TECHNISCHE
UNIVERSITÄT
WIEN
Vienna | Austria

DISSERTATION

Flexible Path Planning for Robotic Industrial Manufacturing Processes

Ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften (Dr. techn.)

unter der Leitung von
Univ.Prof. Dipl.-Ing. Dr.techn. Andreas KUGI
E376
Institut für Automatisierungs- und Regelungstechnik

eingereicht an der
Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von
Thomas WEINGARTSHOFER
Matrikelnummer: 01126457

Wien, Mai, 2024

Studiendekan:	Univ.Prof. Dr.sc. Silvan SCHMID
Betreuer:	Univ.Prof. Dipl.-Ing. Dr.techn. Andreas KUGI
Tag des Rigorosums:	21.05.2024
Prüfungsvorsitzender:	Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Thilo SAUTER
Erster Gutachter:	Prof. Dr.-Ing. Knut GRAICHEN
Zweiter Gutachter:	Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Markus VINCZE

Vorwort

Die vorliegende Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Automatisierungs- und Regelungstechnik an der Technischen Universität Wien entstanden. Die abwechslungsreiche Arbeit am Institut hat mir außerordentlich viel Freude bereitet, seien es die theoretischen Arbeiten während der Entstehung einer Publikation, die praktischen Tätigkeiten im Labor, der Austausch mit Projektpartnern um reale Probleme in der Industrie zu lösen oder die Zusammenarbeit mit motivierten Studenten. Bei diesen Tätigkeiten und durch den Austausch mit verschiedenen Personen habe ich sehr viel dazugelernt. All das hat schlussendlich diese Dissertation ermöglicht und daher möchte ich diesen besonderen Menschen im Folgenden danken.

Zuallererst meinem Doktorvater Prof. Dr. Andreas Kugi, der mich schon während meines Studiums für die Automatisierungstechnik begeistern konnte. Vor allem das Zusammenspiel von Mathematik mit Algorithmen zur Regelung von realen Systemen hat mich sofort fasziniert. Vielen Dank für jedwede Unterstützung, deinen Weitblick und die Möglichkeit, sowohl in der Forschung als auch in der Lehre einen Beitrag leisten zu dürfen. Ebenso möchte ich Dr. Christian Hartl-Nesic als meinen direkten Ansprechpartner von ganzem Herzen danken. Du hast mich während der gesamten Zeit meiner Dissertation mit deiner fachlichen Kompetenz und deinem kollegialen Umgang unterstützt, umso mehr in stressigen Zeiten. Des Weiteren möchte ich all meinen Kollegen am Institut für die schöne Zeit und die unzähligen fachlichen und privaten Gespräche danken. Es freut mich außerordentlich, dass mittlerweile so viele Freundschaften daraus entstanden sind. Ein besonderer Dank gilt meinen Kollegen der industriellen Robotik Gruppe und meinen langjährigen Kommilitonen.

Zu guter Letzt möchte ich mich bei meiner Familie für ihre Unterstützung und ihr Vertrauen bedanken. Meine Eltern haben mir schon früh die unterschiedlichen Wege einer beruflichen Laufbahn aufgezeigt und mich stets in all meinen Vorhaben unterstützt. Ohne euch wäre diese Dissertation nicht möglich gewesen. Danke!

Kurzzusammenfassung

Die Herausforderungen von flexibler Produktion mit hoch automatisierten Fertigungslinien ist der Trend zur Personalisierung von Produkten bis hin zur vollständigen Individualisierung. Mit diesem Trend müssen Pfadplanungsalgorithmen für industrielle Roboter mithalten. In dieser Arbeit werden flexible Planungsalgorithmen vorgestellt, die die automatische Erzeugung von Roboterprogrammen in der flexiblen Fertigung unterstützen und komplexe Pfadplanungsprobleme für industrielle Prozesse auf 3D-Freiformoberflächen lösen.

In der Industrie ist computerunterstützte Offline-Roboterprogrammierung Stand der Technik, bei der Fertigungspfade manuell oder teilautomatisiert generiert werden. In dieser Arbeit wird eine vollautomatische Generierung von Roboterprogrammen für 3D-Werkstücke basierend auf benutzergenerierten 2D-Eingangsmustern vorgestellt. Dafür werden zwei Projektionsmethoden von 2D auf 3D evaluiert, eine einfache parallele Projektion und eine konforme Abbildung basierend auf dem kleinsten Fehlerquadrat. Um die Genauigkeit der Projektionsmethoden zu zeigen, wird ein experimenteller Zeichenprozess mit einem industriellen Roboter durchgeführt. Dabei wird eine reine Positionsregelung mit einer hybriden Kraft-/Positionsregelung im Aufgabenraum verglichen. Mit der hybriden Kraft-/Positionsregelung wird das genaueste Zeichenergebnis erzielt, da auch die Normalkraft des Stiftes auf die Oberfläche des 3D Werkstücks geregelt werden kann.

In einer flexiblen Produktion muss eine Roboterarbeitszelle eine Vielzahl unterschiedlicher Produkte bearbeiten können. Das beinhaltet auch Produkte, die während der Konzeptionierung der Arbeitszelle noch nicht bekannt waren. Wenn bestimmte Fertigungspfade nicht ausführbar sind, sind aufwändige Anpassungen der Roboterplatzierung oder des Fertigungspfades notwendig. In dieser Arbeit wird gezeigt, dass auch ein mit geringerem Aufwand verbundenes Anpassen der Halterung des Werkzeuges am Endeffektor zu ausführbaren Fertigungspfaden führen kann. Dafür wird ein Optimierungsalgorithmus entwickelt, der mit einem Pfadplaner kombiniert wird, um im Konfigurationsraum die optimale Befestigung des Werkzeuges zu berechnen. Dabei werden unter anderem die Anzahl der möglichen Pfade im Konfigurationsraum sowie der Abstand zu den mechanischen Achslimits maximiert. Des Weiteren ist dieser Algorithmus zur Berechnung der optimalen Roboterplatzierung verwendbar. Das entwickelte Konzept wird anhand eines industriellen Schneidprozesses in der Schuhindustrie validiert, bei dem eine Reihe von unterschiedlichen Fertigungspfaden abgefahren werden muss.

Um die Flexibilität einer vorhandenen Arbeitszelle weiter zu erhöhen, können die speziellen Prozesseigenschaften eines Fertigungsprozesses gezielt genutzt werden und dadurch den Lösungsraum der Pfadplanung erheblich vergrößern. Diese Prozesseigenschaften können redundante Freiheitsgrade und zulässige Abweichungen vom Fertigungspfad (Toleranzen und Prozessfenster) sein. In dieser Arbeit wird ein neuer Pfadplaner entwickelt, der diese Prozesseigenschaften und eine Kollisionsvermeidung systematisch in einem Optimierungsproblem berücksichtigt. Dabei werden mehrere Pfade im Konfigurationsraum parallel berechnet, um den optimalen Pfad zu finden. Zwei spezielle Prozesse, ein Zeichenprozess und ein Prozess zum Spritzlackieren, werden mit diesem Algorithmus optimiert und komplexe Pfadplanungsprobleme auf 3D-Freiformoberflächen gelöst, die mit dem Stand der Technik nicht gelöst werden können.

In dieser Arbeit werden die entwickelten Algorithmen an einem experimentellen Zeichenprozess und in Simulation für einen Schneid- und Spritzlackierprozess demonstriert, die repräsentativ für andere industrielle Prozesse auf 3D-Freiformoberflächen sind. Darüber hinaus sind alle Methoden allgemein formuliert, damit diese für verschiedenste Prozesse angewandt werden können, z.B. für Schweißen, Spritzlackieren, Fräsen, Polieren oder für Textilfertigungsprozesse wie Schneiden, Nähen und Kleben.

Abstract

The current challenge in flexible production with highly automated production lines is the ongoing trend toward customization of products up to full individualization. Consequently, path-planning algorithms for industrial robots have to keep pace with this trend. This thesis presents flexible planning algorithms to support the automatic generation of robot programs in flexible automation to solve complex path-planning problems in industrial processes on freeform 3D surfaces.

In industry, offline robot programming approaches using computer-aided workflows, where manufacturing paths are generated manually or semi-automatically, are state-of-the-art. This work investigates a fully automatic generation of robot programs for 3D workpieces based on user-generated 2D input patterns. For this, two projection methods from 2D to 3D, i.e., a simple parallel projection and a least-squares conformal mapping, are evaluated. In order to show the accuracy of the projection approaches, a drawing process with an industrial robot is demonstrated in an experimental setup with two task-space control concepts, i.e., a motion control and a hybrid force/motion control. With the hybrid force/motion control, the normal contact force of the pen with the workpiece's surface is controlled, yielding the most accurate drawing result.

In flexible production, a robotic work cell must be able to execute an industrial process on a wide range of products, including ones not known during the design phase of the work cell. If specific manufacturing paths are not executable, laborious adaptions of the robot placement or manufacturing path are necessary. In some instances, adapting the tool mounting on the end-effector can also lead to executable robot trajectories, as shown in this work. Therefore, an optimization algorithm is developed and combined with a joint-space path planner to compute the optimal tool mounting while maximizing, e.g., the number of joint-space path solutions and distance to the mechanical joint limits. This algorithm is also applicable for finding the optimal robot base placement. It is validated in an industrial trimming process from the shoe industry, where a set of manufacturing paths must be executed.

To further increase the flexibility of given work cells, the distinct properties of the manufacturing process, i.e., redundant degrees of freedom and allowed deviations from the manufacturing path (tolerances and process windows), can significantly enlarge the path planning search space. Hence, an optimization-based joint-space path planner is developed, which systematically includes these process properties and a collision avoidance strategy. Multiple joint-space paths are computed in parallel to find the optimal path. The proposed algorithm optimizes two distinct processes, i.e., a drawing process and a spray-painting process. It is shown that complex path-planning problems can be solved on freeform 3D surfaces where state-of-the-art concepts fail.

In this thesis, the developed algorithms are demonstrated experimentally for a drawing task and in simulation for a trimming and spray-painting task, which are manufacturing processes representative of industrial processes on freeform 3D surfaces. Additionally, the proposed methods are formulated in a general way such that they can be easily applied to other processes, e.g., welding, spray painting, milling, polishing, or textile fabrication processes like cutting, sewing, and gluing.

Contents

Nomenclature

Acronyms and Abbreviations

2D	2-Dimensional
3D	3-Dimensional
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
CPU	Central Processing Unit
DoF	Degree of Freedom
FDM	Fused Deposition Modeling
fmincon	find minimum of constrained nonlinear multivariable function
IK	Inverse Kinematics
IoT	Internet of Things
KDL	Kinematics and Dynamics Library
LSCM	Least-Squares Conformal Mapping
OMPL	Open Motion Planning Library
PRM	Probabilistic Roadmap Methods
RAM	Random-Access Memory
RRT	Rapidly-exploring Random Trees
STOMP	Stochastic Trajectory Optimization for Motion Planning
surrogateopt	surrogate optimization for global minimization of time-consuming objective functions
TCP	Tool Center Point
TORM	Trajectory Optimization of a Redundant Manipulator
Trajopt	Trajectory Optimization for Motion Planning
uniquetol	unique values within tolerance
V-Clip	VORONOI-clip

General Notation

$a,\ \gamma,\ A,\ \Gamma,\ \dots$	scalars, sets
$\mathbf{a},\ \boldsymbol{\gamma},\ \dots$	vectors
$\mathbf{A},\ \boldsymbol{\Gamma},\ \dots$	matrices
$\mathbf{a}^{\mathrm{T}},\ \mathbf{A}^{\mathrm{T}}$	transpose of vectors and matrices
a_i	i^{th} element of the vector $\mathbf{a}$
A_{ij}	element in i^{th} row and j^{th} column of the matrix $\mathbf{A}$ with $i, j \in \mathbb{N}$
U	maps
$\mathcal{B}$	coordinate frames
Ⓟ, Ⓣ	annotation of points or timestamps
$\dot{a}$	total derivative with respect to time
$\ddot{a}$	second-order total derivative with respect to time
$\frac{\partial}{\partial a}$	partial derivative with respect to variable a
$\lvert\cdot\rvert$	absolute value of a scalar
$\lvert\cdot\rvert_{\mathrm{c}}$	cardinality of a set
$\lVert\cdot\rVert_2$	Euclidean 2-norm
$\lVert\cdot\rVert_\infty$	infinity norm
$(\cdot)^{\mathcal{Z}}_{\mathcal{Y}}$	mathematical objects describing the geometric relation of the frame $\mathcal{Z}$ w.r.t. the frame $\mathcal{Y}$, expressed in the frame $\mathcal{Y}$

Maps

U	inverse conformal map
X	conformal map

Coordinate Frames

$\mathcal{B}$	robot base frame
$\mathcal{E}$	end-effector frame
$\mathcal{L}_h$	frame of link h
$\mathcal{L}_{\mathrm{c},h}$	frame in center of mass of link h
$\mathcal{M}$	manufacturing frame
$\mathcal{P}$	workpiece frame
$\mathcal{T}$	tool center point (TCP) frame, tool frame
$\mathcal{V}$	vertex frame
$\mathcal{W}$	world frame

Latin Symbols

$\mathbf{0}$	zero vector or zero matrix of appropriate dimension
$\mathbf{A}$	weighting matrix of joint-space planner
A	area
$\mathbf{a}$	acceleration vector
a	weighting scalar of joint-space planner
$\mathbf{b}$	vector of barrier functions
$\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$	CORIOLIS matrix
C	index set of continuous joint-space paths
$\mathbf{c}$	vector of constraints
c	index of continuous joint-space path
$\mathbf{d}$	distance vector
D	index set of joint-space solutions
d	index of joint-space solution
E	energy
e	number of joint-space solutions
$\mathbf{f}$	vector of forces
$f(\cdot)$	function
$\mathbf{g}(\mathbf{q})$	vector of gravitational forces
$\mathbf{H}_{\mathcal{Y}}^{\mathcal{Z}}$	homogeneous transformation of frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$
H	set of homogeneous transformations
$\mathbf{h}(\mathbf{q})$	forward kinematics function
$\mathbf{h}^{-1}(\mathbf{q})$	inverse kinematics function
$\mathbf{I}$	identity matrix of appropriate dimension
i	imaginary unit
$\mathbf{J}_{\mathcal{Y}}^{\mathcal{Z}}$	geometric Jacobian of frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$
$\mathbf{J}_{\mathcal{Y},\mathrm{v}}^{\mathcal{Z}}$	translational part of the geometric Jacobian
$\mathbf{J}_{\mathcal{Y},\omega}^{\mathcal{Z}}$	rotational part of the geometric Jacobian
$\mathbf{K}$	gain matrix
$\mathbf{k}$	residual vector
$\mathbf{l}$	rotation vector of angle-axis representation
l	signed distance between obstacles
m	mass of a rigid body

$\mathbf{M}(\mathbf{q})$	mass matrix
$\mathbf{N}$	sparse matrix of vertices
$\mathbf{n}$	surface normal vector
n	number of degrees of freedom of a manipulator
n_c	number of objective functions
n_l	number of collision objects
n_m	number of manufacturing path poses
n_p	number of continuous paths
n_t	number of triangles
n_v	number of vertices
$\mathbf{o}_\mathcal{Y}^\mathcal{Z}$	unit quaternion $\mathbf{o}^\mathrm{T} = \left[\eta\, \boldsymbol{\varepsilon}^\mathrm{T}\right]$ of frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$
$\mathbf{P}$	projection matrix
P	set of continuous paths
$\mathbf{p}_\mathcal{Y}^\mathcal{Z}$	Cartesian position $\mathbf{p}^\mathrm{T} = \left[x\ y\ z\right]$ in 3D of the origin of frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$
Q	set of joint configurations
$\mathbf{q}$	joint configuration
$\mathbf{R}_\mathcal{Y}^\mathcal{Z}$	rotation matrix of frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$
$\mathbf{r}$	vector-valued index of rotational grid points
$\mathbf{S}(\cdot)$	skew-symmetric operator
S	mesh segment
s	path progress
$\mathbf{T}$	inertia tensor
$\mathbf{t}$	vector-valued index of translational grid points
T	triangle
t	time
$\mathbf{U}$	coordinates of all vertices in 2D plane
U	coordinates of vertex $U = u + \mathrm{i}v$ in 2D plane as complex number
$\mathbf{u}$	coordinates of a vertex in 2D plane as position vector
u	coordinate u of a vertex in 2D plane
$\mathbf{V}$	selection matrix
V	Lyapunov function
$\mathbf{v}$	coordinates of a vertex in 3D space as position vector

v	coordinate v of a vertex in 2D plane
$\mathbf{W}$	matrix of coordinate differences of vertices
W	coordinate differences of vertices defining a triangle
$\mathbf{w}$	control input
w	sharpness criterion
$\mathbf{x}_{\mathcal{Y}}^{\mathcal{Z}}$	Cartesian position $\mathbf{x}^{\mathrm{T}} = \begin{bmatrix} x & y \end{bmatrix}$ in 2D of the origin of frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$
x	Cartesian position coordinate x
$\mathbf{Y}$	constant selection matrix
$\mathbf{y}_{\mathcal{Y},\mathrm{o}}^{\mathcal{Z}}$	pose of frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$ in task space with the orientation represented in quaternions, $(\mathbf{y}_{\mathrm{o}})^{\mathrm{T}} = \begin{bmatrix} \mathbf{p}^{\mathrm{T}} & \mathbf{o}^{\mathrm{T}} \end{bmatrix}$
$\mathbf{y}_{\mathcal{Y},\varphi}^{\mathcal{Z}}$	pose of frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$ in task space with the orientation represented in Roll-Pitch-Yaw angles, $(\mathbf{y}_{\varphi})^{\mathrm{T}} = \begin{bmatrix} \mathbf{p}^{\mathrm{T}} & \boldsymbol{\varphi}^{\mathrm{T}} \end{bmatrix}$
y	Cartesian position coordinate y
z	Cartesian position coordinate z

Greek Symbols

$\boldsymbol{\alpha}$	matrix of least-squares problem
$\boldsymbol{\beta}, \boldsymbol{\gamma}$	vectors of least-squares problem
$\boldsymbol{\delta}$	vector of forces and moments $\boldsymbol{\delta}^{\mathrm{T}} = \begin{bmatrix} \mathbf{f}^{\mathrm{T}} & \boldsymbol{\mu}^{\mathrm{T}} \end{bmatrix}$
$\boldsymbol{\varepsilon}$	vector part of unit quaternion
$\boldsymbol{\zeta}$	position error state
η	scalar part of unit quaternion
θ	rotation angle
κ	number of slack variables
$\boldsymbol{\lambda}$	vector of forces and moments for subspace
λ	barrier parameter
$\boldsymbol{\mu}$	vector of moments
$\boldsymbol{\nu}$	vector of end-effector velocities for subspace
ν	weight of objective functions
$\boldsymbol{\xi}$	quaternion error state
ρ	scale of 2D path
$\boldsymbol{\sigma}$	slack variable

$\boldsymbol{\tau}$	vector of torques
$\boldsymbol{\upsilon}$	vector of end-effector velocities
$\boldsymbol{\varphi}_{\mathcal{Y}}^{\mathcal{Z}}$	Roll-Pitch-Yaw angles of frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$
χ	subset of cost functions
$\boldsymbol{\omega}_{\mathcal{Y}}^{\mathcal{Z}}$	vector of angular velocities frame $\mathcal{Z}$ w.r.t. frame $\mathcal{Y}$, expressed in $\mathcal{Y}$

Diacritics

$\hat{\bullet}$	labeling an estimated variable $\bullet$
$\bar{\bullet}$	labeling a normalized variable $\bullet$
$\tilde{\bullet}$	labeling an error variable $\bullet$

Subscripts

$_{k,h}$	referring to the entry of a sequence
$_{\mathrm{min}}$	labeling the lower limit
$_{\mathrm{max}}$	labeling the upper limit
$_{\mathrm{mean}}$	labeling the mean value

Superscripts

*	optimal solution
$^{\mathrm{H}}$	Hermitian conjugate
$^{\mathrm{I}}$	imaginary part
$^{\mathrm{R}}$	real part
†	right pseudoinverse
$^{\mathrm{T}}$	transpose

Chapter 1 Introduction

In automated production lines, an increasing number of industrial robots are put into operation every year [1]. Besides improving productivity [2], the main driver for this trend is the growing product diversity in the industry, which approaches full individualization with a high degree of automation [3]. Therefore, the demand for flexible automated production systems has increased tremendously [4]. In some production sectors like the clothing, shoe, and apparel industry, end consumers can customize and personalize products during the ordering process, and the size, appearance, and location of custom labels, logos, and symbols can be specified. This raises the demand for flexible production systems and automated workflows [5].

In order to manufacture a product intelligently, the functional interaction between the aspects of smart design, machines, monitoring, control, and scheduling must be smoothly aligned [6]. Beginning with smart design and Computer-Aided Design (CAD), the blueprints and schematics for manufacturing a customized product are developed. These plans must be executed with smart machines that consider all important process parameters. With smart monitoring with the help of sensors and Internet of Things (IoT) solutions, a cloud-based control of manufacturing line and superordinate scheduling of individual tasks is possible. Furthermore, smart commissioning in factories and warehouses with autonomous transport vehicles is crucial to building flexible production lines [6]. This thesis focuses on smart designs, i.e. CAD-based manufacturing task definitions, and smart machines, i.e., industrial robot work cells with automatic planning and execution of manufacturing tasks with high quality demands.

Major parts of this chapter have been published in the author's works [7, 8, 9, 10, 11] and are adapted for this thesis.

1.1 Automatic and Flexible Manufacturing

In industrial automation, frequently implemented manufacturing processes include welding, spray painting, milling, drilling, sanding, polishing, grinding, chamfering, see, e.g., [12, 13, 14], and also textile fabrication processes like cutting, sewing, gluing, or drawing on workpieces, see, e.g., [15, 16, 17]. In many cases, executing these manufacturing processes with an industrial robot requires a robotic tool to follow a given manufacturing path most accurately. Often, industrial processes must be executed continuously to lower the execution time and guarantee the desired manufacturing quality, e.g., a continuous welding seam or a uniform coating with spray paint. Each robot in such a production system must be programmed accordingly to achieve the specific manufacturing task. These robot programs either describe the robot movements as a task-space path, i.e., a Cartesian tool path, or as a joint-space path and can be generated using online or offline programming approaches [14]. In online programming, the robot is taught the desired motions on-site [18]. In contrast, offline programming uses a software representation of the work cell, and the robot programs are generated in a simulated environment [14], [19].

In order to keep up with the trend of flexible manufacturing, online teach-in of robots becomes infeasible due to the laborious task and the necessary set-up time in the manufacturing line. Therefore, CAD-based offline programming using fully automated Computer-Aided Manufacturing (CAM)-based production workflows is required [20], which has to keep pace with the recent developments in terms of flexibility, complexity, and computational performance [21]. In the CAM-based workflow, the manufacturing paths are generated manually by the user or automatically, e.g., with coverage algorithms or machine learning approaches [22]. In the literature, automatic offline robot programming algorithms for specific manufacturing processes and known object geometries have been published [23], e.g., uniform spray painting [24, 25, 26], polishing the workpiece surface [27, 28] or drawing on a moving workpiece [7]. In contrast, the manual design of the manufacturing paths in the CAM environment, e.g., to weld seams along certain geometric paths, requires application-specific knowledge and preparation time, which is especially cumbersome if the manufacturing paths need to be adapted frequently. Automated offline path planning based on user inputs and known object geometry would allow fast adaptions of the manufacturing process, replacing the manual regeneration of the robot program in the CAM workflow. This reduces downtimes of the manufacturing line and, therefore, enables flexible manufacturing.

In the context of flexible manufacturing, various manufacturing paths have to be executed, which are often not known during the design of the robotic work cell. The robot has to move the tool along the user-provided manufacturing paths, although the kinematic and dynamic properties of the manipulator are limited, e.g., mechanical axis or torque limits [29]. If the requirements of the intended manufacturing process exceed the capabilities of the robotic system or the planning algorithm, the manufacturing paths cannot be executed. Hence, the paths have to be adapted, i.e., changing the user input and replanning the path in the CAM workflow [30], a different robot has to be used, or the current robotic setup has to be adjusted, e.g., the robot is positioned differently relative to the workpiece [31]. Adapting the manufacturing path in the CAM environment requires time and often does not fully consider the robot's capabilities. A complete change or repositioning of the robot is costly, time-consuming, and laborious. Hence, approaches for executing manufacturing processes without adapting the manufacturing path and minimal adaptions on the robotic work cell are needed to fulfill the flexibility demands. An example of a minimal adaption could be the optimal (re-)mounting of the robotic tool on the end-effector instead of changing the robot position in the robotic setup. Additionally, smart path planners for industrial robots are important to increase flexibility in a manufacturing facility, e.g., [32, 33].

In order to consider the manufacturing processes as a whole, the geometric path and the set of process properties must be specified. Process properties characterize a specific manufacturing process in terms of tolerances, process windows, and additional degrees of freedom (DoF) provided by the robotic tool. In a sanding process, an example of a process property is the maximum allowed misalignment between the workpiece surface and the tool. The drilling axis of a drill tool constitutes a redundant DoF, around which the tool may be rotated freely without degrading the process execution. Also, position deviations might be tolerable to a certain extent in a spray process. Therefore, incorporating those specific process properties into the path planning can significantly increase the flexibility of the given robotic work cell and make the manufacturing line more adaptive to different manufacturing paths. Thus, manufacturing paths could be executed without changing the robotic setup. So far, this has been investigated for specific manufacturing processes only, e.g., welding [32] and chamfering [34].

1.2 Aim of this Work

The work aims to investigate and develop flexible and fully automatic path-planning algorithms for executing complex manufacturing processes, which cannot be planned with state-of-the-art concepts. In the previous section, the demand for flexible path planning in complex manufacturing processes is motivated. This thesis focuses on answering the following three research questions.

- In order to meet the demand for flexible offline programming, a robot program for a manufacturing process should be generated fully automatically from a user-created 2D input pattern. This pattern should be automatically transferred to a 3D workpiece to plan and execute the desired manufacturing process. This raises the question of which algorithms and workflows are best suited to execute this task and, at the same time, achieve the best manufacturing quality, e.g., introducing the least distortion of the 2D pattern on the 3D workpiece. Furthermore, a total execution pipeline should be developed and demonstrated in a laboratory environment, beginning from the input of the 2D pattern to the implementation with an industrial robot on the 3D workpiece. Additionally, control concepts for the manufacturing process should explored to demonstrate the developed workflow.
- In order to execute a process according to a user-defined manufacturing path with an industrial robot, the location of the robot base relative to the workpiece has to be chosen appropriately in the production cell. In flexible manufacturing, however, products and robot tool paths may change frequently, possibly exceeding the capabilities of the work cell with a fixed robot base. Adapting the complete setup or remounting the robot is often costly and time-consuming. Therefore, the research question is raised: Can the time-consuming remounting of the robot be avoided by adapting the tool center point (TCP) of the robotic tool?
- In order to achieve an even higher degree of flexibility provided by a robotic work cell, the following research question should be answered: How can the specific process properties of a manufacturing process be exploited in the path planning? Therefore, an advanced path-planning algorithm should systematically incorporate the process properties. Path planning problems,

which require a robot to perform continuous motions along complex manufacturing paths, may yield planning solutions that state-of-the-art concepts fail to find. This could potentially handle highly complex industrial path planning problems with significant flexibility.

The presented open research questions have been investigated in the following publications of the author, and significant parts of these publications are used in this thesis:

[7] T. Weingartshofer, M. Schwegel, C. Hartl-Nesic, T. Glück, and A. Kugi, "Collaborative Synchronization of a 7-Axis Robot", *IFAC-PapersOnLine*, vol. 52, no. 15, pp. 507–512, 2019, © IFAC.

[8] T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, "Optimal TCP and Robot Base Placement for a Set of Complex Continuous Paths", in *IEEE International Conference on Robotics and Automation*, 2021, pp. 9659-9665, © IEEE.

[9] T. Weingartshofer, A. Haddadi, C. Hartl-Nesic, and A. Kugi, "Flexible Robotic Drawing on 3D Objects with an Industrial Robot", in *IEEE Conference on Control Technology and Applications*, 2022, pp. 29-36, © IEEE.

[10] T. Weingartshofer, B. Bischof, M. Meiringer, C. Hartl-Nesic, and A. Kugi, "Optimization-based path planning framework for industrial manufacturing processes with complex continuous paths", *Robotics and Computer-Integrated Manufacturing*, vol. 82, no. 102516, pp. 1-16, 2023, © Authors, CC BY 4.0.

[11] T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, "Automatic and Flexible Robotic Drawing on Complex Surfaces with an Industrial Robot", *IEEE Transactions on Control Systems Technology*, 2023, © Authors, CC BY 4.0, in press.

1.3 Outline of the Thesis

Chapter 2 "Mathematical Model of Industrial Robots and Manufacturing Processes" introduces the mathematical model of a general industrial robot, used in the following chapters. This model comprises the robot's forward and inverse kinematics and dynamics. Further, the mathematical definition of a manufacturing process and a manufacturing path is given.

In Chapter 3 “Flexible Robotic Manufacturing based on User Inputs”, two projection methods to map the 2D input pattern to the 3D object are presented. Based on those projections, robot trajectories are generated with an automatic workflow. This workflow is validated by an experimental drawing task with an industrial robot. Additionally, two control concepts, i.e., pure motion control and hybrid force/motion control, are investigated, and a contact force estimation is implemented to guarantee a constant contact force during the drawing process. The proposed automated workflow applies to various industrial processes, e.g., spray painting, cutting, (laser) engraving.

Chapter 4 “Optimal TCP and Robot Base Placement” focuses on avoiding costly and time-consuming repositioning of the robot by merely adapting the TCP in high-mix/low-volume scenarios with complex continuous paths. To this end, an algorithm for the optimal TCP placement for a set of manufacturing paths is proposed. This algorithm is based on a fast joint-space path planner capable of moving through kinematic singularities. Because the robot base placement of an industrial robot in flexible production lines is crucial, the proposed concept is also applied to finding the optimal robot base placement. The feasibility of the approach is demonstrated for a trim application in shoe production for 44 complex continuous tool paths.

In Chapter 5 “Path Planning using Process Properties”, the flexibility of a robotic work cell is further improved by systematically incorporating the process properties of the manufacturing process in a path-planning framework. Consequently, planning and executing a larger variety of manufacturing paths becomes possible without adaptions of the robotic setup. The proposed path planning framework is demonstrated for experimental drawing and simulated spraying processes. Here, the planner can solve complex planning problems with continuous manufacturing paths by systematically exploiting the process properties by considering collisions and the ability to move through singular configurations. This leads to planning solutions that state-of-the-art concepts cannot find.

In Chapter 6 “Conclusions and Outlook”, this thesis is summarized, possible combinations of the concepts are presented, and an outlook on future work is given.

Chapter 2 Mathematical Model of Industrial Robots and Manufacturing Processes

In order to execute a given manufacturing task, e.g., welding or spray painting, the industrial robot has to move a tool relative to a workpiece. Planning and controlling industrial robots requires suitable mathematical models of the robot and the processes to be executed. Hence, this chapter gives the mathematical formulations of an industrial robot and a manufacturing process. These definitions will be used for several applications in this thesis.

The first section of this chapter focuses on the mathematical model of an industrial robot. For this, the robot's forward kinematics is introduced with homogeneous transformations. Then, a general formulation of the Jacobian and the inverse kinematics is presented, followed by the derivation of the dynamic robot model. In the second section of this chapter, a general manufacturing process is defined mathematically. The notation of a geometric manufacturing path and the specific process properties and necessary DoF are presented.

Major parts of this chapter have been published in the author's work [10] and are adapted for this thesis.

2.1 Robot Model

In this section, the kinematic and dynamic models of a general industrial robot are presented. The forward kinematics is computed using homogeneous transformations, followed by defining the manipulator Jacobian and inverse kinematics function. The specific industrial robots used in this thesis are described in Appendix A.

Major parts of this section have been published in the author's work [10] and are adapted for this thesis.

2.1.1 Homogeneous Transformations

The pose of a coordinate frame or rigid body comprises the position and orientation w.r.t. a reference frame. The Cartesian position of the origin of a coordinate frame is denoted by $\left(\mathbf{p}_{\mathcal{X}}^{\mathcal{Y}}\right)^{\mathrm{T}} = \left[x_{\mathcal{X}}^{\mathcal{Y}}\ y_{\mathcal{X}}^{\mathcal{Y}}\ z_{\mathcal{X}}^{\mathcal{Y}}\right] \in \mathbb{R}^3$. Note that here and in the following, the notation $(\cdot)_{\mathcal{X}}^{\mathcal{Y}}$ refers to mathematical objects describing the geometric relation of the frame $\mathcal{Y}$ w.r.t. $\mathcal{X}$, expressed in $\mathcal{X}$, see, e.g., [29]. For describing the Cartesian orientation of a coordinate frame, multiple representations are used in this thesis and are defined in the following, see, e.g., [29]:

- Rotation matrix: The orientation of the coordinate frame $\mathcal{Y}$ w.r.t. the coordinate frame $\mathcal{X}$ is described by the rotation matrix $\mathbf{R}_{\mathcal{X}}^{\mathcal{Y}} \in \mathrm{SO}(3)$. A rotation matrix has the mathematical properties

$$(\mathbf{R}_{\mathcal{X}}^{\mathcal{Y}})^{-1} = (\mathbf{R}_{\mathcal{X}}^{\mathcal{Y}})^{\mathrm{T}} = \mathbf{R}_{\mathcal{Y}}^{\mathcal{X}} \ . \tag{2.1}$$

  An elementary rotation by the angle θ around the local coordinate axis $i \in \{x, y, z\}$ is denoted as $\mathbf{R}_{i,\theta}$. Consecutive rotations around different coordinate axes are computed by postmultiplication of the elementary rotations.

- Roll-Pitch-Yaw angles: A minimal representation of the orientation is given by the Roll-Pitch-Yaw angles $\left(\boldsymbol{\varphi}_{\mathcal{X}}^{\mathcal{Y}}\right)^{\mathrm{T}} = \left[\varphi_x\ \varphi_y\ \varphi_z\right]$. The coordinate frames are always rotated around the axes of the fixed coordinate frame $\mathcal{X}$. The rotation matrix corresponding to $\boldsymbol{\varphi}_{\mathcal{X}}^{\mathcal{Y}}$ is calculated by premultiplication of the elementary rotations with

$$\mathbf{R}_{\mathcal{X}}^{\mathcal{Y}}(\boldsymbol{\varphi}_{\mathcal{X}}^{\mathcal{Y}}) = \mathbf{R}_{z,\varphi_z}\mathbf{R}_{y,\varphi_y}\mathbf{R}_{x,\varphi_x} \ . \tag{2.2}$$

Note that consecutive rotations around the axes in the order z, y, and x, i.e., postmultiplications of the elementary rotations, are equivalent to consecutive rotations around the fixed axes x, y, and z [29].

- Unit quaternion: Another way to describe an orientation is the unit quaternion $\left(\mathbf{o}_{\mathcal{X}}^{\mathcal{Y}}\right)^{\mathrm{T}} = \begin{bmatrix}\eta & \boldsymbol{\varepsilon}^{\mathrm{T}}\end{bmatrix} \in \mathbb{R}^4$, with the scalar part η and the vector part $\boldsymbol{\varepsilon}^{\mathrm{T}} = \begin{bmatrix}\varepsilon_x & \varepsilon_y & \varepsilon_z\end{bmatrix}$. The elements of the unit quaternion $\mathbf{o}_{\mathcal{X}}^{\mathcal{Y}}$ satisfy the normalization condition
$$\eta^2 + \varepsilon_x^2 + \varepsilon_y^2 + \varepsilon_z^2 = 1 \ . \tag{2.3}$$
Further, mathematical definitions and relations of unit quaternions, which are used throughout this thesis, are stated in Appendix B.

