}$ are shown in Figure 5.6, which contain high amounts of measurement noise. Figure 5.7 shows the measured joint angles $\mathbf{q}$ during the drawing on the whiteboard. It is seen that in regions where the joint angles change faster, the corresponding joint torques are larger. Due to the fact that during the drawing process on the whiteboard the orientation of the end-effector is controlled to be constant, the joint angles q_5 and q_7 change in opposite direction to each other, to maintain a constant pen orientation.

The trajectory error is depicted in Figure 5.8. It can be observed that the position error $\tilde{\mathbf{p}}$ is in the sub-millimeter range. The orientation error $\mathbf{e}_o$ is given as the vector part of the quaternion error and therefore has no unit.

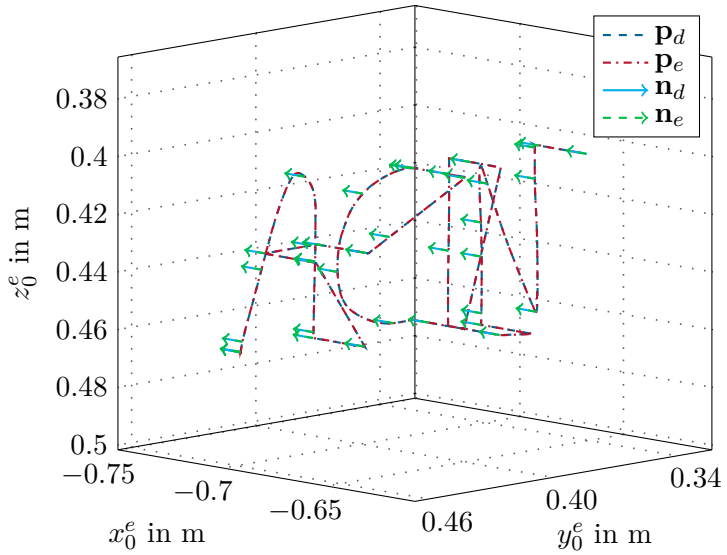


Figure 5.4: 3D trajectory of drawing pattern on the whiteboard and its desired values expressed in the base coordinate frame, including the transition phases between pattern segments.

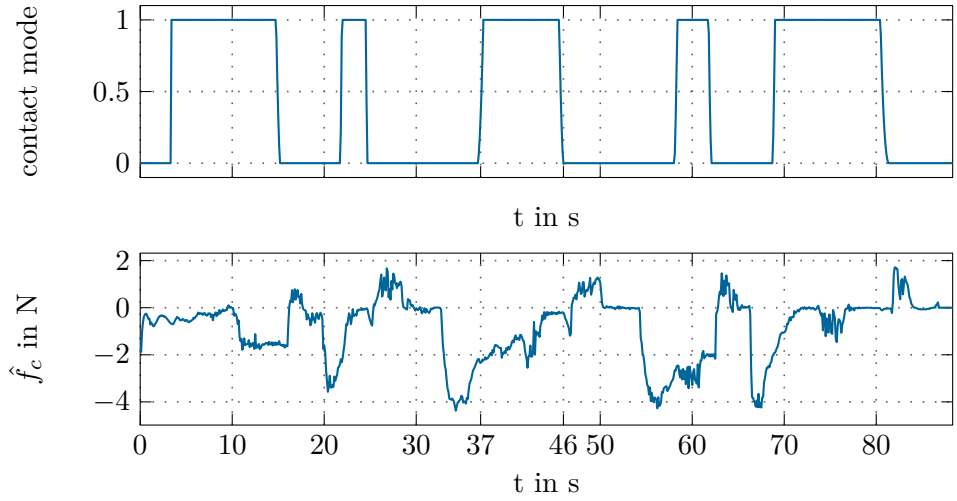


Figure 5.5: Desired trajectory of contact mode and estimated contact force $\hat{f}_c$ during drawing on the whiteboard with motion control.

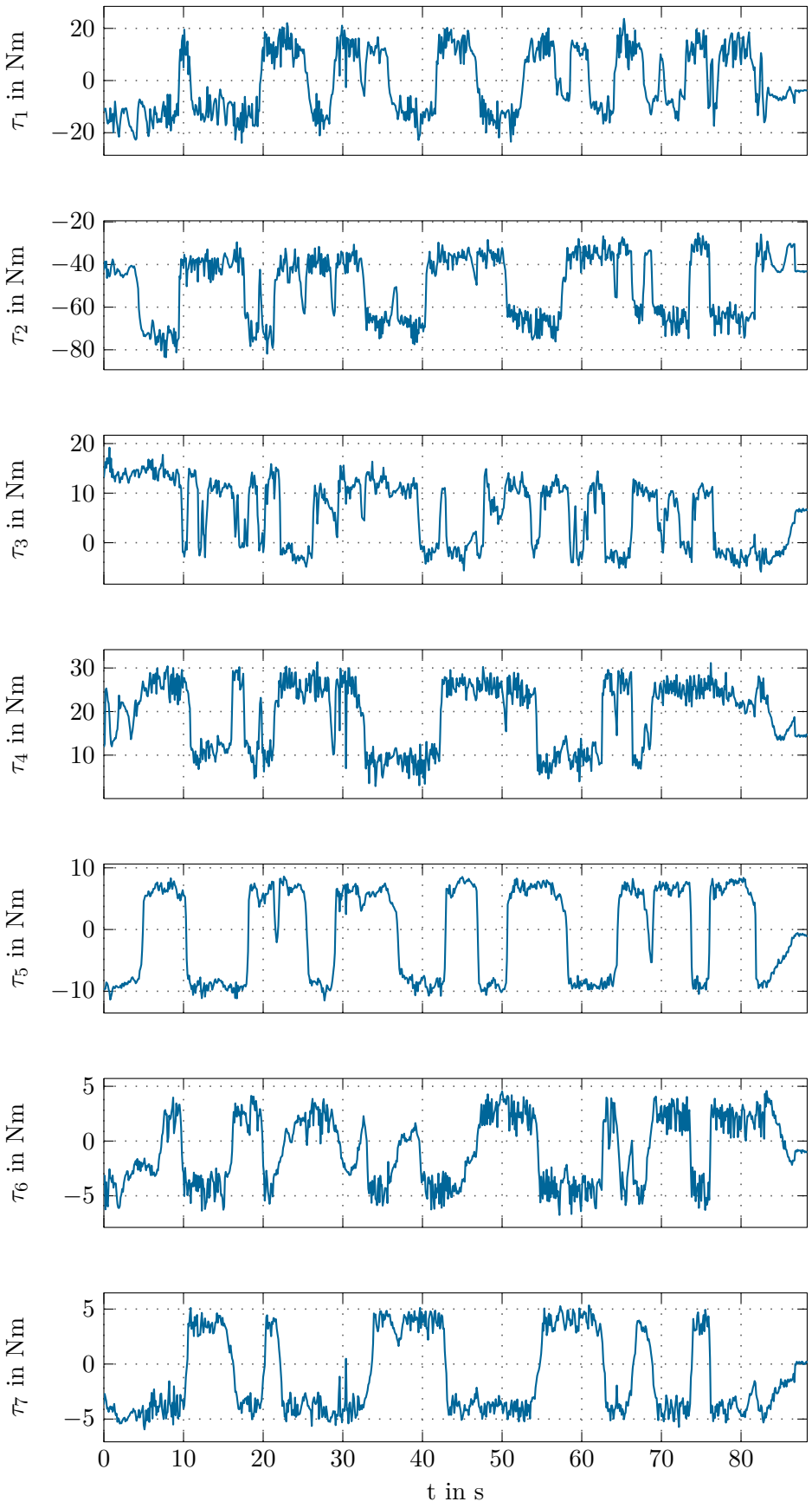


Figure 5.6: Measured joint torques $\boldsymbol{\tau}$ during drawing on the whiteboard with motion control.

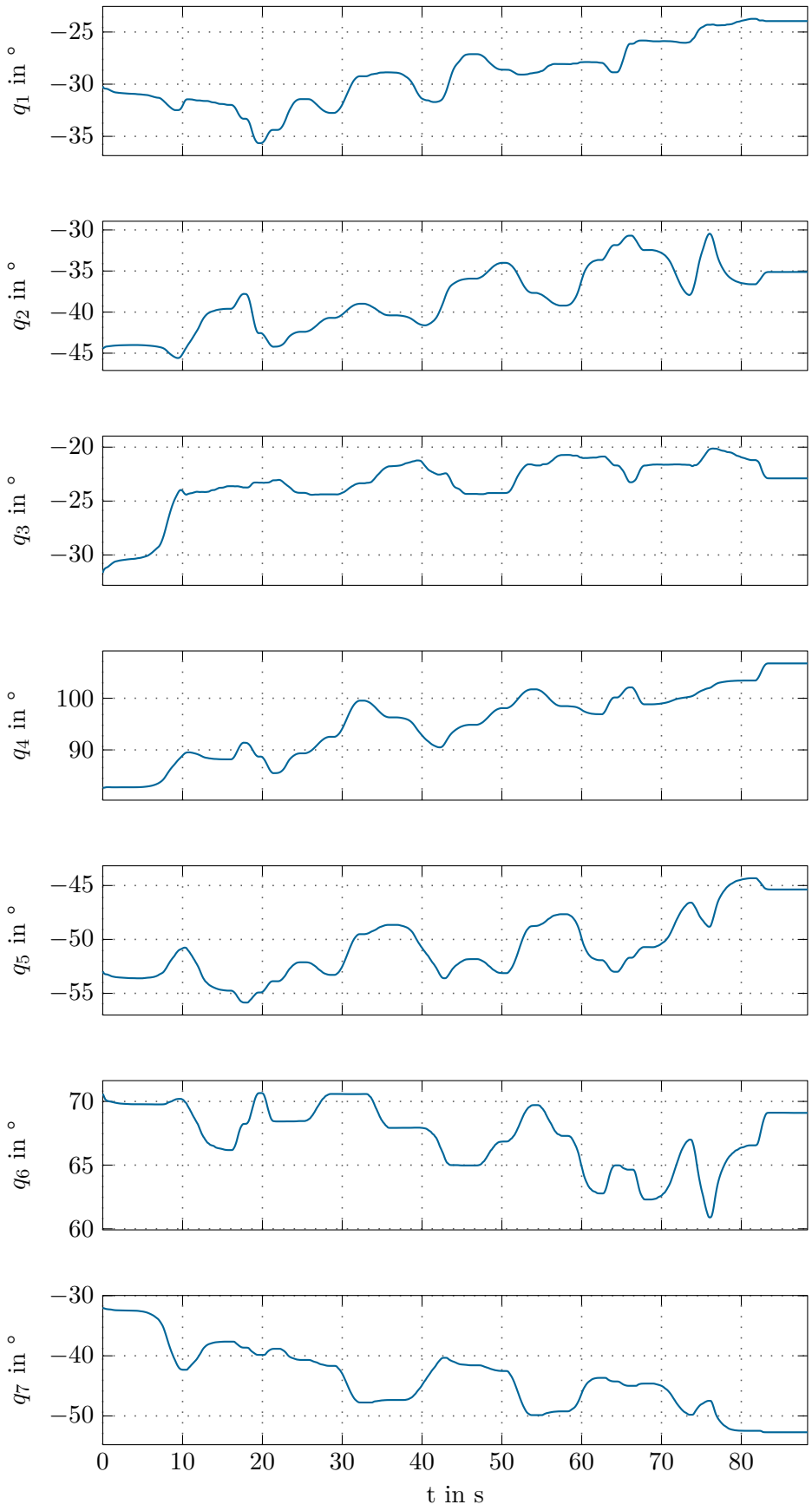


Figure 5.7: Joint angles **q** during drawing on the whiteboard with motion control.

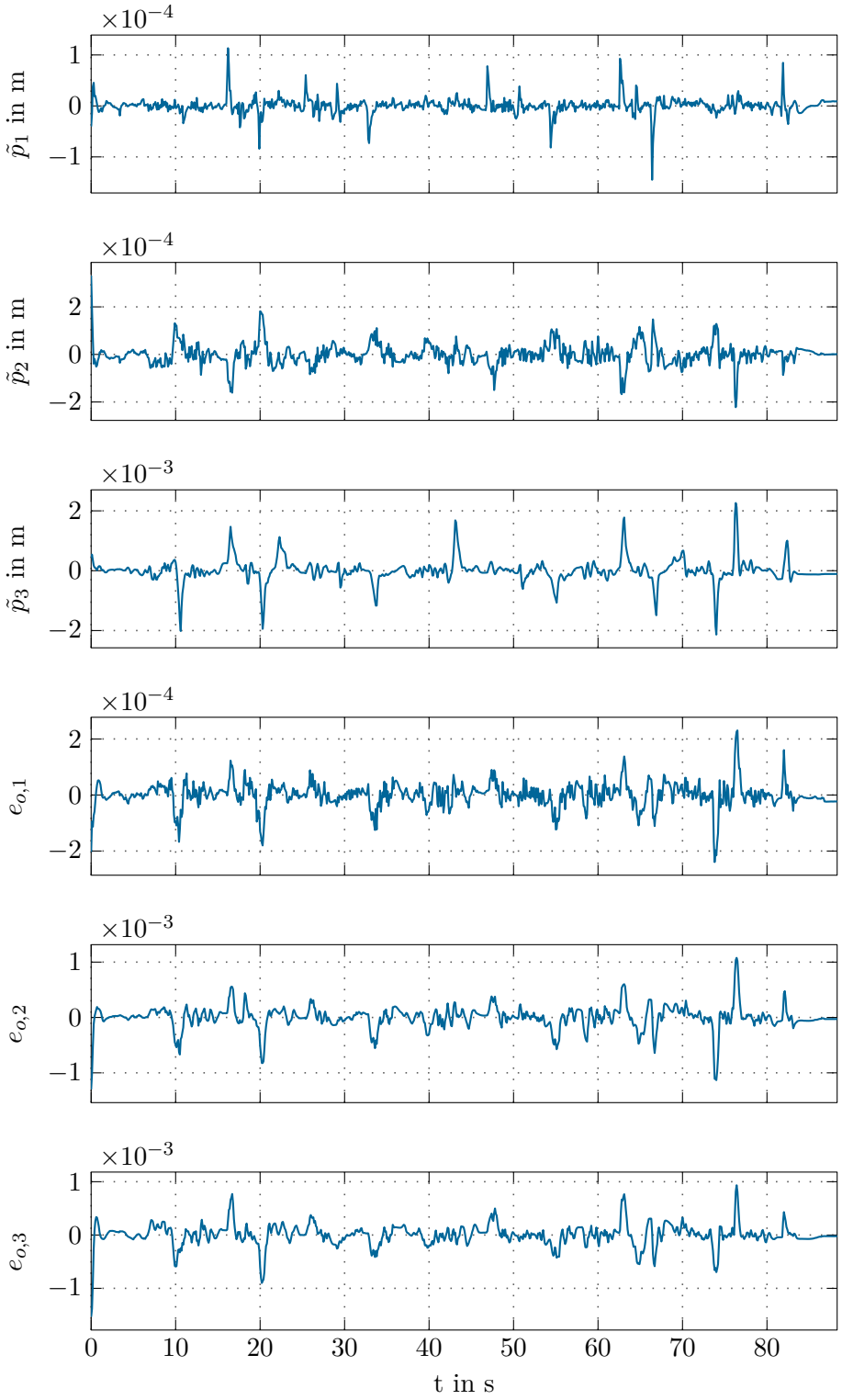


Figure 5.8: Trajectory error during drawing on the whiteboard with motion control.

Drawing on the Whiteboard Using Hybrid Force/Motion Control and Without End-Effector Tracking

In this section, the impact of the force control on the drawing results is examined. The hybrid force/motion control law (4.44) from Section 4.2.2 is used to follow the desired drawing trajectory while maintaining a constant contact force on the whiteboard plane. The contact force is in negative z-direction of the end-effector coordinate frame. Hence, the resulting force has a negative sign. In this experimental scenario, the desired contact force f_d during drawing is set to -3 N.

The estimated contact force $\hat{f}_c$ and its desired value f_d during drawing using hybrid force/motion control are illustrated in Figure 5.9. As can be observed, the contact force is controlled to the desired constant value with a small remaining error. In the transition phases, where no contact is present, the estimated external force is nearly zero, once the transient period of the force observer has decayed. Note that during the transition phases the pure motion control is used.

Since the measured joint torques $\boldsymbol{\tau}$ and joint angles $\mathbf{q}$ are similar to the case without force control, they are not depicted here. The resulting trajectory error is shown in Figure 5.10. The trajectory position error $\tilde{\mathbf{p}}$ is in the sub-millimeter range similar to Figure 5.8, where the pure motion control was used. The orientation error $\mathbf{e}_o$ is slightly larger compared to the previous experiment. This can be explained by the fact that in this experiment a larger contact force between the pen tip and the whiteboard exists, which also increases the friction forces. In the presence of friction, the force and motion control are not decoupled anymore. Consequently, the end-effector orientation cannot change as fast as before. Note that the used parameters for the position and orientation control are the same as in the experiment with the pure motion control, see Appendix A.3. By using larger gain values for the orientation control, a lower orientation error can be achieved.

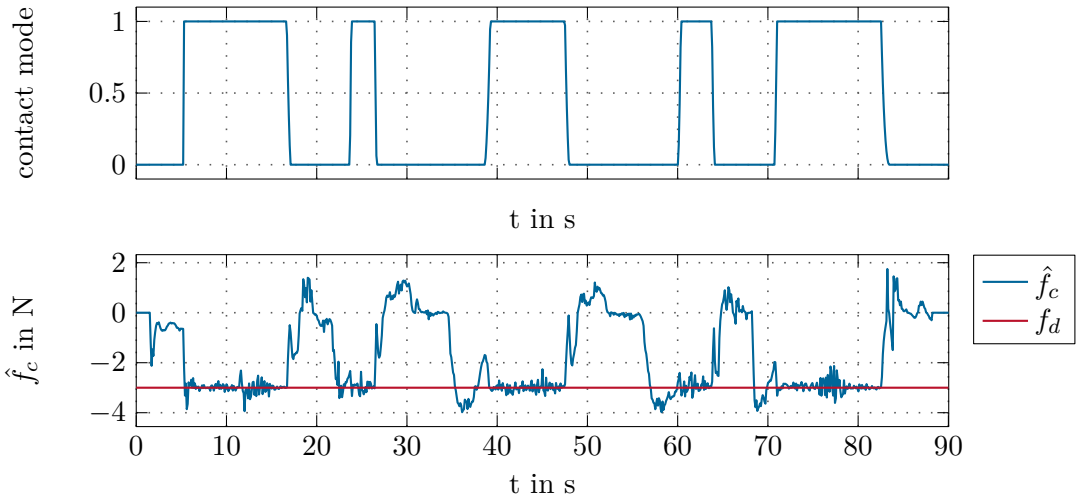


Figure 5.9: Desired trajectory of contact mode and estimated contact force $\hat{f}_c$ during drawing on the whiteboard with hybrid force/motion control.