The homogeneous transformation

$$\mathbf{H}_{\mathcal{X}}^{\mathcal{Y}} = \begin{bmatrix}\mathbf{R}_{\mathcal{X}}^{\mathcal{Y}} & \mathbf{p}_{\mathcal{X}}^{\mathcal{Y}} \\ \mathbf{0} & 1\end{bmatrix} \tag{2.4}$$

describes the pose of a coordinate frame $\mathcal{Y}$ w.r.t. the coordinate frame $\mathcal{X}$. The corresponding inverse transformation is analytically given by

$$\left(\mathbf{H}_{\mathcal{X}}^{\mathcal{Y}}\right)^{-1} = \begin{bmatrix}\left(\mathbf{R}_{\mathcal{X}}^{\mathcal{Y}}\right)^{-1} & -\left(\mathbf{R}_{\mathcal{X}}^{\mathcal{Y}}\right)^{-1}\mathbf{p}_{\mathcal{X}}^{\mathcal{Y}} \\ \mathbf{0} & 1\end{bmatrix} = \begin{bmatrix}\mathbf{R}_{\mathcal{Y}}^{\mathcal{X}} & -\mathbf{R}_{\mathcal{Y}}^{\mathcal{X}}\mathbf{p}_{\mathcal{X}}^{\mathcal{Y}} \\ \mathbf{0} & 1\end{bmatrix} = \begin{bmatrix}\mathbf{R}_{\mathcal{Y}}^{\mathcal{X}} & \mathbf{p}_{\mathcal{Y}}^{\mathcal{X}} \\ \mathbf{0} & 1\end{bmatrix} = \mathbf{H}_{\mathcal{Y}}^{\mathcal{X}} \ , \tag{2.5}$$

where $\mathbf{0}$ denotes a zero vector or zero matrix with appropriate size [29]. Additionally, a sequence of transformations $\mathbf{H}_{\mathcal{X}}^{\mathcal{Y}}$ and $\mathbf{H}_{\mathcal{Y}}^{\mathcal{Z}}$ is composed by multiplying the homogeneous transformations in the form

$$\mathbf{H}_{\mathcal{X}}^{\mathcal{Z}} = \mathbf{H}_{\mathcal{X}}^{\mathcal{Y}}\mathbf{H}_{\mathcal{Y}}^{\mathcal{Z}} \ . \tag{2.6}$$

Using the homogeneous transformation (2.4), the position of the origin of the coordinate frame $\mathcal{Z}$ described in the coordinate frame $\mathcal{Y}$, i.e. $\mathbf{p}_{\mathcal{Y}}^{\mathcal{Z}}$, can be transformed into the coordinate frame $\mathcal{X}$ using, see [29]

$$\begin{bmatrix}\mathbf{p}_{\mathcal{X}}^{\mathcal{Z}} \\ 1\end{bmatrix} = \mathbf{H}_{\mathcal{X}}^{\mathcal{Y}}\begin{bmatrix}\mathbf{p}_{\mathcal{Y}}^{\mathcal{Z}} \\ 1\end{bmatrix} \ . \tag{2.7}$$

2.1.2 Forward Kinematics

The relationship for homogeneous transformations (2.6) is used to describe the forward kinematics of the end-effector frame $\mathcal{E}$ w.r.t. the robot base frame $\mathcal{B}$ of a robot with n DoF in serial kinematics, see [29]

$$\mathbf{H}_{\mathcal{B}}^{\mathcal{E}}(\mathbf{q}) = \mathbf{H}_{\mathcal{B}}^{\mathcal{L}_1}(q_1)\mathbf{H}_{\mathcal{L}_1}^{\mathcal{L}_2}(q_2)\cdots\mathbf{H}_{\mathcal{L}_{n-1}}^{\mathcal{L}_n}(q_n)\mathbf{H}_{\mathcal{L}_n}^{\mathcal{E}} \ . \tag{2.8}$$

In (2.8), the coordinate frames attached to the robot links are denoted by $\mathcal{L}_h, h = 1, \ldots, n$, and the joint configuration $\mathbf{q}^{\mathrm{T}} = \begin{bmatrix} q_1 & \cdots & q_n \end{bmatrix}$ contains the generalized coordinates of the robot, i.e., the positions of prismatic joints and the angles of revolute joints.

Augmented Forward Kinematics

In this work, industrial robots are used to execute manufacturing processes on workpieces. Therefore, additional coordinate frames at the tool center point (TCP) $\mathcal{T}$, at the origin of the workpiece $\mathcal{P}$, and a world frame $\mathcal{W}$ are added to consider all components of the work cell. In general, the kinematics of the manufacturing process is described by an augmented forward kinematics $\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}$, which denotes the pose of the TCP frame $\mathcal{T}$ w.r.t. the workpiece frame $\mathcal{P}$. Depending on the manufacturing process, the tool may be mounted on the end-effector of the robot and the workpiece is stationary or the tool may be stationary in the world frame $\mathcal{W}$ and the workpiece is mounted on the end-effector and manipulated by the robot, see Fig. 2.1. Hence, depending on the mounting of the tool, the augmented forward kinematics $\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}$ is specified differently, as discussed in the following.

Tool mounted on the end-effector When the tool is mounted on the end-effector, the pose of the TCP frame $\mathcal{T}$ w.r.t. the end-effector frame $\mathcal{E}$ is given by the homogeneous transformation $\mathbf{H}_{\mathcal{E}}^{\mathcal{T}}$. Additionally, the stationary poses of the robot base $\mathcal{B}$ and the workpiece $\mathcal{P}$ w.r.t. the world frame $\mathcal{W}$ are specified by $\mathbf{H}_{\mathcal{W}}^{\mathcal{B}}$ and $\mathbf{H}_{\mathcal{W}}^{\mathcal{P}}$, respectively. Consequently, the pose $\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}$ of the manufacturing system is computed in the form

$$\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}) = \left(\mathbf{H}_{\mathcal{W}}^{\mathcal{P}}\right)^{-1}\mathbf{H}_{\mathcal{W}}^{\mathcal{B}}\mathbf{H}_{\mathcal{B}}^{\mathcal{E}}(\mathbf{q})\mathbf{H}_{\mathcal{E}}^{\mathcal{T}} \ , \tag{2.9}$$

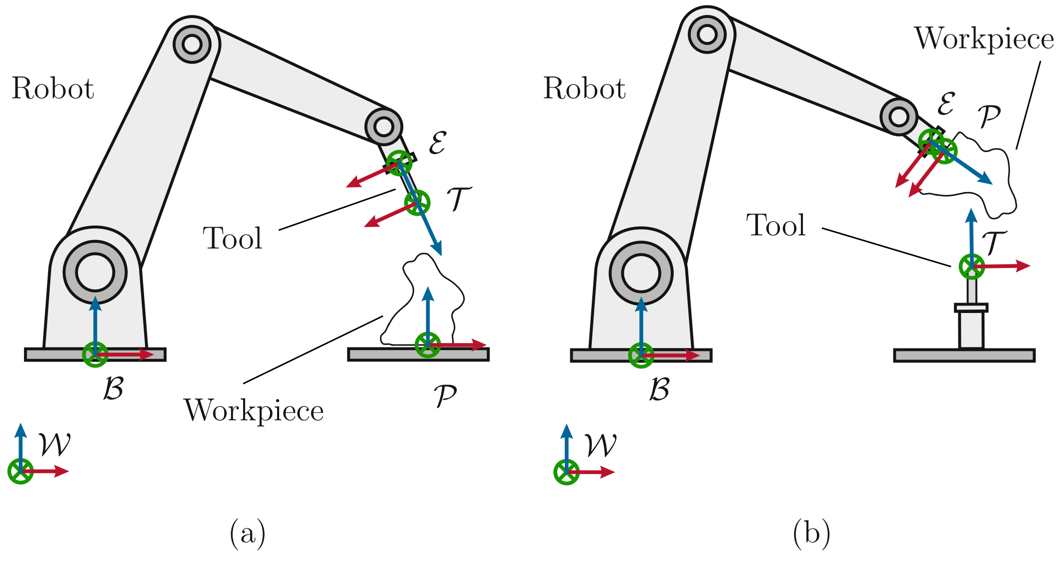


Figure 2.1: Kinematics of the robot: (a) Tool mounted on the end-effector, (b) Stationary tool; adapted from [10].

which is the augmented forward kinematics of the tool $\mathcal{T}$ w.r.t. the workpiece $\mathcal{P}$, see Fig. 2.1a.

Stationary tool If the tool is stationary, the workpiece is mounted on the end-effector flange. Then, the pose $\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}$ of the manufacturing system is

$$\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}) = \mathbf{H}_{\mathcal{P}}^{\mathcal{E}}(\mathbf{H}_{\mathcal{B}}^{\mathcal{E}}(\mathbf{q}))^{-1}\mathbf{H}_{\mathcal{B}}^{\mathcal{W}}\mathbf{H}_{\mathcal{W}}^{\mathcal{T}} , \tag{2.10}$$

utilizing the known geometric relations between the workpiece frame $\mathcal{P}$, the end-effector frame $\mathcal{E}$, the TCP frame $\mathcal{T}$, the world frame $\mathcal{W}$, and the robot base frame $\mathcal{B}$, described by $\mathbf{H}_{\mathcal{P}}^{\mathcal{E}}$, $\mathbf{H}_{\mathcal{W}}^{\mathcal{T}}$, and $\mathbf{H}_{\mathcal{B}}^{\mathcal{W}}$, see Fig. 2.1b.

The pose of the tool frame (2.9) and (2.10) can also be written as forward kinematics function $\mathbf{h}_{\mathrm{H}}(\mathbf{q})$ with

$$\mathbf{H}_{\mathcal{P}}^{\mathcal{T}} = \mathbf{h}_{\mathrm{H}}(\mathbf{q}) , \tag{2.11}$$

and

$$\mathbf{y}_{\mathcal{P},\mathrm{o}}^{\mathcal{T}} = \begin{bmatrix} \mathbf{p}_{\mathcal{P}}^{\mathcal{T}} \\ \mathbf{o}_{\mathcal{P}}^{\mathcal{T}} \end{bmatrix} = \mathbf{h}_{\mathrm{o}}(\mathbf{q}) , \tag{2.12a}$$

$$\mathbf{y}^{\mathcal{T}}_{\mathcal{P},\varphi} = \begin{bmatrix} \mathbf{p}^{\mathcal{T}}_{\mathcal{P}} \\ \boldsymbol{\varphi}^{\mathcal{T}}_{\mathcal{P}} \end{bmatrix} = \mathbf{h}_{\varphi}(\mathbf{q}) \ , \tag{2.12b}$$

where the vector on the left-hand side consists of the Cartesian position vector $\mathbf{p}^{\mathcal{T}}_{\mathcal{P}}$ and an orientation representation. In $\mathbf{y}^{\mathcal{T}}_{\mathcal{P},\mathrm{o}}$, the orientation is expressed as unit quaternion $\mathbf{o}^{\mathcal{T}}_{\mathcal{P}}$, and in $\mathbf{y}^{\mathcal{T}}_{\mathcal{P},\varphi}$, it is given as Roll-Pitch-Yaw angles $\boldsymbol{\varphi}^{\mathcal{T}}_{\mathcal{P}}$, see Section 2.1.1. With the forward kinematics (2.12), the joint configuration $\mathbf{q}$ in the joint space is mapped into the task space, i.e., the coordinates to describe the pose of the tool frame $\mathcal{T}$ w.r.t. the workpiece frame $\mathcal{P}$ in which the task is defined, see [29].

2.1.3 Manipulator Jacobian

The instantaneous Cartesian and angular velocity of the tool frame $\mathcal{T}$ w.r.t. the workpiece frame $\mathcal{P}$ is combined in the velocity $\boldsymbol{v}^{\mathcal{T}}_{\mathcal{P}}$ and given by, see, e.g., [29]

$$\boldsymbol{v}^{\mathcal{T}}_{\mathcal{P}} = \begin{bmatrix} \dot{\mathbf{p}}^{\mathcal{T}}_{\mathcal{P}} \\ \boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{q}} \mathbf{p}^{\mathcal{T}}_{\mathcal{P}} \\ \frac{\partial}{\partial \dot{\mathbf{q}}} \boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}} \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q}) \dot{\mathbf{q}} = \begin{bmatrix} \mathbf{J}^{\mathcal{T}}_{\mathcal{P},\mathrm{v}}(\mathbf{q}) \\ \mathbf{J}^{\mathcal{T}}_{\mathcal{P},\omega}(\mathbf{q}) \end{bmatrix} \dot{\mathbf{q}} \ , \tag{2.13}$$

with the geometric manipulator Jacobian $\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})$ and the joint velocity $\dot{\mathbf{q}}$, i.e., the total time derivative $\frac{\mathrm{d}\mathbf{q}}{\mathrm{d}t}$, and $\frac{\partial}{\partial \cdot}$ refers to the partial derivative. The Jacobian is composed of the Jacobian $\mathbf{J}^{\mathcal{T}}_{\mathcal{P},\mathrm{v}}(\mathbf{q})$ related to the linear velocity $\dot{\mathbf{p}}^{\mathcal{T}}_{\mathcal{P}}$ and the Jacobian $\mathbf{J}^{\mathcal{T}}_{\mathcal{P},\omega}(\mathbf{q})$ related to the angular velocity $\boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}}$. The geometric Jacobian $\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})$ in (2.13) is computed with the skew-symmetric operator, see, e.g., [35]

$$\mathbf{S}(\boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}}) = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.14}$$

and the relations

$$\mathbf{S}(\boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}}) = \dot{\mathbf{R}}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q}) \left(\mathbf{R}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\right)^{\mathrm{T}} = \sum_{h=1}^{n} \left(\frac{\partial \mathbf{R}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})}{\partial q_h} \right) \left(\mathbf{R}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\right)^{\mathrm{T}} \dot{q}_h \tag{2.15}$$

and

$$\dot{\mathbf{p}}^{\mathcal{T}}_{\mathcal{P}} = \sum_{h=1}^{n} \left(\frac{\partial \mathbf{p}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})}{\partial q_h} \right) \dot{q}_h \ . \tag{2.16}$$

2.1.4 Inverse Kinematics

The inverse of the function $\mathbf{h}(\mathbf{q})$ in (2.11) and (2.12) is the inverse kinematics, formally defined as

$$Q = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_e\} = \mathbf{h}_{\mathrm{o}}^{-1}(\mathbf{y}_{\mathcal{P},\mathrm{o}}^{\mathcal{T}}) = \mathbf{h}_{\varphi}^{-1}(\mathbf{y}_{\mathcal{P},\varphi}^{\mathcal{T}}) = \mathbf{h}_{\mathrm{H}}^{-1}(\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}) \ , \tag{2.17}$$

where Q is the set of all possible solutions $\mathbf{q}_i, i = 1, \ldots, e$, where $e = |Q|_{\mathrm{c}}$ is the number of joint-space solutions and $|\cdot|_{\mathrm{c}}$ denotes the cardinality of the set $\cdot$, cf. [8]. Note that if the dimension of the task space is smaller than the robot's DoF n, the robot is kinematically redundant, and an infinite number of inverse kinematics solutions exist; see [29]. In order to obtain a finite number of solutions e in (2.17) again, the redundant DoF are sampled.

2.1.5 Dynamic Model

The dynamic model of a manipulator is calculated with the Euler-Lagrange equations using the Lagrangian $E_{\mathrm{L}} = E_{\mathrm{T}} - E_{\mathrm{V}}$, see, e.g., [29],

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial E_{\mathrm{L}}}{\partial \dot{q}_h}\right) - \frac{\partial E_{\mathrm{L}}}{\partial q_h} = \tau_h \ , \quad h = 1, \ldots, n \ , \tag{2.18}$$

where E_{T} represents the kinetic energy

$$E_{\mathrm{T}}(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2}\dot{\mathbf{q}}^{\mathrm{T}} \underbrace{\sum_{h=1}^{n} \left(m_h \left(\mathbf{J}_{\mathcal{W},\mathrm{v}}^{\mathcal{L}_{\mathrm{c},h}}\right)^{\mathrm{T}} \mathbf{J}_{\mathcal{W},\mathrm{v}}^{\mathcal{L}_{\mathrm{c},h}} + \left(\mathbf{J}_{\mathcal{W},\omega}^{\mathcal{L}_{\mathrm{c},h}}\right)^{\mathrm{T}} \mathbf{T}_{\mathcal{W}}^{\mathcal{L}_{\mathrm{c},h}} \mathbf{J}_{\mathcal{W},\omega}^{\mathcal{L}_{\mathrm{c},h}} \right)}_{\mathbf{M}(\mathbf{q})} \dot{\mathbf{q}} \ , \tag{2.19}$$

and E_{V} is the potential energy

$$E_{\mathrm{V}}(\mathbf{q}) = -\sum_{h=1}^{n} (\mathbf{a}_{\mathrm{g}})^{\mathrm{T}} \mathbf{p}_{\mathcal{W}}^{\mathcal{L}_{\mathrm{c},h}}(\mathbf{q}) m_h \ . \tag{2.20}$$

The mass of the individual robot links is $m_h, h = 1, \ldots, n$, and $\mathbf{a}_{\mathrm{g}}$ denotes the vector of gravitational acceleration. The inertia tensor $\mathbf{T}_{\mathcal{W}}^{\mathcal{L}_{\mathrm{c},h}}$, the position $\mathbf{p}_{\mathcal{W}}^{\mathcal{L}_{\mathrm{c},h}}$, and the Jacobian $\mathbf{J}_{\mathcal{W}}^{\mathcal{L}_{\mathrm{c},h}}$, see also (2.13), are formulated in the frame $\mathcal{L}_{\mathrm{c},h}$. Each frame $\mathcal{L}_{\mathrm{c},h}, h = 1, \ldots, n$, is located at the center of mass of the individual robot link h, cf. (2.8). The dynamic model of the manipulator is derived using the EULER-LAGRANGE equations (2.18) and is rearranged as

$$\sum_{j=1}^{n} M_{hj}(\mathbf{q})\ddot{q}_j + \sum_{j=1}^{n}\sum_{k=1}^{n} C_{kjh}(\mathbf{q})\dot{q}_k\dot{q}_j + g_h(\mathbf{q}) = \tau_h + \tau_{\text{ext},h} \ , \quad h = 1, \dots, n \ , \tag{2.21}$$

with M_{hj} denoting the element in row h and column j of the positive definite mass matrix $\mathbf{M}(\mathbf{q})$, the CHRISTOFFEL symbols

$$C_{kjh} = \frac{1}{2}\left(\frac{\partial M_{hj}(\mathbf{q})}{\partial q_k} + \frac{\partial M_{hk}(\mathbf{q})}{\partial q_j} - \frac{\partial M_{kj}(\mathbf{q})}{\partial q_h}\right) \ , \tag{2.22}$$

and

$$g_h(\mathbf{q}) = \frac{\partial E_\text{V}(\mathbf{q})}{\partial q_h} \ . \tag{2.23}$$

Equation (2.21) is then rewritten with the vector of gravitational forces $\mathbf{g}^\text{T}(\mathbf{q}) = \begin{bmatrix} g_1(\mathbf{q}) & \cdots & g_h(\mathbf{q}) \end{bmatrix}$ and the joint acceleration $\ddot{\mathbf{q}}$ as rigid-body model in vector form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} + \boldsymbol{\tau}_\text{ext} \ , \tag{2.24}$$

with the CORIOLIS matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ comprising the elements, see, e.g., [35]

$$C_{hj}(\mathbf{q}, \dot{\mathbf{q}}) = \sum_{k=1}^{n} C_{kjh}\dot{q}_k \ . \tag{2.25}$$

The vector of generalized torques $\boldsymbol{\tau}^\text{T} = \begin{bmatrix} \tau_1 & \cdots & \tau_n \end{bmatrix}$ is the control input, and the external torques acting on the robot are denoted by $\boldsymbol{\tau}_\text{ext}$.

2.2 Manufacturing Process

In this work, a manufacturing process is specified by a continuous geometric manufacturing path on the workpiece and the process properties, including process DoF, redundant DoF, constraints, process tolerances, and process windows. This general concept of manufacturing processes is defined mathematically in this section.

Major parts of this section have been published in the author's work [10] and are adapted for this thesis.

2.2.1 Manufacturing Path

A continuous geometric manufacturing path is given as a sequence of poses $H_{\mathcal{P}}^{\mathcal{M}} = \{\mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}, k = 1, \ldots, n_{\mathrm{m}}\}$, where each pose $\mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}$ describes a manufacturing frame $\mathcal{M}$ along a path w.r.t. the workpiece frame $\mathcal{P}$, i.e., the desired tool pose during process execution. Hence, the manufacturing process is executed by moving the robotic tool, described by the TCP $\mathcal{T}$, along the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ while respecting the process properties defined in the next section. Note that the manufacturing path is assumed to be sufficiently smooth to generate continuous robot movements. Additionally, the spatial resolution is chosen according to the desired accuracy of the manufacturing result.

2.2.2 Process Properties

The Cartesian DoF of the robotic tool, which are needed to accomplish the manufacturing task, see [36], are called *process DoF*. Most processes demand all 6 DoF of the 3D Cartesian space, e.g., cutting and sewing. Some processes only require 5 DoF or less, and the remaining DoF is referred to as a *redundant process DoF* in this work. For example, the rotation axis of a rotating tool like a drill or a polishing disk is considered a redundant process DoF. Furthermore, some processes allow for deviations from the manufacturing path. In the following, the term *process window* refers to allowed tool deviations from the manufacturing path, which do not degrade the process quality, e.g., [36]. On the other hand, deviations that degrade the process quality to a tolerable extent but should be avoided if possible will be referred to as *process tolerances*. The process tolerances may be constrained additionally by defining a *tolerance band*. In the remainder of this work, the generic term *process properties* encompasses process tolerances, process windows, constraints, and redundant process DoF. Process tolerances and windows can be specified in one or multiple process DoF, i.e., allowing certain position deviations in specific Cartesian directions or orientation deviations w.r.t. specific axes.

Many industrial processes allow for process tolerances or process windows during process execution. Hence, the tool frame $\mathcal{T}$ may deviate from the manufacturing frame $\mathcal{M}$, see Section 2.2.1, to a certain predefined extent. For example, in a polishing process, the orientation of the robotic polishing tool may deviate slightly

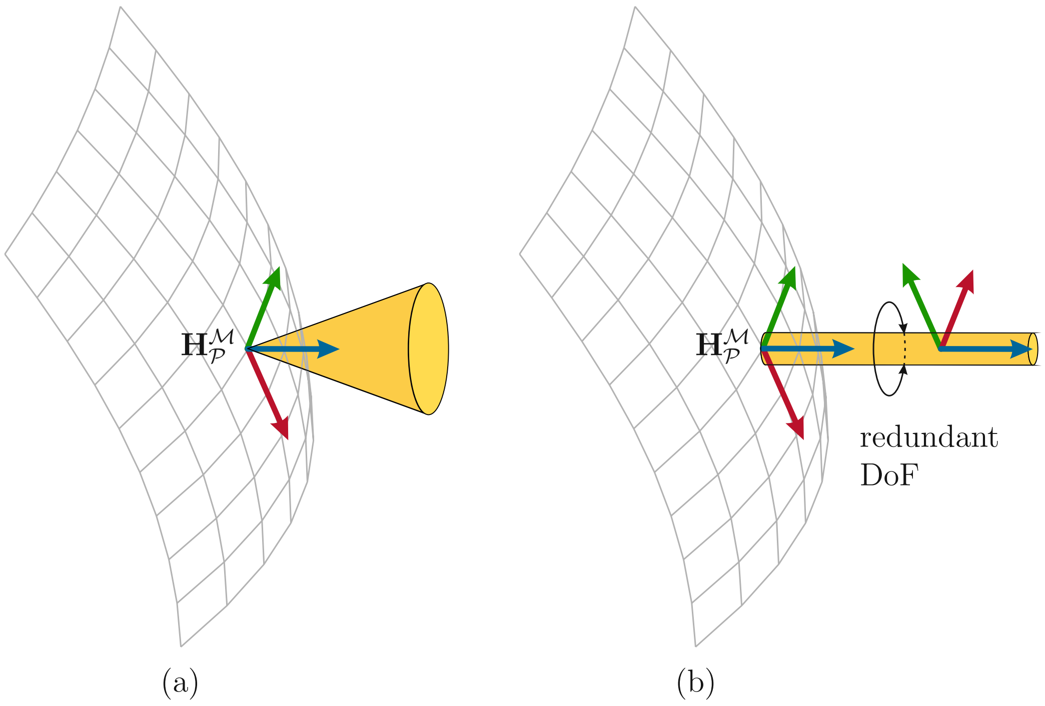
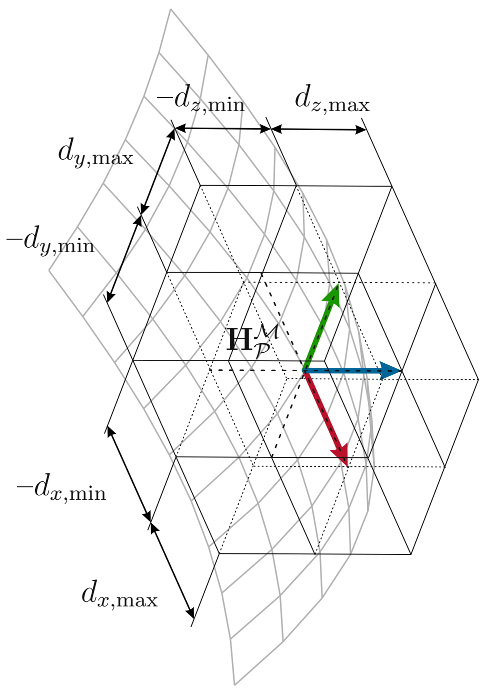
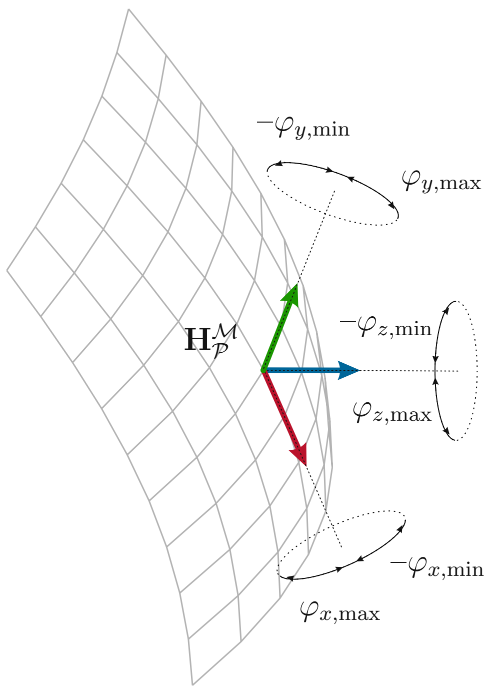

$\mathbf{H}_{\mathcal{P}}^{\mathcal{M}}$
$\mathbf{H}_{\mathcal{P}}^{\mathcal{M}}$
redundant
DoF
(a)
(b)

Figure 2.2: Tolerances for manufacturing processes (highlighted in yellow): (a) cutting process, (b) spray-painting process; adapted from [10].

from the surface normal, but its position should exactly follow the manufacturing path. Additionally, the rotating disk of the polishing tool represents a redundant process DoF around which the tool may be rotated arbitrarily. Similarly, in a robotic ultrasonic cutting process, the position of the knife must follow the given manufacturing path exactly, while the knife tilt may vary within a certain process window, which results in a cone of admissible orientations, see Fig. 2.2a. Note that no redundant DoF exists in a cutting process with a knife. In contrast, in a spray-painting process, the spray nozzle has to perfectly align with the surface normal vector with the lateral position coordinates matching exactly the manufacturing frame $\mathcal{M}$, while the distance to the object's surface should stay within a given tolerance band. Furthermore, if a rotationally symmetric spray jet is used, the manufacturing process becomes invariant to the rotation of the spray jet around the surface normal vector. Thus, in the latter case, a redundant DoF is present in the process, see Fig. 2.2b.



(a)



$-\varphi_{y,\mathrm{min}}$
$\varphi_{y,\mathrm{max}}$
$\mathbf{H}_{\mathcal{P}}^{\mathcal{M}}$
$-\varphi_{z,\mathrm{min}}$
$\varphi_{z,\mathrm{max}}$
$-\varphi_{x,\mathrm{min}}$
$\varphi_{x,\mathrm{max}}$

(b)

Figure 2.3: Tolerance bands and process windows: (a) displacements, (b) orientation deviations of the tool frame $\mathcal{T}$ w.r.t. the manufacturing frame $\mathcal{M}$; adapted from [10].

The process windows or tolerance bands of the process DoF are specified by the minimum and maximum allowed displacements, see Fig. 2.3a,

$$\mathbf{d}_{\mathcal{M},\mathrm{min}}^{\mathcal{T}} = \begin{bmatrix} d_{x,\mathrm{min}} \\ d_{y,\mathrm{min}} \\ d_{z,\mathrm{min}} \end{bmatrix}, \qquad \mathbf{d}_{\mathcal{M},\mathrm{max}}^{\mathcal{T}} = \begin{bmatrix} d_{x,\mathrm{max}} \\ d_{y,\mathrm{max}} \\ d_{z,\mathrm{max}} \end{bmatrix}, \tag{2.26}$$

and the minimum and maximum rotations in terms of Roll-Pitch-Yaw angles, defined in Section 2.1.1, see Fig. 2.3b,

$$\boldsymbol{\varphi}_{\mathcal{M},\mathrm{min}}^{\mathcal{T}} = \begin{bmatrix} \varphi_{x,\mathrm{min}} \\ \varphi_{y,\mathrm{min}} \\ \varphi_{z,\mathrm{min}} \end{bmatrix}, \qquad \boldsymbol{\varphi}_{\mathcal{M},\mathrm{max}}^{\mathcal{T}} = \begin{bmatrix} \varphi_{x,\mathrm{max}} \\ \varphi_{y,\mathrm{max}} \\ \varphi_{z,\mathrm{max}} \end{bmatrix} \tag{2.27}$$

of the tool frame $\mathcal{T}$ w.r.t. the manufacturing frame $\mathcal{M}$. Note that the simple (linear) box constraints (2.26) and (2.27) may also be replaced by nonlinear constraints, e.g., circular or spherical distance constraints.

Chapter 3 Flexible Robotic Manufacturing based on User Inputs

This chapter presents an automatic workflow to generate and execute a manufacturing process on different 3D workpieces based on 2D user inputs. For this, a flexible path-planning algorithm and a tailored control concept are developed. The workflow is demonstrated for a drawing process in a laboratory environment, which is representative of other industrial manufacturing processes, e.g., welding, cutting, or milling.

First, a literature review is presented for automatic user-based robot programming for industrial robots. Second, two path projection methods to map the user input on a workpiece are shown, and the robot trajectory generation is explained. In order to execute the drawing process with an industrial robot, two control concepts, i.e., pure motion control and hybrid force/motion control, are presented and compared. Finally, the whole automatic path-planning and execution pipeline is demonstrated in multiple experiments. Preliminary work was developed in the author's diploma thesis [37] and in a subsequent diploma thesis [38] supervised by the author.

Major parts of this chapter have been published in the author's works [7, 9, 11] and are adapted for this thesis.

3.1 Literature Review

In the literature, the automatic generation of robot programs from user input has been considered in several works. In many publications, a drawing process is a demonstration example to show the capabilities of the proposed algorithms. In most cases, the considered task can be easily adapted to conceptually similar different manufacturing processes, e.g., engraving, laser cutting, or painting.

Major parts of this section have been published in the author's works [9, 11] and are adapted for this thesis.

In [39, 40], an edge detection algorithm is used to extract features from portraits of humans, which are then drawn on a flat canvas by a humanoid robot. In order to perform this task, a task-space path is generated first, for which a joint-space path is computed using inverse kinematics and then executed by the robot. Furthermore, [41] considers an industrial robot drawing on a flat whiteboard, where robot programs are automatically generated to draw the edges and important features of ordinary photos and images. Similarly, in [42, 43, 44, 45], robotic drawing has been presented emphasizing artistic and stylistic algorithms for path generation. Drawing on 2.5D (terrain-like) objects is demonstrated in [46], where a Kuka LBR iiwa 7 R800 is utilized. The drawing is interpolated with Bézier curves, and an impedance control is used to draw on unknown non-planar objects. In [47], a force/torque sensor is mounted on the end-effector of the drawing robot. With this information, the robot can draw on objects whose relative position to the robot is unknown. Even on 3D surfaces with small curvature, drawing is successful because the pen orientation is controlled to remain normal to the surface. Because the force information is only available during the drawing process and the object's geometry is unknown, the projection can cause distortions and is unpredictable. Most of the works discussed so far focus on the artistic aspects of robotic drawing on planar surfaces, but the manufacturing aspects of 3D objects are not considered in detail.

Automatic path planning to work directly on 3D workpieces has been examined for processes like spraying [24], polishing [28], and draping [36]. In those works, algorithms generate 3D paths automatically based on the CAD data without further user input. In most manufacturing processes, the workpiece geometry is already known, e.g., [48], or state-of-the-art 3D scanners and algorithms can be employed to obtain the shape of the workpiece [49].

Another way of planning and executing robot motions from user input is interactive teach-in methods. A state-of-the-art concept based on an instrumented pointing tool was developed by Wandelbots [50]. With the so-called *TracePen*,

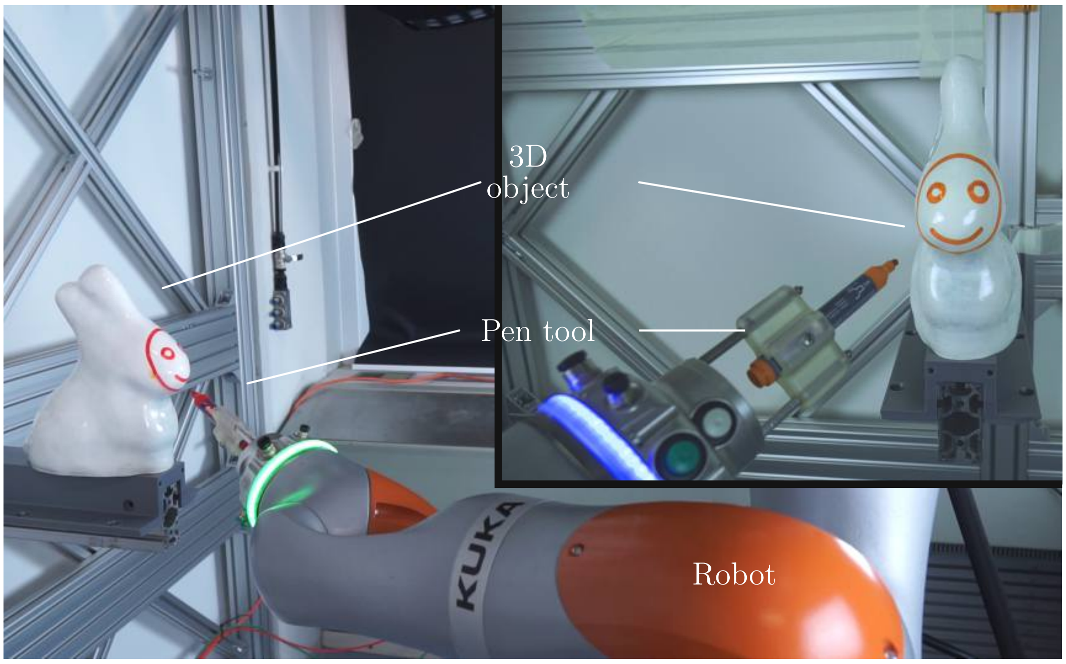


Figure 3.1: Experimental setup of the drawing process with a Kuka LBR iiwa 14 R820; adapted from [11].

robot motions are demonstrated by the user, and a robot program is generated automatically. In [51], an instrumented tool records the position, orientation, and tension force of a rope winding task. This demonstrated trajectory is then executed by an industrial robot. Inspired by the gaming industry, a representation of a robotic environment in virtual/mixed reality is used to teach an industrial task [52, 53]. However, those teach-in methods need a trained person to generate the robot program, and this is especially cumbersome for frequently changing user inputs or small lot sizes.

A different way to customize the visual design of products is to use inkjet printer heads mounted on the industrial robot's end-effector to perform 2D printing on 3D surfaces directly. The printer head prints custom designs on shoes [54] or cars [55] and thus provides easy customization. In this process, the robot motion is generated for each product only once because the robot trajectory stays the same for printing arbitrary 2D input patterns on the same surface area. The print quality on curved surfaces increases for small 2D input patterns because the print head has to be at a constant distance from the product.

This chapter aims to demonstrate customization of products in an industrial manufacturing process. To this end, a robotic drawing task based on user inputs for known arbitrary 3D objects is presented, see Fig. 3.1, which is the main contribution of this chapter. This process demands the robotic system to achieve the required flexibility and replication accuracy on the 3D surface. A user specifies the drawing and its exact location, size, and orientation on the object. Subsequently, the robot trajectory is planned based on accurate mapping, and the drawing procedure is executed. In order to improve the drawing quality, the contact force of the pen is adjusted using a hybrid force/torque controller, which is further extended to control the position and orientation of the pen simultaneously.

This automatic pipeline applies to different manufacturing processes. Robot-assisted additive manufacturing processes, like material extrusion, e.g., fused deposition modeling (FDM), material jetting, directed energy deposition as shown in [56], and spray painting, e.g. [25], can also be performed using the proposed automatic trajectory planning and robot execution pipeline. Further examples are automated milling and laser engraving, where a user input pattern has to be engraved into large 3D objects. This pipeline can also be employed for subtractive processes, including laser or ultrasonic cutting.

3.2 Path Projection and Trajectory Generation

In this section, two path projection methods to transfer a user-provided 2D input pattern onto a 3D object are presented, and the robot trajectory generation is explained. First, a least-squares conformal mapping (LSCM) is introduced and explained in detail. Next, a simple parallel projection is presented for comparison. Finally, the robot trajectory is generated based on the result of a projection method.

Major parts of this section have been published in the author's works [9, 11] and are adapted for this thesis.

3.2.1 Path Projection using Least-Squares Conformal Mapping

A flow chart illustrating the individual steps of the path projection with LSCM is shown in Fig. 3.2. In order to handle meshes of high complexity, the mesh of the

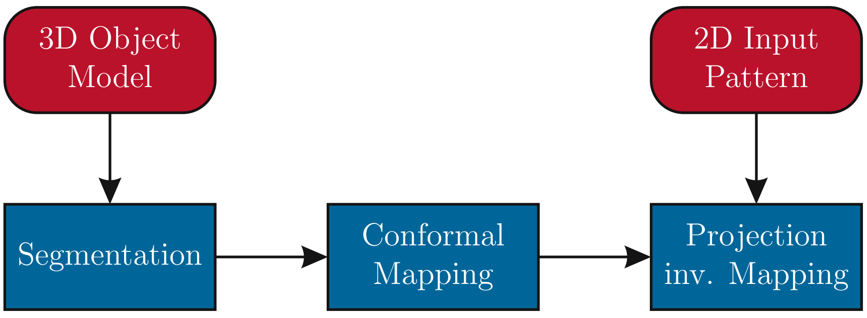


Figure 3.2: Flow chart of LSCM to project the 2D input pattern; adapted from [9, 11].

3D object is first decomposed into multiple mesh segments. This preparation step limits the local distortions of the transferred 2D input pattern in the subsequent workflow. Next, the mesh segments are flattened using a LSCM approach. The 2D input pattern is projected on the flattened surface, and then an inverse mapping transfers the pattern back onto the 3D object, which is discussed at the end of this section. Preliminary work was developed in the diploma thesis [38] which was supervised by the author. The methods in this subsection are summarized from [57, 58] and are tightly integrated into the path-planning workflow. Subsequently, these methods are utilized to compare the two projection methods, i.e., the LSCM-based and parallel projection.

Segmentation

As a first preparation step, the mesh segmentation algorithm presented in [57] is applied to the mesh of the 3D object before flattening the individual mesh segments. This way, distortions in the projected 2D paths are minimized, and the replication accuracy on the 3D object is improved, see [38].

The algorithm [57] first finds the boundaries between two segments by computing the so-called sharpness criterion

$$w_{i,j} = \arccos\left(\frac{\mathbf{n}_i^{\mathrm{T}}\mathbf{n}_j}{\|\mathbf{n}_i\|_2\,\|\mathbf{n}_j\|_2}\right) \tag{3.1}$$

for each edge of the mesh, where $\mathbf{n}_i$ and $\mathbf{n}_j$ denote the normal vectors of two adjacent triangles T_i and T_j, respectively, see [59]. Thereby, $\|\cdot\|_2$ denotes the Euclidean 2-norm of the vector $\cdot$. Edges for which the angle $w_{i,j}$ in (3.1) is above

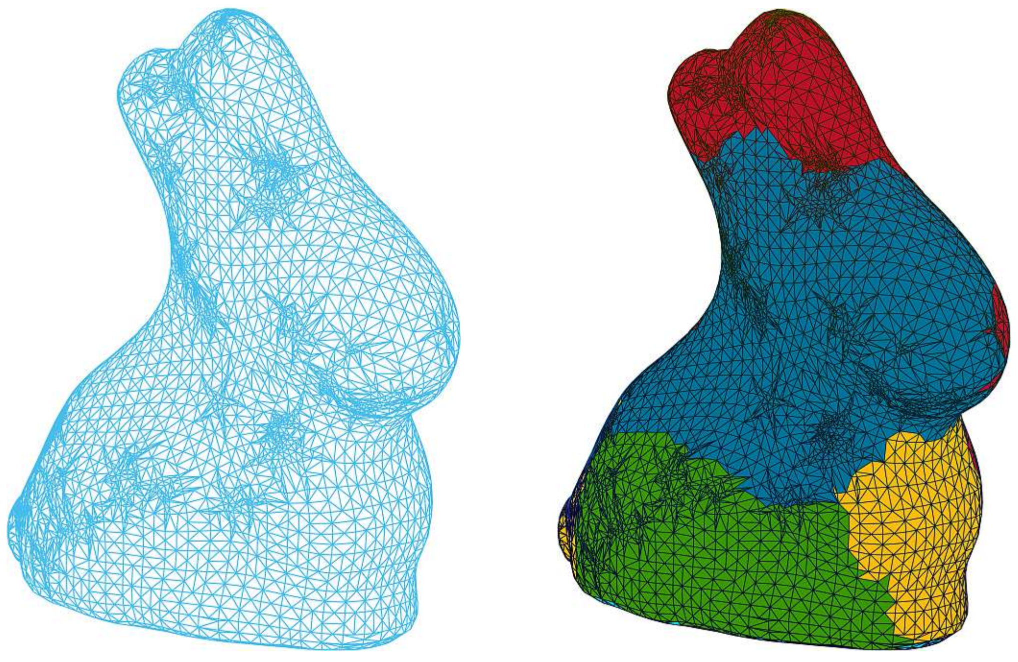

(a) (b)

Figure 3.3: Segmentation of a 3D object before flattening: (a) Mesh of the 3D object, (b) Result of the segmentation using [57]; adapted from [9, 11].

a particular threshold value are combined to feature curves. Next, the triangles with the maximum geodesic distance to a feature curve are determined. These triangles are subsequently used as seeds for a region-growing algorithm to obtain the individual mesh segments. An example of a segmented 3D object is depicted in Fig. 3.3.

Least-Squares Conformal Mapping

The LSCM approach [57] is used to flatten the mesh segments of the 3D object into 2D meshes. A locally isotropic conformal map $\mathrm{X} : (u, v) \mapsto (x, y)$ preserves the local angles and, therefore, the shape of small figures but generally not their size, see [38].

In this section, the considered mesh segment S consists of n_t triangles T_i, $i = 1, \dots, n_\mathrm{t}$, and n_v vertices. For each triangle T_i, new coordinates $(x_{i,1}, y_{i,1})$, $(x_{i,2}, y_{i,2})$, $(x_{i,3}, y_{i,3})$ of the vertices are computed, with a local orthonormal basis located in the vertex $(x_{i,1}, y_{i,1})$ and the x-axis aligned with the edge to $(x_{i,2}, y_{i,2})$. Then, the conformal mapping for a single triangle fulfills the condition

$$\frac{\partial \mathrm{X}}{\partial u} - \mathrm{i}\frac{\partial \mathrm{X}}{\partial v} = 0 \ , \tag{3.2}$$

with the complex number $\mathrm{X} = x + \mathrm{i}y$, where i denotes the imaginary unit. The inverse conformal mapping $\mathrm{U} : (x, y) \mapsto (u, v)$ reads as [60]

$$\frac{\partial \mathrm{U}}{\partial x} + \mathrm{i}\frac{\partial \mathrm{U}}{\partial y} = 0 \ , \tag{3.3}$$

which is a formulation of the Cauchy-Riemann equations

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial u_i}{\partial x_i} \\ \frac{\partial u_i}{\partial y_i} \end{bmatrix} = \begin{bmatrix} \frac{\partial v_i}{\partial x_i} \\ \frac{\partial v_i}{\partial y_i} \end{bmatrix} \tag{3.4}$$

for complex numbers $\mathrm{U} = u + \mathrm{i}v$, see [61]. Since (3.3) cannot be satisfied for all triangles T_i of the mesh segment S simultaneously, the minimization problem

$$\min_{\mathrm{U}} \sum_{T_i \in S} f_{\mathrm{m}}(T_i) \tag{3.5a}$$

$$f_{\mathrm{m}}(T_i) = 2 \int_{T_i} \left| \frac{\partial \mathrm{U}}{\partial x} + \mathrm{i}\frac{\partial \mathrm{U}}{\partial y} \right|^2 \mathrm{d}A_i \tag{3.5b}$$

is formulated, where $|\cdot|$ denotes the magnitude of the complex number $\cdot$ and A_i is the area of the triangle T_i, $i = 1, \dots, n_{\mathrm{t}}$.

The gradients in (3.5b) for a single triangle T_i read as

$$\begin{bmatrix} \frac{\partial u_i}{\partial x_i} \\ \frac{\partial u_i}{\partial y_i} \end{bmatrix} = \frac{1}{2A_i}\mathbf{W} \begin{bmatrix} u_{i,1} \\ u_{i,2} \\ u_{i,3} \end{bmatrix} \ , \tag{3.6a}$$

$$\begin{bmatrix} \frac{\partial v_i}{\partial x_i} \\ \frac{\partial v_i}{\partial y_i} \end{bmatrix} = \frac{1}{2A_i}\mathbf{W} \begin{bmatrix} v_{i,1} \\ v_{i,2} \\ v_{i,3} \end{bmatrix} \ , \tag{3.6b}$$

with

$$\mathbf{W} = \begin{bmatrix} y_{i,2} - y_{i,3} & y_{i,3} - y_{i,1} & y_{i,1} - y_{i,2} \\ x_{i,3} - x_{i,2} & x_{i,1} - x_{i,3} & x_{i,2} - x_{i,1} \end{bmatrix} \tag{3.7}$$

and

$$2A_i = (x_{i,1}y_{i,2} - y_{i,1}x_{i,2}) + (x_{i,2}y_{i,3} - y_{i,2}x_{i,3}) + (x_{i,3}y_{i,1} - y_{i,3}x_{i,1}) \ . \tag{3.8}$$

The compact formulation

$$\frac{\partial \mathrm{U}}{\partial x} + \mathrm{i}\frac{\partial \mathrm{U}}{\partial y} = \frac{\mathrm{i}}{2A_i}\begin{bmatrix} W_{i,1} & W_{i,2} & W_{i,3}\end{bmatrix}\begin{bmatrix} U_{i,1} \\ U_{i,2} \\ U_{i,3}\end{bmatrix} = 0 \tag{3.9}$$

is found with the coordinates of the vertices $U_{i,j} = u_{i,j} + \mathrm{i}v_{i,j}, j = 1, 2, 3,$ of the corresponding triangle $T_i, i = 1, \ldots, n_\mathrm{t}$, using (3.6) and

$$W_{i,1} = (x_{i,3} - x_{i,2}) + \mathrm{i}(y_{i,3} - y_{i,2}) \ , \tag{3.10a}$$
$$W_{i,2} = (x_{i,1} - x_{i,3}) + \mathrm{i}(y_{i,1} - y_{i,3}) \ , \tag{3.10b}$$
$$W_{i,3} = (x_{i,2} - x_{i,1}) + \mathrm{i}(y_{i,2} - y_{i,1}) \ . \tag{3.10c}$$

Inserting (3.9) into the minimization problem (3.5), the optimization problem for the whole mesh segment S with all triangles $T_i \in S$ is reformulated as

$$\min_{\mathbf{U}} f_\mathrm{m}(\mathbf{U}) = \min_{\mathbf{U}} \sum_{T_i \in S} f_\mathrm{m}(T_i) \tag{3.11a}$$
$$f_\mathrm{m}(T_i) = \left| \frac{1}{2A_i} \begin{bmatrix} W_{i,1} & W_{i,2} & W_{i,3}\end{bmatrix}\begin{bmatrix} U_{i,1} \\ U_{i,2} \\ U_{i,3}\end{bmatrix} \right|^2 \ , \tag{3.11b}$$

with the vector of all vertices $\mathbf{U}^\mathrm{T} = \begin{bmatrix} U_1 & \cdots & U_{n_\mathrm{v}}\end{bmatrix}$. Note that a single vertex may be contained in multiple triangles.

In order to solve the optimization problem (3.11), the cost function $f_\mathrm{m}(\mathbf{U})$ is written as

$$f_\mathrm{m}(\mathbf{U}) = \mathbf{U}^\mathrm{H}\mathbf{N}^\mathrm{H}\mathbf{N}\mathbf{U} \ , \tag{3.12}$$

with the sparse matrix $\mathbf{N} \in \mathbb{R}^{n_\mathrm{t} \times n_\mathrm{v}}$ and the Hermitian conjugation denoted by the superscript $\cdot^\mathrm{H}$. The elements of $\mathbf{N}$ are calculated as

$$N_{ij} = \begin{cases} \frac{W_{i,j}}{\sqrt{2A_i}} & \text{if vertex } U_j \text{ belongs to triangle } T_i \\ & \text{(consisting of the vertices } U_j, U_k, U_l) \\ 0 & \text{otherwise ,} \end{cases} \tag{3.13}$$

with $W_{i,j} = (x_{i,l} - x_{i,k}) + \mathrm{i}(y_{i,l} - y_{i,k})$, cf. (3.10). Equation (3.12) is rewritten in the quadratic form

$$f_\mathrm{m}(\mathbf{U}) = \|\mathbf{N}\mathbf{U}\|_2^2 = \|\mathbf{N}_\mathrm{f}\mathbf{U}_\mathrm{f} + \mathbf{N}_\mathrm{p}\mathbf{U}_\mathrm{p}\|_2^2 \ , \tag{3.14}$$

with $\mathbf{U}^\mathrm{T} = \left[\mathbf{U}_\mathrm{f}^\mathrm{T} \ \mathbf{U}_\mathrm{p}^\mathrm{T}\right]$, where $\mathbf{U}_\mathrm{f} = \mathbf{U}_\mathrm{f}^\mathrm{R} + \mathrm{i}\mathbf{U}_\mathrm{f}^\mathrm{I}$ denotes the vector of free (unknown) and $\mathbf{U}_\mathrm{p} = \mathbf{U}_\mathrm{p}^\mathrm{R} + \mathrm{i}\mathbf{U}_\mathrm{p}^\mathrm{I}$ of pinned (given) vertex coordinates. As suggested in the original work [57], the two vertices with the maximum distance to each other are pinned for each mesh segment. This way, (3.11) is reformulated as a least-squares problem in the unknown variables

$$\boldsymbol{\gamma}^\mathrm{T} = \left[\left(\mathbf{U}_\mathrm{f}^\mathrm{R}\right)^\mathrm{T} \quad \left(\mathbf{U}_\mathrm{f}^\mathrm{I}\right)^\mathrm{T}\right] \tag{3.15}$$

and

$$f_\mathrm{m}(\boldsymbol{\gamma}) = \|\boldsymbol{\alpha}\boldsymbol{\gamma} - \boldsymbol{\beta}\|_2^2 \ , \tag{3.16}$$

with

$$\boldsymbol{\alpha} = \begin{bmatrix} \mathbf{N}_\mathrm{f}^\mathrm{R} & -\mathbf{N}_\mathrm{f}^\mathrm{I} \\ \mathbf{N}_\mathrm{f}^\mathrm{I} & \mathbf{N}_\mathrm{f}^\mathrm{R} \end{bmatrix} \ , \tag{3.17a}$$

$$\boldsymbol{\beta} = -\begin{bmatrix} \mathbf{N}_\mathrm{p}^\mathrm{R} & -\mathbf{N}_\mathrm{p}^\mathrm{I} \\ \mathbf{N}_\mathrm{p}^\mathrm{I} & \mathbf{N}_\mathrm{p}^\mathrm{R} \end{bmatrix} \begin{bmatrix} \mathbf{U}_\mathrm{p}^\mathrm{R} \\ \mathbf{U}_\mathrm{p}^\mathrm{I} \end{bmatrix} \ . \tag{3.17b}$$

In (3.14) and (3.17), $\mathbf{N}_\mathrm{f}$ and $\mathbf{N}_\mathrm{p}$ denote the sparse index matrices for the free and pinned vertices, respectively, and the superscripts $\cdot^\mathrm{R}$ and $\cdot^\mathrm{I}$ in (3.15) and (3.17) refer to the real and imaginary part of the complex-valued vectors and matrices.

Finally, the least-squares problem with the cost function (3.16) is solved in MATLAB using a numerical solver to find the coordinates of the free vertices $\mathbf{U}_\mathrm{f}$ under the inverse conformal mapping U. For a more detailed explanation of the mapping algorithm, the reader is referred to [57].

2D Path Projection and Inverse Mapping

In this section, the user-provided 2D input pattern is transferred to a user-specified location and size onto the 3D object. To this end, the 2D input pattern is first projected on the corresponding flattened mesh segment. Subsequently, it is mapped back onto the 3D object by the inverse conformal mapping U using barycentric coordinates [58], see [38].

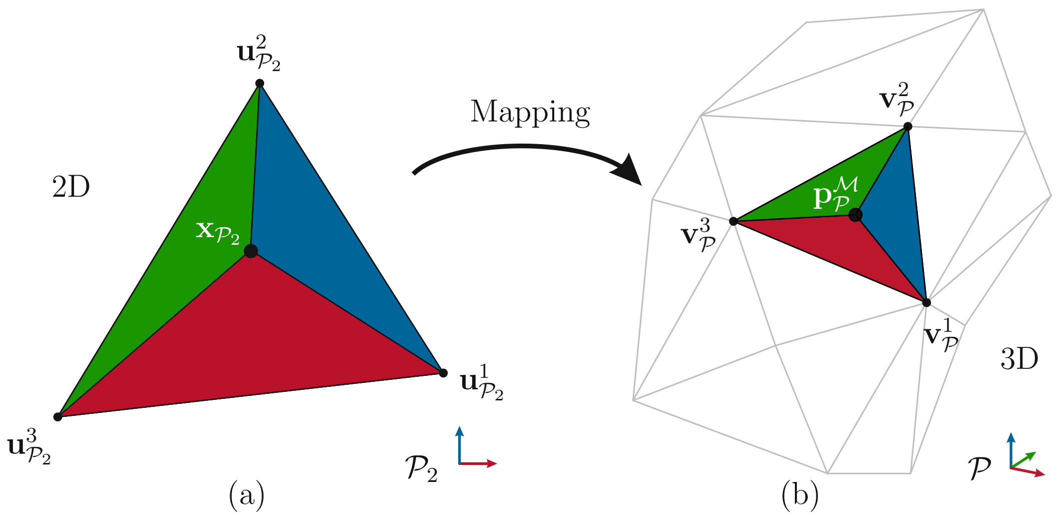
The 2D input pattern is created by the user with a touchscreen, digitizer, or mouse interface and comprises the discrete path points $(\mathbf{x}_{\mathcal{W}_2,k})^{\mathrm{T}} = \left[x_{\mathcal{W}_2,k}\ y_{\mathcal{W}_2,k}\right]$, $k = 1,\dots,n_{\mathrm{m}}$, i.e., the individual discrete path point k w.r.t. a 2D world frame $\mathcal{W}_2$. Furthermore, the user defines the location $\left[\Delta x\ \Delta y\right]^{\mathrm{T}}$, rotation θ, and scale ρ of the desired pattern on the flattened mesh segment. Note that this information can also be calculated from the 3D object by selecting the vertices in which the 2D pattern should be located. The resulting path points of the 2D path $\mathbf{x}_{\mathcal{P}_2,k}$ w.r.t. a 2D workpiece frame $\mathcal{P}_2$, $k = 1,\dots,n_{\mathrm{m}}$, are computed using the transformation

$$\mathbf{x}_{\mathcal{P}_2,k} = \rho \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \mathbf{x}_{\mathcal{W}_2,k} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} , \tag{3.18}$$

see [29]. If the 2D path covers multiple mesh segments, those segments are combined and flattened again.

Next, the correspondence between the flattened and the 3D mesh segment is established via the triangles of both meshes. For each 2D path point $\mathbf{x}_{\mathcal{P}_2,k}$, $k = 1,\dots,n_{\mathrm{m}}$, from the flattened mesh segment, the associated triangle T_k is determined. This is performed using the efficient bin-based algorithm published in [58]; see also [57]. Note that a triangle in the flattened 2D mesh segment directly corresponds to the 3D representation of this triangle.

Finally, the 2D path points $\mathbf{x}_{\mathcal{P}_2,k}$ are mapped onto the 3D mesh segment using barycentric coordinates inside the corresponding triangles T_k, $k = 1,\dots,n_{\mathrm{m}}$. In general, barycentric coordinates map between two arbitrary triangles based on the corresponding vertices, see, e.g., [62]. A given point $\mathbf{x}_{\mathcal{P}_2,k}$ is mapped from one triangle in the workpiece frame $\mathcal{P}_2$ in 2D space, described with the positions of



$\mathbf{u}^2_{\mathcal{P}_2}$
Mapping
2D
$\mathbf{x}_{\mathcal{P}_2}$
$\mathbf{u}^1_{\mathcal{P}_2}$
$\mathbf{u}^3_{\mathcal{P}_2}$
$\mathcal{P}_2$
(a)
$\mathbf{v}^2_{\mathcal{P}}$
$\mathbf{p}^{\mathcal{M}}_{\mathcal{P}}$
$\mathbf{v}^3_{\mathcal{P}}$
$\mathbf{v}^1_{\mathcal{P}}$
3D
$\mathcal{P}$
(b)

Figure 3.4: Mapping from the triangle of the flattened object to the 3D object: (a) Triangle on the flattened object, (b) Triangle on the 3D object; adapted from [9, 11], cf. [58].

the vertices $(\mathbf{u}^1_{\mathcal{P}_2}, \mathbf{u}^2_{\mathcal{P}_2}, \mathbf{u}^3_{\mathcal{P}_2})$ with $(\mathbf{u}^i_{\mathcal{P}_2})^{\mathrm{T}} = \left[x^i_{\mathcal{P}_2}\ y^i_{\mathcal{P}_2}\right]$, $i = 1, 2, 3$, to the triangle in the workpiece frame $\mathcal{P}$ in 3D space, described with the positions of the vertices $(\mathbf{v}^1_{\mathcal{P}}, \mathbf{v}^2_{\mathcal{P}}, \mathbf{v}^3_{\mathcal{P}})$ with $(\mathbf{v}^i_{\mathcal{P}})^{\mathrm{T}} = \left[x^i_{\mathcal{P}}\ y^i_{\mathcal{P}}\ z^i_{\mathcal{P}}\right]$, $i = 1, 2, 3$, in the form, see Fig. 3.4, [58]

$$\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k} = \breve{A}_1 \mathbf{v}^1_{\mathcal{P}} + \breve{A}_2 \mathbf{v}^2_{\mathcal{P}} + \breve{A}_3 \mathbf{v}^3_{\mathcal{P}} \ , \quad k = 1, \dots, n_{\mathrm{m}} \ , \tag{3.19}$$

with

$$\breve{A}_1 = \frac{f_{\mathrm{a}}(\mathbf{x}_{\mathcal{P}_2,k}, \mathbf{u}^2_{\mathcal{P}_2}, \mathbf{u}^3_{\mathcal{P}_2})}{f_{\mathrm{a}}(\mathbf{u}^1_{\mathcal{P}_2}, \mathbf{u}^2_{\mathcal{P}_2}, \mathbf{u}^3_{\mathcal{P}_2})} \ , \tag{3.20a}$$

$$\breve{A}_2 = \frac{f_{\mathrm{a}}(\mathbf{u}^1_{\mathcal{P}_2}, \mathbf{x}_{\mathcal{P}_2,k}, \mathbf{u}^3_{\mathcal{P}_2})}{f_{\mathrm{a}}(\mathbf{u}^1_{\mathcal{P}_2}, \mathbf{u}^2_{\mathcal{P}_2}, \mathbf{u}^3_{\mathcal{P}_2})} \ , \tag{3.20b}$$

$$\breve{A}_3 = \frac{f_{\mathrm{a}}(\mathbf{u}^1_{\mathcal{P}_2}, \mathbf{u}^2_{\mathcal{P}_2}, \mathbf{x}_{\mathcal{P}_2,k})}{f_{\mathrm{a}}(\mathbf{u}^1_{\mathcal{P}_2}, \mathbf{u}^2_{\mathcal{P}_2}, \mathbf{u}^3_{\mathcal{P}_2})} \ . \tag{3.20c}$$

In (3.20), the function $f_{\mathrm{a}}(\cdot, \cdot, \cdot)$ calculates the area of the enclosed triangle. Using (3.19), all 3D path points $\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k}$, $k = 1, \dots, n_{\mathrm{m}}$, are determined. For each 3D path point, a coordinate frame on the surface with the origin $\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k}$ and the orientation $\mathbf{R}^{\mathcal{M}}_{\mathcal{P},k}$ is constructed such that the z-axis is parallel to the local surface normal vector. The remaining orientation results from a suitable reference orientation,

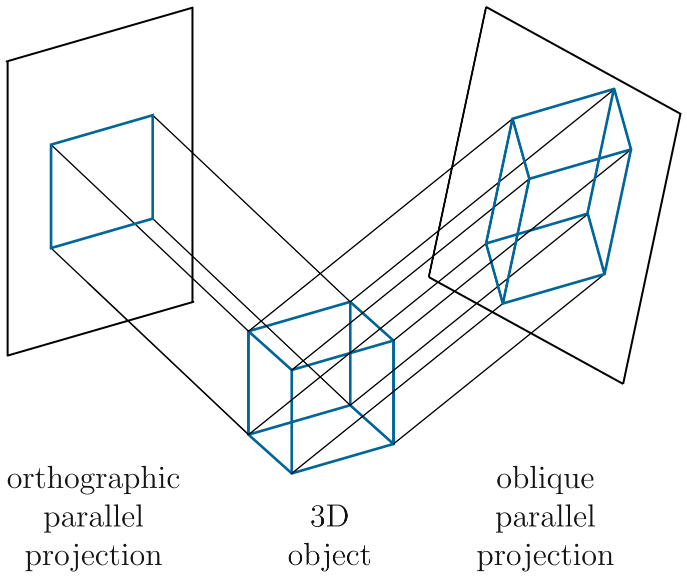


Figure 3.5: Examples of parallel projections from a 3D object to a 2D plane; adapted from [11], cf. [63].

e.g., the x-axis is set horizontally to the floor, and the y-axis is computed to obtain a right-oriented coordinate system.

3.2.2 Parallel Path Projection

Parallel projection, a simple method serving as comparison approach for performance evaluation in the experimental results, is explained in this section. Parallel projection is mainly used to project 3D objects on 2D planes by projecting the points along parallel projection rays, see Fig. 3.5. If the 2D plane is perpendicular to the projection rays, this parallel projection is called *orthographic* and otherwise *oblique*, see [63].

In this work, the inverse orthographic projection is needed, i.e., the 2D pattern of the user input has to be mapped to the 3D object to obtain the 3D path points $\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k}$ in the workpiece frame $\mathcal{P}$. Therefore, the points of the user input $\mathbf{x}_{\mathcal{W}_2,k}$ are given in a 2D plane. This 2D plane can be defined in front of the 3D object by the user or automatically computed based on the mean value of the normal vectors of the corresponding triangles. Then parallel projection rays perpendicular to this 2D plane are generated from each position $\mathbf{x}_{\mathcal{W}_2,k}$. At the intersection points of the parallel rays with the surface of the 3D object, the 3D path points $\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k}$

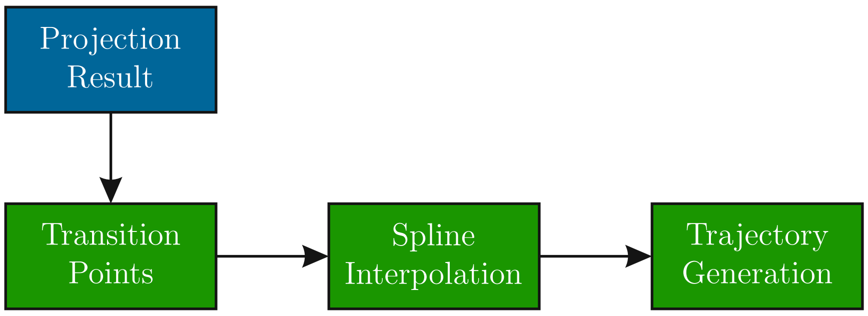


Figure 3.6: Flow chart of generating a robot trajectory based on the projection result; adapted from [9, 11].

corresponding to the 2D input $\mathbf{x}_{\mathcal{W}_2,k}$ are found. The intersection points can be computed with, e.g., the *Ray-Triangle Intersection* algorithm [64]. Analogous to the LSCM, local rotation matrices $\mathbf{R}^{\mathcal{M}}_{\mathcal{P},k}$ are generated for each 3D path point $\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k}$ to construct the manufacturing frames $\mathbf{H}^{\mathcal{M}}_{\mathcal{P},k}$, see Section 2.2.1. The direction of the parallel projection rays is used as the z-axis of this frame. The remaining orientations are again computed based on a reference orientation. Note that the orientations of all path points $\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k}$ are equal since all projection rays are parallel due to the used parallel projection, cf. [11].

3.2.3 Cartesian Robot Trajectory

The projection result from Subsections 3.2.1 and 3.2.2 is described by the sequence of 3D path points $\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k}$, $k = 1, \ldots, n_{\mathrm{m}}$ located on the surface of the 3D object and the corresponding orientations $\mathbf{R}^{\mathcal{M}}_{\mathcal{P},k}$. Along this 3D path, the robot has to draw the user-defined pattern. The resulting sequence of manufacturing poses is composed of the path points and orientations in the form

$$H^{\mathcal{M}}_{\mathcal{P}} = \left\{ \mathbf{H}^{\mathcal{M}}_{\mathcal{P},k} = \begin{bmatrix} \mathbf{R}^{\mathcal{M}}_{\mathcal{P},k} & \mathbf{p}^{\mathcal{M}}_{\mathcal{P},k} \\ \mathbf{0} & 1 \end{bmatrix}, \quad k = 1, \ldots, n_{\mathrm{m}} \right\} . \tag{3.21}$$

A flow chart of the necessary computation steps is shown in Fig. 3.6. If a user-provided 2D input pattern contains multiple disconnected path segments, additional transition points are added to the sequence of 3D path points. Between two segments, the robot retracts the pen from the surface by performing a linear motion. Subsequently, the robot moves from the end point of one segment to the starting point of the next path segment and approaches the surface again. The

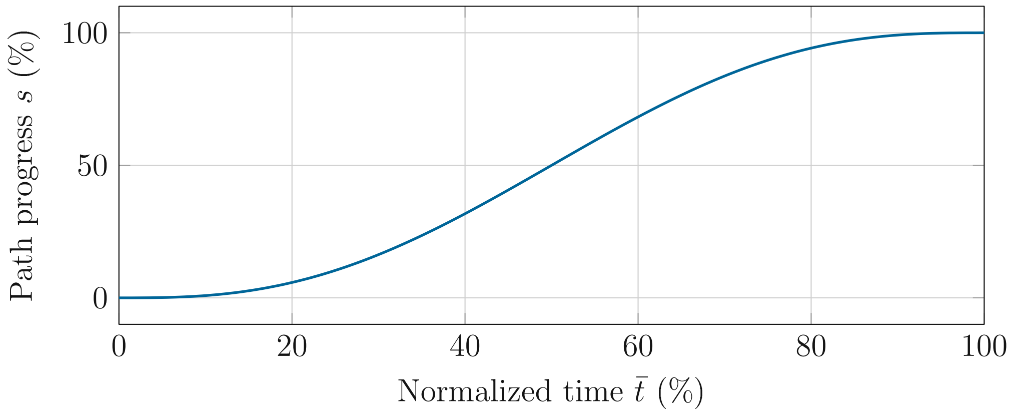


Figure 3.7: Path progress of the spline curve for a single segment; adapted from [7, 9].

individual sequences of 3D path points are interpolated according to [65], where a 5th-order polynomial is used for the time parametrization, see Fig. 3.7. This way, the drawing process of the robot starts and ends smoothly, see [38].

3.3 Control Concept

In order to realize the robotic drawing process with a pen on the 3D object, the robot executes the generated trajectory using a suitable control concept, which is detailed in this section. First, a standard task-space controller is described to control the motion of the pen on the surface of the 3D object. Second, contact force estimation [66] is introduced. Third, a hybrid force/motion controller is adapted from [29, 67] to control the contact force during the drawing process and improve the robotic drawing task. The standard task-space controller and the hybrid force/motion controller compute a torque $\boldsymbol{\tau}_1$, added to the torque $\boldsymbol{\tau}_2$ generated from a subsequently presented null-space controller. This results in the final control torque $\boldsymbol{\tau} = \boldsymbol{\tau}_1 + \boldsymbol{\tau}_2$. Preliminary work was developed in the diploma theses [37, 38].

Major parts of this section have been published in the author's works [7, 9, 11] and are adapted for this thesis.

3.3.1 Motion Control

Applying the inverse dynamics control to the dynamic robot model in (2.24) and neglecting the external torques $\boldsymbol{\tau}_{\mathrm{ext}}$, see [29]

$$\boldsymbol{\tau}_1 = \mathbf{M}(\mathbf{q})\mathbf{w}_{\mathrm{n}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \tag{3.22}$$

yields the new linear system dynamics in the form

$$\ddot{\mathbf{q}} = \mathbf{w}_{\mathrm{n}} \ , \tag{3.23}$$

with the new input $\mathbf{w}_{\mathrm{n}}$. A standard task-space controller is implemented by choosing the new input $\mathbf{w}_{\mathrm{n}}$ as

$$\mathbf{w}_{\mathrm{n}} = \left(\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})\right)^{\dagger} \left(\mathbf{w}_{\mathrm{m}} - \dot{\mathbf{J}}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})\dot{\mathbf{q}}\right) \ , \tag{3.24}$$

with the control input, see [29]

$$\mathbf{w}_{\mathrm{m}} = \begin{bmatrix} \ddot{\mathbf{p}}_{\mathcal{P}}^{\mathcal{M}} + \mathbf{K}_{\mathrm{D}}\,\dot{\tilde{\mathbf{p}}}_{\mathcal{P}} + \mathbf{K}_{\mathrm{P}}\,\tilde{\mathbf{p}}_{\mathcal{P}} + \mathbf{K}_{\mathrm{I}}\int \tilde{\mathbf{p}}_{\mathcal{P}}\,\mathrm{d}t \\ \dot{\boldsymbol{\omega}}_{\mathcal{P}}^{\mathcal{M}} + \mathbf{K}_{\omega}\,\tilde{\boldsymbol{\omega}}_{\mathcal{P}} + K_{\mathrm{o}}\,\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}} \end{bmatrix} \ , \tag{3.25}$$

where $\mathbf{A}^{\dagger} = \mathbf{A}^{\mathrm{T}}\left(\mathbf{A}\mathbf{A}^{\mathrm{T}}\right)^{-1}$ is the right pseudo-inverse of the matrix $\mathbf{A}$ and $\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})$ the augmented Jacobian of the tool frame $\mathcal{T}$ w.r.t. the workpiece frame $\mathcal{P}$ from (2.13). The spline-interpolated desired Cartesian trajectory $\left[\left(\mathbf{p}_{\mathcal{P}}^{\mathcal{M}}(t)\right)^{\mathrm{T}} \ \left(\mathbf{o}_{\mathcal{P}}^{\mathcal{M}}(t)\right)^{\mathrm{T}}\right]$ of the manufacturing frame $\mathcal{M}$ was introduced in Section 3.2.3, where $\mathbf{o}_{\mathcal{P}}^{\mathcal{M}}$ denotes the unit quaternion related to the rotation matrix $\mathbf{R}_{\mathcal{P}}^{\mathcal{M}}$. Hence, the position error of the controller (3.25) is computed as $\tilde{\mathbf{p}}_{\mathcal{P}} = \mathbf{p}_{\mathcal{P}}^{\mathcal{M}} - \mathbf{p}_{\mathcal{P}}^{\mathcal{T}}$, the velocity error as $\dot{\tilde{\mathbf{p}}}_{\mathcal{P}} = \dot{\mathbf{p}}_{\mathcal{P}}^{\mathcal{M}} - \dot{\mathbf{p}}_{\mathcal{P}}^{\mathcal{T}}$, and the angular velocity error as $\tilde{\boldsymbol{\omega}}_{\mathcal{P}} = \boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{M}} - \boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{T}}$ in the workpiece frame $\mathcal{P}$. The vector part of the quaternion error $\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}$ is computed with the quaternion product $\otimes$, defined in Appendix B.1, as

$$\mathbf{o}_{\mathcal{T}}^{\mathcal{M}} = \begin{bmatrix} \eta_{\mathcal{T}}^{\mathcal{M}} \\ \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}} \end{bmatrix} = \mathbf{o}_{\mathcal{P}}^{\mathcal{M}} \otimes \left(\mathbf{o}_{\mathcal{P}}^{\mathcal{T}}\right)^{-1} = \begin{bmatrix} \eta_{\mathcal{P}}^{\mathcal{M}} \\ \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} \end{bmatrix} \otimes \begin{bmatrix} \eta_{\mathcal{P}}^{\mathcal{T}} \\ -\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} \end{bmatrix} = \begin{bmatrix} \eta_{\mathcal{P}}^{\mathcal{M}}\eta_{\mathcal{P}}^{\mathcal{T}} + \left(\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}}\right)^{\mathrm{T}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} \\ \eta_{\mathcal{P}}^{\mathcal{T}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} - \eta_{\mathcal{P}}^{\mathcal{M}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} - \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} \times \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} \end{bmatrix} \ , \tag{3.26}$$

see [29, 68]. The quaternion error (3.26) describes the orientation deviation of the manufacturing frame $\mathcal{M}$ w.r.t. the tool frame $\mathcal{T}$, i.e., the quaternion representation of the rotation matrix $\mathbf{R}_{\mathcal{T}}^{\mathcal{M}} = \mathbf{R}_{\mathcal{P}}^{\mathcal{M}}(\mathbf{R}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}))^{\mathrm{T}}$, see [29]. The controller and the proof of the closed-loop stability is published in the author's publications [7, 37] and presented for the sake of completeness in the following two subsections.

Position Error Dynamics

The position error dynamics of (2.24) with (3.22)-(3.25), with the state $\boldsymbol{\xi}^{\mathrm{T}} = \left[(\tilde{\mathbf{p}}_{\mathcal{P}})^{\mathrm{T}}\ (\dot{\tilde{\mathbf{p}}}_{\mathcal{P}})^{\mathrm{T}}\ (\ddot{\tilde{\mathbf{p}}}_{\mathcal{P}})^{\mathrm{T}}\right]$, reads as

$$\dot{\boldsymbol{\xi}} = \check{\mathbf{K}}\boldsymbol{\xi} \quad \text{with} \quad \check{\mathbf{K}} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \\ -\mathbf{K}_{\mathrm{I}} & -\mathbf{K}_{\mathrm{P}} & -\mathbf{K}_{\mathrm{D}} \end{bmatrix} . \tag{3.27}$$

If $\mathbf{K}_{\mathrm{I}}$, $\mathbf{K}_{\mathrm{P}}$, and $\mathbf{K}_{\mathrm{D}}$ are chosen to render $\check{\mathbf{K}}$ a Hurwitz matrix, the position error dynamics (3.27) becomes exponentially stable, see [29].