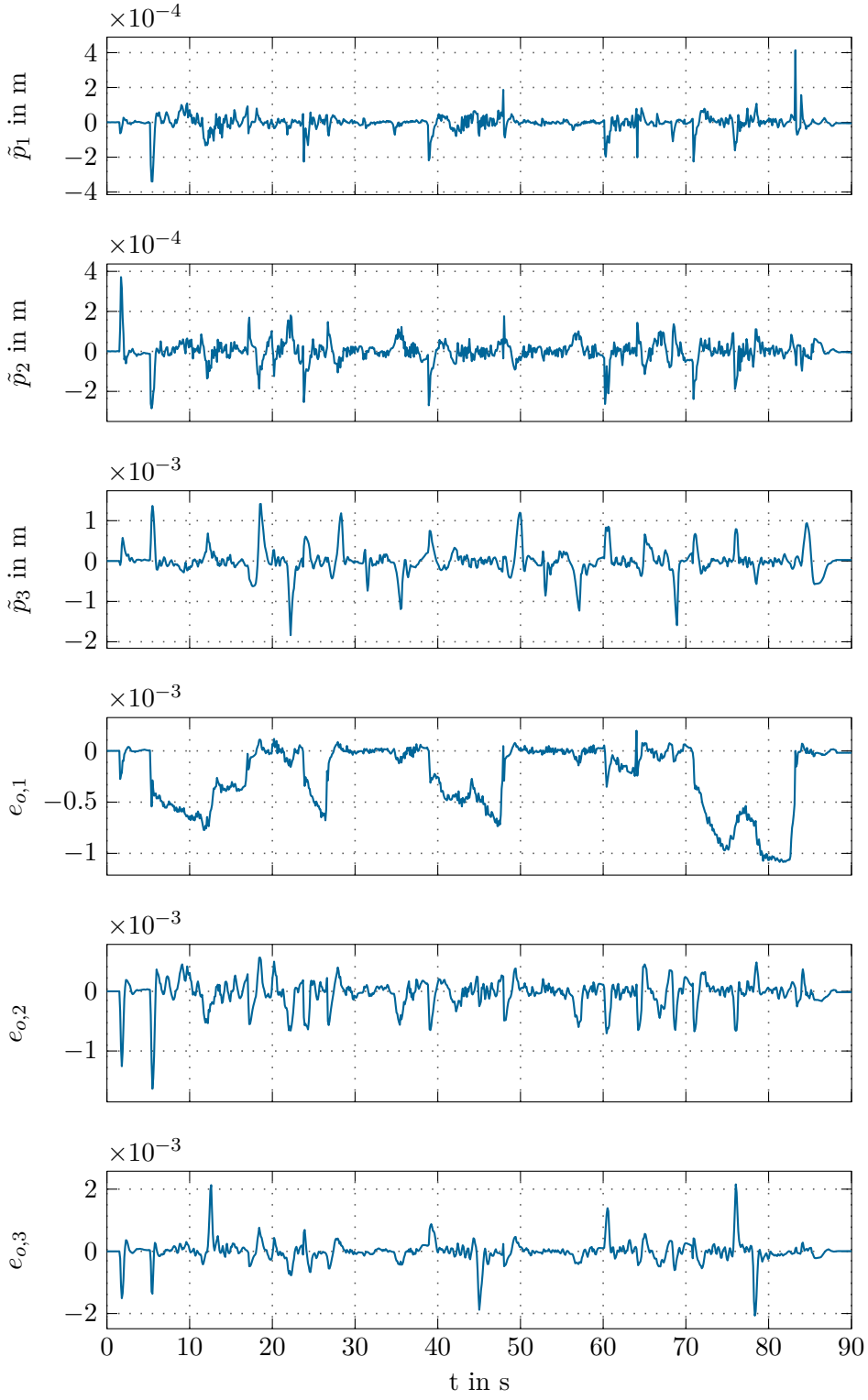


Figure 5.10: Trajectory error during drawing on the whiteboard with hybrid force/motion control.

Drawing on the Whiteboard with Hybrid Force/Motion Control and Using End-Effector Tracking

In this experiment, the hybrid control law (4.44) is used again. In addition the accuracy of the robot end-effector is evaluated and improved by tracking the end-effector with OPTITRACK cameras. The trajectory position error $\tilde{\mathbf{p}}$ is therefore the difference between the desired $\mathbf{p}_d$ and with the OPTITRACK measured position $\hat{\mathbf{p}}_0^e$ of the end-effector. Similarly, the trajectory orientation error is computed as the vector part of the quaternion error between the desired Q_d and the measured quaternions $\hat{Q}_e$. Due to the high amounts of noise in the measured signals, the linear and angular velocities, which are determined using the filter (5.1), are not used as controlled variable. Instead, like in the case without end-effector tracking, the end-effector velocities are computed from the joint velocities $\dot{\mathbf{q}}$ using the geometric Jacobian matrix. The desired contact force f_d during drawing is set to -3 N.

The end-effector absolute position error due to the limited absolute accuracy of the robot is measured with OPTITRACK and is shown in Figure 5.11. This absolute position error $\breve{\mathbf{p}}$ is defined as deviation between the measured pose of the robot end-effector by OPTITRACK and the pose of the end-effector determined using the encoder measurements with the forward kinematics. The orientation error $\breve{\mathbf{e}}_o$ is defined similarly and is expressed as the vector part of the quaternion error. The largest position error can be observed in y direction and is up to 2 mm. Note that in this scenario, the shown position error is avoided during the drawing task. Figure 5.12 shows the estimated contact force $\hat{f}_c$ and its desired value f_d.

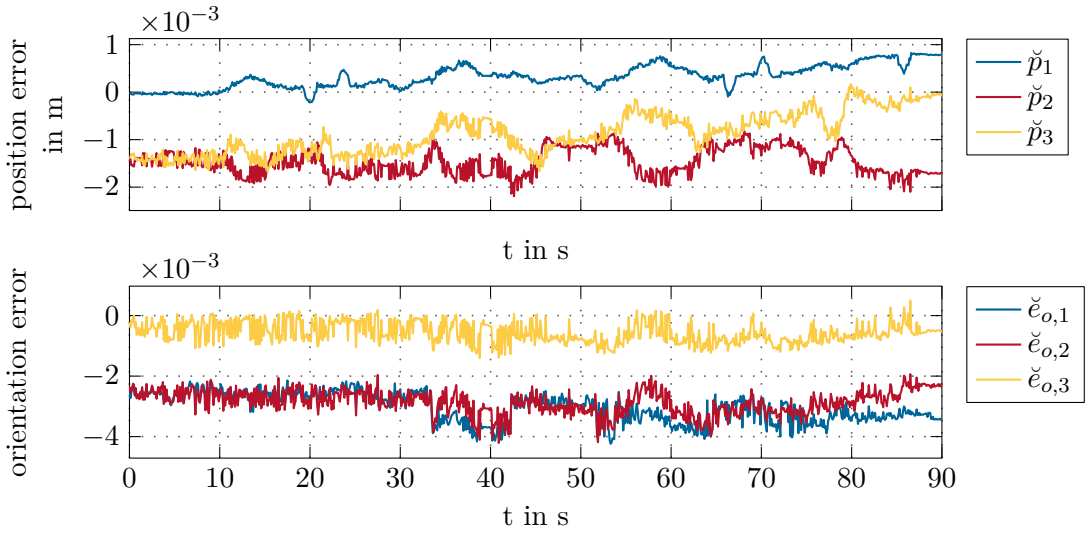


Figure 5.11: End-effector position and orientation error during drawing on the whiteboard with hybrid force/motion control and using OPTITRACK for the end-effector tracking.

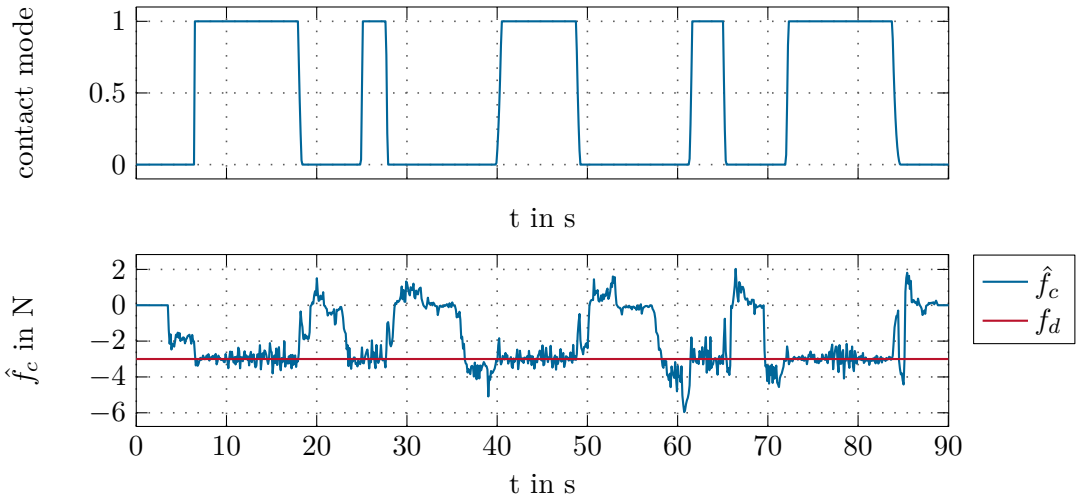


Figure 5.12: Desired trajectory of contact mode and estimated contact force $\hat{f}_c$ during drawing on the whiteboard with hybrid force/motion control and using OPTITRACK for the end-effector tracking.

5.2.4 Drawing on the Rabbit

In this section, the results of drawing on the 3D-printed rabbit are given. The input 2D pattern is a smiley, which is projected onto the rabbit's surface. The 2D pattern and the projection result are shown in Figure 5.13. Drawing results of different experiments are depicted in Figure 5.14. In the following, the results are analysed.

(a) 2D pattern (b) projection result

Figure 5.13: The input 2D pattern and its projection onto the rabbit model.

(a) Motion control

(b) Hybrid control without Optitrack

(c) Hybrid control with Optitrack

Figure 5.14: Drawing results on the 3D-printed rabbit.

Drawing on the Rabbit with Motion Control

In the first experiment, the robot draws the smiley pattern on the rabbit, while no force control is used. The actual 3D drawing path $\mathbf{p}_e$ including the transition phases between its segments is shown in Figure 5.15. The desired drawing path $\mathbf{p}_d$ is also depicted. It is observed that the actual positions correspond to the desired ones. The vectors $\mathbf{n}_e$ and $\mathbf{n}_d$ denote the actual and desired normal vectors of the end-effector orientation.

Figure 5.16 illustrates the estimated contact force $\hat{f}_c$. In this case, the contact force is not constant during the drawing because no force control was used. Therefore, quite large forces up to 8 N can be seen, which are undesirable. In industrial applications, excessive contact forces can damage the working object and the tool at the end-effector.

Figure 5.17 shows the measured joint torques $\boldsymbol{\tau}$ during the drawing. The joint angles $\mathbf{q}$ are seen in Figure 5.18. The most significant angle changes occur between $t = 10\,\mathrm{s}$ and $t = 30\,\mathrm{s}$. In this time interval, the big circle of the smiley pattern is drawn. The resulting trajectory error during the drawing is depicted in Figure 5.19. The position error $\tilde{\mathbf{p}}$ is in the sub-millimeter range.

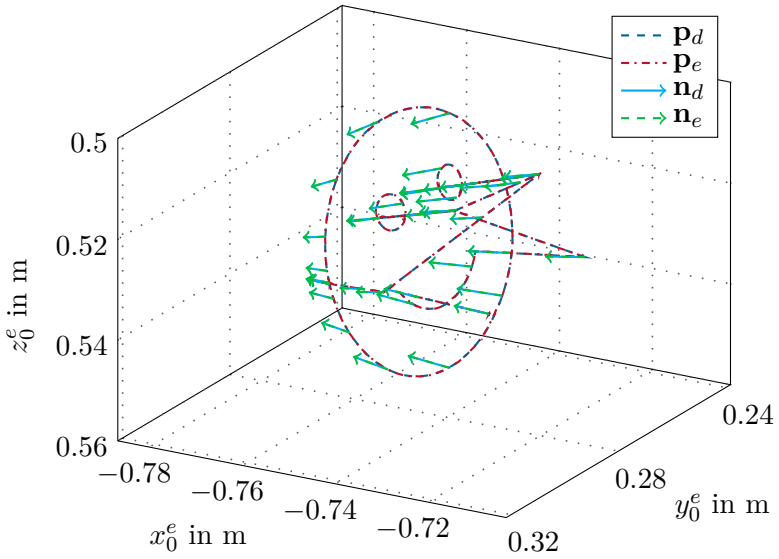


Figure 5.15: 3D trajectory of drawing pattern on the rabbit and its desired values expressed in the base coordinate frame, including the transition phases.

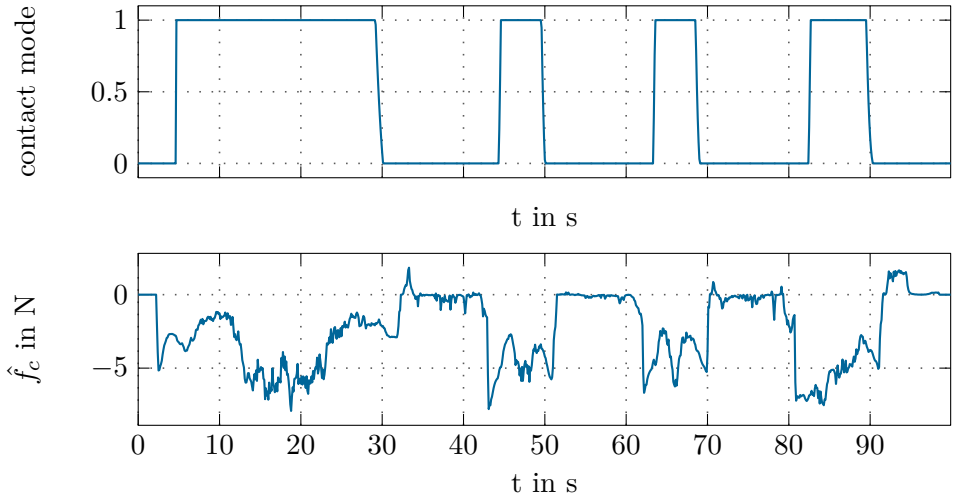


Figure 5.16: Desired trajectory of contact mode and estimated contact force $\hat{f}_c$ during the drawing on the 3D-printed rabbit with motion control.

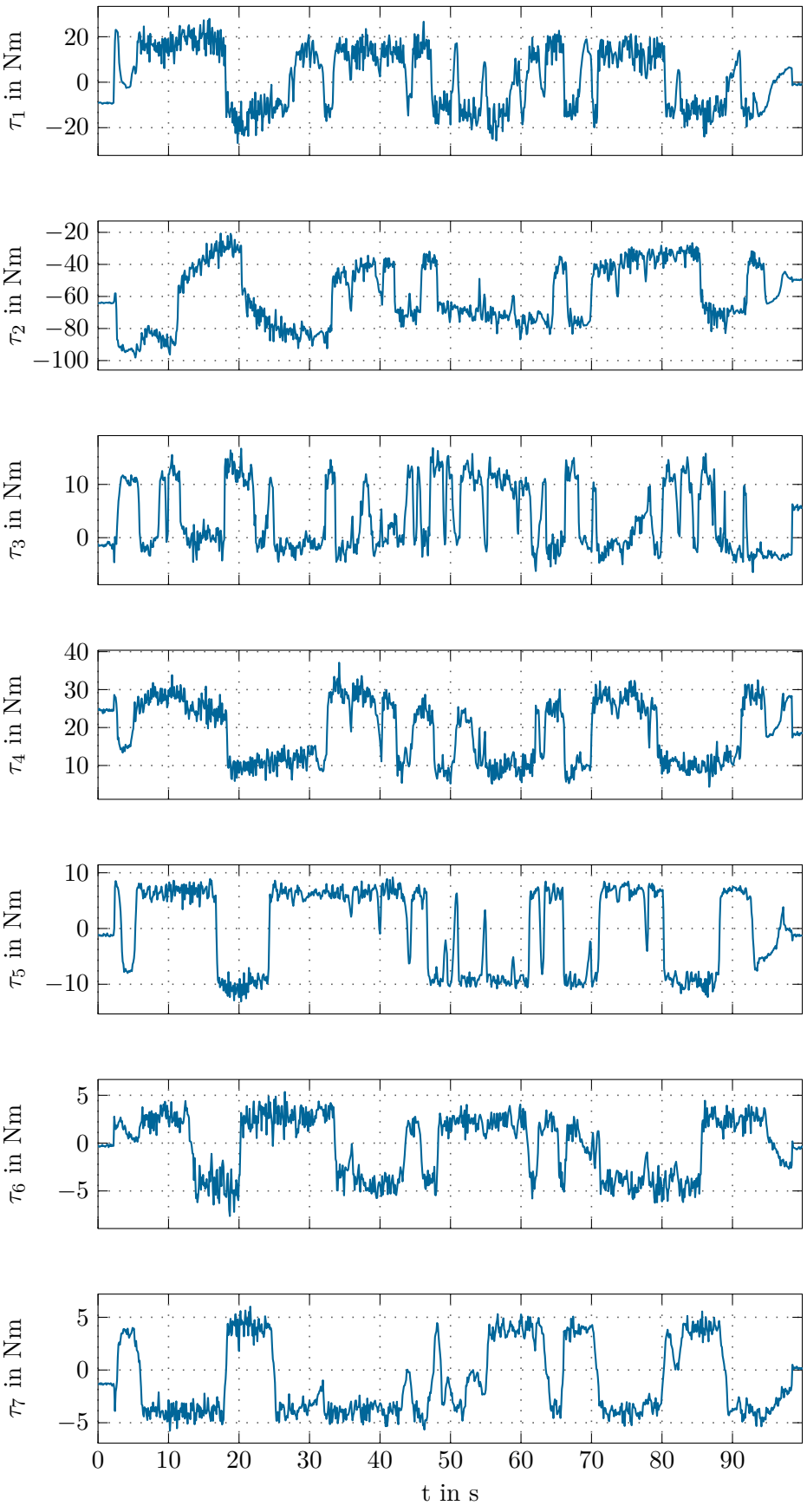


Figure 5.17: Measured joint torques $\boldsymbol{\tau}$ during the drawing on the 3D-printed rabbit with motion control.