Orientation Error Dynamics

In order to prove the stability of the orientation error dynamics of (2.24) with (3.22)-(3.25), the positive definite LYAPUNOV function

$$V(\boldsymbol{\zeta}) = K_{\mathrm{o}}\left(\tilde{\eta}_{\mathcal{P}}^2 + (\tilde{\boldsymbol{\varepsilon}}_{\mathcal{P}})^{\mathrm{T}}\tilde{\boldsymbol{\varepsilon}}_{\mathcal{P}}\right) + \frac{1}{2}\left(\tilde{\boldsymbol{\omega}}_{\mathcal{P}}(\boldsymbol{\zeta})\right)^{\mathrm{T}}\tilde{\boldsymbol{\omega}}_{\mathcal{P}}(\boldsymbol{\zeta}) \tag{3.28}$$

is chosen with the state $\boldsymbol{\zeta}^{\mathrm{T}} = \left[\tilde{\eta}_{\mathcal{P}}\ (\tilde{\boldsymbol{\varepsilon}}_{\mathcal{P}})^{\mathrm{T}}\ \dot{\tilde{\eta}}_{\mathcal{P}}\ (\dot{\tilde{\boldsymbol{\varepsilon}}}_{\mathcal{P}})^{\mathrm{T}}\right]$ and the relations $\tilde{\eta}_{\mathcal{P}} = \eta_{\mathcal{P}}^{\mathcal{M}} - \eta_{\mathcal{P}}^{\mathcal{T}}$ and $\tilde{\boldsymbol{\varepsilon}}_{\mathcal{P}} = \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} - \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}}$. A similar approach without $K_{\mathrm{o}} > 0$ can be found in [68]. The time derivative of (3.28) results in, see Appendix B.2,

$$\dot{V}(\boldsymbol{\zeta}) = -\left(\tilde{\boldsymbol{\omega}}_{\mathcal{P}}(\boldsymbol{\zeta})\right)^{\mathrm{T}}\mathbf{K}_{\omega}\tilde{\boldsymbol{\omega}}_{\mathcal{P}}(\boldsymbol{\zeta}) , \tag{3.29}$$

which is negative semi-definite for $\mathbf{K}_{\omega} > 0$. Because the largest invariant set in $\{\boldsymbol{\zeta} : \dot{V}(\boldsymbol{\zeta}) = 0\}$ is the point $\boldsymbol{\zeta} = \mathbf{0}$, LASALLE'S invariance theorem proves the asymptotic stability of the equilibrium point $\boldsymbol{\zeta} = \mathbf{0}$, see, e.g., [69].

3.3.2 Contact Force Estimation

In this work, no external force/torque sensor is available to directly measure the interacting forces between the pen and the 3D object. Instead, the contact force estimation [66] is employed and summarized in this section, see [38].

Based on the dynamic robot model (2.24), the work [66] shows that the residual vector

$$\mathbf{k}(t) = \mathbf{K}_{\mathrm{C}} \left(\mathbf{M}(\mathbf{q})\dot{\mathbf{q}} - \int_0^t \boldsymbol{\tau} + \mathbf{C}^{\mathrm{T}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) + \mathbf{k}\,\mathrm{d}s \right) \tag{3.30}$$

leads to the residual dynamics

$$\dot{\mathbf{k}}(t) = \mathbf{K}_{\mathrm{C}}(\boldsymbol{\tau}_{\mathrm{ext}} - \mathbf{k}(t)) \ , \tag{3.31}$$

with a large positive definite gain matrix $\mathbf{K}_{\mathrm{C}}$, from which the external torques $\boldsymbol{\tau}_{\mathrm{ext}} \approx \mathbf{k}$ are estimated.

In general, the relation between the vector of contact forces $\mathbf{f}_{\mathcal{P}}$ and moments $\boldsymbol{\mu}_{\mathcal{P}}$ expressed in the workpiece frame $\mathcal{P}$, i.e., $(\boldsymbol{\delta}_{\mathcal{P}})^{\mathrm{T}} = \left[(\mathbf{f}_{\mathcal{P}})^{\mathrm{T}}\ (\boldsymbol{\mu}_{\mathcal{P}})^{\mathrm{T}}\right]$, and the external torques $\boldsymbol{\tau}_{\mathrm{ext}}$ is established with the geometric Jacobian $\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})$ of the tool frame $\mathcal{T}$, e.g., [29]

$$\boldsymbol{\tau}_{\mathrm{ext}} = -\left(\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})\right)^{\mathrm{T}} \boldsymbol{\delta}_{\mathcal{P}} \ . \tag{3.32}$$

The estimated contact force $\hat{\mathbf{f}}_{\mathcal{P}}$ and moment $\hat{\boldsymbol{\mu}}_{\mathcal{P}}$ is computed as

$$\begin{bmatrix} \hat{\mathbf{f}}_{\mathcal{P}} \\ \hat{\boldsymbol{\mu}}_{\mathcal{P}} \end{bmatrix} = -\left(\left(\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})\right)^{\mathrm{T}} \right)^{\dagger} \mathbf{k} \ , \tag{3.33}$$

while the moment $\boldsymbol{\mu}_{\mathcal{P}}$ on the tool frame $\mathcal{T}$ vanishes due to point contact of the pen with the object's surface. The contact force w.r.t. the tool frame $\mathcal{T}$ reads as

$$\hat{\mathbf{f}}_{\mathcal{T}} = \left(\mathbf{R}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})\right)^{\mathrm{T}} \hat{\mathbf{f}}_{\mathcal{P}} = \begin{bmatrix} \hat{f}_{\mathcal{T},x} \\ \hat{f}_{\mathcal{T},y} \\ \hat{f}_{\mathcal{T},z} \end{bmatrix} \ . \tag{3.34}$$

3.3.3 Hybrid Force/Motion Control

In this section, the hybrid force/motion controller proposed in [29, 67] is extended for the robotic drawing task. To this end, the pen's lateral position and orientation are controlled simultaneously, and the estimated contact force $\hat{\mathbf{f}}_{\mathcal{T}}$ at the pen tip along the surface normal vector is regulated to the desired value $\mathbf{f}_{\mathrm{d}}$.

The inverse dynamics control law for (2.24) is applied, which reads as

$$\boldsymbol{\tau}_1 = \mathbf{M}(\mathbf{q})\mathbf{w}_{\mathrm{h}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) \ , \tag{3.35}$$

with a new control input $\mathbf{w}_\mathrm{h}$. The new control input $\mathbf{w}_\mathrm{h}$ is chosen as

$$\mathbf{w}_\mathrm{h} = (\mathbf{J}_\mathcal{P}^\mathcal{T}(\mathbf{q}))^\dagger \mathbf{V}_\mathrm{m} \breve{\mathbf{w}}_\mathrm{m} + \mathbf{M}^{-1}(\mathbf{q})(\mathbf{J}_\mathcal{P}^\mathcal{T}(\mathbf{q}))^\mathrm{T} \mathbf{V}_\mathrm{f} \mathbf{w}_\mathrm{f} - (\mathbf{J}_\mathcal{P}^\mathcal{T}(\mathbf{q}))^\dagger \dot{\mathbf{J}}_\mathcal{P}^\mathcal{T}(\mathbf{q})\dot{\mathbf{q}} + (\mathbf{J}_\mathcal{P}^\mathcal{T}(\mathbf{q}))^\dagger \dot{\mathbf{V}}_\mathrm{m} \boldsymbol{\nu}_\mathcal{P}^\mathcal{T}, \tag{3.36}$$

with the motion control input $\breve{\mathbf{w}}_\mathrm{m} \in \mathbb{R}^6$, the force control input $\mathbf{w}_\mathrm{f} \in \mathbb{R}^6$, and the velocity vector $\boldsymbol{\nu}_\mathcal{P}^\mathcal{T}$ to be defined later, cf. [11, 29, 38]. The matrices $\mathbf{V}_\mathrm{m}$ and $\mathbf{V}_\mathrm{f}$ utilized in (3.36) are computed as

$$\mathbf{V}_\mathrm{m} = \mathbf{T}_\mathcal{P}^\mathcal{T}(\mathbf{q})\mathbf{Y}_\mathrm{m}(\mathbf{T}_\mathcal{P}^\mathcal{T}(\mathbf{q}))^\mathrm{T} \tag{3.37a}$$
$$\mathbf{V}_\mathrm{f} = \mathbf{T}_\mathcal{P}^\mathcal{T}(\mathbf{q})\mathbf{Y}_\mathrm{f}(\mathbf{T}_\mathcal{P}^\mathcal{T}(\mathbf{q}))^\mathrm{T} \tag{3.37b}$$

with the constant selection matrices

$$\mathbf{Y}_\mathrm{m} = \mathrm{diag}(1, 1, 0, 1, 1, 1) \ , \tag{3.38a}$$
$$\mathbf{Y}_\mathrm{f} = \mathrm{diag}(0, 0, 1, 0, 0, 0) \ , \tag{3.38b}$$

and the transformation matrix $\mathbf{T}_\mathcal{P}^\mathcal{T}(\mathbf{q})$ given by

$$\mathbf{T}_\mathcal{P}^\mathcal{T}(\mathbf{q}) = \begin{bmatrix} \mathbf{R}_\mathcal{P}^\mathcal{T}(\mathbf{q}) & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \ , \tag{3.39}$$

in order to apply the control inputs $\breve{\mathbf{w}}_\mathrm{m}$ and $\mathbf{w}_\mathrm{f}$ in their specific coordinate axes w.r.t. the tool frame $\mathcal{T}$. Because the matrices $\mathbf{V}_\mathrm{f}$ and $\mathbf{V}_\mathrm{m}$ satisfy

$$(\mathbf{V}_\mathrm{f})^\mathrm{T} \mathbf{V}_\mathrm{m} = \mathbf{0} \ , \tag{3.40}$$

the force and motion control loops are decoupled with the inverse dynamics control law (3.35) and (3.36) which is shown in the following, see [14, 29].

With the assumption of a rigid contact between the pen's tip and the object's surface, no mechanical work is done, and the kinostatic relationship

$$(\boldsymbol{\delta}_\mathcal{P})^\mathrm{T} \boldsymbol{\upsilon}_\mathcal{P}^\mathcal{T} = 0 \ , \tag{3.41}$$

is satisfied, see [29]. Velocities only emerge in the corresponding velocity subspace and read as

$$\boldsymbol{\upsilon}_\mathcal{P}^\mathcal{T} = \mathbf{V}_\mathrm{m} \boldsymbol{\nu}_\mathcal{P}^\mathcal{T} \tag{3.42}$$

and all forces and moments appear only in the force subspace

$$\boldsymbol{\delta}_{\mathcal{P}} = \mathbf{V}_{\mathrm{f}}\boldsymbol{\lambda}_{\mathcal{P}} \tag{3.43}$$

with the velocity vector $\boldsymbol{\nu}_{\mathcal{P}}^{\mathcal{T}}$ and the force vector $\boldsymbol{\lambda}_{\mathcal{P}}$ w.r.t. the workpiece frame $\mathcal{P}$. The time derivative of (3.42) using (2.13) is

$$\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})\dot{\mathbf{q}} = \dot{\mathbf{V}}_{\mathrm{m}}\boldsymbol{\nu}_{\mathcal{P}}^{\mathcal{T}} + \mathbf{V}_{\mathrm{m}}\dot{\boldsymbol{\nu}}_{\mathcal{P}}^{\mathcal{T}} \ . \tag{3.44}$$

Solving (3.44) for $\ddot{\mathbf{q}}$ and inserting the result into (2.24) with $\boldsymbol{\tau}_{\mathrm{ext}} = -(\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}))^{\mathrm{T}}\boldsymbol{\delta}_{\mathcal{P}} = -(\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}))^{\mathrm{T}}\mathbf{V}_{\mathrm{f}}\boldsymbol{\lambda}_{\mathcal{P}}$ yields

$$\begin{aligned}\mathbf{M}(\mathbf{q})(\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}))^{\dagger}\mathbf{V}_{\mathrm{m}}\dot{\boldsymbol{\nu}}_{\mathcal{P}}^{\mathcal{T}} = \boldsymbol{\tau} - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) - \mathbf{M}(\mathbf{q})(\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}))^{\dagger}\dot{\mathbf{V}}_{\mathrm{m}}\boldsymbol{\nu}_{\mathcal{P}}^{\mathcal{T}} + \\ \mathbf{M}(\mathbf{q})(\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}))^{\dagger}\dot{\mathbf{J}}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})\dot{\mathbf{q}} - (\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}))^{\mathrm{T}}\mathbf{V}_{\mathrm{f}}\boldsymbol{\lambda}_{\mathcal{P}} \ .\end{aligned} \tag{3.45}$$

By premultiplying (3.45) with $((\mathbf{J}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}))^{\dagger}\mathbf{V}_{\mathrm{m}})^{\mathrm{T}}$ and using (3.40), the last term vanishes. After inserting $\boldsymbol{\tau} = \boldsymbol{\tau}_1$ from (3.35) and (3.36), the new motion dynamics reads as

$$\dot{\boldsymbol{\nu}}_{\mathcal{P}}^{\mathcal{T}} = \breve{\mathbf{w}}_{\mathrm{m}} \ , \tag{3.46}$$

see [29]. This equation shows that the force subspace does not influence the subspace of the new motion dynamics. The new control input $\breve{\mathbf{w}}_{\mathrm{m}}$ from (3.46) is chosen similarly to $\mathbf{w}_{\mathrm{m}}$ from (3.25) as, see, e.g. [29],

$$\breve{\mathbf{w}}_{\mathrm{m}} = \begin{bmatrix} \ddot{\mathbf{p}}_{\mathcal{P}}^{\mathcal{M}} + (\mathbf{R}_{\mathcal{P}}^{\mathcal{T}})\mathbf{K}_{\mathrm{D}}(\mathbf{R}_{\mathcal{P}}^{\mathcal{T}})^{\mathrm{T}}\,\dot{\tilde{\mathbf{p}}}_{\mathcal{P}} + (\mathbf{R}_{\mathcal{P}}^{\mathcal{T}})\mathbf{K}_{\mathrm{P}}(\mathbf{R}_{\mathcal{P}}^{\mathcal{T}})^{\mathrm{T}}\,\tilde{\mathbf{p}}_{\mathcal{P}} + (\mathbf{R}_{\mathcal{P}}^{\mathcal{T}})\mathbf{K}_{\mathrm{I}}(\mathbf{R}_{\mathcal{P}}^{\mathcal{T}})^{\mathrm{T}}\int\tilde{\mathbf{p}}_{\mathcal{P}}\,\mathrm{d}t \\ \dot{\boldsymbol{\omega}}_{\mathcal{P}}^{\mathcal{M}} + \mathbf{K}_{\omega}\,\tilde{\boldsymbol{\omega}}_{\mathcal{P}} + K_{\mathrm{o}}\,\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}} \end{bmatrix} \tag{3.47}$$

with transforming the gain matrices $\mathbf{K}_{\mathrm{D}}$, $\mathbf{K}_{\mathrm{P}}$, and $\mathbf{K}_{\mathrm{I}}$ to render an active compliance control in the tool frame $\mathcal{T}$. Again, the exponential stability of the position dynamics can be ensured with the state $\boldsymbol{\xi}^{\mathrm{T}} = \left[(\tilde{\mathbf{p}}_{\mathcal{P}})^{\mathrm{T}}\ (\dot{\tilde{\mathbf{p}}}_{\mathcal{P}})^{\mathrm{T}}\ (\ddot{\tilde{\mathbf{p}}}_{\mathcal{P}})^{\mathrm{T}}\right]$,

$$\dot{\boldsymbol{\xi}} = \breve{\mathbf{K}}\boldsymbol{\xi} \quad \text{with} \quad \breve{\mathbf{K}} = (\mathbf{R}_{\mathcal{P}}^{\mathcal{T}})\check{\mathbf{K}}(\mathbf{R}_{\mathcal{P}}^{\mathcal{T}})^{\mathrm{T}} \ , \tag{3.48}$$

where $\check{\mathbf{K}}$ in (3.27) is a Hurwitz matrix, which is also true for $\breve{\mathbf{K}}$. Note that the orientation terms $\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}$ and $\tilde{\boldsymbol{\omega}}_{\mathcal{P}}$ stay the same because of the identity matrix in $\mathbf{T}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})$, cf. (3.39). Hence, the orientation error dynamics is unchanged and is asymptotically stable with the Lyapunov function (3.28), see [7, 37].

The control input (3.47) can also be interpreted within the tool frame $\mathcal{T}$ by transforming $\tilde{\mathbf{p}}_{\mathcal{P}}, \tilde{\boldsymbol{\varepsilon}}_{\mathcal{P}}, \dot{\tilde{\mathbf{p}}}_{\mathcal{P}}$, and $\tilde{\boldsymbol{\omega}}_{\mathcal{P}}$ with

$$\begin{bmatrix}\tilde{\mathbf{p}}_{\mathcal{T}}\\ \boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{T}}\end{bmatrix} = \left(\mathbf{T}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\right)^{\mathrm{T}}\begin{bmatrix}\tilde{\mathbf{p}}_{\mathcal{P}}\\ \boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{T}}\end{bmatrix} , \tag{3.49a}$$

$$\begin{bmatrix}\dot{\tilde{\mathbf{p}}}_{\mathcal{T}}\\ \tilde{\boldsymbol{\omega}}_{\mathcal{P}}\end{bmatrix} = \left(\mathbf{T}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\right)^{\mathrm{T}}\begin{bmatrix}\dot{\tilde{\mathbf{p}}}_{\mathcal{P}}\\ \tilde{\boldsymbol{\omega}}_{\mathcal{P}}\end{bmatrix} , \tag{3.49b}$$

resulting in

$$\breve{\mathbf{w}}_{\mathcal{T},\mathrm{m}} = (\mathbf{T}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q}))^{\mathrm{T}}\breve{\mathbf{w}}_{\mathrm{m}} = \begin{bmatrix}(\mathbf{R}^{\mathcal{T}}_{\mathcal{P}})^{\mathrm{T}}\ddot{\mathbf{p}}^{\mathcal{M}}_{\mathcal{P}} + \mathbf{K}_{\mathrm{D}}\,\dot{\tilde{\mathbf{p}}}_{\mathcal{T}} + \mathbf{K}_{\mathrm{P}}\,\tilde{\mathbf{p}}_{\mathcal{T}} + \mathbf{K}_{\mathrm{I}}\int\tilde{\mathbf{p}}_{\mathcal{T}}\,\mathrm{d}t\\ \dot{\boldsymbol{\omega}}^{\mathcal{M}}_{\mathcal{P}} + \mathbf{K}_{\omega}\,\tilde{\boldsymbol{\omega}}_{\mathcal{P}} + K_{\mathrm{o}}\,\boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{T}}\end{bmatrix} . \tag{3.50}$$

Hence, the position errors are rotated into the tool frame $\mathcal{T}$ according to, see (2.5)

$$\tilde{\mathbf{p}}_{\mathcal{T}} = \left(\mathbf{R}^{\mathcal{T}}_{\mathcal{P}}\right)^{\mathrm{T}}\tilde{\mathbf{p}}_{\mathcal{P}} = \mathbf{R}^{\mathcal{P}}_{\mathcal{T}}(\mathbf{p}^{\mathcal{M}}_{\mathcal{P}} - \mathbf{p}^{\mathcal{T}}_{\mathcal{P}}) \equiv \mathbf{R}^{\mathcal{P}}_{\mathcal{T}}\mathbf{p}^{\mathcal{M}}_{\mathcal{P}} + \mathbf{p}^{\mathcal{P}}_{\mathcal{T}} = \mathbf{p}^{\mathcal{M}}_{\mathcal{T}} , \tag{3.51}$$

resulting in the position error of the tool frame $\mathcal{T}$ w.r.t. the manufacturing frame $\mathcal{M}$. Furthermore, also the feedforward term $\ddot{\mathbf{p}}^{\mathcal{M}}_{\mathcal{P}}$ is rotated into the tool frame $\mathcal{T}$, while $\dot{\mathbf{R}}^{\mathcal{T}}_{\mathcal{P}}$ is neglected in (3.49), see [67].

The constraints (3.41) of the force-controlled subspace read with (3.43) and (2.13) as

$$(\mathbf{V}_{\mathrm{f}})^{\mathrm{T}}\boldsymbol{v}^{\mathcal{T}}_{\mathcal{P}} = (\mathbf{V}_{\mathrm{f}})^{\mathrm{T}}\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\dot{\mathbf{q}} = \mathbf{0} . \tag{3.52}$$

Solving (2.24) for $\ddot{\mathbf{q}}$ with the external torque $\boldsymbol{\tau}_{\mathrm{ext}} = -(\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q}))^{\mathrm{T}}\mathbf{V}_{\mathrm{f}}\boldsymbol{\lambda}_{\mathcal{P}}$ and inserting the result into the time derivative of (3.52), i.e.,

$$(\dot{\mathbf{V}}_{\mathrm{f}})^{\mathrm{T}}\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\dot{\mathbf{q}} + (\mathbf{V}_{\mathrm{f}})^{\mathrm{T}}\dot{\mathbf{J}}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\dot{\mathbf{q}} + (\mathbf{V}_{\mathrm{f}})^{\mathrm{T}}\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\ddot{\mathbf{q}} = \mathbf{0} , \tag{3.53}$$

yields

$$\begin{aligned}\boldsymbol{\lambda}_{\mathcal{P}} = &\left((\mathbf{V}_{\mathrm{f}})^{\mathrm{T}}\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\mathbf{M}^{-1}(\mathbf{q})(\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q}))^{\mathrm{T}}\mathbf{V}_{\mathrm{f}}\right)^{-1}\\ &\left((\mathbf{V}_{\mathrm{f}})^{\mathrm{T}}\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\mathbf{M}^{-1}(\mathbf{q})\left(\boldsymbol{\tau} - \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{q}(\mathbf{q})\right) + (\mathbf{V}_{\mathrm{f}})^{\mathrm{T}}\dot{\mathbf{J}}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\dot{\mathbf{q}} + (\dot{\mathbf{V}}_{\mathrm{f}})^{\mathrm{T}}\mathbf{J}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q})\dot{\mathbf{q}}\right) .\end{aligned} \tag{3.54}$$

By inserting (3.35) and (3.36) in (3.54) and using (3.40), the identity $(\dot{\mathbf{V}}_\mathrm{f})^\mathrm{T}\mathbf{V}_\mathrm{m} = -(\mathbf{V}_\mathrm{f})^\mathrm{T}\dot{\mathbf{V}}_\mathrm{m}$, and (3.42) the new force dynamics results in

$$\boldsymbol{\lambda}_\mathcal{P} = \mathbf{w}_\mathrm{f} \ , \tag{3.55}$$

see [29]. This equation shows that the velocity subspace does not influence the subspace of the new force dynamics. The force control input $\mathbf{w}_\mathrm{f}$ is chosen as

$$\mathbf{w}_\mathrm{f} = \mathbf{T}_\mathcal{P}^\mathcal{T}(\mathbf{q}) \begin{bmatrix} \mathbf{w}_{\mathcal{T},\mathrm{f}} \\ \mathbf{0} \end{bmatrix} \ , \tag{3.56}$$

where

$$\mathbf{w}_{\mathcal{T},\mathrm{f}} = \mathbf{f}_\mathrm{d} + \mathbf{K}_\mathrm{Pf}(\mathbf{f}_\mathrm{d} - \hat{\mathbf{f}}_\mathcal{T}) + \mathbf{K}_\mathrm{If} \int \mathbf{f}_\mathrm{d} - \hat{\mathbf{f}}_\mathcal{T} \mathrm{d}t \ , \tag{3.57}$$

with the desired force $(\mathbf{f}_\mathrm{d})^\mathrm{T} = \begin{bmatrix} 0 & 0 & f_{\mathrm{d},z} \end{bmatrix}$, see [29, 38]. The contact force $\hat{\mathbf{f}}_\mathcal{T}$ is assumed to appear only in the z-direction of the pen's tip tool frame $\mathcal{T}$, i.e., $(\hat{\mathbf{f}}_\mathcal{T})^\mathrm{T} = \begin{bmatrix} 0 \ 0 \ \hat{f}_{\mathcal{T},z} \end{bmatrix}$, cf. (3.38b). The force control input (3.56) and (3.57) with the force error $\tilde{\mathbf{f}}_\mathcal{T} = \mathbf{f}_\mathrm{d} - \hat{\mathbf{f}}_\mathcal{T}$, and with the assumption of a correct estimation of the contact force $\hat{\mathbf{f}}_\mathcal{P} \approx \mathbf{f}_\mathcal{P}$ leads to the force error dynamics

$$(\mathbf{I} + \mathbf{K}_\mathrm{Pf})\dot{\tilde{\mathbf{f}}}_\mathcal{T} + \mathbf{K}_\mathrm{If}\tilde{\mathbf{f}}_\mathcal{T} = \mathbf{0} \ , \tag{3.58}$$

which is exponentially stable with positive definite diagonal gain matrices $\mathbf{I} + \mathbf{K}_\mathrm{Pf}$ and $\mathbf{K}_\mathrm{If}$.

3.3.4 Null-Space Controller

Kinematically redundant robots exhibit an additional null-space in task-space control, which is stabilized using the null space control law

$$\boldsymbol{\tau}_2 = \mathbf{M}(\mathbf{q})\mathbf{P}\big(-\mathbf{b}(\mathbf{q}) - \mathbf{K}_\mathrm{Dn}\dot{\mathbf{q}} - \mathbf{K}_\mathrm{Pn}(\mathbf{q} - \mathbf{q}_\mathrm{mean})\big) \ , \tag{3.59}$$

with the projection matrix $\mathbf{P} = \mathbf{I} - (\mathbf{J}_\mathcal{P}^\mathcal{T}(\mathbf{q}))^\dagger \mathbf{J}_\mathcal{P}^\mathcal{T}(\mathbf{q})$, see [11, 29, 38, 70], and the positive definite diagonal gain matrices $\mathbf{K}_\mathrm{Dn}$ and $\mathbf{K}_\mathrm{Pn}$. In (3.59), the barrier function $\mathbf{b}^\mathrm{T}(\mathbf{q}) = \begin{bmatrix} b_1(q_1) \ b_2(q_2) \ \cdots \ b_n(q_n) \end{bmatrix}$ is defined as

$$b_h(q_h) = \frac{b_\mathrm{max}}{(q_h - q_{h,\mathrm{max}})^2} - \frac{b_\mathrm{min}}{(q_{h,\mathrm{min}} - q_h)^2} \ , \quad h = 1, \dots, n \ , \tag{3.60}$$

to avoid reaching the mechanical axis limits

$$\mathbf{q}_{\mathrm{max}}^{\mathrm{T}} = \begin{bmatrix} q_{1,\mathrm{max}} & q_{2,\mathrm{max}} & \cdots & q_{n,\mathrm{max}} \end{bmatrix} \quad \text{and} \tag{3.61a}$$

$$\mathbf{q}_{\mathrm{min}}^{\mathrm{T}} = \begin{bmatrix} q_{1,\mathrm{min}} & q_{2,\mathrm{min}} & \cdots & q_{n,\mathrm{min}} \end{bmatrix} \ , \tag{3.61b}$$

see Appendix A and [71]. The parameters b_{max} and b_{min} are used to tune the barrier functions, and $\mathbf{q}_{\mathrm{mean}}$ denotes the mid point between the axis limits, i.e.

$$\mathbf{q}_{\mathrm{mean}} = \frac{1}{2}\begin{bmatrix} q_{1,\mathrm{max}} + q_{1,\mathrm{min}} & q_{2,\mathrm{max}} + q_{2,\mathrm{min}} & \cdots & q_{n,\mathrm{max}} + q_{n,\mathrm{min}} \end{bmatrix} . \tag{3.62}$$

3.4 Experimental Results

In this section, the experimental results for the robotic drawing process are presented and discussed. First, the experimental setup and the input pattern to be drawn are presented. Second, the parallel path projection and the path projection using the LSCM from Section 3.2 are evaluated in simulation with the user-provided 2D input pattern. Third, the optimal placement of the robot w.r.t. the workpiece is determined. Fourth, drawing trajectories are generated and executed experimentally on the robot using the motion controller from Section 3.3.1, and fifth, the hybrid force/motion controller from Section 3.3.3 is evaluated experimentally. Finally, the drawing results and measurements are compared. A video of the experimental drawing process is provided at `www.acin.tuwien.ac.at/c1eb` .
Major parts of this section have been published in the author's work [9, 11] and are adapted for this thesis.

3.4.1 Experimental Setup

The experimental setup for the drawing process is shown in Fig. 3.1. In this setup, the ceiling-mounted industrial robot KUKA LBR iiwa 14 R820 has a pen tool attached to the end-effector according to the kinematic arrangement in Fig. 2.1a. The tool comprises a passive compliance mechanism to account for absolute positioning errors and model uncertainties of the robot and its environment. The kinematic parameters of the KUKA LBR iiwa 14 R820 are shown in Appendix A.1. The workpiece in the experiments is a 3D-printed rabbit on which the robot has to draw the input pattern.

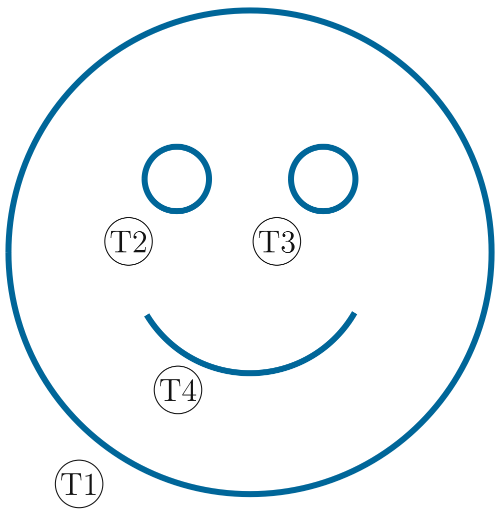


Figure 3.8: User-provided 2D input pattern for the experimental drawing process with the disconnected segments (T1)$\cdots$(T4); adapted from [9, 11].

Input Pattern

The 2D input pattern is provided by the user using a computer mouse, a touch-screen, or by drawing patterns with a digitizer on a tablet device. Furthermore, a path may also be generated from parametric equations. In the following experimental drawing process, the 2D input pattern with four disconnected segments (T1)$\cdots$(T4) shown in Fig. 3.8 is used, i.e., a smiley symbol.

3.4.2 Path Projection

In the following, the user-provided 2D input pattern in Fig. 3.8 is projected at a user-specified location on the 3D object, which is the face of the 3D-printed rabbit, see Fig. 3.9. Both projection methods introduced in Section 3.2, i.e., the parallel projection and the proposed projection based on the LSCM, are used. Subsequently, the projection results are compared and robot trajectories for the process execution are generated.

Parallel Projection Method

For the simple parallel projection from Section 3.2.2, parallel green rays are generated originating at each path point $\mathbf{x}_{\mathcal{W}_2,k}$, $k = 1, \ldots, n_\mathrm{m}$, from the blue 2D input

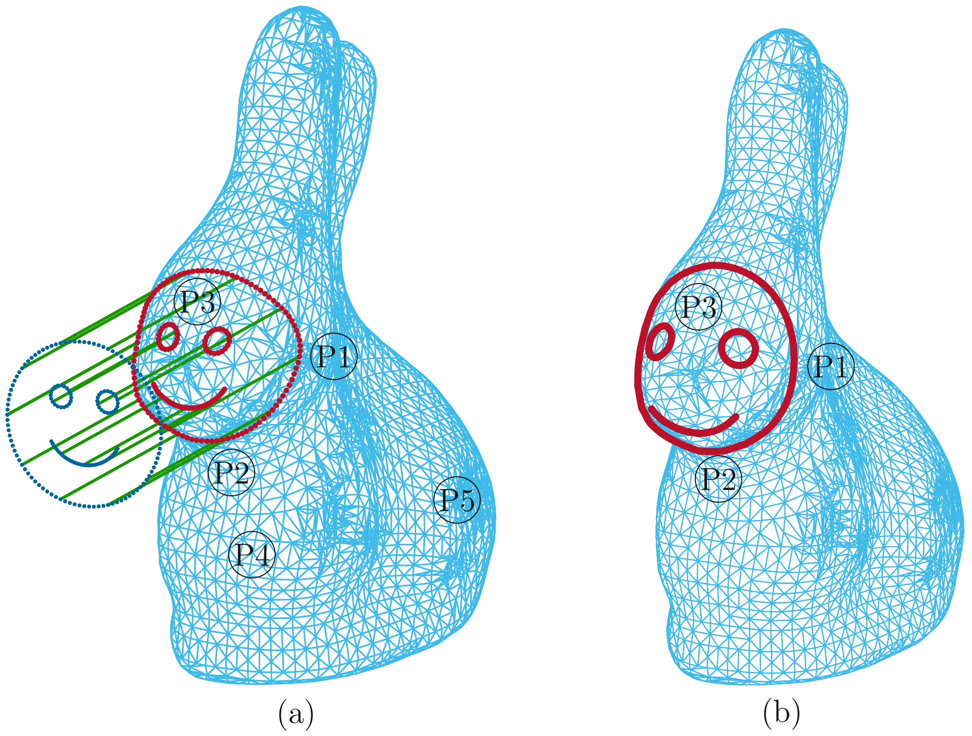


Figure 3.9: 2D input path projected on the 3D object with (a) parallel projection (b) LSCM-based projection; adapted from [11].

pattern, see Fig. 3.9a. Then, the path points $\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k}, k = 1, \dots, n_{\mathrm{m}}$, on the 3D object are found at the intersection points of the green rays with the 3D object's mesh using the implementation of the *Ray-Triangle Intersection* algorithm [72]. At each path point $\mathbf{p}^{\mathcal{M}}_{\mathcal{P},k}, k = 1, \dots, n_{\mathrm{m}}$, a desired orientation for the manufacturing frame $\mathcal{M}$, i.e., $\mathbf{R}^{\mathcal{M}}_{\mathcal{P},k}$, is generated from the path planning. The z-axis of the pen tool is aligned with the green ray and the x- and y-axes are chosen according to a reference orientation. The orientation is equal for every path point.

Examining the parallel projection result in Fig. 3.9a, a large distortion can be seen at point $\textcircled{\text{P1}}$ due to the high curvature of the 3D object and the large inclination angle of the parallel rays. In comparison, the eyes of the smiley symbol at $\textcircled{\text{P3}}$ are less distorted because of the smaller curvature. Nevertheless, the eyes are in an elliptic shape and not properly placed. If the parallel projection is used to project a 2D input pattern to areas with small curvature, e.g., $\textcircled{\text{P4}}$ or $\textcircled{\text{P5}}$ in Fig. 3.9a, the drawing result of the whole 2D input would be significantly better without notable distortions, cf. [9].

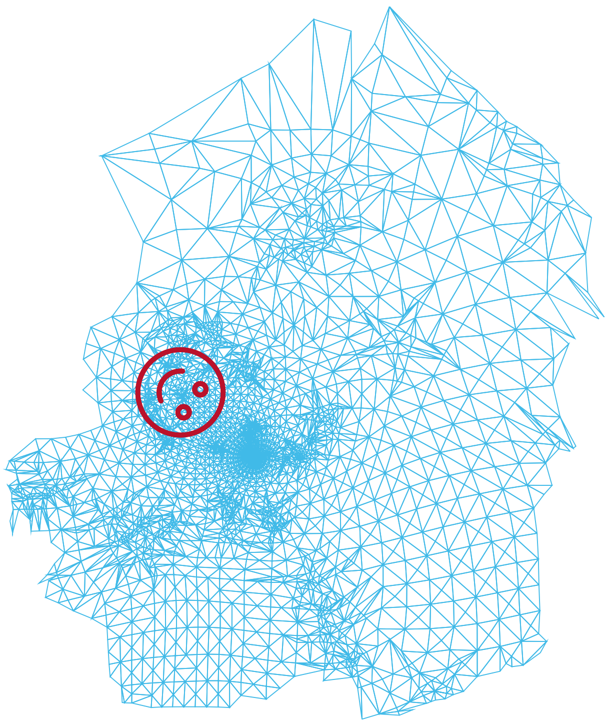


Figure 3.10: Projection of the 2D input pattern on the red and blue flattened segments of the 3D object in Fig. 3.3; adapted from [11].

Since the parallel projection is computationally inexpensive, the computation is finished after approximately 50 ms on an Intel Core i7-8700K at 3.70 GHz.

Path Projection using Least-Squares Conformal Mapping

The path projection on the 3D-printed rabbit according to Section 3.2.1 is computed by segmenting the object, resulting in the segmentation shown in Fig. 3.3. Then, the individual segments are flattened using conformal mapping. In the next step, the 2D input pattern is projected on the flattened segments, i.e., the face of the 3D-printed rabbit, see Fig. 3.10. The red and blue segments located at the face of the 3D-printed rabbit in Fig. 3.3b must be combined to be able to apply the LSCM-based projection method for the whole face area. The 2D input pattern is transferred back to the 3D object using the barycentric coordinates introduced in Section 3.2.1. The projection result of the 2D input pattern on the 3D surface of the workpiece is illustrated in Fig. 3.9b. According to (3.21), coordinate systems are attached to the 3D projection points of the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$. The z-direction of the pen tool's coordinate systems is chosen based on the local normal vector of the surface, and the remaining directions are derived from a reference orientation. The x-axis is set horizontally to the floor, while the y-axis is computed to obtain a right-oriented coordinate system.

The shape of the pattern, in particular the outer circle of the smiley symbol at $\textcircled{\scriptsize P1}$ and the circular eyes at $\textcircled{\scriptsize P3}$ in Fig. 3.9b, is projected on the 3D object with minimum distortions.

The path projection using the conformal mapping method is performed on the 3D-printed rabbit, which comprises approximately 13000 faces and 6500 vertices. The computation is executed on an Intel Core i7-8700K at 3.70 GHz. The most time-consuming computation of this offline planning is the segmentation, which takes around 5 s. The LSCM-based projection can be calculated in 0.3 s. Note that those steps must be executed only once for each 3D object, see Fig. 3.2.

Properties of Projection Methods

In this section, further simulations to compare the parallel projection with the LSCM-based projection are presented. The 2D input pattern from Fig. 3.8 is projected on the ear of the 3D-printed rabbit. Due to the high curvature of the area around the ear, projections are challenging, and local distortions are likely to occur, Fig. 3.11.

In the first experiment, the parallel projection is used to project the smiley symbol with the 2D plane located on the side of the 3D-printed rabbit, see Fig. 3.11a. It is clearly seen that some of the green rays do not intersect with the 3D object. Hence, only parts of the input pattern are projected on the 3D-printed rabbit, and the trajectory generation fails.

In the second experiment, the parallel projection is employed with the 2D plane for the input pattern located on top of the 3D-printed rabbit. In this experiment, large distortions occur due to the high curvature; see Fig. 3.11b. Additionally, the projection result is dissected in multiple parts due to the concave areas on the side. Although, theoretically, the trajectory can be computed, the drawing process with the robot would fail.

The third experiment shows the projection of the 2D input pattern from Fig. 3.8 on the ear of the 3D-printed rabbit with the LSCM-based method; see Fig. 3.11c. Using this mapping, the 2D input pattern is properly projected on the workpiece.

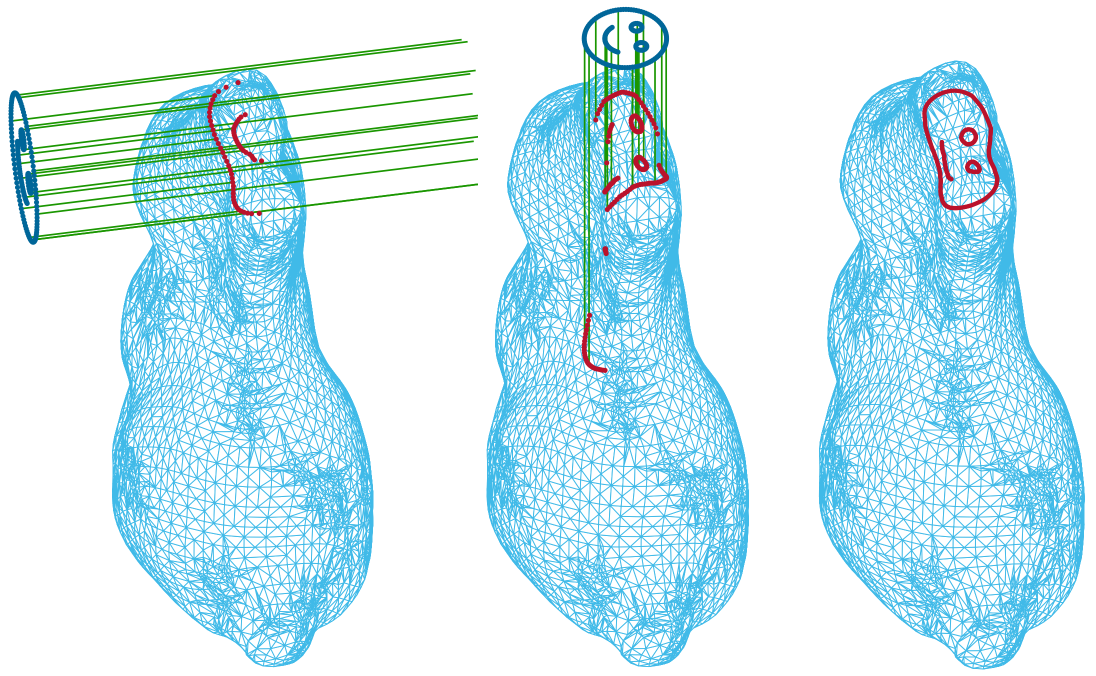


Figure 3.11: 2D input path projected on the ear of the 3D-printed rabbit with (a) parallel projection from the side (b) parallel projection from the top (c) LSCM-based projection; adapted from [11].

Flattening the segments of the 3D object into 2D form brings along that the 2D input pattern is wrapped around the ear of the 3D-printed rabbit, see also Fig. 3.10. Thus, the projection result contains the complete user-provided pattern and the trajectory for the robotic drawing process is computable.

3.4.3 Optimal Robot Base Placement

The optimal placement of the robot base is crucial to execute the continuous drawing path with the robot since the robot's workspace is limited due to the mechanical joint limits. Therefore, an optimal robot base placement w.r.t. the world frame is determined, see Fig. 2.1a. In this example, the world frame $\mathcal{W}$ coincides with the workpiece frame $\mathcal{P}$.

In order to find this optimal placement, the total number of feasible inverse kinematics solutions of all manufacturing-path poses in $H_{\mathcal{P}}^{\mathcal{M}}$ from (3.21) is maximized. The set Q_k of joint-space solutions is computed for a specific path pose $\mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}$ with the analytical inverse kinematics [73] as

$$Q_k = \mathbf{h}_{\mathrm{H}}^{-1}(\mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}) \ , \tag{3.63}$$

see (2.9) and (2.17). The Kuka LBR iiwa 14 R820 has $n = 7$ DoF, i.e., the manipulator is kinematically redundant and an infinite number of inverse kinematics solutions exist for each feasible path pose $\mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}$. Therefore, the inverse kinematics solutions are reduced to a finite number of solutions by filtering Q_k from (3.63) according to

$$\begin{aligned} Q_k^{\mathrm{f}} &= f_{\mathrm{f}}(Q_k) = \{\mathbf{q}_{\mathrm{f},1}, \mathbf{q}_{\mathrm{f},2}, \cdots, \mathbf{q}_{\mathrm{f},e_{\mathrm{f},k}}\} \\ &= \Big\{\mathbf{q}_i, \mathbf{q}_j \in Q_k \Big| q_{\mathrm{dist}} < \|\mathbf{q}_i - \mathbf{q}_j\|_\infty, i,j = 1,\dots,e_k\Big\} \ , \end{aligned} \tag{3.64}$$

where q_{dist} determines the minimum distance between two solutions in the filtered set Q_k^{f}. In (3.64), $\|\cdot\|_\infty$ denotes the maximum norm of the vector $\cdot$, and $e_{\mathrm{f},k}$ and e_k are the total numbers of solutions in Q_k^{f} and Q_k, respectively. The optimal robot base placement relative to the world frame $\mathbf{p}_{\mathcal{W}}^{\mathcal{B}*}$ is then computed with (3.64) by solving the optimization problem

$$\mathbf{p}_{\mathcal{W}}^{\mathcal{B}*} = \underset{\mathbf{p}_{\mathcal{W}}^{\mathcal{B}} \in \mathbb{R}^3}{\arg\max} \sum_{k=1}^{n_{\mathrm{m}}} e_{\mathrm{f},k}(\mathbf{p}_{\mathcal{W}}^{\mathcal{B}}) \ , \tag{3.65}$$

i.e., by maximizing the number of distinguishable inverse kinematics solutions, see the individual coordinate frames in Fig. 2.1a.

In the experimental setup, the robot base is placed as accurately as possible relative to the world frame, according to the result of (3.65). The actual robot base placement in the experimental setup is determined using a calibration procedure during the initial setup, see Fig. 3.12. In this calibration procedure, the robot is equipped with calibration pins at the end-effector (red rectangles in Fig. 3.12), which tightly fit into the holes at the base of the 3D-printed rabbit (red circles in Fig. 3.12). The actual robot base placement is obtained from the measurement of the robot configuration $\mathbf{q}$ and the forward kinematics (2.8), which accurately calibrates the pose of the robot base w.r.t. the world frame for subsequent path planning and trajectory execution.

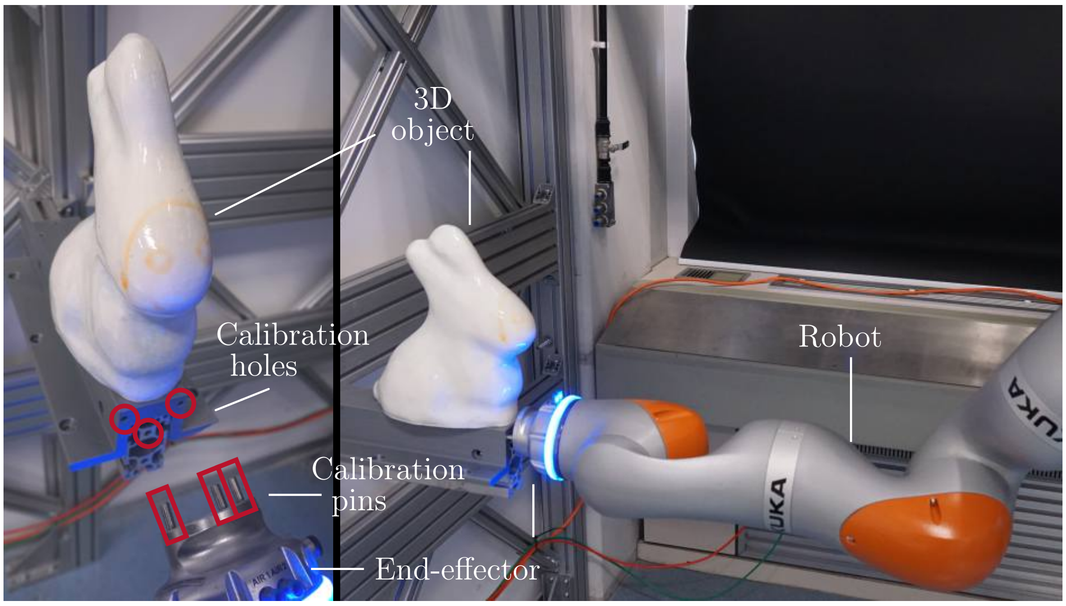


Figure 3.12: Calibration of the experimental setup with the KUKA LBR iiwa 14 R820; adapted from [11].

3.4.4 Drawing Process with Motion Control

In this section, the drawing process using the two presented path projection results from Section 3.4.2 is executed with the motion control concept introduced in Section 3.3.1. In the following, the planning and measurement results and the pen motions are discussed in terms of 3D paths, position control errors, and joint-space paths.

Planned and Executed 3D Paths

The drawing process is planned with both projection methods, transition paths are added, and a spline interpolation is computed with a suitable time parametrization, see Section 3.2.3. The trajectories are then executed on the KUKA LBR iiwa 14 R820 using the motion control concept of Section 3.3.1. The resulting 3D paths are depicted in Fig. 3.13 for the parallel projection and in Fig. 3.14 for the LSCM-based projection. For both methods, the desired trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{M}}$ of the tool frame $\mathcal{T}$, i.e., the pen tip, the corresponding desired pen orientations $\mathbf{n}_{\mathcal{B}}^{\mathcal{M}}$, and the actual trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{T}}$ w.r.t. the robot's base frame $\mathcal{B}$, computed using the forward kinematics (2.8), are shown. The automatically generated transition paths are also visible.

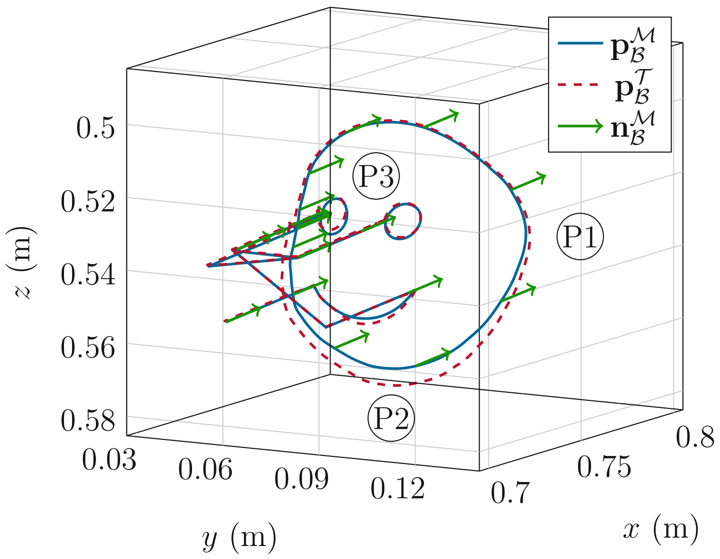


Figure 3.13: Desired and executed 3D paths $\mathbf{p}_{\mathcal{B}}^{\mathcal{M}}$ and $\mathbf{p}_{\mathcal{B}}^{\mathcal{T}}$ and the corresponding surface normal vectors $\mathbf{n}_{\mathcal{B}}^{\mathcal{M}}$ using parallel projection with motion control; adapted from [11].

For the parallel projection in Fig. 3.13, the pen orientation $\mathbf{R}_{\mathcal{B}}^{\mathcal{M}}$ is constant and aligned with the projection direction $\mathbf{n}_{\mathcal{B}}^{\mathcal{M}}$, while it remains aligned with the surface normal vector using the LSCM-based projection. Consequently, a high position deviation evolves for the parallel projection between the desired and the actual trajectory, which becomes clearly visible around P2. In this area, the local curvature of the 3D object is very high, see also Fig. 3.9a. Due to the constant orientation $\mathbf{R}_{\mathcal{B}}^{\mathcal{M}}$, the angle between the surface normal vector and the pen at the tool surface becomes large. Due to this large approach angle, the pen slips on the object's surface, which makes the pen tip deviate from the desired manufacturing path $\mathbf{p}_{\mathcal{B}}^{\mathcal{M}}$ and results in a large position error. In comparison, the approach angle around the eyes at point P3 in Fig. 3.13 is much smaller; therefore, this effect is less pronounced, see Fig. 3.9.

For the LSCM-based projection, see Fig. 3.14, the motion controller has a good closed-loop performance when following the desired trajectory in the experimental setup. Even at the points P1 and P2 with high curvature, cf. Fig. 3.13, the pen tip motion $\mathbf{p}_{\mathcal{B}}^{\mathcal{T}}$ does not deviate from the desired manufacturing path $\mathbf{p}_{\mathcal{B}}^{\mathcal{M}}$. Because the pen axis is aligned with the surface normal vector, i.e., $\mathbf{n}_{\mathcal{B}}^{\mathcal{M}}$, the pen does not slip on the surface due to the perpendicular approach angle. The position errors

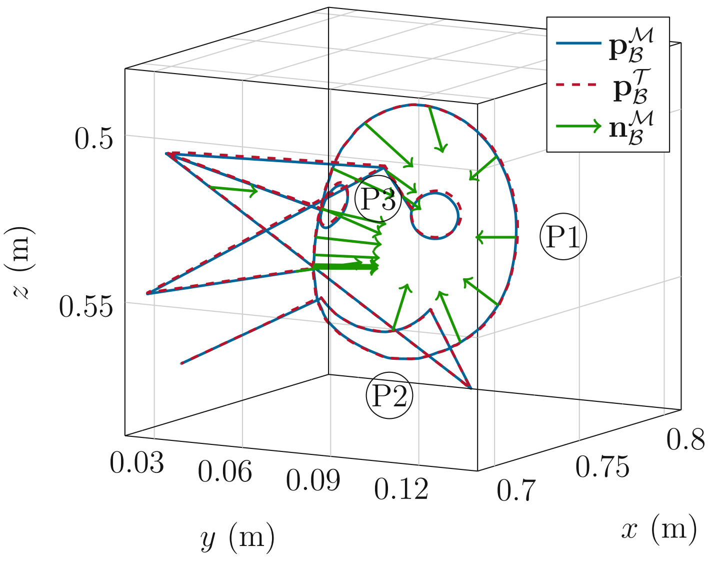


Figure 3.14: Desired and executed 3D paths $\mathbf{p}_{\mathcal{B}}^{\mathcal{M}}$ and $\mathbf{p}_{\mathcal{B}}^{\mathcal{T}}$ and the corresponding surface normal vectors $\mathbf{n}_{\mathcal{B}}^{\mathcal{M}}$ using the LSCM-based projection with motion control; adapted from [11].

remain low throughout the trajectory, even in areas with high curvature, which will be discussed quantitatively in the next section.

Position Control Error

In Fig. 3.15, the position control errors $\tilde{\mathbf{p}}_{\mathcal{T}}$ in the tool frame $\mathcal{T}$ for both experiments are presented, see (3.49). The topmost graph shows the contact state of the pen, where intervals with the value 1 indicate that the pen is in contact with the 3D object, and the value of the contact state is 0 when the pen is retracted from the surface. The variable s denotes the path progress from 0 % to 100 %. During the first interval (T1), the outer circle of the smiley symbol; at (T2) and (T3), the eyes; and during (T4), the mouth is drawn, cf. Fig. 3.8. In between those intervals, the robot's end-effector moves from the end of one segment to the starting point of the next segment.

Fig. 3.15 shows that the position control errors $\tilde{\mathbf{p}}_{\mathcal{T}}$ are large within the intervals (T1) ($s \in [5\,\%, 20\,\%]$)) and (T4) ($s \in [80\,\%, 90\,\%]$) because of the large approach angles. Small position errors emerge during the intervals (T2) and (T3) due to the small local curvature and the small approach angle. Due to the passive compliance of the pen holder along the z-direction, the position control error in this direction

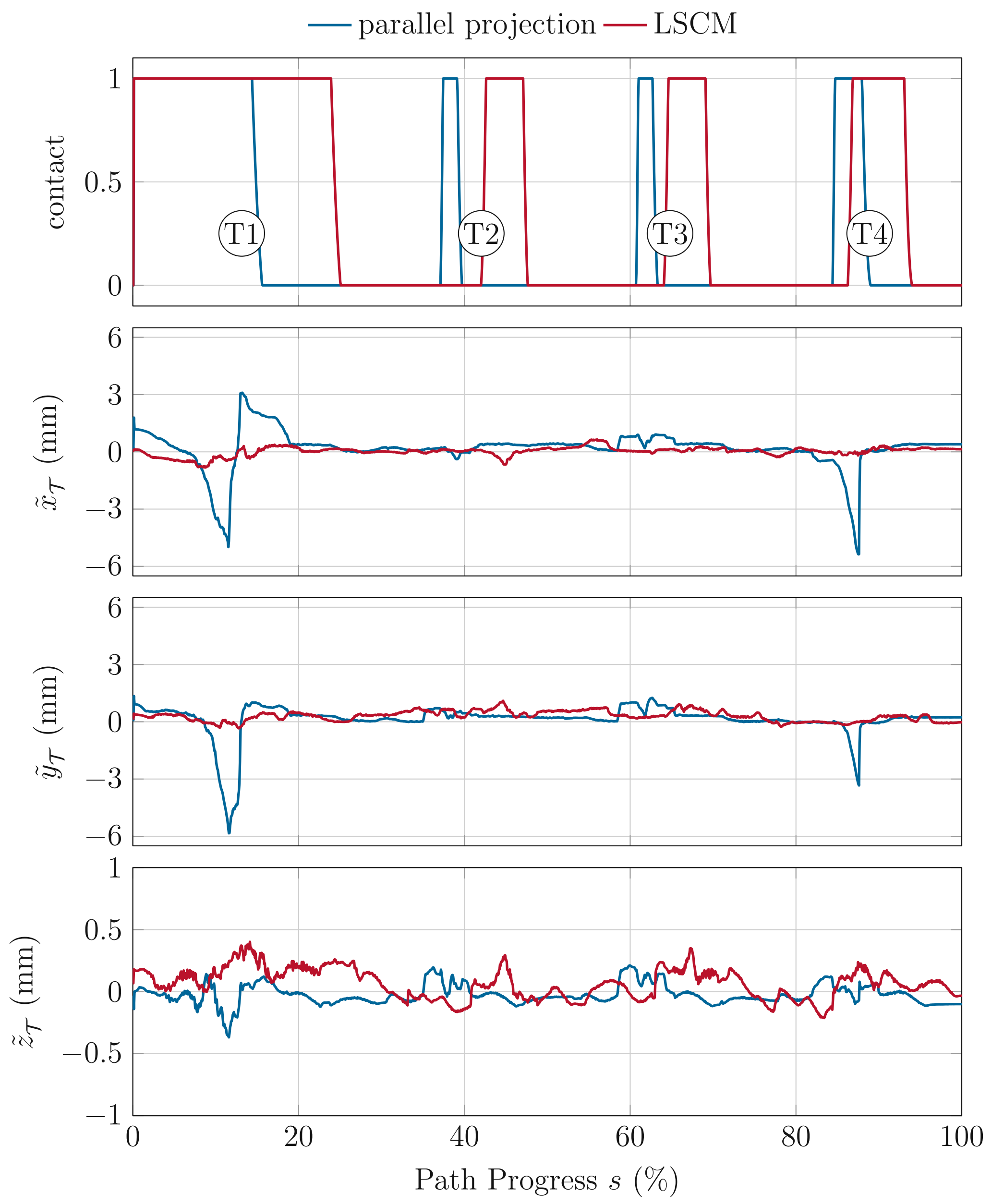


Figure 3.15: Evolution of the pen/surface contact (top) and position control errors $(\tilde{\mathbf{p}}_{\mathcal{T}})^{\mathrm{T}} = \left[\tilde{x}_{\mathcal{T}}\ \tilde{y}_{\mathcal{T}}\ \tilde{z}_{\mathcal{T}}\right]$ using the parallel projection and the LSCM-based projection and motion control; adapted from [11].

Table 3.1: Gain matrices of the position, orientation, and null-space control law in (3.25), (3.59) and (3.60).

Position Control	Orientation Control	Null-Space Control
$\mathbf{K}_\mathrm{D} = 10\,\mathrm{diag}(8,8,8)$	$\mathbf{K}_\omega = 10\,\mathrm{diag}(8,8,8)$	$\mathbf{K}_\mathrm{Dn} = 10\,\mathrm{diag}(8,8,8)$
$\mathbf{K}_\mathrm{P} = 10^2\,\mathrm{diag}(16,16,16)$	$K_\mathrm{o} = 1600$	$\mathbf{K}_\mathrm{Pn} = 10^2\,\mathrm{diag}(16,16,16)$
$\mathbf{K}_\mathrm{I} = \mathrm{diag}(0,0,0)$		$b_\mathrm{max} = -b_\mathrm{min} = 10$

is small. In contrast, the position control errors $\tilde{\mathbf{p}}_\mathcal{T}$ of the LSCM-based experiment in Fig. 3.15 (red lines) remain small during the entire execution time. Quantitatively, the position control error $\tilde{\mathbf{p}}_\mathcal{T}$ remains below 1 mm for all Cartesian position coordinates.

Note that the lengths of the intervals $\textcircled{\scriptsize T1}\cdots\textcircled{\scriptsize T4}$ for the two projection methods differ. This is due to the fact that the range of the axes motion during the drawing process depends on the planned trajectory. Using the LSCM-based projection, the pen orientation is always normal to the surface. Therefore, the robot has to perform wide motions in the joint space and, consequently, the time intervals are longer. The total execution time is around 300 s. The positive definite diagonal gain matrices $\mathbf{K}_\omega, \mathbf{K}_\mathrm{Pn}, \mathbf{K}_\mathrm{Dn}$, and $K_\mathrm{o} > 0$ in (3.25) and (3.59) are chosen empirically, and the gain matrices $\mathbf{K}_\mathrm{D}$, $\mathbf{K}_\mathrm{P}$, $\mathbf{K}_\mathrm{I}$ in (3.25) are found using pole placement. The parameters of the gain matrices are given in Tab. 3.1. Note that the feedforward terms $\ddot{\mathbf{p}}_\mathcal{P}^\mathcal{M}$ and $\dot{\boldsymbol{\omega}}_\mathcal{P}^\mathcal{M}$ in the motion control law (3.25) are neglected due to the high gain matrices $\mathbf{K}_\mathrm{D}$ and $\mathbf{K}_\mathrm{P}$.

Joint-Space Paths

The joint-space paths $\bar{\mathbf{q}}^\mathrm{T}(s) = \begin{bmatrix}\bar{q}_1 & \bar{q}_2 & \cdots & \bar{q}_7\end{bmatrix}$ for the process execution with both projections are depicted in Fig. 3.16, where the individual joint angles $\bar{q}_h$ are normalized to their respective axis limits $q_{h,\mathrm{min}}$ and $q_{h,\mathrm{max}}$ in the form

$$\bar{q}_h = \frac{2q_h - (q_{h,\mathrm{max}} + q_{h,\mathrm{min}})}{q_{h,\mathrm{max}} - q_{h,\mathrm{min}}}\ , \quad h = 1,\ldots,n\ . \tag{3.66}$$

As the orientation of the pen during the parallel projection experiment remains constant, only small changes of the joint angles are required to follow the corresponding desired path, see the blue lines in Fig. 3.16.

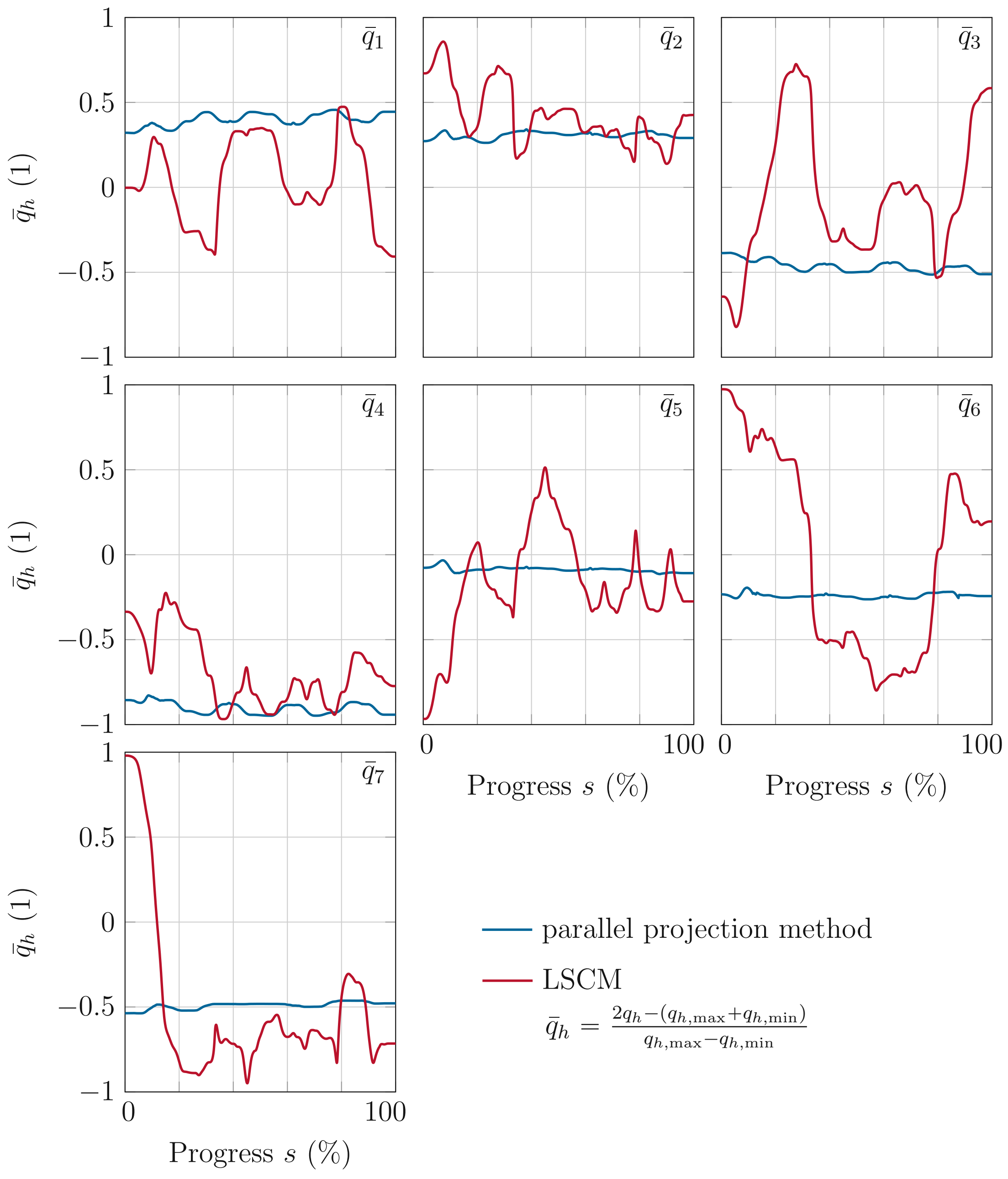


Figure 3.16: Joint-space path $\bar{\mathbf{q}}(s)$ using the parallel projection and the LSCM-based projection with motion control. The paths of the individual joints are normalized to their respective axis limits $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$; adapted from [11].

The joint-space path $\bar{\mathbf{q}}(s)$ for the LSCM-based experiment (red lines in Fig. 3.16) shows that, in this case, significantly larger robot movements are performed. Most joints come close to the respective mechanical axis limit during process execution. Therefore, applying the advanced null-space control law (3.59) is necessary to execute this process, even though the relative position of the robot base $\mathcal{B}$ w.r.t. the workpiece frame $\mathcal{P}$ is already chosen optimally with (3.65). Consequently, executing the planned trajectory is more challenging using the LSCM-based projection, but it yields more accurate results for the drawing process, cf. Figs. 3.13 and 3.14.

3.4.5 Drawing Process with Hybrid Force/Motion Control

In this experiment, the hybrid force/motion controller from Section 3.3.3 is employed to perform the robotic drawing process using the planned trajectory of the LSCM-based projection. The results are discussed in terms of the planned and executed 3D paths, position control errors, and contact forces.

Planned and Executed 3D Paths

The desired and actual paths of the pen tip $\mathbf{p}_{\mathcal{B}}^{\mathcal{M}}$ and $\mathbf{p}_{\mathcal{B}}^{\mathcal{T}}$ are shown in Fig. 3.17. It can be clearly seen that the actual trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{T}}$ deviates from the desired trajectory $\mathbf{p}_{\mathcal{B}}^{\mathcal{M}}$ in z-direction because the force control law (3.56) is applied in this direction. The position in z-direction must deviate from the planned trajectory to guarantee a constant contact force. During the transition phase from the end point of one segment to the starting point of the next segment, the motion controller of Section 3.4.4 is used. In order to ensure a smooth transition between the hybrid force/motion control and the motion control, slightly smaller diagonal entries of the positive definite gain matrix $\mathbf{K}_\omega$ and $K_\mathrm{o} > 0$ and smaller diagonal entries of the matrices $\mathbf{K}_\mathrm{D}$ and $\mathbf{K}_\mathrm{P}$ compared to the experiment with the motion controller in Section 3.4.4 were chosen, see Tab. 3.2 and compare with Tab. 3.1. In the hybrid force/motion control law (3.36), the last term is omitted due to a sufficiently slow change of the selection matrix $\mathbf{V}_\mathrm{m}$, and the feedforward terms $\ddot{\mathbf{p}}_{\mathcal{P}}^{\mathcal{M}}$ and $\dot{\boldsymbol{\omega}}_{\mathcal{P}}^{\mathcal{M}}$ in the motion control law (3.47) are negligible due to the high gain matrices $\mathbf{K}_\mathrm{D}$ and $\mathbf{K}_\mathrm{P}$. In the force control law (3.57), the feedforward term $\mathbf{f}_\mathrm{d}$ is also negligible because of the high gain matrix $\mathbf{K}_\mathrm{Pf}$. Additionally, the term $-\mathbf{K}_\mathrm{Df}\left(\mathbf{R}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})\right)^\mathrm{T}\dot{\mathbf{p}}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q})$ is added with the positive definite diagonal matrix $\mathbf{K}_\mathrm{Df}$ to damp the motion along

Table 3.2: Gain matrices of the position, orientation, and force control law in (3.47) and (3.57), cf. Tab. 3.1.

Position Control	Orientation Control	Force Control
$\mathbf{K}_\mathrm{D} = 10\,\mathrm{diag}(4,4,4)$	$\mathbf{K}_\omega = 10\,\mathrm{diag}(4,4,4)$	$\mathbf{K}_\mathrm{Df} = 10^2\,\mathrm{diag}(0,0,8)$
$\mathbf{K}_\mathrm{P} = 10^2\,\mathrm{diag}(4,4,4)$	$K_\mathrm{o} = 400$	$\mathbf{K}_\mathrm{Pf} = 10^2\,\mathrm{diag}(0,0,1)$
$\mathbf{K}_\mathrm{I} = \mathrm{diag}(0,0,0)$		$\mathbf{K}_\mathrm{If} = 10^2\,\mathrm{diag}(0,0,1)$

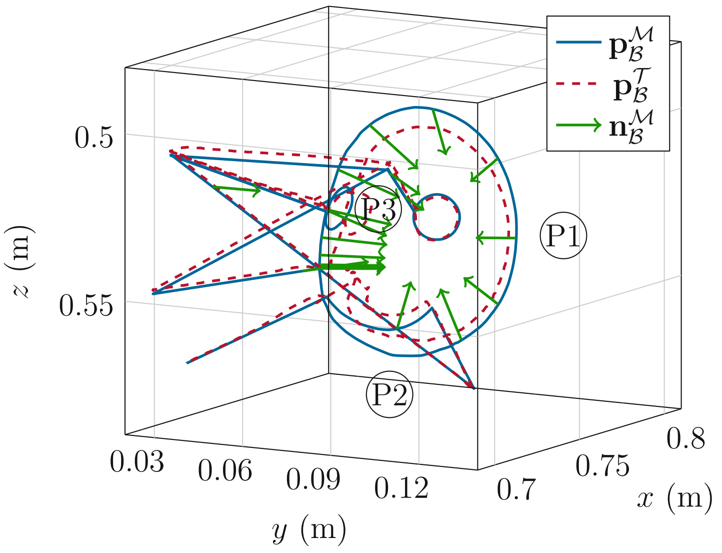


Figure 3.17: Desired and executed 3D path $\mathbf{p}_\mathcal{B}^\mathcal{M}$ and $\mathbf{p}_\mathcal{B}^\mathcal{T}$ and the corresponding surface normal vectors $\mathbf{n}_\mathcal{B}^\mathcal{M}$ using the LSCM-based projection with hybrid force/motion control; adapted from [11].

the z-direction of the tool frame $\mathcal{T}$ and generate smooth robot motions during the experiment, see [67]. This term does not affect the proof of stability from Section 3.3.3 because no end-effector velocity appears in z-direction with ideal contact, see (3.42). The gain matrices of the null-space controller are the same as those of the motion controller.

Contact Force

The estimated contact force $\hat{f}_{\mathcal{T},z}$ is depicted in Fig. 3.18 for the LSCM-based trajectory with the hybrid force/motion control and the motion control of Section 3.4.4. The contact force parallel to the surface normal $\hat{f}_{\mathcal{T},z}$ is controlled to the desired value of $-3\,\mathrm{N}$ by the hybrid force/motion controller (red line). At

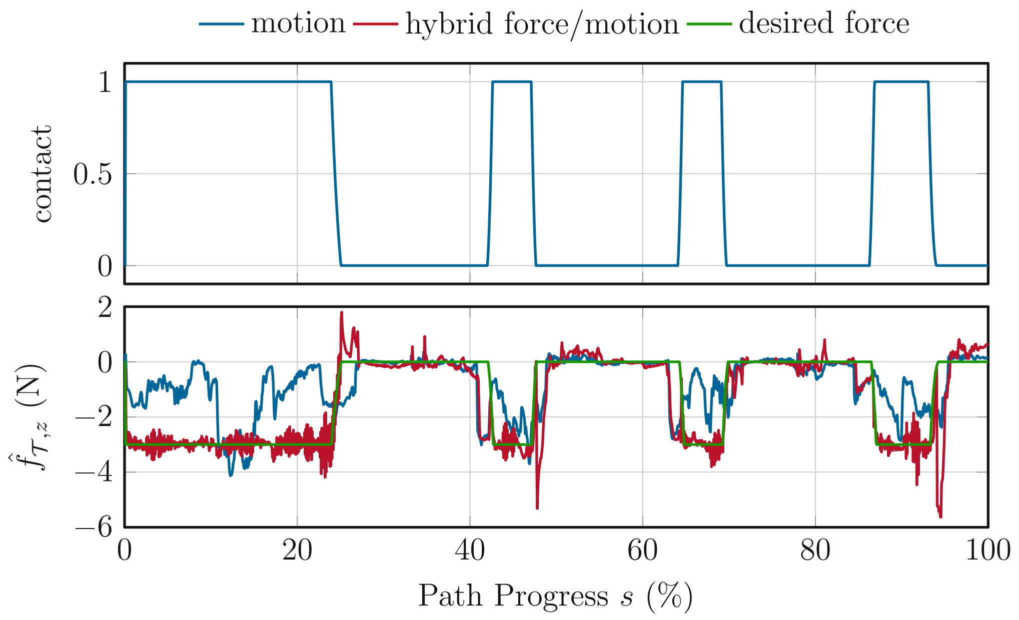


Figure 3.18: Evolution of the pen/surface contact state (top) and the estimated contact force $\hat{f}_{\mathcal{T},z}$ in z-direction of the tool frame $\mathcal{T}$ (bottom) during the drawing process with the motion control and the hybrid force/motion control; adapted from [11].

every pen/surface contact state change, the controller is switched from the hybrid force/motion controller to the motion controller and vice versa. The small contact force peaks in $\hat{f}_{\mathcal{T},z}$, see Fig. 3.18, originate from these controller switching operations. In contrast, the contact force $\hat{f}_{\mathcal{T},z}$ is not controlled by the motion controller and, therefore, it varies between -4 N and nearly zero in this case (blue line). Additionally, a loss of the pen/surface contact can occur due to misalignment of the workpiece and inaccuracies of the robot. For both controllers, the estimated forces are approximately zero in phases without contact.

Remark 1 *The contact force estimation [66] requires a precisely calibrated dynamic robot model* (2.24). *Otherwise, significant estimation errors may occur, even if no pen/surface contact is present. Alternatively, the contact force estimation can be calibrated for a specific robot trajectory by performing the experiment without any pen/surface contact as a reference motion with* $\hat{\mathbf{f}}_{\mathcal{T}} = \mathbf{0}$.

Position Control Error

In order to evaluate the performance of the hybrid force/motion control, the contact state and the position control error $\tilde{\mathbf{p}}_{\mathcal{T}}$ of the drawing process are depicted in Fig. 3.19. Overall, small control errors are observed for $\tilde{x}_{\mathcal{T}}$ and $\tilde{y}_{\mathcal{T}}$, whereas the peaks originate from the controller switching, cf. Fig. 3.18. During the intervals with pen/surface contact, the position control error $\tilde{\mathbf{p}}_{\mathcal{T}}$ in z-direction is higher because the force controller adapts the position in z-direction to control the contact force $\hat{f}_{\mathcal{T},z}$.

3.4.6 Comparison of the Drawing Results

This section presents and compares the experimental drawing results of both projection methods introduced in Section 3.2 and the two control concepts from Section 3.3. The resulting drawing patterns on the 3D-printed rabbit using the parallel projection and the LSCM-based projection with motion control are depicted in Fig. 3.20a as blue line and Fig. 3.20b as red line. The other patterns (the yellow line for the LSCM-based projection in Fig. 3.20a and the blue line for the parallel projection in Fig. 3.20b) are only depicted for comparison. Comparing the two projection methods, distortions from the parallel projection are seen, especially at areas with high curvature, e.g., at (P1). The robot is not able to accurately follow the desired trajectory at areas with a large approach angle using the parallel projection with motion control, e.g.,(P2), cf. Fig. 3.13 and Fig. 3.14.