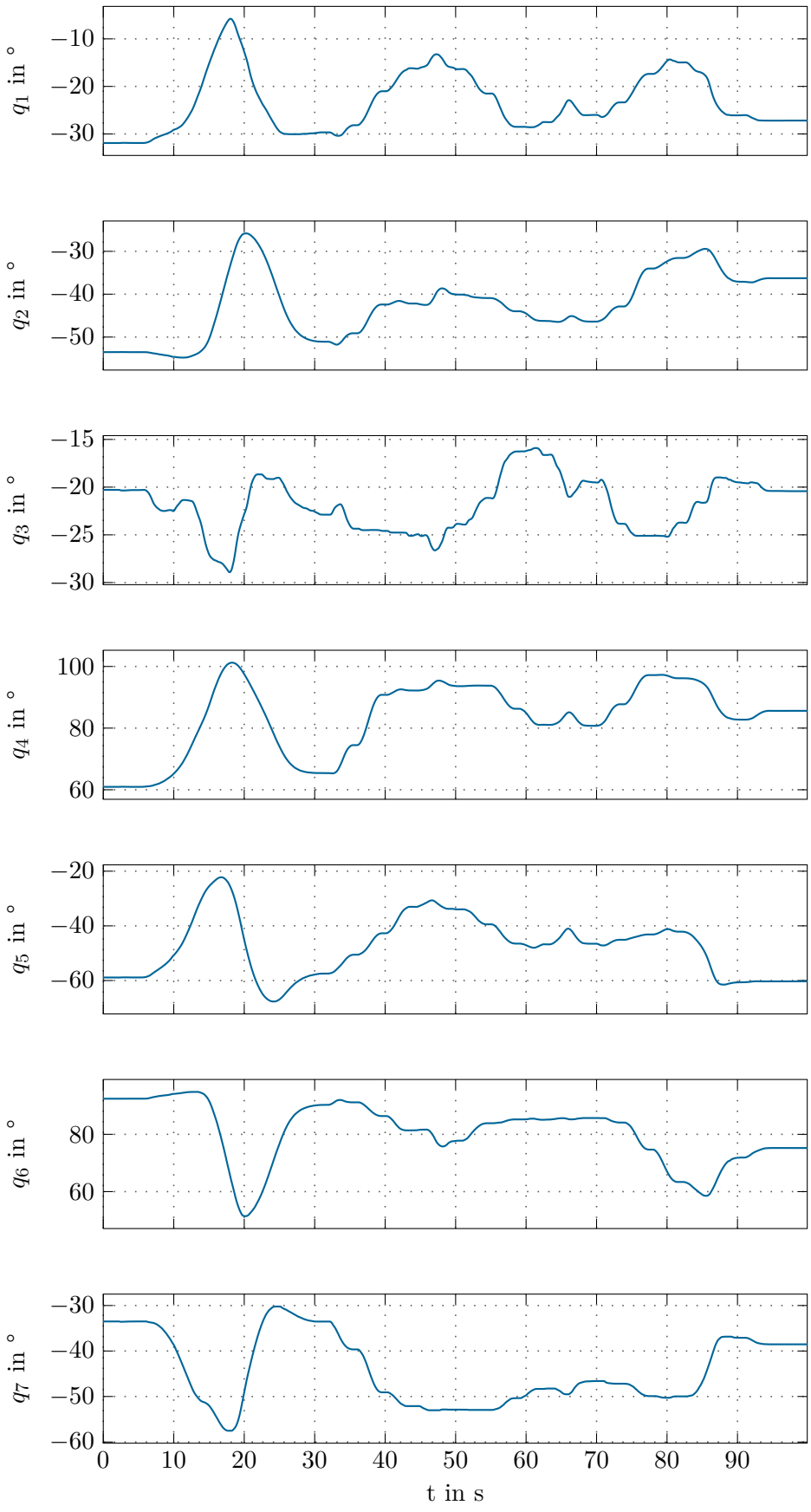


Figure 5.18: Joint angles **q** during the drawing on the 3D-printed rabbit with motion control.

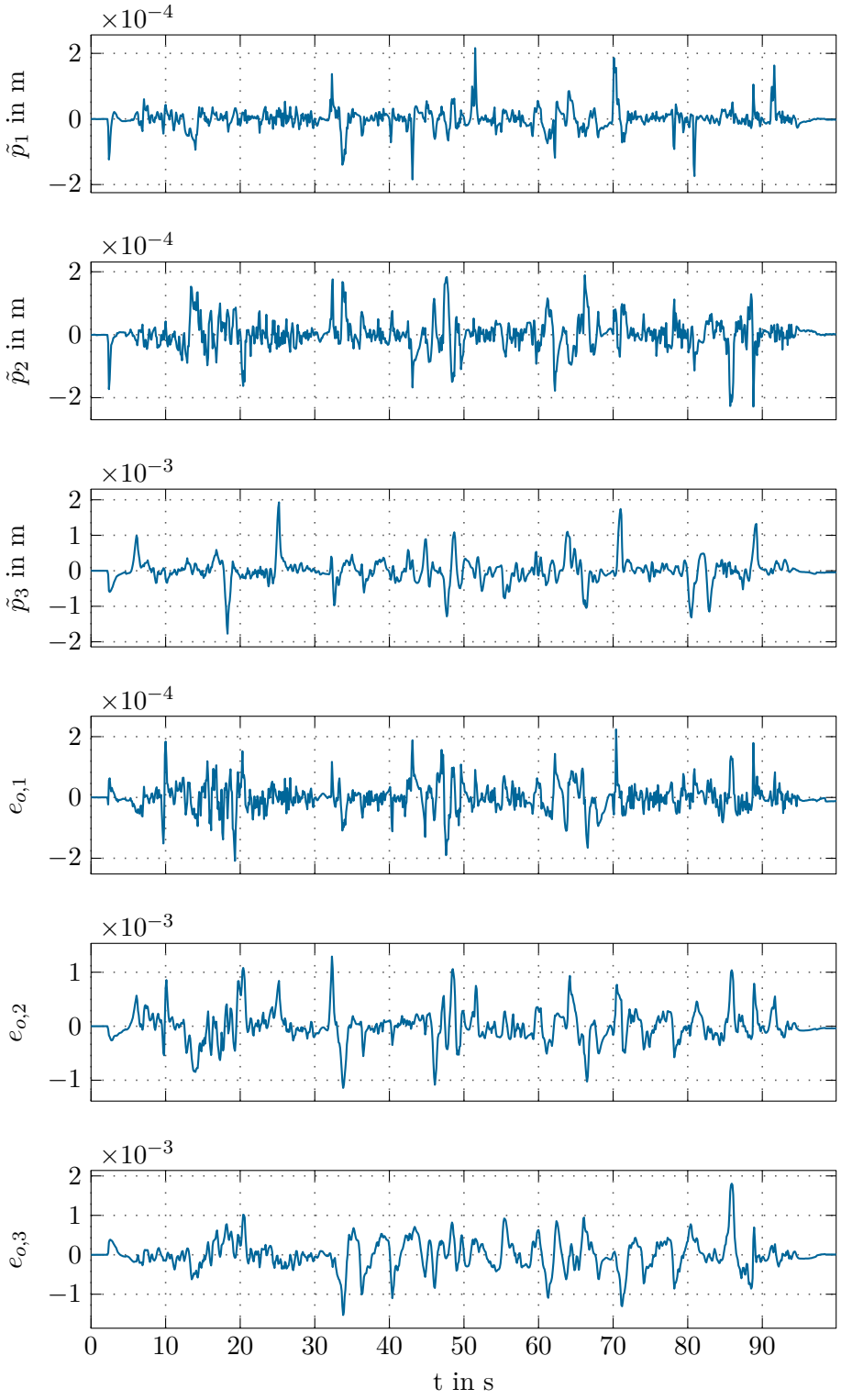


Figure 5.19: Trajectory error during the drawing on the 3D-printed rabbit with motion control.

Drawing on the Rabbit Using Hybrid Force/Motion Control and Without End-Effector Tracking

In this section, the desired contact force f_d during the drawing is set to -4 N, which is achieved with the end-effector using the hybrid force/motion controller.

Figure 5.20 shows the estimated contact force $\hat{f}_c$ and its desired value f_d during the drawing. It is seen that the contact force is maintained constant at the desired level. At the time $t = 90\,\mathrm{s}$, where the last part of the drawing is over, a peak is observed in the contact force. This force peak shows the impact caused by the relatively fast switching between hybrid and motion control at the end of the drawing stage. It should be noted that the limited absolute accuracy of the end-effector has also an influence on the impact effect. Suppose that the limited trajectory planning accuracy for the end-effector causes the programmed value for the pen tip position to be slightly under the physical surface of the object. In this case, since a spring-loaded pen gripper is used, the spring is compressed and the resulting spring force leads to a larger contact force. Hence, at the time of controller switching to pure motion control the impact force occurs. Notice that a higher position control accuracy could cause even larger force peaks, see [10]. The impact effect must be avoided to prevent high forces exerted on the object, which might damage the robot end-effector and the object.

The measured joint torques $\boldsymbol{\tau}$ and angles $\mathbf{q}$ are similar to the case without force control, and therefore not shown again. The resulting trajectory error is seen in Figure 5.21. Comparing to the drawing with the motion control in Section 5.2.4, two peaks can be observed at around $t = 50\,\mathrm{s}$ and $t = 90\,\mathrm{s}$ in the position error $\tilde{p}_1$. Those peaks result from the switching between the hybrid and motion controllers. This effect can be avoided by using a softer controller switching method. The end-effector orientation as unit quaternion is shown in Figure 5.22 together with its desired values. It can be seen that the largest orientation change emerges between $t = 10\,\mathrm{s}$ and $t = 30\,\mathrm{s}$, where the robot draws the main circle of the smiley.