Next, the drawing results with the two presented control concepts are compared for the LSCM-based trajectory, see Fig. 3.21. In the drawing result with the motion controller in Fig. 3.21a, a fluctuating line thickness is observed. In particular, at point (P6), nearly no pen/surface contact is present, emerging from inaccuracies of the robot kinematics. This thin line results from a low contact force of the pen tip which can also be seen in the estimated contact force $\hat{f}_{\mathcal{T},z}$ in Fig. 3.18 at $s \approx 10\,\%$. In contrast, a uniform line thickness is achieved with the hybrid force/motion control in Fig. 3.21b. The experiments for this drawing process are executed without an absolute calibration of the robot and without optical measurement of the actual Cartesian end-effector pose.

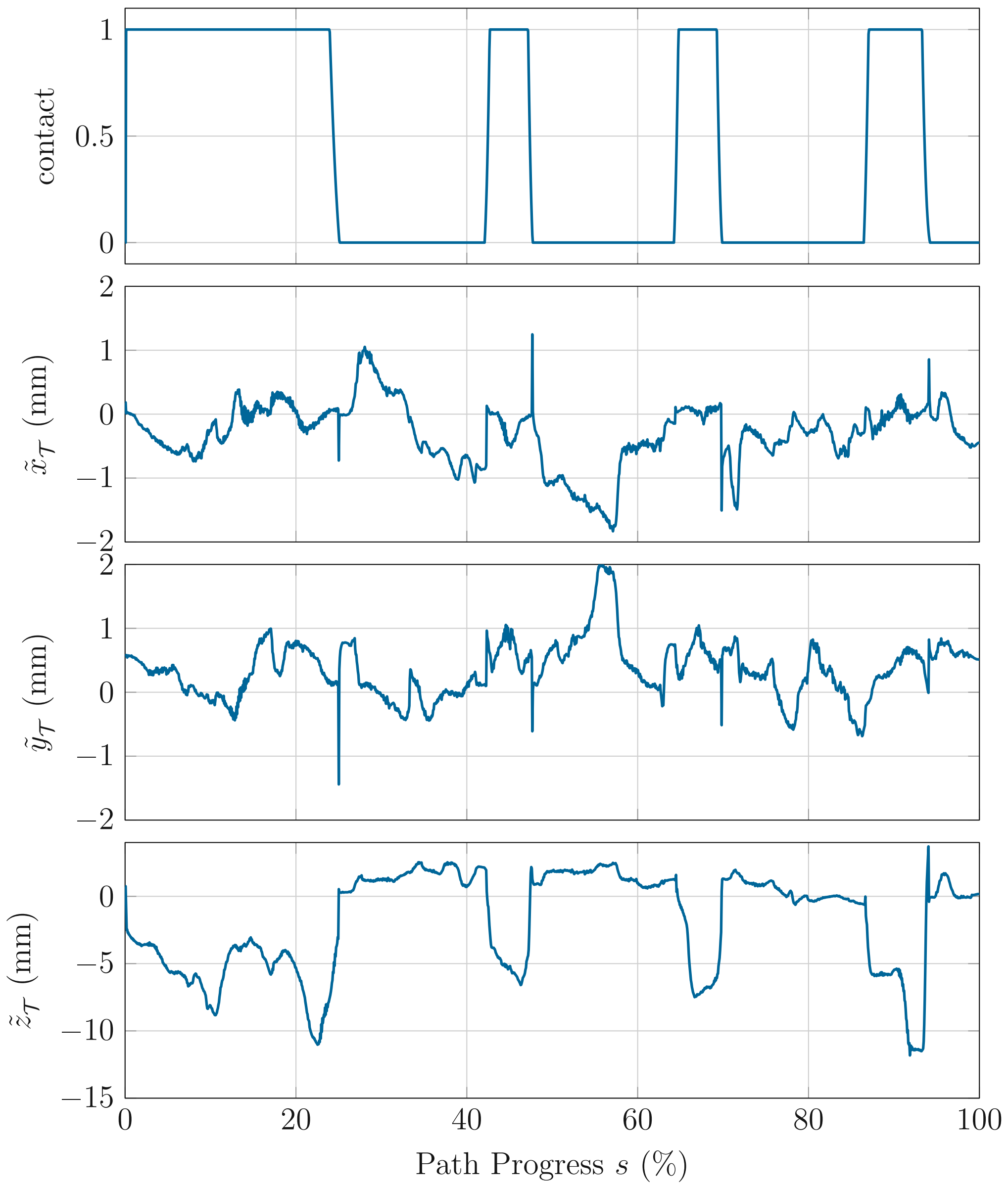
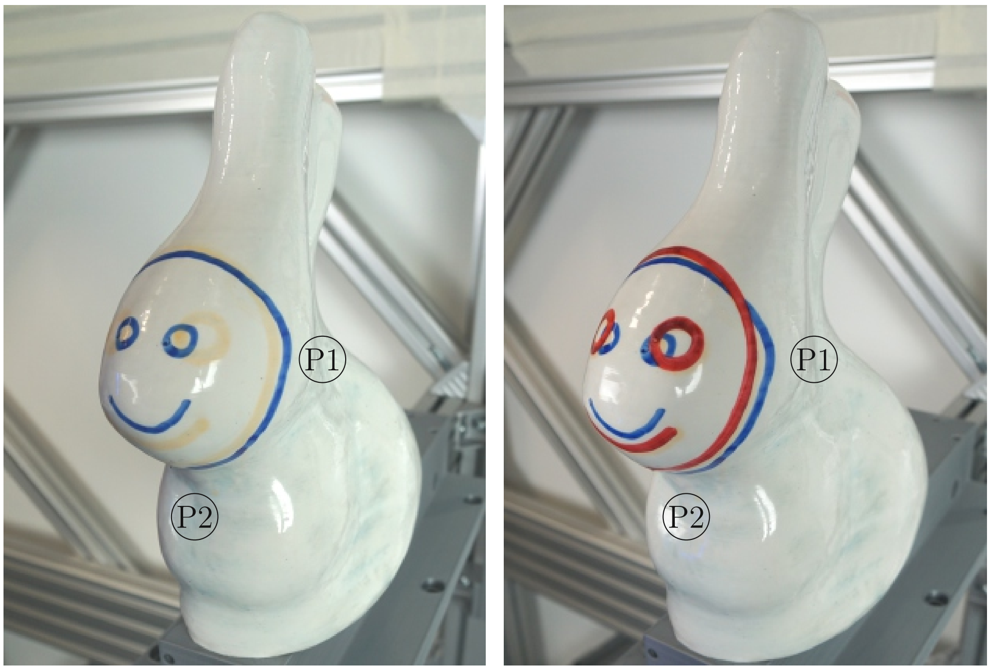


Figure 3.19: Evolution of the pen/surface contact (top) and position control error $\tilde{\mathbf{p}}_{\mathcal{T}}$ using the LSCM-based projection with the hybrid force/motion control; adapted from [11].



(a) (b)

Figure 3.20: Resulting patterns on the 3D object with motion control and (a) parallel projection (blue line) (b) LSCM-based projection (red line); adapted from [11].

3.5 Conclusions

In this chapter, an automated workflow for the customization of products using robotic manufacturing is presented on the basis of a drawing process on a complex surface of a 3D object. In this process, a 2D input pattern is provided by a user together with the desired size, location, and rotation on the 3D object and then drawn on the 3D object using an industrial robot.

The workflow starts with a projection procedure of the 2D input pattern to the 3D object, for which two different projection methods are presented and explained, i.e., the least-squares conformal mapping (LSCM) projection and a simple parallel projection approach. The conformal mapping procedure comprises a segmentation step, an LSCM to flatten the segments, and an inverse map using barycentric coordinates. This way, distortions of the 2D input pattern are minimized, and a 3D path on the 3D object is obtained. Although the parallel projection can be used in areas with small curvature and low complexity a visually proper pro-

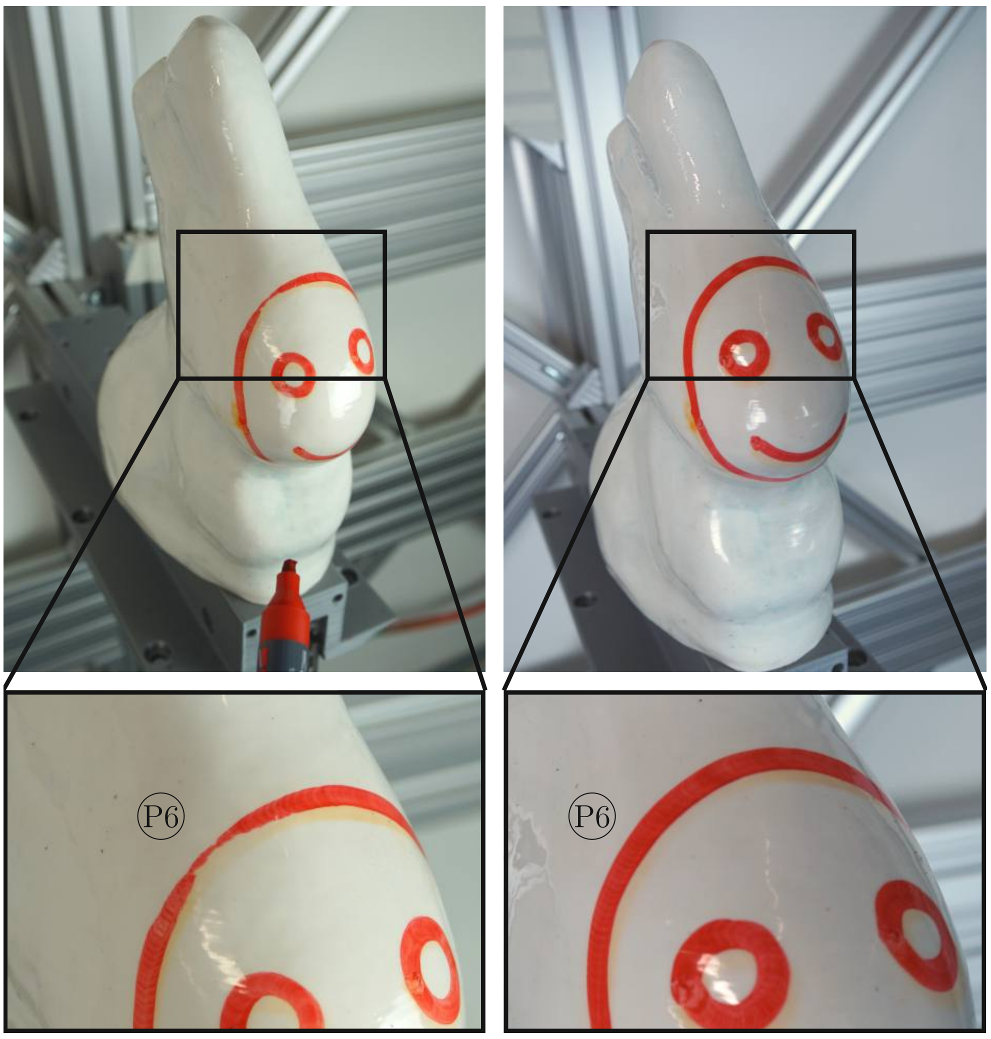


Figure 3.21: Resulting patterns on the 3D object with the LSCM-based projection using (a) motion control (b) hybrid force/motion control; adapted from [11].

jection is achieved only with the more advanced conformal mapping procedure. Based on the result of the two presented projection methods, robot trajectories are planned and executed in an experimental setup with an industrial robot. For the planned trajectories, two different control concepts, i.e., motion control and hybrid force/motion control, are presented and employed in the experiments. Additionally, the contact force during the drawing process is estimated. The motion controller can execute the trajectory with small errors. However, only the hybrid force/motion controller is able to maintain the desired contact force normal to the surface during the whole task execution, which is necessary for achieving a high production quality. This chapter presented the drawing results using both projection procedures and the two control concepts.

In industry, this approach allows the automatic mapping of 2D manufacturing paths to different 3D objects while maintaining the required position and contact force accuracy. This automated workflow directly applies to other manufacturing processes like laser engraving, milling, or ultrasonic cutting.

Major parts of this section have been published in the author's works [7, 9, 11] and are adapted for this thesis.

Chapter 4 Optimal TCP and Robot Base Placement

In the previous chapter, a flexible way to draw 2D user-input patterns fully automatically on complex 3D workpieces with an industrial robot was shown. In order to guarantee high flexibility, the robot's kinematic structure needs to fulfill the kinematic requirements of the drawing task for many possible input patterns. Executing the trajectory generated with the least-squares conformal mapping approach may be challenging due to the high curvature of the workpiece. The robot placement is crucial to execute the desired complex continuous robot trajectories. Although the robot placement is optimally designed and an advanced null-space controller is used for the drawing task, some axes are occasionally close to their limits. If a joint limit is violated during the drawing task, continuous robot execution is no longer possible.

In this chapter, the optimal robot base placement for multiple manufacturing paths is investigated. An emphasis is laid on optimizing the mounting of the TCP on the robot's end-effector, which has gained little attention in the literature so far. This approach allows the adaption of the robot cell in a manufacturing line after deployment or during operation if challenging robot trajectories cannot be executed with the TCP mounting of the original robot work cell.

First, a literature review for the optimal robot base and TCP placement is discussed. Then, a fast joint-space planner for continuous paths is presented, designed to be embedded in a superordinate optimization problem. Subsequently, this planner is utilized in the algorithm to find the optimal TCP pose, formulated as an optimization problem. Additionally, this algorithm is adapted to find the optimal robot base placement. The proposed approach is applied to a trim application, where the optimal robot base and the TCP placement are calculated for 44 complex continuous paths.

Major parts of this chapter have been published in the author's work [8] and are adapted for this thesis.

4.1 Literature Review

The robot base placement is crucial for the execution of certain manufacturing paths. Most papers in the literature deal with base pose optimization, considering different properties of the robot, i.e., kinematics [74], dynamics [75], manipulability [76], stiffness [77], and time optimality [78].

Major parts of this section have been published in the author's work [8] and are adapted for this thesis.

In one example in the literature, the stiffness and deformation of individual axes are considered for the base placement in milling tasks [79] to satisfy the high accuracy requirements. In some applications, i.e., mostly for milling and welding tasks, continuous end-effector paths are investigated. In [80] and [81], the elastostatic model of the robot is used to find the optimal stationary workpiece placement for continuous paths by minimizing the distance between the robot tool and its desired path. In [82], genetic algorithms that consider kinematic and dynamic manipulability and milling forces for a given trajectory are used. A fast search for a feasible stationary workpiece pose is shown in [83] for a 7-DoF robot with a tool axis redundancy by successively considering additional constraints in an optimization problem. Other works concerned with robot base placement only consider individual poses instead of continuous paths, e.g., for spot welding [84, 85], short seam welding [86], or pick-and-place tasks using mobile robots [87]. Further works focus on capability and reachability maps, which are mostly used to position the base of mobile robots in grasping tasks, e.g., [88], [89], and [90].

For a general 6-DoF robot, up to 16 inverse kinematics solutions exist, which result in the same end-effector pose, see [29]. Industrial robots with 6 DoF and a spherical wrist are used in most applications. If the range of one or more rotational axes is beyond 180°, additional inverse kinematics solutions must be considered in the planning. In [91], the feasibility of the whole path is investigated while focusing mainly on cycle times. The work [31] combines two optimization loops, wherein, the inner loop, a redundant DoF is used to avoid singularities and the dynamical limits of the robot. In the outer loop, a particle swarm optimization searches for the global optimal robot base pose. Although wide axis ranges are considered in both works, the same solution of inverse kinematics is employed throughout a single-path execution.

Because the search for the optimal robot base placement is closely related to path planning, a survey of sampling-based path planning methods that systematically incorporate constraints is given in [92]. The graph-based DESCARTES algorithm described in [93] considers movements through multiple inverse kinematics solutions. However, this planner only outputs one solution for each path planning problem and exhibits comparably long computation times. More advanced path-planning algorithms like [94] and [95] cover a wider range of applications but also suffer from higher computational costs.

Nevertheless, a robotic work cell is only productive if all desired manufacturing paths are plannable and executable. This chapter aims to show that by adapting the TCP on the end-effector, the robotic work cell can improve its flexibility, and a necessary repositioning of the robot base can be avoided. This adaptation of the TCP could be implemented in practice using multiple tools with different TCP and a tool changer. In order to find the optimal TCP pose, a fast path planner generating all possible solutions is developed, considering axis ranges over 180° and movements through singular configurations. Furthermore, this algorithm is formulated in a general way and applies to finding the optimal robot base placement while taking a set of paths into account for optimization. To the best of the authors' knowledge, such an approach has not been proposed in the literature yet.

4.2 Fast Kinematic Joint-Space Path Planner

In this section, a fast joint-space path planner is proposed which considers all feasible kinematics solutions. Additionally, this planner is capable of generating multiple joint-space paths and planning through kinematic singularities. Based on the output of the joint-space path planner, the optimal TCP of the end-effector is determined as the solution of an optimization problem in Section 4.3, considering joint-space motions and kinematic robot limits.

Algorithm 1 Joint-space path planner

```
Require: $H_{\mathcal{P}}^{\mathcal{M}} = \{\mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}, k = 1, \dots, n_{\mathrm{m}}\}$
  INITIALIZE:
  for $k = 1$ to $n_{\mathrm{m}}$ do
    $\mathbf{H}_{\mathcal{P},k}^{\mathcal{T}} \leftarrow \mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}$
    $Q_k \leftarrow \mathbf{h}_{\mathrm{H}}^{-1}\left(\mathbf{H}_{\mathcal{P},k}^{\mathcal{T}}\right)$
    $D_k \leftarrow \{1, \dots, |Q_k|_{\mathrm{c}}\}$
  end for
  SEARCH CONTINUOUS JOINT-SPACE PATHS:
  $C \leftarrow D_1$
  $\forall c \in C : P_c \leftarrow \{\mathbf{q}_{c,1}\}$
  for $k = 2$ to $n_{\mathrm{m}}$ do
    for all $P_c$, $c \in C$ do
      $d_{\min} \leftarrow \arg\min_{d \in D_k} \|\mathbf{q}_{d,k} - \mathrm{lastElement}(P_c)\|_2$
      if $\|\mathbf{q}_{d_{\min},k} - \mathrm{lastElement}(P_c)\|_2 < q_{\mathrm{dist}}$ then
        $P_c \leftarrow P_c \cup \{\mathbf{q}_{d_{\min},k}\}$
      else
        $C \leftarrow C \setminus c$
      end if
    end for
  end for
```

Major parts of this section have been published in the author's work [8] and are adapted for this thesis.

The objective of the path planner is to find all feasible continuous joint-space paths P for a given manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$, see Section 2.2.1. The joint-space path planner is summarized in Algorithm 1.

The industrial robot must consecutively follow the manufacturing path poses $\mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}$ contained in $H_{\mathcal{P}}^{\mathcal{M}}$. The augmented forward kinematics introduced in Section 2.1.2 compute the robot configuration depending on whether the tool is attached to the end-effector (2.9)

$$\mathbf{H}_{\mathcal{P},k}^{\mathcal{T}}(\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}}) = \mathbf{H}_{\mathcal{P}}^{\mathcal{W}}\mathbf{H}_{\mathcal{W}}^{\mathcal{B}}\mathbf{H}_{\mathcal{B}}^{\mathcal{E}}(\mathbf{q}_k)\mathbf{H}_{\mathcal{E}}^{\mathcal{T}}(\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}}) \tag{4.1}$$

or is stationary (2.10)

$$\mathbf{H}_{\mathcal{P},k}^{\mathcal{T}}(\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{P}}) = \left(\mathbf{H}_{\mathcal{E}}^{\mathcal{P}}(\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{P}})\right)^{-1}\left(\mathbf{H}_{\mathcal{B}}^{\mathcal{E}}(\mathbf{q}_k)\right)^{-1}\mathbf{H}_{\mathcal{B}}^{\mathcal{W}}\mathbf{H}_{\mathcal{W}}^{\mathcal{T}}\,. \tag{4.2}$$

The augmented forward kinematics (4.1) is parametrized with the pose $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}$ of the tool mounting w.r.t. the end-effector, i.e., the position $\mathbf{p}^{\mathcal{T}}_{\mathcal{E}}$ and orientation given as Roll-Pitch-Yaw angles $\boldsymbol{\varphi}^{\mathcal{T}}_{\mathcal{E}}$ from (2.12b). The latter is indicated by the index φ in $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}$. Similarly, if the tool is stationary, the workpiece is mounted on the end-effector, and the augmented forward kinematics (4.2) is parametrized with the pose $\mathbf{y}^{\mathcal{P}}_{\mathcal{E},\varphi}$ of the workpiece mounting w.r.t. the end-effector. With these parametrizations, the geometric dimensions of the mechanical attachment on the end-effector can be found by using the poses $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}$ or $\mathbf{y}^{\mathcal{P}}_{\mathcal{E},\varphi}$ as optimization variables in the optimization problem.

In the initialization phase of Algorithm 1, the inverse kinematics (2.17) is applied to each pose of the manufacturing path $\mathbf{H}^{\mathcal{T}}_{\mathcal{P},k}$ of the augmented forward kinematics (4.1) or (4.2),

$$Q_k = \mathbf{h}_{\mathrm{H}}^{-1}\left(\mathbf{H}^{\mathcal{T}}_{\mathcal{P},k}\right) \ , \tag{4.3}$$

for $k = 1, \dots, n_{\mathrm{m}}$, to obtain the sets Q_k containing $e_k = |Q_k|_{\mathrm{c}} = |D_k|_{\mathrm{c}} \in \mathbb{N}$ inverse kinematics solutions $\mathbf{q}_{d,k} \in Q_k,\ d \in D_k$. Here, D_k denotes the index set of the solutions in Q_k, which only contains feasible solutions, i.e., all inverse kinematics solutions that satisfy the kinematic constraints, like mechanical joint limits. If the DoF of the robot is greater than the process DoF, i.e., the dimension of the task space in (2.12), the sets Q_k are sampled according to (3.64), which results in the sets Q^{f}_k, see, e.g., [93]. The set Q_k is also used to cover this case in the following.

Initially, $C = D_1$ is chosen, i.e., at the beginning, all starting solutions $c \in D_1$ are assumed to yield a feasible sequence P_c. Then, starting with $c \in D_1$, all possible sequences of feasible continuous joint-space paths $P_c, c \in C \subseteq D_1$ are determined, where C is the index set of feasible continuous joint-space paths. The first element of each sequence P_c corresponds to the inverse kinematics solution $\mathbf{q}_{c,1}$, $c \in D_1$. The second element of each sequence P_c is obtained by minimizing the Euclidean distance $\|\mathbf{q}_{d,2} - \mathbf{q}_{c,1}\|_2$ for all $d \in D_2$. The index of the solution with minimum distance is denoted by $d_{\min}$. If $\|\mathbf{q}_{d_{\min},2} - \mathbf{q}_{c,1}\|_2 < q_{\mathrm{dist}}$, with a user-defined threshold q_{dist}, the second element of the sequence P_c is $\mathbf{q}_{d_{\min},2}$, otherwise P_c is discarded, and a new index set C is defined as $C = D_1 \setminus c$. This procedure is now sequentially applied to all further path points $k = 3, \dots, n_{\mathrm{m}}$. With this approach, multiple continuous joint-space paths P_c, $c \in C$, are found for a given task-space manufacturing path $H^{\mathcal{M}}_{\mathcal{P}}$. This joint-space path planner can consider movements through kinematic singularities and solutions resulting from wide turning axis ranges since all feasible joint-space configurations of each path pose $\mathbf{H}^{\mathcal{T}}_{\mathcal{P},k}$ are accounted for to

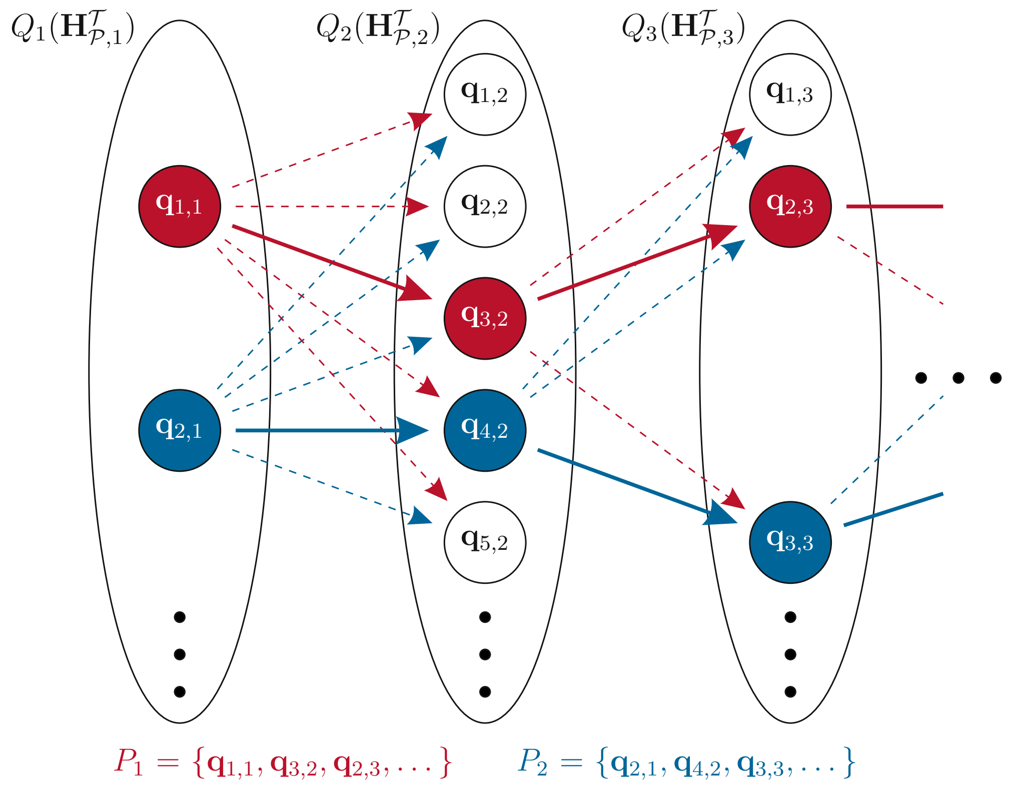

$Q_1(\mathbf{H}^{\mathcal{T}}_{\mathcal{P},1})$
$Q_2(\mathbf{H}^{\mathcal{T}}_{\mathcal{P},2})$
$Q_3(\mathbf{H}^{\mathcal{T}}_{\mathcal{P},3})$
$\mathbf{q}_{1,1}$
$\mathbf{q}_{2,1}$
$\mathbf{q}_{1,2}$
$\mathbf{q}_{2,2}$
$\mathbf{q}_{3,2}$
$\mathbf{q}_{4,2}$
$\mathbf{q}_{5,2}$
$\mathbf{q}_{1,3}$
$\mathbf{q}_{2,3}$
$\mathbf{q}_{3,3}$
$P_1 = \{\mathbf{q}_{1,1}, \mathbf{q}_{3,2}, \mathbf{q}_{2,3}, \dots\}$
$P_2 = \{\mathbf{q}_{2,1}, \mathbf{q}_{4,2}, \mathbf{q}_{3,3}, \dots\}$

Figure 4.1: Example of the path planner searching for continuous paths. The solid arrows represent transitions with the shortest Euclidean distance, while the dashed arrows represent discarded transitions between feasible joint configurations; adapted from [8].

find the path with the least joint movement. A joint-space controller has to be used if the robot is moved through a singular configuration.

A visual representation of the search for continuous join-space paths is shown in Fig. 4.1. In this figure, the nodes represent feasible joint configurations. The shortest Euclidean distances are indicated as arrows with solid lines, and the dashed lines represent discarded transitions.

4.3 Optimal TCP Placement

This section describes the algorithm to determine the optimal pose $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}$ of the TCP frame $\mathcal{T}$ or the optimal pose $\mathbf{y}^{\mathcal{P}}_{\mathcal{E},\varphi}$ of the workpiece frame $\mathcal{P}$ w.r.t. the end-effector frame $\mathcal{E}$. By adapting the end-effector mounting using an optimization-based approach, redesigning the robot work cell and repositioning the robot can be avoided. First, a set of paths is introduced as input for the optimization. Second, the objective function and the optimization criteria are presented, and third, the optimization algorithm is described in detail. Finally, the algorithm is adapted for optimal robot base placement. In the following, only the TCP pose w.r.t. the end-effector $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}$ is used.

Major parts of this section have been published in the author's work [8] and are adapted for this thesis.

4.3.1 Set of Paths

The set of manufacturing paths $\{H^{\mathcal{M}}_{\mathcal{P},1}, \cdots, H^{\mathcal{M}}_{\mathcal{P},n_\mathrm{p}}\}$ with the number of paths n_p is considered as input for the algorithm. The joint-space path-planning algorithm from Section 4.2 is executed for all manufacturing paths $H^{\mathcal{M}}_{\mathcal{P},i}$, $i = 1, \dots, n_\mathrm{p}$, for a given TCP pose $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}$, resulting in the feasible joint-space paths $\{P^i_c \mid c \in C_i\}, i = 1, \dots, n_\mathrm{p}$, for all manufacturing paths. Note that C_i is the index set for the feasible solutions of the corresponding manufacturing path i. Each manufacturing path $H^{\mathcal{M}}_{\mathcal{P},i}$ consists of $n_{\mathrm{m},i}$ poses, cf. (4.1) or (4.2).

4.3.2 Objective Function and Optimization Criteria

The objective function $f_\mathrm{p}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$ computes a scalar measure for a given TCP pose $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}$. The results of the joint-space path planner from Section 4.2 are used as qualitative and quantitative measure in the objective function of the optimization.

The objective function $f_\mathrm{p}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$ consists of four terms

$$f_\mathrm{p}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}) = \frac{\nu_1 f_{\mathrm{p},1}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}) + \nu_2 f_{\mathrm{p},2}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}) + \nu_3 f_{\mathrm{p},3}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}) + \nu_4 f_{\mathrm{p},4}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})}{\nu_1 + \nu_2 + \nu_3 + \nu_4} , \tag{4.4}$$

where the coefficients $\nu_j > 0$, $j = 1, \dots, 4$, are manually tuned weights of the terms $f_{\mathrm{p},j}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$. The terms $f_{\mathrm{p},j}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$ read as follows:

- The term $f_{\mathrm{p},1}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$ accounts for the number of feasible inverse kinematics solutions $e_{k,i}$ for every path pose $k = 1, \dots, n_{\mathrm{m},i}$ of every manufacturing path

$H^{\mathcal{M}}_{\mathcal{P},i}$, $i = 1, \dots, n_{\mathrm{p}}$, i.e.,

$$f_{\mathrm{p},1}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}) = \frac{1}{e_{\max} n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} \frac{1}{n_{\mathrm{m},i}} \sum_{k=1}^{n_{\mathrm{m},i}} e_{k,i} \ . \tag{4.5}$$

The pre-factor in (4.5) normalizes the function. The constant $e_{\max}$ denotes the maximum number of inverse kinematics solutions for the specific robot. For non-redundant robots this constant is the number of distinguishable joint-space solutions and for redundant robots the infinite number of possible joint-pace solutions is sampled, see Section 2.1.4. The term (4.5) locally generates a gradient in a meaningful direction.

- The function $f_{\mathrm{p},2}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$ corresponds to the number of feasible continuous joint-space solutions for each manufacturing path $H^{\mathcal{M}}_{\mathcal{P},i}$, $i = 1, \dots, n_{\mathrm{p}}$, and is chosen as

$$f_{\mathrm{p},2}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}) = \frac{1}{e_{\max} n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} |C_i|_{\mathrm{c}} \ . \tag{4.6}$$

  The higher the objective term (4.6), the more manufacturing paths are continuously executable by the robot.

- By including $f_{\mathrm{p},3}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$ in the objective function, the joint movement between consecutive manufacturing path poses is minimized for continuous paths. The objective term $f_{\mathrm{p},3}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$ is defined as

$$f_{\mathrm{p},3}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}) = \frac{1}{n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} \frac{1}{(n_{\mathrm{m},i} - 1)\,|C_i|_{\mathrm{c}}} \sum_{c \in C_i} \sum_{k=1}^{n_{\mathrm{m},i}-1} \|\mathbf{q}^i_{c,k+1} - \mathbf{q}^i_{c,k}\|_2^{-1} \ , \tag{4.7}$$

  where $\mathbf{q}^i_{c,k}$ is the k^{th} element of the sequence P^i_c. Again, the pre-factors normalize the objective term (4.7).

- A robot setup is more flexible if it can execute paths for which the TCP was not optimized for. Hence, the last term $f_{\mathrm{p},4}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$ considers the reserves of the joint positions to the respective mechanical joint limits $q_{h,\min}$ and $q_{h,\max}$, $h = 1, \dots, n$, during the robot motion and is given by

$$f_{\mathrm{p},4}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}) = \frac{1}{n_{\mathrm{p}}} \sum_{i=1}^{n_{\mathrm{p}}} \frac{1}{n_{\mathrm{m},i}\,|C_i|_{\mathrm{c}}} \sum_{c \in C_i} \sum_{k=1}^{n_{\mathrm{m},i}} \|\bar{\mathbf{q}}(\mathbf{q}^i_{c,k})\|_2^{-1} \ , \tag{4.8}$$

with $\mathbf{q}_{c,k}^i \in P_c^i$, where the vector-valued function $\bar{\mathbf{q}}^{\mathrm{T}}(\mathbf{q}) = \begin{bmatrix}\bar{q}_1 & \cdots & \bar{q}_n\end{bmatrix}$ corresponds to (3.66) $\bar{q}_h = \frac{2q_h - (q_{h,\max} + q_{h,\min})}{q_{h,\max} - q_{h,\min}}$, $h = 1, \dots, n$.

4.3.3 Optimization Algorithm

Using the objective function (4.4), the optimization problem for the optimal TCP placement $\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}*}$ reads as

$$\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}*} = \underset{\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}} \in \mathbb{R}^6}{\arg\max} \; f_{\mathrm{p}}(\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}}) \tag{4.9a}$$

$$\text{s.t. } \mathbf{y}_{\mathcal{E},\varphi,\min}^{\mathcal{T}} \leq \mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}} \leq \mathbf{y}_{\mathcal{E},\varphi,\max}^{\mathcal{T}} \quad , \tag{4.9b}$$

with the boundaries $(\mathbf{y}_{\mathcal{E},\varphi,\min}^{\mathcal{T}})^{\mathrm{T}} = \left[(\mathbf{p}_{\mathcal{E},\min}^{\mathcal{T}})^{\mathrm{T}} \; (\boldsymbol{\varphi}_{\mathcal{E},\min}^{\mathcal{T}})^{\mathrm{T}}\right]$ and $(\mathbf{y}_{\mathcal{E},\varphi,\max}^{\mathcal{T}})^{\mathrm{T}} = \left[(\mathbf{p}_{\mathcal{E},\max}^{\mathcal{T}})^{\mathrm{T}} \; (\boldsymbol{\varphi}_{\mathcal{E},\max}^{\mathcal{T}})^{\mathrm{T}}\right]$, see (2.12b) and Fig. 2.3. Note that $f_{\mathrm{p}}(\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}})$ is not continuously differentiable. Furthermore, the proposed path-planning algorithm is computationally expensive, particularly for large path sets. Therefore, a surrogate optimization approach is chosen. Specifically, the MATLAB implementation *surrogateopt* is used to solve (4.9), which uses an approximation of the objective function to search for the global optimum [96]. Additionally, the parallel nature of this algorithm is utilized to reduce the computation time.

4.3.4 Robot Base Placement

By considering a new optimization variable, the problem (4.9) is adapted in a straightforward way for optimal robot base placement, which is described in this section. To this end, the homogeneous transformation $\mathbf{H}_{\mathcal{P},k}^{\mathcal{T}}$ in (4.1) or (4.2) is reformulated as

$$\mathbf{H}_{\mathcal{P},k}^{\mathcal{T}}(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}}) = \mathbf{H}_{\mathcal{P}}^{\mathcal{W}} \left(\mathbf{H}_{\mathcal{B}}^{\mathcal{W}}(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}})\right)^{-1} \mathbf{H}_{\mathcal{B}}^{\mathcal{E}}(\mathbf{q}_k) \mathbf{H}_{\mathcal{E}}^{\mathcal{T}} \;, \tag{4.10}$$

or

$$\mathbf{H}_{\mathcal{P},k}^{\mathcal{T}}(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}}) = \mathbf{H}_{\mathcal{P}}^{\mathcal{E}} \left(\mathbf{H}_{\mathcal{B}}^{\mathcal{E}}(\mathbf{q}_k)\right)^{-1} \mathbf{H}_{\mathcal{B}}^{\mathcal{W}}(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}}) \mathbf{H}_{\mathcal{W}}^{\mathcal{T}} \;, \tag{4.11}$$

with the 2D robot base position $(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}})^{\mathrm{T}} = \begin{bmatrix}x_{\mathcal{B}}^{\mathcal{W}} & y_{\mathcal{B}}^{\mathcal{W}}\end{bmatrix}$ as new optimization variable and a constant height $z_{\mathcal{B}}^{\mathcal{W}} = 0.193\,\mathrm{m}$. The optimization variable $\mathbf{x}_{\mathcal{B}}^{\mathcal{W}}$ is the position of the world frame $\mathcal{W}$ w.r.t. the robot base frame $\mathcal{B}$. The optimization problem

(4.9) is adapted in the form

$$\mathbf{x}_{\mathcal{B}}^{\mathcal{W}*} = \underset{\mathbf{x}_{\mathcal{B}}^{\mathcal{W}} \in \mathbb{R}^2}{\arg\max} \; f_{\mathrm{p}}(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}}) \tag{4.12a}$$

$$\text{s.t. } \mathbf{x}_{\mathcal{B},\min}^{\mathcal{W}} \leq \mathbf{x}_{\mathcal{B}}^{\mathcal{W}} \leq \mathbf{x}_{\mathcal{B},\max}^{\mathcal{W}} \quad , \tag{4.12b}$$

with the minimum and maximum positions $\mathbf{x}_{\mathcal{B},\min}^{\mathcal{W}}$ and $\mathbf{x}_{\mathcal{B},\max}^{\mathcal{W}}$ in terms of the available workspace. The optimization (4.12) is again solved with the surrogate optimization approach. Every other pose, position, or orientation of the forward kinematics can be found using a different optimization variable and reformulating the optimization problem.