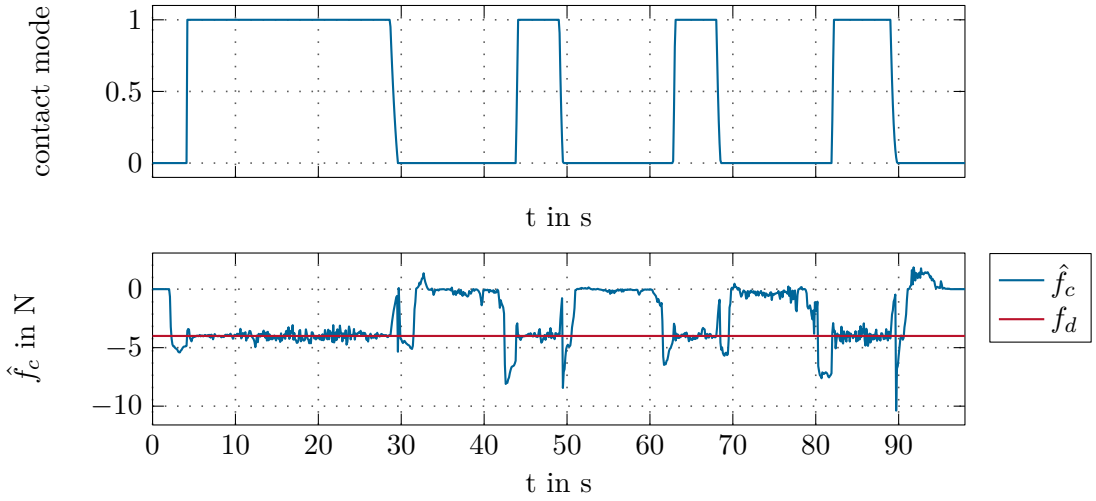


Figure 5.20: Desired trajectory of contact mode and estimated contact force $\hat{f}_c$ during the drawing on the 3D-printed rabbit with hybrid force/motion control.

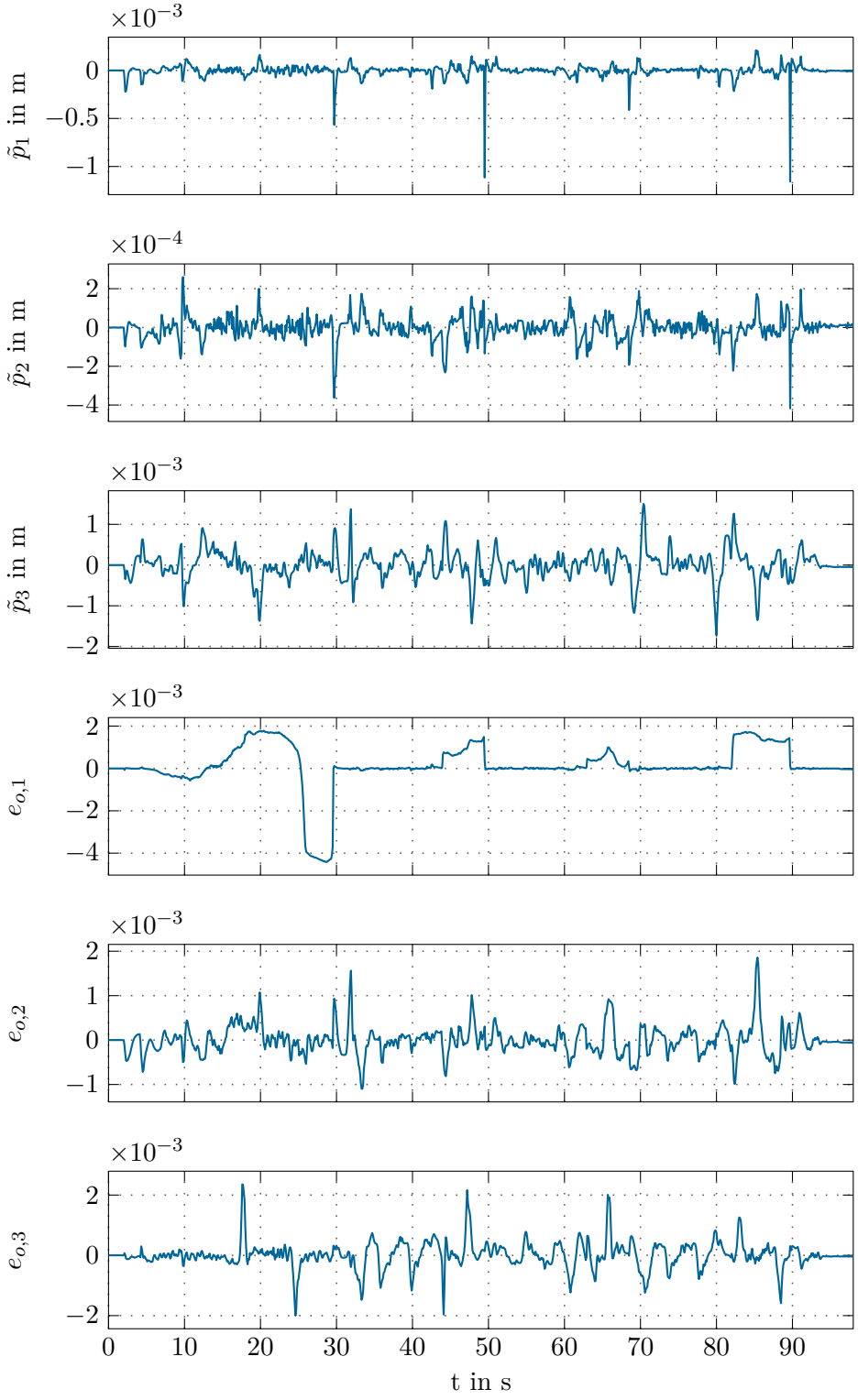


Figure 5.21: Trajectory error during the drawing on the 3D-printed rabbit with hybrid force/motion control.

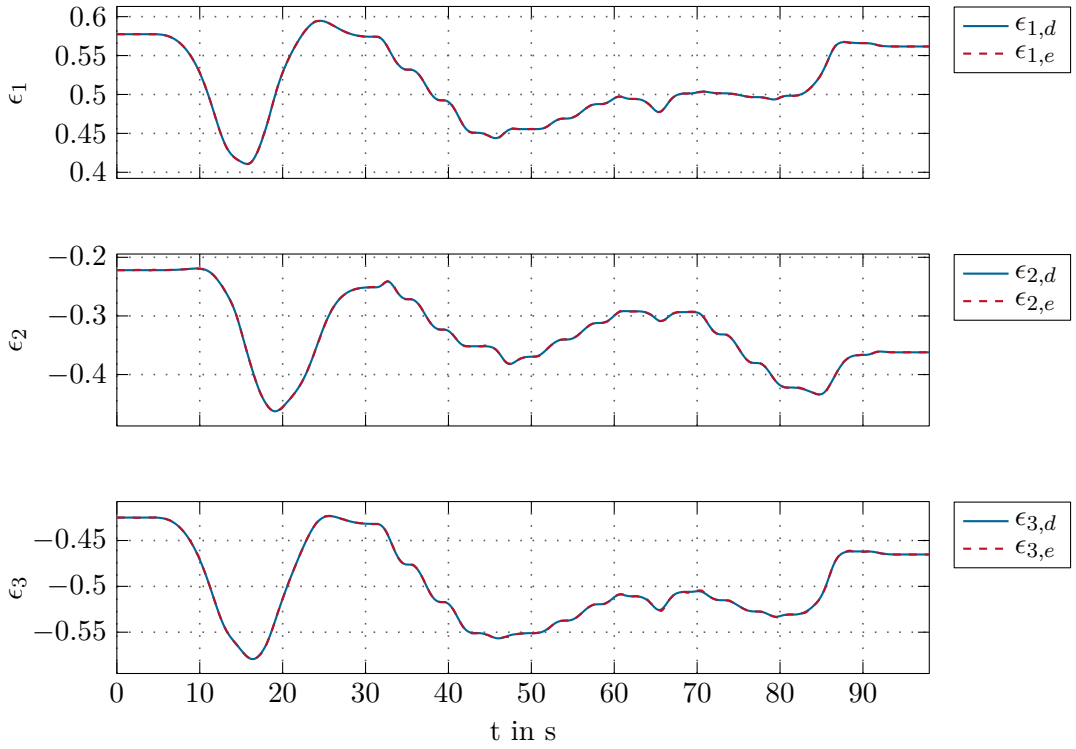


Figure 5.22: End-effector orientation as vector part of the unit quaternions and its desired value during the drawing on the 3D-printed rabbit with hybrid force/motion control.

Drawing on the Rabbit with Hybrid Force/Motion Control and Using End-Effector Tracking

In the last experiment, the OPTITRACK measuring system is utilized to improve the absolute position accuracy of the end-effector during the drawing on the 3D object. For the trajectory tracking, the hybrid force/motion control (4.44) is applied, where the measured end-effector pose is used as a controlled variable. The desired contact force f_d is chosen as -4 N.

Figure 5.23 illustrates the absolute position error $\breve{\mathbf{p}}$ of the robot end-effector during the drawing, measured by the OPTITRACK system. This error would occur, if no end-effector tracking was used. The orientation error $\breve{\mathbf{e}}_o$ is defined similarly and computed as the vector part of the quaternion error.

The estimated contact force $\hat{f}_c$ and its desired value f_d are shown in Figure 5.24. In this experiment, as a result of the improved positioning accuracy of the end-effector, the previous mentioned impact effect is not present anymore. The resulting trajectory error is depicted in Figure 5.25. In this case, the two peaks in the position tracking error $\tilde{p}_1$, which are observed in Figure 5.21, are avoided. This is achieved due to the increased positioning accuracy.

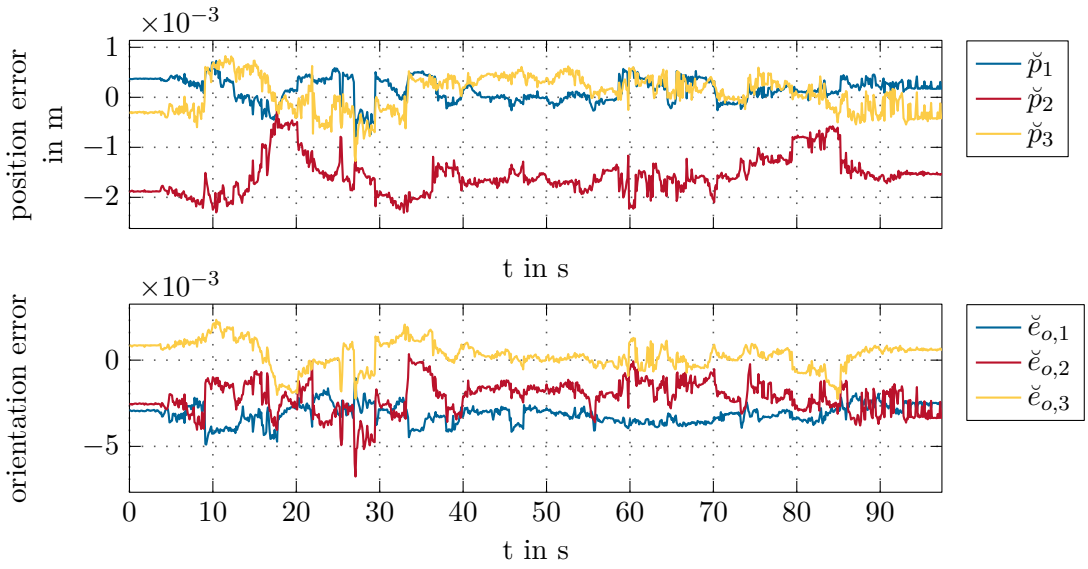


Figure 5.23: End-effector position and orientation error during the drawing on the 3D-printed rabbit with hybrid force/motion control and using OPTITRACK for the end-effector tracking.

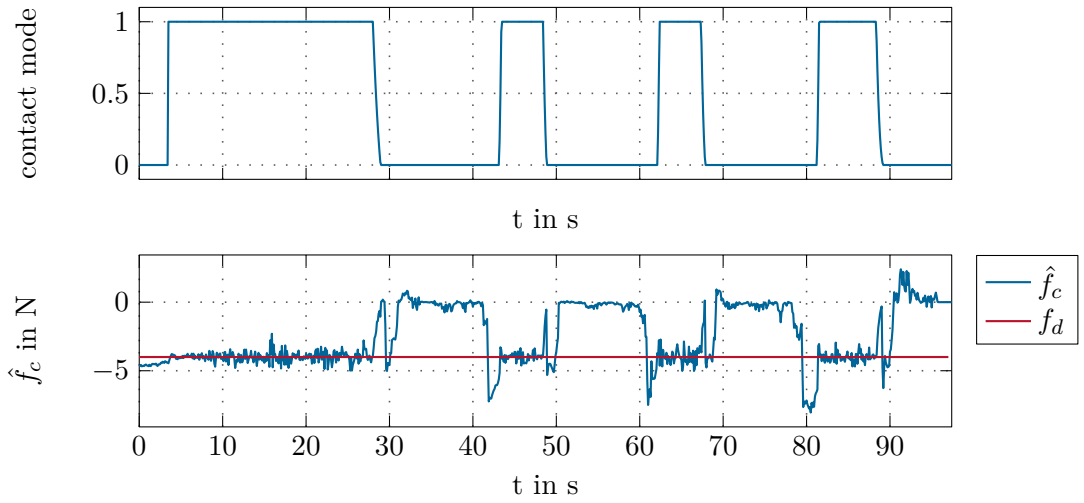


Figure 5.24: Desired trajectory of contact mode and estimated contact force $\hat{f}_c$ during the drawing on the 3D-printed rabbit with hybrid force/motion control and using OPTITRACK for the end-effector tracking.

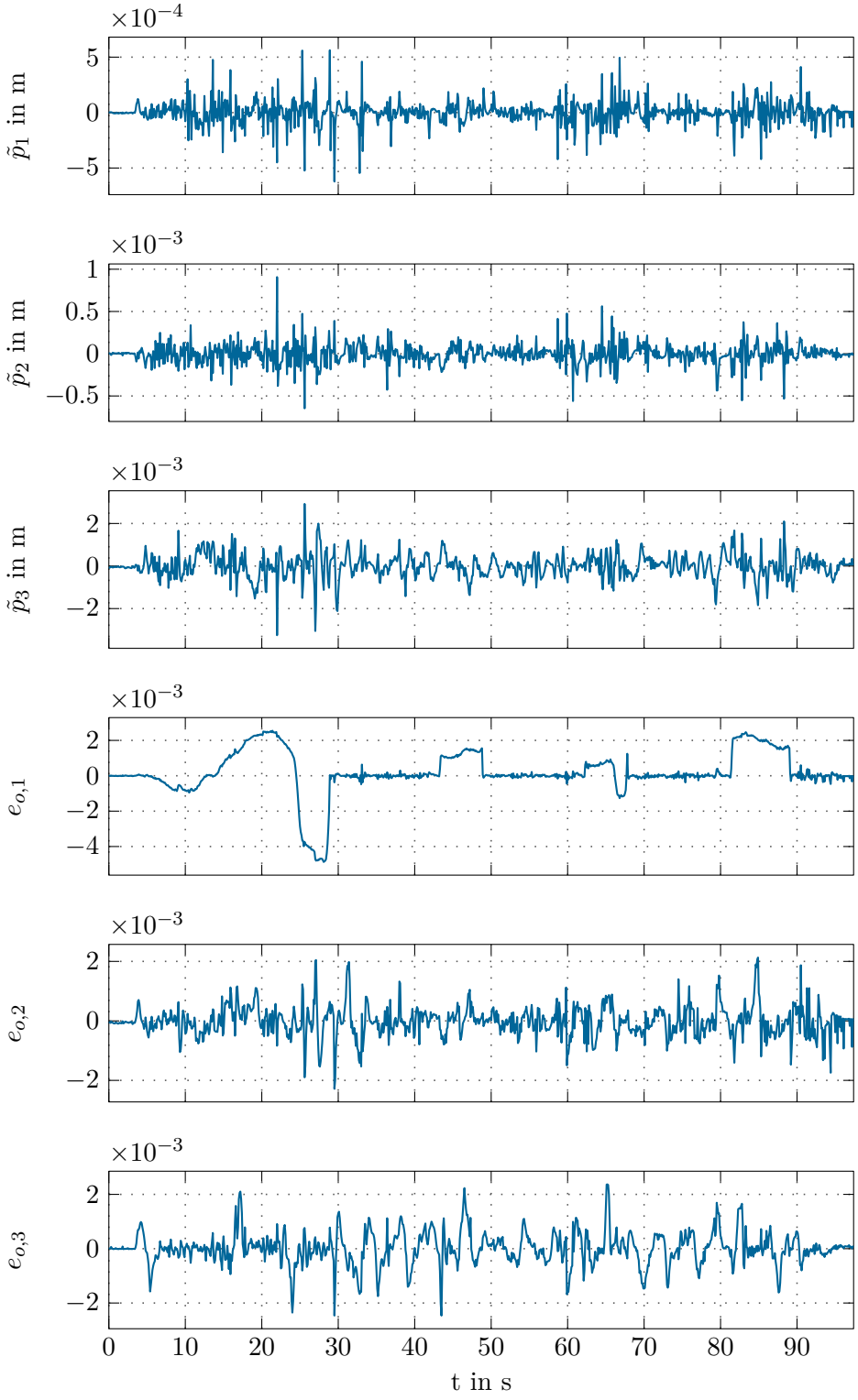


Figure 5.25: Trajectory error during the drawing on the 3D-printed rabbit with hybrid force/motion control and using OPTITRACK for the end-effector tracking.

6 Conclusions

In this thesis, a robotic drawing system was presented, which is able to draw a 2D input pattern on an arbitrarily shaped non-planar 3D object with known CAD model. To also control the contact force, the robot was controlled with a hybrid force/motion controller.

Chapter 2 deals with the mathematical model of the used robot KUKA LWR iiwa 14 R820. First, the direct and inverse kinematics of the robot is discussed. Then, by using the Euler-Lagrange approach a rigid-body model of the manipulator is derived. Based on singular perturbation theory, the dynamic model is extended to obtain a robot model with flexible joints.

In order to project the 2D pattern onto the 3D object surface, the surface is parameterized. After a segmentation method is applied to the surface, shown in Chapter 3, a least-squares conformal mapping approach is used to flatten the surface. Then the projection is performed using linear interpolation with barycentric coordinates. For trajectory planning, a cubic spline curve which passes through the projected 3D points is constructed.

The 3D object can be placed arbitrarily in the workspace of the robot. In order to obtain the position and orientation of the object, an optical measuring system is used. To improve the absolute accuracy of the robot's end-effector, the measuring system is utilized to provide the pose information of the end-effector with sub-millimeter accuracy.

The hybrid force/motion control scheme is described in Chapter 4. This controller enables the end-effector to follow the desired drawing trajectory and simultaneously maintaining a constant contact force between the pen tip and the object surface. For force control, a model-based estimation of the contact force is used. Additionally, to deal with the redundancy of the manipulator, a null-space controller is designed.

In Chapter 5, the proposed method is verified with two main experiments. In the first experiment, the drawing task is applied to a whiteboard. In the second experiment, a pattern is drawn on a 3D-printed rabbit by the robot. Using the optical tracking of the end-effector increases the absolute positional accuracy by 2 mm in all spatial directions. The use of the hybrid force/motion control enables a constant contact force and therefore, the drawing is performed more uniformly compared to pure motion control. Moreover, large undesirable contact forces, which might damage the working object or end-effector, are avoided.

For future works, the contact force estimation can be improved by using e. g. a Kalman filter or using a force/torque sensor attached to the robot wrist. Furthermore, using a time-optimal trajectory planning method can speed up the whole drawing process.

To achieve robustness against occlusions of the end-effector during the optical tracking, both the pose information from the robot joint encoders and the optical measuring system could be combined using a multi-sensor data fusion technique. This would lead to a further improvement of the absolute positioning accuracy.