4.4 Simulation Results

In this section, the optimal placement of the TCP and the robot base is applied to a robot work cell performing a trim application. A robot moves a trimming tool along the edge of a shoe sole to trim excess material arising from injection molding, which is a common task in semi-automated shoe manufacturing. The kinematic arrangement is shown in Fig. 2.1a. In this application, the 6-DoF robot KUKA Cybertech KR8 R1620 with a spherical wrist is used, see Appendix A.2. The range of joint 3 is constrained by the workshop floor and the energy chain of the tool from $q_{3,\min} = -137°$ to $q_{3,\min} = 60°$, cf. Tab. A.2. The robot base and the TCP must be placed such that 22 trim paths for the left and the right shoe, i.e., 44 paths, are executable. Two of the trim paths are shown in Fig. 4.2, where the robot base frame $\mathcal{B}$ is located at the origin. The trim paths are described in the workpiece frame $\mathcal{P}$, which is also the world frame $\mathcal{W}$, i.e., $\mathbf{H}_{\mathcal{P}}^{\mathcal{W}} = \mathbf{I}$. All feasible joint-space solutions for one trim path with $n_{\mathrm{m}} = 638$ manufacturing path frames are found in less than 100 ms using the fast joint-space path planner from Section 4.2 on an INTEL Core i7-8700K with 16 GB RAM. A video of the experimental trimming process is provided at `www.acin.tuwien.ac.at/9c5f` .

Major parts of this section have been published in the author's work [8] and are adapted for this thesis.

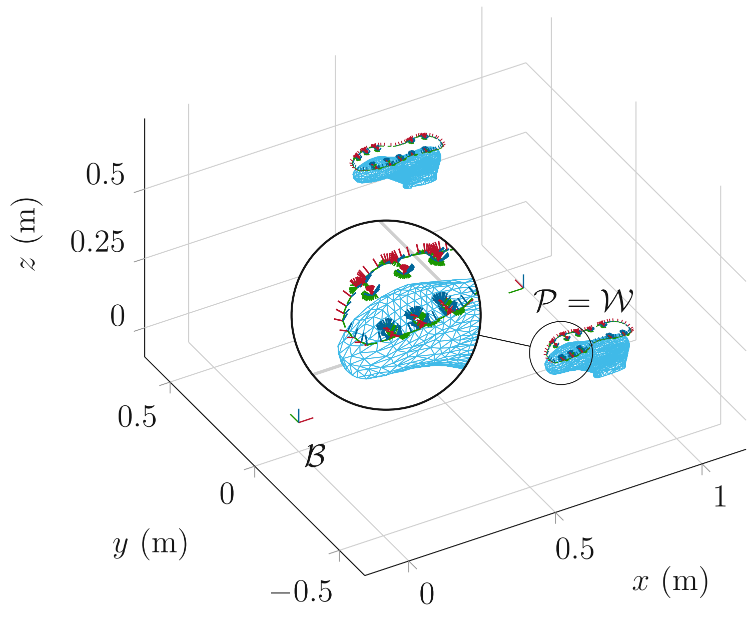


Figure 4.2: Example trim paths for the left and the right shoe given in the workpiece frame $\mathcal{P}$, which equals the world frame $\mathcal{W}$; adapted from [8].

4.4.1 Optimal Robot Base Placement

In this section, the placement of the robot base for the trim application is investigated using the optimization problem (4.12), which is analogous to the workpiece placement. The homogeneous transformation $\mathbf{H}_{\mathcal{E}}^{\mathcal{T}}$ in (4.10) is taken from the existing tool

$$(\mathbf{y}_{\mathcal{E},\varphi,\mathrm{ex}}^{\mathcal{T}})^{\mathrm{T}} = \begin{bmatrix} -0.066\,\mathrm{m} & 0 & 0.269\,\mathrm{m} & \pi & 0.719\,\mathrm{rad} & -\pi \end{bmatrix} . \tag{4.13}$$

In order to illustrate and discuss the objective function $f_{\mathrm{p}}(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}})$ for the surrogate optimization algorithm, a MONTE-CARLO simulation in the admissible range

$$(\mathbf{x}_{\mathcal{B},\min}^{\mathcal{W}})^{\mathrm{T}} = \begin{bmatrix} 0 & -1\,\mathrm{m} \end{bmatrix} \text{ and} \tag{4.14a}$$

$$(\mathbf{x}_{\mathcal{B},\max}^{\mathcal{W}})^{\mathrm{T}} = \begin{bmatrix} 1.5\,\mathrm{m} & 1\,\mathrm{m} \end{bmatrix} \tag{4.14b}$$

is performed and shown in Fig. 4.3. In this figure, dark blue areas represent high objective function values, and red areas indicate low values. Furthermore, Fig. 4.3 reveals three distinct plateaus originating from the objective term $f_{\mathrm{p},2}(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}})$ from

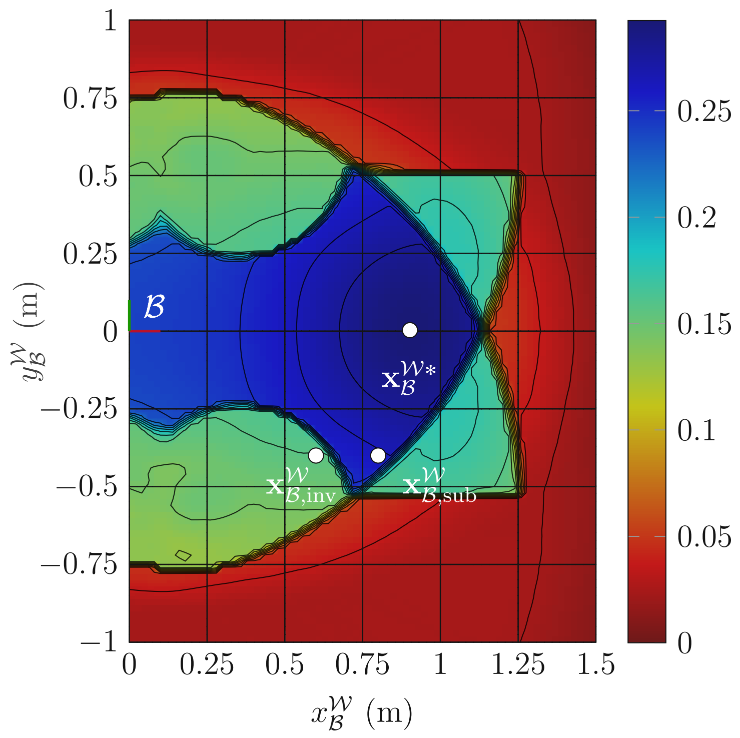


Figure 4.3: Grid-based MONTE-CARLO simulation with contour lines with the optimal robot base placement $\mathbf{x}_{\mathcal{B}}^{\mathcal{W}*}$; adapted from [8].

(4.6). For a robot base placement $\mathbf{x}_{\mathcal{B}}^{\mathcal{W}}$ in the blue area, the algorithm can solve the joint-space paths for all trim paths. The green area marks positions where only a subset of trim paths can be solved, and in the red area, none of the trim paths are executable by the robot.

Thus, with the boundaries (4.14), the optimal robot base placement reads as

$$(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}*})^{\mathrm{T}} = \begin{bmatrix} 0.902\,\mathrm{m} & 0.003\,\mathrm{m} \end{bmatrix} . \tag{4.15}$$

This optimal point $\mathbf{x}_{\mathcal{B}}^{\mathcal{W}*}$ is found by applying the surrogate optimization to (4.12), in which the objective function (4.4) is evaluated about 25 times. The weights of (4.4) are empirically tuned and are chosen as $\nu_1 = 5, \nu_2 = 10, \nu_3 = 0.1$, and $\nu_4 = 1$. If the robot base is placed at this optimal point $\mathbf{x}_{\mathcal{B}}^{\mathcal{W}*}$, all 44 trim paths can be executed with the existing tool with the TCP (4.13) mounted on the robot.

For a single trim path, Fig. 4.4 shows a comparison of the joint-space paths for two different robot base placements $\mathbf{x}_{\mathcal{B}}^{\mathcal{W}}$, i.e., the optimal point $\mathbf{x}_{\mathcal{B}}^{\mathcal{W}*}$ from (4.15) shown in blue and a suboptimal point $(\mathbf{x}_{\mathcal{B},\mathrm{sub}}^{\mathcal{W}})^{\mathrm{T}} = \begin{bmatrix} 0.8\,\mathrm{m} & -0.4\,\mathrm{m} \end{bmatrix}$ depicted in red,

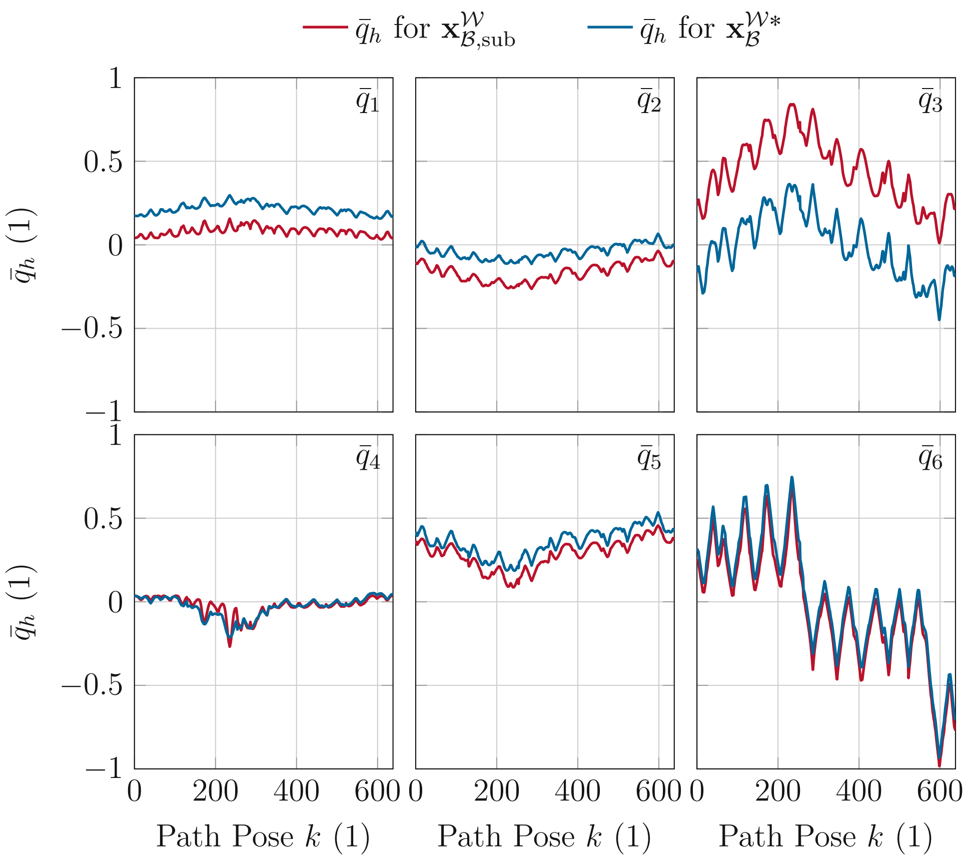


Figure 4.4: Joint-space paths normalized to the respective joint limit, see (3.66), for the optimal robot base placement $(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}*})^{\mathrm{T}} = \begin{bmatrix}0.902\,\mathrm{m} & 0.003\,\mathrm{m}\end{bmatrix}$ in blue and for the suboptimal placement $(\mathbf{x}_{\mathcal{B},\text{sub}}^{\mathcal{W}})^{\mathrm{T}} = \begin{bmatrix}0.8\,\mathrm{m} & -0.4\,\mathrm{m}\end{bmatrix}$ in red; adapted from [8].

see Fig. 4.3. The joint angles are normalized with (3.66) in the range $[\mathbf{q}_{\min}, \mathbf{q}_{\max}]$ according to Tab. A.2 with the reduced joint limit of $q_{3,\min} = 60°$. Although all desired trim paths can be executed for both placements, a distinct difference is visible in the joint-space paths. Particularly for joint q_3, the joint motions for the suboptimal placement provide less reserves to the joint limits of the robot. This is taken into account by the objective function term $f_{\mathrm{p},4}(\mathbf{x}_{\mathcal{B}}^{\mathcal{W}})$ in (4.8), which penalizes joint-space path points close to the joint limits. Furthermore, Fig. 4.4 reveals that the wide turning range of joint 6 from $q_{6,\min} = -350°$ to $q_{6,\max} = 350°$ is necessary to perform the continuous motion along the complex trim path.

4.4.2 Optimal TCP Placement

In this section, the optimal TCP placement is investigated for all 44 trim paths, considering the suboptimal robot placement $(\mathbf{x}^{\mathcal{W}}_{\mathcal{B},\mathrm{inv}})^{\mathrm{T}} = \begin{bmatrix}0.6\,\mathrm{m} & -0.4\,\mathrm{m}\end{bmatrix}$. With this placement, only a subset of all trim paths can be executed with the available tool $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi,\mathrm{ex}}$ introduced in (4.13), see Fig. 4.3. The optimization problem (4.9) is solved with the boundaries

$$(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi,\min})^{\mathrm{T}} = \begin{bmatrix}-0.2\,\mathrm{m} & -0.2\,\mathrm{m} & 0.2\,\mathrm{m} & -\pi & -\frac{\pi}{2} & -\pi\end{bmatrix} \tag{4.16}$$

and

$$(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi,\max})^{\mathrm{T}} = \begin{bmatrix}0.2\,\mathrm{m} & 0.2\,\mathrm{m} & 0.4\,\mathrm{m} & \pi & \frac{\pi}{2} & \pi\end{bmatrix}\;, \tag{4.17}$$

which arise from the mechanical limits for the construction of the trim tool and the allowed specifications of the robot. The weights ν_j, $j = 1, \ldots, 4$, are chosen as in Section 4.4.1.

The optimal TCP pose $\mathbf{y}^{\mathcal{T}*}_{\mathcal{E},\varphi}$ for the optimization problem (4.9) is found as

$$\begin{aligned}(\mathbf{y}^{\mathcal{T}*}_{\mathcal{E},\varphi})^{\mathrm{T}} &= \begin{bmatrix}-0.079\,\mathrm{m} & 0.039\,\mathrm{m} & 0.309\,\mathrm{m} & 2.849\,\mathrm{rad} & 0.805\,\mathrm{rad} & 3.040\,\mathrm{rad}\end{bmatrix}\\ &= \begin{bmatrix}(\mathbf{p}^{\mathcal{T}*}_{\mathcal{E}})^{\mathrm{T}} & (\boldsymbol{\varphi}^{\mathcal{T}*}_{\mathcal{E}})^{\mathrm{T}}\end{bmatrix}\;,\end{aligned} \tag{4.18}$$

given by the optimal TCP position $\mathbf{p}^{\mathcal{T}*}_{\mathcal{E}}$ and orientation $\boldsymbol{\varphi}^{\mathcal{T}*}_{\mathcal{E}}$ as Roll-Pitch-Yaw angles. With this optimal TCP pose, all trim paths can be realized, and a repositioning of the robot can be avoided. In Fig. 4.5, the poses of the existing tool $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi,\mathrm{ex}}$ from (4.13) and optimal tool $\mathbf{y}^{\mathcal{T}*}_{\mathcal{E},\varphi}$ from (4.18) are compared, for which different tool mounts are used. Figure 4.6 illustrates the updated MONTE-CARLO simulation from Fig. 4.3 with the optimal TCP pose $\mathbf{y}^{\mathcal{T}*}_{\mathcal{E},\varphi}$ from (4.18). In this figure, the investigated point $\mathbf{x}^{\mathcal{W}}_{\mathcal{B},\mathrm{inv}}$ is now located in the blue area, which again refers to the area where all desired trim paths can be executed by the robot. This point was located in the green area in Fig. 4.3 using the existing tool with the TCP pose $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi,\mathrm{ex}}$.

In order to validate this result qualitatively, the admissible area of the TCP pose is investigated by visualizing the objective function $f_{\mathrm{p}}(\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi})$ from (4.4) for the 6-dimensional space $\mathbf{y}^{\mathcal{T}}_{\mathcal{E},\varphi}$ using two 3-dimensional MONTE-CARLO simulations for the position and orientation separately. In Fig. 4.7, the MONTE-CARLO simulation of the Cartesian position $\mathbf{p}^{\mathcal{T}}_{\mathcal{E}}$ is shown, whereas the orientation of the TCP $\boldsymbol{\varphi}^{\mathcal{T}}_{\mathcal{E}}$ is

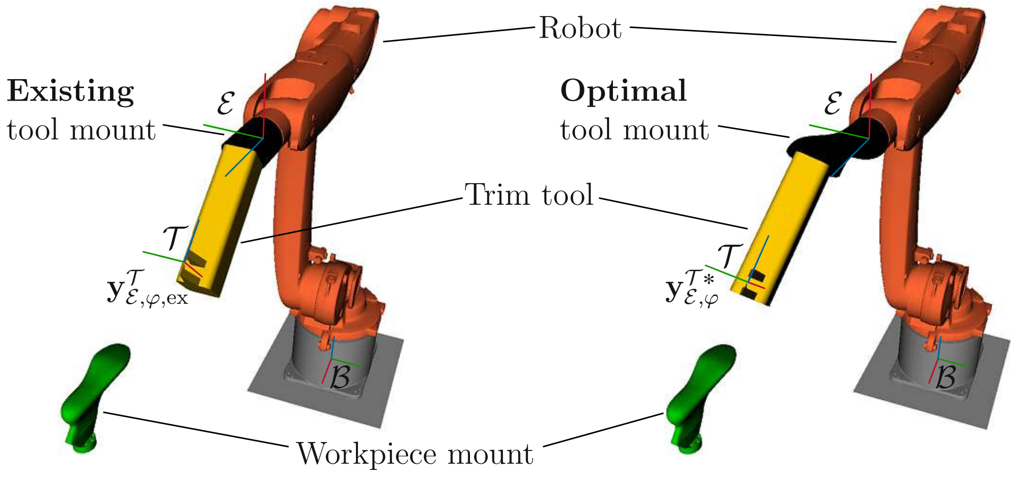


Figure 4.5: Comparison of the pose of the (a) existing tool $\mathbf{y}_{\mathcal{E},\varphi,\mathrm{ex}}^{\mathcal{T}}$ (b) optimal tool $\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}*}$ and their corresponding tool mounts.

fixed to the optimum value $\boldsymbol{\varphi}_{\mathcal{E}}^{\mathcal{T}*}$. Note that only TCP positions $\mathbf{p}_{\mathcal{E}}^{\mathcal{T}}$ for which all trim paths can be executed are plotted. The blue dots mark high values of the objective function and low values are depicted in red. A gradient in the objective function is visible, which guides the optimization algorithm towards the optimal position $\mathbf{p}_{\mathcal{E}}^{\mathcal{T}*}$. Similarly, a Monte-Carlo simulation of the Roll-Pitch-Yaw angles $\boldsymbol{\varphi}_{\mathcal{E}}^{\mathcal{T}}$ using the optimal TCP position $\mathbf{p}_{\mathcal{E}}^{\mathcal{T}*}$ from (4.18) is depicted in Fig. 4.8. Again, only data points $\boldsymbol{\varphi}_{\mathcal{E}}^{\mathcal{T}}$ are shown with which the robot can execute all 44 desired trim paths. The feasible areas in Fig. 4.8 are more distinct compared to Fig. 4.7 for the TCP position $\mathbf{p}_{\mathcal{E}}^{\mathcal{T}}$. The apparent four separated areas are connected due to the periodicity of the Roll-Pitch-Yaw angles $\varphi_{\mathcal{E},z}^{\mathcal{T}}$ and $\varphi_{\mathcal{E},x}^{\mathcal{T}}$. Therefore, the objective function $f_{\mathrm{p}}(\mathbf{y}_{\mathcal{E},\varphi}^{\mathcal{T}})$ exhibits a distinct maximum which is found by the surrogate optimization.

4.5 Conclusions

In order to improve the flexibility of robot work cells, an algorithm for the optimal TCP and robot base placement for a given set of complex manufacturing paths is presented. To this end, a fast joint-space path planner is developed, which calculates all feasible continuous joint-space paths. This planner considers all possible inverse kinematics solutions, including wide axis ranges, and allows the planning of

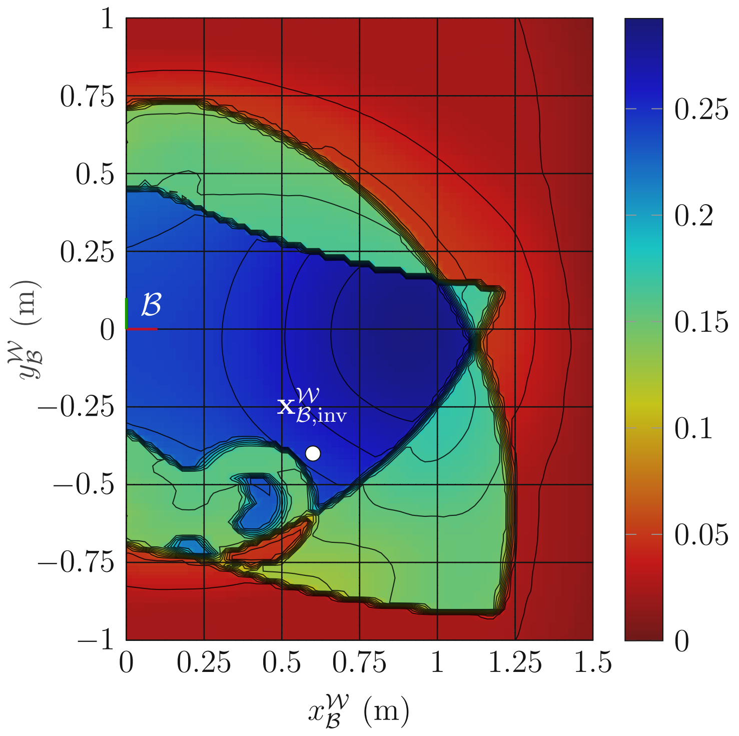


Figure 4.6: Grid-based MONTE-CARLO simulation with contour lines using the TCP pose $\mathbf{y}^{\mathcal{T}*}_{\mathcal{E},\varphi}$; adapted from [8].

robot motions through singular points. The optimization-based placement of the TCP and the robot base is performed using a surrogate optimization and systematically considers the reserves to the respective joint limits. The proposed algorithm is applied in a trim application in shoe manufacturing to find the optimal robot base placement and the optimal TCP placement in the case of a suboptimally placed robot base. This work shows that, in some cases, it is sufficient to adapt the TCP of the robot tool on the end-effector instead of redesigning the robot work cell. Optimally placing the robot base also increases the reserves to the respective robot joint limits.

Major parts of this section have been published in the author's work [8] and are adapted for this thesis.

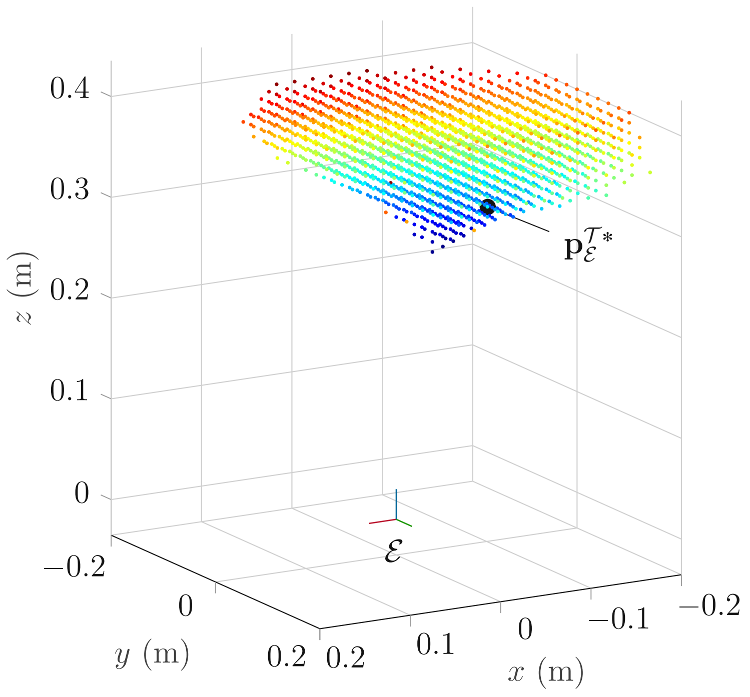


Figure 4.7: Grid-based MONTE-CARLO simulation of the TCP position $\mathbf{p}_{\mathcal{E}}^{\mathcal{T}}$ with optimum orientation $\boldsymbol{\varphi}_{\mathcal{E}}^{\mathcal{T}*}$. High objective term values are shown as blue dots; adapted from [8].

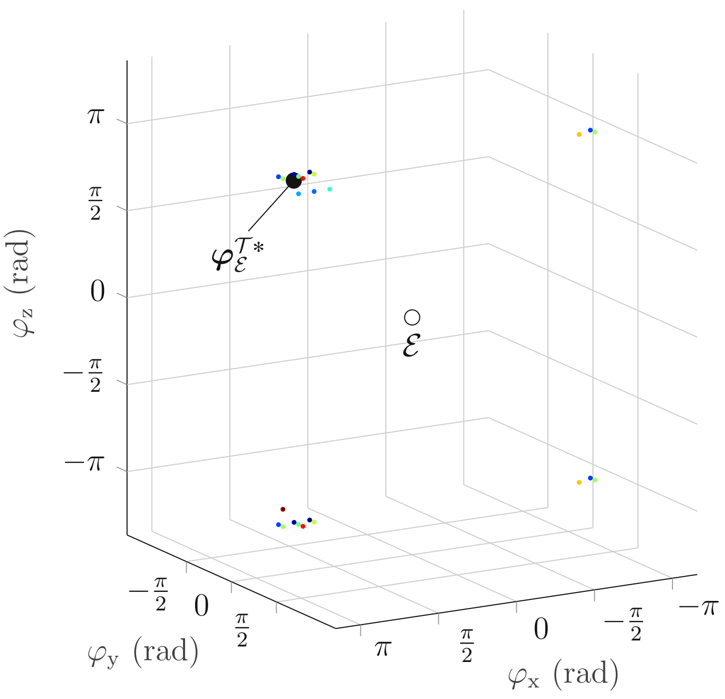


Figure 4.8: Grid-based MONTE-CARLO simulation of the TCP orientation $\boldsymbol{\varphi}_{\mathcal{E}}^{\mathcal{T}}$ with optimum position $\mathbf{p}_{\mathcal{E}}^{\mathcal{T}*}$. High objective term values are shown as blue dots; adapted from [8].

Chapter 5 Path Planning using Process Properties

The previous chapter presented the optimal TCP and robot base placement for a set of complex manufacturing paths. The location of the robot base and the TCP are crucial for designing a manufacturing line for a specific product. In a flexible production line, the product or manufacturing path changes frequently. If these manufacturing paths exceed the kinematic capabilities of the robotic setup, an adaption of the manufacturing line would be necessary, which, in some cases, can be achieved by adaptions of the TCP on the end-effector. Since the adaption of the TCP is usually only possible within geometrical and load limits, this approach cannot be applied in all situations to obtain executable manufacturing paths. In some cases, additional aspects of the production process must be considered to achieve the required flexibility.

In this chapter, the strict specifications of the manufacturing path can be softened by exploiting the *process properties*, which significantly increases the search space for path planning. This chapter discusses how the *process properties* of the manufacturing path, i.e., process tolerances, process windows, constraints, and redundant process DoF, can be used to solve complex path-planning problems. Incorporating those *process properties* into a path-planning framework leads to solutions for continuous robot motions for manufacturing paths, which cannot be found otherwise. Hence, an adaption of the robotic setup could be avoided. Therefore, considering these advanced path-planning capabilities allows us to handle highly complex industrial path-planning problems with significant flexibility.

First, a literature review of the broad field of path planning is discussed. Second, a general collision avoidance scheme is presented. The following section describes a general path-planning framework using this collision avoidance scheme. A cost function is designed based on the process properties, and an optimization is formulated to solve the path-planning problem. Next, the proposed path planner is used to determine an optimal robot base placement. Finally, the path planner is applied to two different manufacturing processes.

Major parts of this chapter have been published in the author's work [10] and are adapted for this thesis.

5.1 Literature Review

In this section, the state-of-the-art path-planning algorithms for industrial robots are summarized and discussed. An emphasis is laid on path-planning methods that consider tolerances, process windows, constraints, and redundant DoF, as introduced in Section 2.2.2.

Major parts of this section have been published in the author's work [10] and are adapted for this thesis.

In the literature, several path planning algorithms tailored to specific manufacturing processes have been published, which incorporate only the application-specific process properties, e.g., welding [32, 33], surface inspection [97], sanding [98], and chamfering [34]. In this work, a general deterministic path-planning framework is proposed, which systematically considers all process properties. Additionally, the planning algorithm takes into account collision avoidance and can plan through singular joint configurations, which is mostly avoided with different concepts in state-of-the-art path planners, e.g., [32, 36, 94, 99].

Note that this review focuses on path planning in joint space for a given Cartesian tool path. Generally, the tool path may be designed manually or automatically based on user inputs, e.g., with the concepts of Section 3.2, or be the result of an automatic path-generation algorithm based on the workpiece geometry, e.g., [24]. Path-planning algorithms are distinguished by the underlying method used, i.e., sampling or optimization-based, which are discussed in the following. Additionally, the pathwise inverse kinematics problem is addressed as a special form of the inverse kinematics problem.

Sampling-based Path Planners

A detailed survey of sampling-based path planners that systematically incorporate constraints is given in [92]. The most prominent library of sampling-based path planners is the *Open Motion Planning Library (OMPL)* [100], in which several sampling-based approaches are included, such as *Probabilistic Roadmap Methods (PRM)* and *Rapidly-exploring Random Trees (RRT)*. As the main focus of those approaches is solving point-to-point motion planning problems, the intermediate Cartesian path cannot be provided beforehand. Furthermore, sampling-based path planners mostly require smoothing as a post-processing step to remove redundant and jerky motions, e.g., [101]. Additionally, including multiple constraints, redundant DoF, and tolerances in the computation of a joint-space path increases the problem size drastically, which becomes infeasible to solve.

The popular DESCARTES algorithm [93] is a semi-constrained offline path planner that considers tolerances. Therein, for every Cartesian path point, multiple inverse kinematics solutions are computed using inverse kinematics solvers such as the *Kinematics and Dynamics Library (KDL)* [102] or *Inverse Kinematics Fast (IKFast)* [103]. Process tolerances are considered by sampling the allowed tolerance band with an equidistant grid, for which all corresponding inverse kinematics solutions are computed. Hence, the number of possible inverse kinematics solutions for each path point increases, particularly if tolerances are allowed for multiple DoF. This leads to high memory usage and high computation time when searching the optimal joint-space path in a graph search with the DIJKSTRA algorithm. In [104], an RRT-based planner computes collision-free paths without an explicit goal configuration. Instead, a goal region is described by workspace goal criteria, and a path with the lowest tolerance utilization is computed. Considering multiple constraints and costs in sampling-based methods is challenging, especially if optimal paths must be found, see [105]. In [106], a sampling-based planner is introduced to solve point-to-point problems. The planner incorporates kinematic and dynamic constraints to find optimal paths.

In many works, path planners tailored to a special industrial process are designed, e.g., for welding applications in [33]. Therein, the redundant process DoF of the tool is discretized, and a collision-free path is generated by a modified RRT* algorithm. Another example is presented in [32], where the planning of a complex welding path for a 6-DoF robot is performed in two steps. First, the joint configurations of the end-effector's start and end poses are determined. Second, the continuous robot motion between those configurations is computed. In this process, the rotation around the welding torch is a redundant process DoF, cf. Section 2.2.2. This DoF is sampled with an equidistant grid, and the corresponding joint configurations are computed. If no solution of the inverse kinematics problem is found, a different inclination angle of the torch is used. Furthermore, the joint movement defines costs to be minimized by the beam search algorithm while near-singular configurations of the robot are avoided. The above path planners [32, 33] sample the redundant DoF and, therefore, must reduce the search space by sampling to solve the planning problem.

Optimization-based Path Planners

In optimization-based path planners, an optimization scheme is employed to plan feasible and locally or globally optimal joint-space paths. Constraints can be systematically incorporated, e.g., [107]. A sequential convex optimization algorithm called *Trajectory Optimization for Motion Planning (Trajopt)* is presented in [108]. The *Trajopt* algorithm guesses one or multiple initial solutions for a trajectory and optimizes the joint-space path length while considering kinematic constraints. Process tolerances are not taken into account by this algorithm. The *covariant Hamiltonian optimization for motion planning (CHOMP)* algorithm in [109] is used to find smooth collision-free trajectories, and the *stochastic trajectory optimization for motion planning (STOMP)* algorithm in [110] is a gradient-free optimization with the possibility to include constraints. The above path planners compute an intermediate joint-space path between two configurations and, therefore, are suited for solving point-to-point planning problems only.

Optimization-based approaches can consider additional criteria for path planning, i.e., energy consumption [111, 112], time [113, 114, 115], time and jerk [116, 117], and a combination of multiple criteria [118, 119]. The dynamic model of the robot is also taken into account in [115, 120]. Additionally, the path optimization can be executed in a post-processing step. For example, in [121], the joint-space path length is minimized from an initial guess.

In [122], it is shown that using tolerances of the manufacturing path results in lower jerks in the planned trajectory. Nevertheless, not all DoF of the robot are used, and a distinct inverse kinematics solution is employed, i.e., the elbow-up solution, to find a trajectory with the lowest possible maximum jerk based on a heuristic approach.

A robotic layup task is investigated in [36] for a multi-robot scenario. The redundant process DoF of the roller tool is utilized to generate robot trajectories for this task. The process window is discretized and an optimization-based approach is applied to solve the trajectory planning problem while considering process constraints and avoiding singular configurations.

In [123], a time-optimal or energy-optimal joint-space trajectory is searched, where end-effector position and orientation tolerances are allowed within a tube in the task space, i.e., within a tolerance band. Although tolerances are considered, other process properties like process windows, (redundant) process DoF, and collisions are not systematically included.

Inverse Kinematics and Pathwise Inverse Kinematics

A general review of methods to compute the inverse kinematics is given in [124]. The inverse kinematics computes joint configurations for one specific end-effector pose, see (2.17). In [95], a continuous multivariate inverse function is described to find the global inverse kinematics of redundant manipulators, i.e., a unique joint configuration for a given end-effector pose despite the kinematic redundancy. The *TRAC-Inverse Kinematics (TRAC-IK)* solver, proposed in [125], considers tolerances in each Cartesian dimension and improves the calculation times compared to *KDL* [102]. Another widely used method to solve the inverse kinematics is to formulate an optimization problem, e.g., [126, 127]. In [128], a genetic algorithm to solve the inverse kinematics problem is proposed, where different soft constraints of the manipulator are weighted in the cost function.

Pathwise inverse kinematics refers to applying an inverse kinematics scheme to a Cartesian end-effector path to find one or more corresponding joint-space paths. Solving the pathwise inverse kinematics problem by computing the inverse kinematics for each path point independently generally leads to discontinuous joint-space paths. An example for an optimization-based scheme is the *trajectory optimization of a redundant manipulator (TORM)* in [129, 130], which computes trajectories for given end-effector paths. A two-stage gradient descent optimization minimizes the position and orientation error of the joint-space path w.r.t. the desired end-effector poses. New trajectories are repeatedly generated to avoid local minima. However, the process properties of the underlying manufacturing process cannot be considered systematically. Another work, which is concerned with solving pathwise inverse kinematics problems, is given in [131], where an anytime graph-based path planner minimizes a geometric task-space criterion, the so-called FRÉCHET distance. This FRÉCHET distance can be interpreted as the utilization of process tolerances. In [132], the inverse kinematics problem is solved with geometric task prioritization if the robot cannot execute the desired motion. In the path planner proposed in [132], the tolerances of the desired geometric task are implemented as soft constraints. Hence, no guarantees in terms of maximum deviations can be given.

The online path planner *Relaxed Inverse Kinematics (RelaxedIK)* [94] computes continuous joint-space paths as a sequence of optimization problems, where the solution of the previous path point is the initial guess for the subsequent optimization problem. The position and orientation deviations from the given path are weighted in the objective function as soft constraints. Because the *RelaxedIK* planner does not accurately reach the given manufacturing paths, the sampling-based *Stampede* algorithm is proposed in [133]. A pathwise inverse kinematics solution with a sampling-based graph search is used, considering process windows. Based on *RelaxedIK*, the path planner proposed in [134] focuses on offline path planning and considers process tolerances. Starting at multiple joint configurations for the first path point, continuous joint-space paths are computed, which account for the tolerances of the end-effector pose. Although self collisions are detected, no general collision avoidance for obstacles in the workspace is considered. In [94, 133, 134], singular configurations are avoided, but there are no guarantees to adhere to and distinguish between tolerance bands and process windows, and the available redundant process DoF are not exploited. Hence, no systematic and simultaneous usage of all process properties is considered.

5.1.1 Comparison and Contribution

Sampling-based path planners are widely used for path planning of industrial processes in spatially constrained environments. By considering (redundant) process DoF, the search space increases exponentially and cannot be covered sufficiently and smoothly using sampling. Pathwise inverse kinematics solvers are based on the inverse kinematics solution of robots, for which every (redundant) process DoF must be discretized. In contrast, in optimization-based path planners, these additional DoF can be considered as continuous optimization variables or additional constraints, e.g., process windows and tolerance bands, and criteria, e.g., time or energy optimality, can be incorporated.