Since in this work the pose of the 3D object can also be tracked by the camera system, as

a further extension, the drawing process could be synchronized with a slowly moving object. In this way, more complex collaborative scenarios could be implemented. In addition, a graphical user interface could be employed to control the whole robotic drawing system, from reading the input 2D pattern to setting the relevant parameters of the drawing process.

In this work, a robotic system is used to draw an arbitrary pattern on a 3D object with known CAD model. This is achieved by simultaneous control of both end-effector motion and contact force. The optical pose measurement of the end-effector and the object increases the accuracy of the drawing task. The proposed method could be useful for similar industrial applications such as polishing, welding and deburring, where precise manufacturing processes in a roughly calibrated environment are needed.

A Parameters

A.1 Robot Parameters

The robot parameters are obtained from [34]. Table A.1 shows the parameters used in the robot model. The joint limits of the KUKA LWR iiwa are listed in Table A.2.

Parameter	Value	Unit
d_1	360	mm
d_3	420	mm
d_5	400	mm
d_e	276	mm

Table A.1: Kinematic parameters of the KUKA LWR iiwa.

i	1	2	3	4	5	6	7	Unit
q_i^{max}	±170	±120	±170	±120	±170	±120	±175	°
τ_i^{max}	320	320	176	176	110	40	40	N m

Table A.2: Range of motion and maximum torque of each joint of the KUKA LWR iiwa.

A.2 Filter Parameters and Translation Vectors

Table A.3 shows the values of the filter parameters applied to the measured end-effector position and orientation. The entities of the translation vectors $\mathbf{d}_r^j$ and $\mathbf{d}_0^m$ used for object pose transformation are listed in Table A.4.

	Parameter	Value
Position filter	b_0	10000
	b_1	200
Orientation filter	b_0	2500
	b_1	100

Table A.3: Filter parameters.

Parameter	Value	Unit
x_r^j	2	mm
y_r^j	62	mm
z_r^j	−2	mm
x_0^m	−945	mm
y_0^m	−45	mm
z_0^m	645	mm

Table A.4: Translation parameter.

A.3 Controller Parameters

The controller parameters used in the experiments are listed in Table A.5. Herein, diag($[d_1, d_2, \cdots, d_n]$) denotes a diagonal matrix with diagonal elements d_1, d_2, …, d_n.

	Parameter	Value
Joint space computed torque	$\mathbf{K}_i$	diag([8000, 8000, 8000, 8000, 8000, 8000, 8000])
	$\mathbf{K}_p$	diag([1200, 1200, 1200, 1200, 1200, 1200, 1200])
	$\mathbf{K}_d$	diag([60, 60, 60, 60, 60, 60, 60])
Operational space computed torque	$\mathbf{K}_I$	diag([8000, 8000, 8000])
	$\mathbf{K}_P$	diag([1200, 1200, 1200])
	$\mathbf{K}_D$	diag([60, 60, 60])
	$\mathbf{K}_o$	diag([972, 972, 972])
	$\mathbf{K}_\omega$	diag([54, 54, 54])
Motion control part of hybrid controller	$\mathbf{K}_I$	diag([8000, 8000])
	$\mathbf{K}_P$	diag([1200, 1200])
	$\mathbf{K}_D$	diag([60, 60])
	$\mathbf{K}_o$	diag([972, 972, 972])
	$\mathbf{K}_\omega$	diag([54, 54, 54])
Force control	K_{If}	8
	K_{Pf}	1.2
	K_{Df}	20
Null space control	$\mathbf{K}_{pn}$	diag([9, 9, 9, 9, 9, 9, 9])
	$\mathbf{K}_{dn}$	diag([10, 10, 10, 10, 10, 10, 10])

Table A.5: Controller parameters.

Bibliography

[1] S. Calinon, J. Epiney, and A. Billard, "A Humanoid Robot Drawing Human Portraits," in *5th IEEE-RAS International Conference on Humanoid Robots*, 2005, pp. 161–166.

[2] C. Y. Lin, L. W. Chuang, and T. T. Mac, "Human Portrait Generation System for Robot Arm Drawing," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 2009, pp. 1757–1762.

[3] G. Jean-Pierre and Z. Said, "The Artist Robot: A Robot Drawing like a Human Artist," in *IEEE International Conference on Industrial Technology*, 2012, pp. 486–491.

[4] P. Tresset and F. Fol Leymarie, "Portrait Drawing by Paul the Robot," *Computers Graphics*, vol. 37, pp. 348–363, 2013.

[5] X. Huang, S. Bi, M. Dong, H. Chen, S. Fang, and N. Xi, "Automatic Feature Extraction and Optimal Path Planning for Robotic Drawing," in *IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems*, 2016, pp. 19–24.

[6] T. Xue and Y. Liu, "Robot Portrait Rendering Based on Multi-Features Fusion Method Inspired by Human Painting," in *IEEE International Conference on Robotics and Biomimetics*, 2017, pp. 2413–2418.

[7] M. Pichkalev, R. Lavrenov, R. Safin, and K. Hsia, "Face Drawing by KUKA 6 Axis Robot Manipulator," in *12th International Conference on Developments in eSystems Engineering*, 2019, pp. 709–714.

[8] S. Jain, P. Gupta, V. Kumar, and K. Sharma, "A Force-Controlled Portrait Drawing Robot," in *IEEE International Conference on Industrial Technology*, 2015, pp. 3160–3165.

[9] D. Song, T. Lee, and Y. J. Kim, "Artistic Pen Drawing on an Arbitrary Surface Using an Impedance-Controlled Robot," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 4085–4090.

[10] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics Modelling, Planning and Control*. London: Springer, 2009.

[11] M. Shimizu, H. Kakuya, W. K. Yoon, K. Kitagaki, and K. Kosuge, "Analytical Inverse Kinematic Computation for 7-DOF Redundant Manipulators With Joint Limits and Its Application to Redundancy Resolution," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1131–1142, 2008.

[12] K. Kreutz-Delgado, M. Long, and H. Seraji, "Kinematic Analysis of 7-DOF Manipulators," *The International Journal of Robotics Research*, vol. 11, no. 5, pp. 469–481, 1992.

[13] J. J. Craig, *Introduction to Robotics: Mechanics and Control*. Upper Saddle River, NJ: Pearson Education, 2005.

[14] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. New York: John Wiley & Sons, 2005.

[15] C. Ott, *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*. Berlin Heidelberg: Springer, 2008, vol. 49.

[16] L. Le Tien, A. Albu-Schäffer, K. Janschek, and G. Hirzinger, "Entkopplungsregelung und Reibungskompensation für einen Roboter mit elastischen verkoppelten Gelenken," *Automatisierungstechnik at*, no. 9, pp. 499–511, 2010.

[17] C. Hartl-Nesic, *Surface-Based Path Following Control on Freeform 3D Objects*. Düren: Shaker Verlag, 2020, vol. 49.

[18] B. Levy, S. Petitjean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *ACM Trans. Graph.*, vol. 21, pp. 362–371, 2002.

[19] G. Arfken, *Mathematical Methods for Physicists*, 3rd edition. Orlando: Academic Press, 1985.

[20] D. Alpay, *A Complex Analysis Problem Book*. Basel: Birkhäuser, 2016.

[21] A. Hubeli and M. Gross, "Multiresolution Feature Extraction for Unstructured Meshes," in *Proceedings Visualization*, vol. 1, 2001, pp. 287–294.

[22] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution Analysis of Arbitrary Meshes," in *22nd Annual Conference on Computer Graphics and Interactive Techniques*, 1995, pp. 173–182.

[23] J. Xu, X. Zhang, S. Wang, and J. Wu, "Tool Path Generation for Pattern Sculpting on Free-Form Surfaces," *The International Journal of Advanced Manufacturing Technology*, vol. 67, pp. 2469–2476, 2012.

[24] A. A. Ungar, *Barycentric Calculus in Euclidean and Hyperbolic Geometry : a Comparative Introduction*. Singapore: World Scientific, 2010.

[25] *Prime 17w, specifications*, Optitrack. [Online]. Available: `https://optitrack.com/products/prime-17w/specs.html` (visited on 04/02/2020).

[26] J. S. Yuan, "Closed-loop Manipulator Control Using Quaternion Feedback," *IEEE Journal on Robotics and Automation*, vol. 4, no. 4, pp. 434–440, 1988.

[27] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Berlin: Springer, 2008.

[28] E. Magrini and A. De Luca, "Hybrid Force/Velocity Control for Physical Human-Robot Collaboration Tasks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 857–863.

[29] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.

[30] E. Magrini, F. Flacco, and A. De Luca, "Estimation of Contact Forces Using a Virtual Force Sensor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2126–2133.

[31] J. Slotine and W. Li, *Applied Nonlinear Control*. New Jersey: Prentice Hall, 1991.

[32] B. Siciliano, "Kinematic Control of Redundant Robot Manipulators: A Tutorial," *Journal of Intelligent and Robotic Systems*, vol. 3, pp. 201–212, 1990.

[33] P. Hsu, J. Hauser, and S. Sastry, "Dynamic Control of Redundant Manipulators," in *American Control Conference*, 1988, pp. 2135–2139.

[34] *Kuka lbr iiwa 7 r800, lbr iiwa 14 r820 specifications*, 8th edition, KUKA GmbH, 2019.

[35] *Tc3 target for matlab/simulink*, Beckhoff Automation GmbH. [Online]. Available: `https://download.beckhoff.com/download/document/automation/twincat3/TE1400_TC3_Target_Matlab_EN.pdf` (visited on 04/22/2020).

Eidesstattliche Erklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct – Regeln zur Sicherung guter wissenschaftlicher Praxis (in der aktuellen Fassung des jeweiligen Mitteilungsblattes der TU Wien), insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Wien, November 2020

Amin Haddadi