In this chapter, an optimization-based path-planning framework that systematically incorporates process tolerances, process windows, constraints, and redundant process DoF is proposed. In the following, industrial processes are described by a geometric manufacturing path on the workpiece and a set of process properties, see Section 2.2. Incorporating process properties into the path-planning algorithm allows the Cartesian path to deviate from the desired reference within the allowed process DoF. This is suitable for many applications including, e.g., spraying, grinding and welding. By considering the process properties and, thus, exploiting the process windows and tolerance bands, the robot can realize a much larger number of feasible manufacturing paths. In a second step, the joint-space solutions are evaluated and the one with the best manufacturing quality is chosen.

The main contribution of the proposed work is the systematic integration of process properties in the optimization-based path-planning algorithm in the form of equality and inequality constraints, which are not available in state-of-the-art path planners. This way, desired manufacturing process properties in the task space are guaranteed, while other DoF are allowed to vary within given process windows or tolerance bands. Due to the optimization-based approach, no analytical inverse kinematics formulation is required, which is especially useful for kinematically redundant manipulators. Furthermore, analytical gradients of the objective function and the equality and inequality constraints significantly improve the calculation time and convergence speed of the path planner. The proposed path planner also incorporates a collision avoidance method, can plan through kinematic robot singularities, and uses parallelization on multiple CPU cores.

Remark 2 *It is worth mentioning that the proposed path planner is also useful for non-redundant robots performing non-redundant processes. Even for non-redundant robots, like industrial robots with 6 DoF and typical kinematics, e.g., Kuka Cybertech KR8 R1620 [135], the inverse kinematics is not unique and provides up to 16 solutions for a given pose. Further non-uniqueness has to be considered if one or more axes have a large or infinite turning range, i.e., robots with axis ranges of more than* $\pm 180°$. *All possible solutions for the inverse kinematics can be handled with the proposed optimization-based path planner.*

5.2 Collision Avoidance

The proposed path-planning framework incorporates the collision detection framework *V-Clip* [136], which is summarized in this section. The *V-Clip* framework finds the pair of globally closest features of convex polyhedrons, of which the signed distance and also the analytical gradient is computed, e.g., [137]. The signed distance and its gradient are utilized in the optimization algorithm in the next section, see [138].

Major parts of this section have been published in the author's work [10] and are adapted for this thesis.

Each collision object in the environment is described as one polyhedron consisting of vertices, edges and faces. A feature is represented by a single vertex, two vertices define an edge, and three vertices are a face. The *V-Clip* algorithm iteratively examines the neighboring features until the pair of closest features is determined. Assuming that the robot movement between two consecutive poses of the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ is small, the pair of closest features barely change. Hence, the previously found pair is chosen as initial guess for the subsequent path planning step. This way, the pair of closest features is mostly found in the first iteration of the algorithm.

With the *V-Clip* framework, arbitrary objects can be included in collision avoidance, e.g., parts of the robot, the tool, the workpiece, and the environment. The collision checks are then tailored to the manufacturing process, the robot, and its environment. If collisions between multiple objects have to be examined, the *V-Clip* algorithm is applied to each pair of collision objects separately. Therefore, the number of collision checks can be easily reduced to lower the computation time by examining only pairs of objects that are critical for the considered manufacturing process. An example of a collision model for a drawing process is shown in Fig. 5.1. In the depicted manufacturing process, the robot's wrist and the tool mounted on the end-effector are checked for collisions with the table and the box on the table. This collision detection method can also be integrated into the joint-space path planner of Chapter 4 and, hence, the robot base placement and TCP pose optimization.

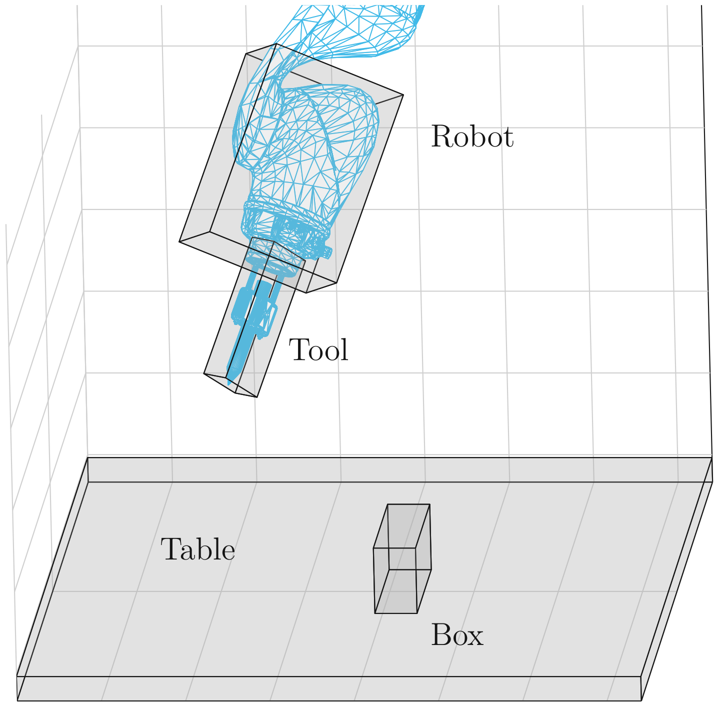


Figure 5.1: Convex polyhedrons as collision objects are checked to avoid collisions of the robot and the tool with the table and the box on the table; adapted from [10].

5.3 Optimization-based Path Planning

The proposed optimization-based path-planning algorithm starts by finding a discrete subset of all feasible robot starting configurations for the first manufacturing frame $\mathbf{H}_{\mathcal{P},1}^{\mathcal{M}}$, considering the available process DoF, redundant DoF, process tolerances, process windows, and constraints. The corresponding joint-space paths are computed by sequentially solving an optimization problem, starting from these initial configurations, considering the process properties and collision avoidance. Finally, the best joint-space path is selected, and a time parametrization is determined to obtain a dynamically feasible robot trajectory. The proposed path planner is deterministic, as no random samples are required to solve the path-planning problem.

In this section, the components of the path planner are explained. First, the starting configurations are discussed. Second, the optimization problem and the sequential solution procedure are introduced. Third, cost function terms and constraint functions are introduced, which can be combined and parametrized to describe the considered manufacturing process. Fourth, the selection process for the best joint-space path is detailed, and finally, the time parametrization of the selected path is explained.

Major parts of this section have been published in the author's work [10] and are adapted for this thesis.

5.3.1 Starting Configurations

A set of feasible initial tool poses is derived from the first manufacturing frame $\mathbf{H}_{\mathcal{P},1}^{\mathcal{M}}$ considering the process, the redundant DoF, the process tolerances, and the process windows. The process windows and tolerance bands defined in (2.26) and (2.27) are sampled with an equidistant grid. The vectors

$$\mathbf{d}_{\mathbf{t}} = \frac{1}{d_{\text{grid}}}\left(\mathbf{d}_{\mathcal{M},\text{max}}^{\mathcal{T}} - \mathbf{d}_{\mathcal{M},\text{min}}^{\mathcal{T}}\right) \circ \mathbf{t} + \mathbf{d}_{\mathcal{M},\text{min}}^{\mathcal{T}} \tag{5.1}$$

and

$$\boldsymbol{\varphi}_{\mathbf{r}} = \frac{1}{\varphi_{\text{grid}}}\left(\boldsymbol{\varphi}_{\mathcal{M},\text{max}}^{\mathcal{T}} - \boldsymbol{\varphi}_{\mathcal{M},\text{min}}^{\mathcal{T}}\right) \circ \mathbf{r} + \boldsymbol{\varphi}_{\mathcal{M},\text{min}}^{\mathcal{T}} \tag{5.2}$$

denote the individual grid points and $\circ$ is the element-wise product, also known as Hadamard product [139]. Thereby, $d_{\text{grid}} + 1$ and $\varphi_{\text{grid}} + 1$ are the number of grid points and the vectors $\mathbf{t}^{\mathrm{T}} = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}$ and $\mathbf{r}^{\mathrm{T}} = \begin{bmatrix} r_x & r_y & r_z \end{bmatrix}$ are composed of the indices $t_x, t_y, t_z = 0, 1, \dots, d_{\text{grid}}$ and $r_x, r_y, r_z = 0, 1, \dots, \varphi_{\text{grid}}$. The indices $\mathbf{t}$ and $\mathbf{r}$ in (5.1) and (5.2) are vector-valued. Using every combination of the indices in $\mathbf{t}$ and $\mathbf{r}$, i.e., every sampled grid point, a set of initial frames H_{g} is derived from the first manufacturing frame $\mathbf{H}_{\mathcal{P},1}^{\mathcal{M}}$ in the form

$$H_{\text{g}} = \left\{ \mathbf{H}_{\mathcal{P},1}^{\mathcal{M}} \begin{bmatrix} \mathbf{R}_{\Delta}(\boldsymbol{\varphi}_{\mathbf{r}}) & \mathbf{d}_{\mathbf{t}} \\ \mathbf{0} & 1 \end{bmatrix} \middle| \begin{matrix} t_x, t_y, t_z = 0, 1, \dots, d_{\text{grid}} \\ r_x, r_y, r_z = 0, 1, \dots, \varphi_{\text{grid}} \end{matrix} \right\} , \tag{5.3}$$

with the rotation matrix for the Roll–Pitch–Yaw angles $(\boldsymbol{\varphi}_{\mathbf{r}})^{\mathrm{T}} = \begin{bmatrix} \varphi_{x,r_x} & \varphi_{y,r_y} & \varphi_{z,r_z} \end{bmatrix}$

$$\mathbf{R}_{\Delta}(\boldsymbol{\varphi}_{\mathbf{r}}) = \begin{bmatrix} \mathrm{c}_x\mathrm{c}_y & \mathrm{c}_x\mathrm{s}_y\mathrm{s}_z - \mathrm{s}_x\mathrm{c}_z & \mathrm{c}_x\mathrm{s}_y\mathrm{c}_z + \mathrm{s}_x\mathrm{s}_z \\ \mathrm{s}_x\mathrm{c}_y & \mathrm{s}_x\mathrm{s}_y\mathrm{s}_z + \mathrm{c}_x\mathrm{c}_z & \mathrm{s}_x\mathrm{s}_y\mathrm{c}_z - \mathrm{c}_x\mathrm{s}_z \\ -\mathrm{s}_y & \mathrm{c}_y\mathrm{s}_z & \mathrm{c}_y\mathrm{c}_z \end{bmatrix} , \tag{5.4}$$

where s_i and c_i denote abbreviations of the trigonometric functions $\sin(\varphi_{i,r_i})$ and $\cos(\varphi_{i,r_i})$, $i = x, y, z$, respectively, see [29].

Next, the discrete subset of all feasible joint-space configurations Q_g for the set of initial frames H_g in (5.3) is computed with the inverse kinematics

$$Q_\mathrm{g} = \{\mathbf{q}_{\mathrm{g},1}, \mathbf{q}_{\mathrm{g},2}, \cdots, \mathbf{q}_{\mathrm{g},e_\mathrm{g}}\} = \{\mathbf{h}_\mathrm{H}^{-1}(\mathbf{H}) \,|\, \forall \mathbf{H} \in H_\mathrm{g}\} \ . \tag{5.5}$$

This yields a total number of e_g joint-space solutions, which are sequentially numbered. This set can be computed via an analytical inverse kinematics solver or numerical inverse kinematics solvers, e.g., *TRAC-IK* [125], cf. [94]. Note that the redundant DoF of kinematically redundant robots have to be sampled as well. In order to reduce the number of initial joint configurations, the joint-space solutions contained in Q_g are filtered using the minimum-distance criterion in (3.64). This yields the reduced set $Q_\mathrm{g}^\mathrm{f} = f_\mathrm{f}(Q_\mathrm{g})$ comprising $e_{\mathrm{f},\mathrm{g}}$ joint-space solutions for the set of initial frames H_g.

Sampling the process windows and computing the inverse kinematics solutions must only be performed with a low grid resolution and only for the first manufacturing frame $\mathbf{H}_{\mathcal{P},1}^{\mathcal{M}}$, which yields different joint-space solutions Q_g^f for the subsequent path-planning problem. In contrast, sampling-based path-planning algorithms require a large number of samples for each pose of the manufacturing path, e.g., the DESCARTES algorithm [93], which becomes infeasible for higher numbers of DoF.

5.3.2 Optimization Problem

Starting from each joint-space solution Q_g^f from Section 5.3.1, a series of optimization problems is solved sequentially, see Fig. 5.2. Each series of optimization problems is given by

$$\mathbf{q}_{d,k}^* = \underset{\mathbf{q}_{d,k} \in \mathbb{R}^n}{\arg\min} \ f_\mathrm{c}(\mathbf{q}_{d,k}, \mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}) \ , \quad k = 1, \dots, n_\mathrm{m} \tag{5.6a}$$

$$\text{s.t. } \mathbf{q}_\mathrm{min} \le \mathbf{q}_{d,k} \le \mathbf{q}_\mathrm{max} \tag{5.6b}$$

$$\mathbf{c}_\mathrm{eq}(\mathbf{q}_{d,k}, \mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}) = \mathbf{0} \tag{5.6c}$$

$$\mathbf{c}_\mathrm{ineq}(\mathbf{q}_{d,k}, \mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}) \le \mathbf{0} \ , \tag{5.6d}$$

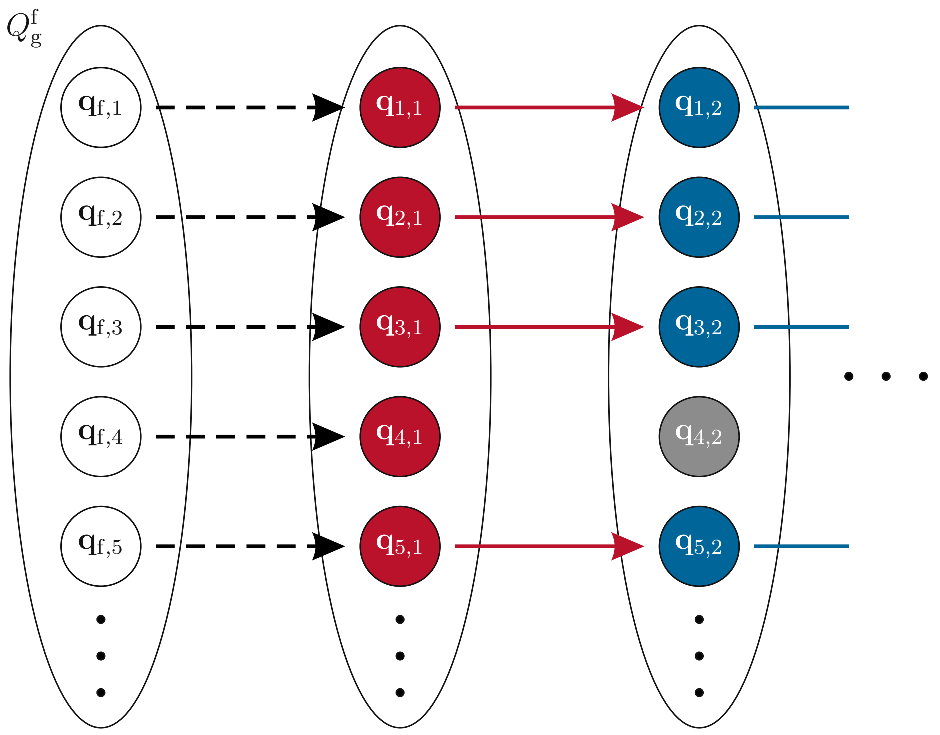


Figure 5.2: Series of optimization problems starting from the initial guesses in $Q_\mathrm{g}^\mathrm{f} = f_\mathrm{f}(Q_\mathrm{g})$ obtained from H_g in (5.3) to find complete continuous joint-space paths. In this example, the initial guess $\mathbf{q}_{\mathrm{f},4}$ does not yield a complete continuous joint-space path; adapted from [10].

with the initial guess for the k^th optimization problem, cf. [94]

$$\mathbf{q}_{d,k,0} = \begin{cases} \mathbf{q}_{\mathrm{f},d} & k = 1 \\ \mathbf{q}^*_{d,k-1} & k > 1 \ , \end{cases} \tag{5.7}$$

where $d = 1, \dots, e_{\mathrm{f,g}}$ selects the joint-space solution from Q_g^f. The optimal joint configuration $\mathbf{q}^*_{d,k}$ for the specific path pose $\mathbf{H}^{\mathcal{M}}_{\mathcal{P},k}$ is found by minimizing the objective function f_c, which is detailed in the next section. The interior-point method from the MATLAB solver *fmincon* is used, e.g., [140], to solve the corresponding problem, cf. (5.6)

$$\begin{bmatrix}\mathbf{q}^*_{d,k} & \boldsymbol{\sigma}^*\end{bmatrix} = \underset{\substack{\mathbf{q}_{d,k}\in\mathbb{R}^n \\ \boldsymbol{\sigma}\in\mathbb{R}^\kappa}}{\arg\min}\ f_\mathrm{c}(\mathbf{q}_{d,k}, \mathbf{H}^{\mathcal{M}}_{\mathcal{P},k}) - \lambda \sum_{i=1}^{\kappa} \ln(\sigma_i) \tag{5.8a}$$

$$\text{s.t. } \boldsymbol{\sigma} \geq \mathbf{0} \tag{5.8b}$$

$$\mathbf{c}_\mathrm{eq}(\mathbf{q}_{d,k}, \mathbf{H}^{\mathcal{M}}_{\mathcal{P},k}) = \mathbf{0} \tag{5.8c}$$

$$\begin{bmatrix}\mathbf{c}_\mathrm{ineq}(\mathbf{q}_{d,k}, \mathbf{H}^{\mathcal{M}}_{\mathcal{P},k}) \\ \mathbf{q}_\mathrm{min} - \mathbf{q}_{d,k} \\ \mathbf{q}_{d,k} - \mathbf{q}_\mathrm{max}\end{bmatrix} + \boldsymbol{\sigma} = \mathbf{0}\ , \tag{5.8d}$$

with the slack variable $\boldsymbol{\sigma}$, the dimension of the inequality constraints vector $\kappa = \dim\left(\begin{bmatrix}\mathbf{c}^\mathrm{T}_\mathrm{ineq} & \mathbf{q}^\mathrm{T}_\mathrm{min} & \mathbf{q}^\mathrm{T}_\mathrm{max}\end{bmatrix}^\mathrm{T}\right)$ and the barrier parameter $\lambda > 0$, see [141, 142]. The last term in (5.8a) are barrier functions to approximate the inequality constraints.

The process and redundant DoF and the process tolerances and windows of the specific manufacturing process are formulated as soft constraints in f_c and hard constraints using the equality constraints $\mathbf{c}_\mathrm{eq}$ and inequality constraints $\mathbf{c}_\mathrm{ineq}$. A number of different objective functions and constraints are provided and explained in the next section. Additionally, the upper and lower joint limits $\mathbf{q}_\mathrm{max}$ and $\mathbf{q}_\mathrm{min}$ are considered as inequality constraints in (5.6b). The initial guess $\mathbf{q}_{d,k,0}$ for each optimization problem in the sequence, see (5.7), is chosen as the solution $\mathbf{q}^*_{d,k-1}$ from the previous optimization. For the first manufacturing frame $\mathbf{H}^{\mathcal{M}}_{\mathcal{P},1}$, the set of joint-space solutions Q^f_g serves as initial guess for the optimization. In order to speed up the solution of the optimization problem, the analytical gradients of the objective function and constraints are utilized. Note that the joint-space solutions in Q^f_g are independent of each other. Hence, multiple joint-space paths can be computed in parallel on a multi-core CPU, which significantly improves the computational performance. Furthermore, the constraints and the objective function can be adapted for each path pose $\mathbf{H}^{\mathcal{M}}_{\mathcal{P},k}$ individually, $k = 1, \ldots, n_\mathrm{m}$, since the optimization problem is solved in series. In this way, for example, lower tolerances can be specified in certain segments of the manufacturing path while wider tolerance bands are used in other segments.

Remark 3 *The solution space may be further increased by considering cross connections between individual joint-space paths, e.g., from $q_{1,1}$ to $q_{2,2}$ in Fig. 5.2, which are not taken into account in this work. This could be implemented using* *a graph representation of the solution space. Consequently, the individual solution paths depend on each other, which makes a parallel computation impossible. Including cross-connections while preserving, at least to a certain extent, parallelism is an interesting future research direction. Note that a similar path-planning approach based on the inverse kinematics is shown in Section 4.2.*

5.3.3 Objective Functions and Constraints

With the proposed optimization-based path-planning approach, the objective functions and constraints are tailored to describe the considered manufacturing process and considering the process properties. This way, individual Cartesian coordinates may be strictly constrained while allowing tolerances in other coordinates.

In general, the objective function consists of n_c cost terms $f_{\mathrm{c},j}, j = 1, \dots, n_\mathrm{c}$, in the form

$$f_\mathrm{c}(\mathbf{q}_{d,k}, \mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}) = \sum_{j=1}^{n_\mathrm{c}} f_{\mathrm{c},j}(\mathbf{q}_{d,k}, \mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}) \ , \tag{5.9}$$

and the equality and inequality constraints are also composed of different components

$$\mathbf{c}_\mathrm{eq} = \begin{bmatrix} \mathbf{c}_{\mathrm{eq},1} \\ \mathbf{c}_{\mathrm{eq},2} \\ \vdots \end{bmatrix} \ , \qquad \mathbf{c}_\mathrm{ineq} = \begin{bmatrix} \mathbf{c}_{\mathrm{ineq},1} \\ \mathbf{c}_{\mathrm{ineq},2} \\ \vdots \end{bmatrix} \ . \tag{5.10}$$

Since the individual terms in (5.9) and (5.10) directly depend on the manufacturing process under consideration, the index j will be omitted in the following to improve the clarity of presentation.

Position Deviation

The deviation $\tilde{\mathbf{p}}_\mathcal{M}$ of the actual tool position $\mathbf{p}_\mathcal{P}^\mathcal{T}$ from the desired position $\mathbf{p}_\mathcal{P}^\mathcal{M}$, described in the manufacturing frame $\mathcal{M}$, is computed as

$$\tilde{\mathbf{p}}_\mathcal{M} = \begin{bmatrix} \tilde{x}_\mathcal{M} \\ \tilde{y}_\mathcal{M} \\ \tilde{z}_\mathcal{M} \end{bmatrix} = \left(\mathbf{R}_\mathcal{P}^\mathcal{M}\right)^\mathrm{T} \left(\mathbf{p}_\mathcal{P}^\mathcal{M} - \mathbf{p}_\mathcal{P}^\mathcal{T}(\mathbf{q})\right) \ , \tag{5.11}$$

using the augmented forward kinematics from Section 2.1.2 and the manufacturing path defined in Section 2.2, analogous to the control error in (3.49), cf. [94].

Depending on the considered manufacturing process, the position deviation may appear as soft constraint in the objective function to implement process tolerances

$$f_\mathrm{c} = \frac{1}{2}(\tilde{\mathbf{p}}_\mathcal{M})^\mathrm{T}\mathbf{A}_\mathrm{p}\tilde{\mathbf{p}}_\mathcal{M} \ , \tag{5.12}$$

with the positive semi-definite weighting matrix $\mathbf{A}_\mathrm{p}$, as hard constraint in the equality constraint to implement a strictly constrained process DoF

$$\mathbf{c}_\mathrm{eq} = \tilde{\mathbf{p}}_\mathcal{M} \ , \tag{5.13}$$

or as inequality constraint to implement process windows

$$\mathbf{c}_\mathrm{ineq} = \begin{bmatrix} \mathbf{d}^\mathcal{T}_{\mathcal{M},\mathrm{min}} - \tilde{\mathbf{p}}_\mathcal{M} \\ \tilde{\mathbf{p}}_\mathcal{M} - \mathbf{d}^\mathcal{T}_{\mathcal{M},\mathrm{max}} \end{bmatrix} \ . \tag{5.14}$$

The position deviation may also appear in both the objective function (5.12) and the inequality constraint (5.14), which yields the implementation of tolerance bands. Instead of the complete vector $\tilde{\mathbf{p}}_\mathcal{M}$, its components $\tilde{x}_\mathcal{M}$, $\tilde{y}_\mathcal{M}$, and $\tilde{z}_\mathcal{M}$ may be used individually as soft, hard, or inequality constraint. Fixing some entries using hard constraints while considering the remaining entries as soft constraints is also possible.

The analytical gradients of (5.13) and (5.14) follow as

$$\frac{\partial \mathbf{c}_\mathrm{eq}}{\partial \mathbf{q}} = \frac{\partial \tilde{\mathbf{p}}_\mathcal{M}}{\partial \mathbf{q}} = -\left(\mathbf{R}^\mathcal{M}_\mathcal{P}\right)^\mathrm{T} \mathbf{J}^\mathcal{T}_{\mathcal{P},\mathrm{v}}(\mathbf{q}) \ , \tag{5.15}$$

$$\frac{\partial \mathbf{c}_\mathrm{ineq}}{\partial \mathbf{q}} = \begin{bmatrix} -\frac{\partial \tilde{\mathbf{p}}_\mathcal{M}}{\partial \mathbf{q}} \\ \frac{\partial \tilde{\mathbf{p}}_\mathcal{M}}{\partial \mathbf{q}} \end{bmatrix} \ , \tag{5.16}$$

with the Jacobian $\mathbf{J}^\mathcal{T}_{\mathcal{P},\mathrm{v}}(\mathbf{q})$ related to the translational velocity of the tool frame $\mathcal{T}$ w.r.t. the workpiece frame $\mathcal{P}$, see (2.13). The gradient of (5.12) reads as

$$\frac{\partial f_\mathrm{c}}{\partial \mathbf{q}} = (\tilde{\mathbf{p}}_\mathcal{M})^\mathrm{T}\mathbf{A}_\mathrm{p}\frac{\partial \tilde{\mathbf{p}}_\mathcal{M}}{\partial \mathbf{q}} \ . \tag{5.17}$$

Orientation Deviation

The orientation deviation $\mathbf{R}_{\mathcal{T}}^{\mathcal{M}} = \mathbf{R}_{\mathcal{P}}^{\mathcal{M}}(\mathbf{R}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}))^{\mathrm{T}}$ is described as unit quaternion $\mathbf{o}_{\mathcal{T}}^{\mathcal{M}} = \mathbf{o}_{\mathcal{P}}^{\mathcal{M}} \otimes (\mathbf{o}_{\mathcal{P}}^{\mathcal{T}})^{-1}$, see (3.26), cf. [94]. If the orientation of the actual tool frame $\mathcal{T}$ exactly matches the orientation of the desired manufacturing frame $\mathcal{M}$, the quaternion $\mathbf{o}_{\mathcal{T}}^{\mathcal{M}}$ in (3.26) becomes $(\mathbf{o}_{\mathcal{T}}^{\mathcal{M}})^{\mathrm{T}} = \begin{bmatrix}1 & \mathbf{0}^{\mathrm{T}}\end{bmatrix}$, see [29]. Thus, the orientation deviation is constructed as soft constraint using the scalar part $\eta_{\mathcal{T}}^{\mathcal{M}}$ of $\mathbf{o}_{\mathcal{T}}^{\mathcal{M}}$ with the positive scalar weight $a_{\mathrm{o}} > 0$ as

$$f_{\mathrm{c}} = \frac{a_{\mathrm{o}}}{2}\left(1 - \eta_{\mathcal{T}}^{\mathcal{M}}\right)^2 \ . \tag{5.18}$$

Consequently, in (5.18), all orientation deviations are penalized equally. If the manufacturing process allows for different orientation tolerances w.r.t. each rotation axis, the soft constraint on the vector part $\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}$ of $\mathbf{o}_{\mathcal{T}}^{\mathcal{M}}$ in the form

$$f_{\mathrm{c}} = \frac{1}{2}(\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}})^{\mathrm{T}}\mathbf{A}_{\mathrm{o}}\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}} \ , \tag{5.19}$$

with the positive semi-definite weighting matrix $\mathbf{A}_{\mathrm{o}}$, is used. In Appendix B.3, the cost terms in (5.18) and (5.19) for the orientation deviation (3.26) are motivated and validated using mathematical and geometric arguments.

The soft constraints (5.18) and (5.19) may also be implemented as hard constraints in the form

$$c_{\mathrm{eq}} = 1 - \eta_{\mathcal{T}}^{\mathcal{M}} \tag{5.20}$$

and

$$\mathbf{c}_{\mathrm{eq}} = \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}} \ . \tag{5.21}$$

When using these hard constraints, rotations around certain axes can be restricted. Moreover, the inequality constraints

$$\mathbf{c}_{\mathrm{ineq}} = \begin{bmatrix} \boldsymbol{\varepsilon}_{\mathcal{M},\mathrm{min}}^{\mathcal{T}} - \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}} \\ \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}} - \boldsymbol{\varepsilon}_{\mathcal{M},\mathrm{max}}^{\mathcal{T}} \end{bmatrix} \ , \tag{5.22}$$

can be employed, where $\boldsymbol{\varepsilon}_{\mathcal{M},\mathrm{min}}^{\mathcal{T}}$ and $\boldsymbol{\varepsilon}_{\mathcal{M},\mathrm{max}}^{\mathcal{T}}$ are computed using (3.26) from the minimum and maximum orientation deviation defined in (2.27). Analogous to the position deviation, only individual components of the orientation error $\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}$

may be included in (5.19), (5.21), and (5.22) by assembling the cost functions and constraints accordingly.

The gradients of the objective function terms (5.18) and (5.19) are computed as

$$\frac{\partial f_{\mathrm{c}}}{\partial \mathbf{q}} = -a_{\mathrm{o}}(1-\eta_{\mathcal{T}}^{\mathcal{M}})\frac{\partial \eta_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \tag{5.23}$$

and

$$\frac{\partial f_{\mathrm{c}}}{\partial \mathbf{q}} = (\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}})^{\mathrm{T}}\mathbf{A}_{\mathrm{o}}\frac{\partial \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \ , \tag{5.24}$$

using the gradients $\frac{\partial \eta_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}}$ and $\frac{\partial \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}}$ of Appendix B.4. The gradient of the constraint (5.20) reads as

$$\frac{\partial c_{\mathrm{eq}}}{\partial \mathbf{q}} = -\frac{\partial \eta_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \ , \tag{5.25}$$

and the gradients of (5.21) and (5.22) yield

$$\frac{\partial \mathbf{c}_{\mathrm{eq}}}{\partial \mathbf{q}} = \frac{\partial \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \ , \tag{5.26}$$

$$\frac{\partial \mathbf{c}_{\mathrm{ineq}}}{\partial \mathbf{q}} = \begin{bmatrix} -\frac{\partial \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \\ \frac{\partial \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \end{bmatrix} \ . \tag{5.27}$$

Collision Avoidance

The *V-Clip* algorithm [136] determines the pair of globally closest features of two convex polyhedrons, for which the signed distance l is computed. The signed distance l is positive if two obstacles are at the distance l from each other, and a negative distance l describes obstacles in collision. For every collision check $i = 1, \dots, n_{\mathrm{l}}$ between two objects, the signed distances $l_i(\mathbf{q})$ of all feature pairs are utilized in the soft constraint as

$$f_{\mathrm{c}} = \frac{1}{2}\begin{bmatrix} \max(0, l_{1,\mathrm{min}} - l_1) \\ \max(0, l_{2,\mathrm{min}} - l_2) \\ \vdots \\ \max(0, l_{n_{\mathrm{l}},\mathrm{min}} - l_{n_{\mathrm{l}}}) \end{bmatrix}^{\mathrm{T}} \mathbf{A}_{\mathrm{v}} \begin{bmatrix} \max(0, l_{1,\mathrm{min}} - l_1) \\ \max(0, l_{2,\mathrm{min}} - l_2) \\ \vdots \\ \max(0, l_{n_{\mathrm{l}},\mathrm{min}} - l_{n_{\mathrm{l}}}) \end{bmatrix} \ , \tag{5.28}$$

with the positive semi-definite weighting matrix $\mathbf{A}_{\mathrm{v}}$ and the respective minimum distances $l_{i,\mathrm{min}}$, see [138]. The signed distances l_i can also be used in inequality constraints as

$$\mathbf{c}_{\text{ineq}} = \begin{bmatrix} -l_1 \\ -l_2 \\ \vdots \\ -l_{n_1} \end{bmatrix} \tag{5.29}$$

to ensure that no collision occurs in the planned joint-space path. Using both constraints (5.28) and (5.29) together improves the convergence of the optimization problem (5.6) significantly.

The analytical gradient of (5.28) reads as

$$\frac{\partial f_{\text{c}}}{\partial \mathbf{q}} = -\begin{bmatrix} \max(0, l_{1,\min} - l_1) \\ \max(0, l_{2,\min} - l_2) \\ \vdots \\ \max(0, l_{n_1,\min} - l_{n_1}) \end{bmatrix}^{\text{T}} \mathbf{A}_{\text{v}} \begin{bmatrix} \frac{\partial l_1}{\partial \mathbf{q}} \\ \frac{\partial l_2}{\partial \mathbf{q}} \\ \vdots \\ \frac{\partial l_{n_1}}{\partial \mathbf{q}} \end{bmatrix} . \tag{5.30}$$

In (5.30), the gradients of the signed distances l_i are computed in the form

$$\frac{\partial l_i}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial l_i}{\partial \mathbf{v}_1} & \frac{\partial l_i}{\partial \mathbf{v}_2} & \cdots & \frac{\partial l_i}{\partial \mathbf{v}_{n_{\text{v}}}} \end{bmatrix} \begin{bmatrix} \mathbf{J}_{\mathcal{P},\text{v}}^{\mathcal{V}_1}(\mathbf{q}) \\ \mathbf{J}_{\mathcal{P},\text{v}}^{\mathcal{V}_2}(\mathbf{q}) \\ \vdots \\ \mathbf{J}_{\mathcal{P},\text{v}}^{\mathcal{V}_{n_{\text{v}}}}(\mathbf{q}) \end{bmatrix} , \tag{5.31}$$

where the vectors $\mathbf{v}_j, j = 1, \ldots, n_{\text{v}}$, denote the positions of the vertices $\mathcal{V}_j$ of the polyhedron, and $\mathbf{J}_{\mathcal{P},\text{v}}^{\mathcal{V}_j}(\mathbf{q})$ are the corresponding Jacobians related to the velocities. Only the vertices defining the closest feature pair are required to compute the gradient, and therefore, most of the entries in the left vector in (5.31) are zero. The gradient of the inequality constraint (5.29) directly follows from (5.31). Although the analytical gradient (5.31) is discontinuous if the closest features change, this does not affect the performance of the optimization algorithm (5.6) in practice.

Joint Limits and Path Continuity

In order to obtain physically feasible robot motions, a joint-space path must be sufficiently smooth and adhere to the joint limits. Hence, to derive continuous joint-space paths, the objective function term

$$f_\mathrm{c} = \frac{1}{2} \left(\mathbf{q} - \mathbf{q}_{d,k,0}\right)^\mathrm{T} \mathbf{A}_\mathrm{s} \left(\mathbf{q} - \mathbf{q}_{d,k,0}\right) \ , \tag{5.32}$$

with the positive semi-definite weighting matrix $\mathbf{A}_\mathrm{s}$, is used to penalize large joint movements. In (5.32), the joint configuration $\mathbf{q}_{d,k,0}$ is the initial guess for the optimization for the k^th manufacturing frame $\mathbf{H}^{\mathcal{M}}_{\mathcal{P},k}$, see (5.7). The corresponding analytical gradient reads as

$$\frac{\partial f_\mathrm{c}}{\partial \mathbf{q}} = \left(\mathbf{q} - \mathbf{q}_{d,k,0}\right)^\mathrm{T} \mathbf{A}_\mathrm{s} \ . \tag{5.33}$$

Robot movements through kinematic singular points of the robot are explicitly allowed with the proposed path-planning framework and are executable with standard controllers of an industrial robot using joint-space control.

The joint positions of the robot are constrained in (5.6b) within their axes limits to compute feasible robot motions. In order to improve convergence, joint positions $\mathbf{q}$ are penalized as soft constraints with the objective function term

$$f_\mathrm{c} = \frac{1}{2} \tilde{\mathbf{q}}^\mathrm{T} \mathbf{A}_\mathrm{l} \tilde{\mathbf{q}} \quad \text{and} \quad \tilde{\mathbf{q}} = \mathbf{q} - \mathbf{q}_\mathrm{mean} \ , \tag{5.34}$$

with $\mathbf{q}_\mathrm{mean}$ from (3.62) and the corresponding gradient

$$\frac{\partial f_\mathrm{c}}{\partial \mathbf{q}} = \tilde{\mathbf{q}}^\mathrm{T} \mathbf{A}_\mathrm{l} \ . \tag{5.35}$$

The diagonal matrix $\mathbf{A}_\mathrm{l}$ is chosen as

$$\mathbf{A}_\mathrm{l} = \frac{a_\mathrm{l}}{2} \operatorname{diag} \left(\mathbf{q}_\mathrm{max} - \mathbf{q}_\mathrm{min}\right)^{-2} \ , \tag{5.36}$$

with the mechanical joint limits of the robot (3.61) and the weighting parameter $a_\mathrm{l} > 0$.

5.3.4 Optimal Joint-Space Path

Multiple joint-space paths are generated by solving the sequence of optimization problems (5.6), starting from all $e_\mathrm{f,g}$ joint-space solutions contained in Q^f_g. If a subproblem of a sequence does not yield a feasible solution, the corresponding joint-space path is discarded, see Fig. 5.2. In order to find the optimal joint-space path, the costs of the objective function terms for the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ are added

$$f_{\mathrm{c}}^{\mathrm{a}}(P_d, H_{\mathcal{P}}^{\mathcal{M}}) = \sum_{k=1}^{n_{\mathrm{m}}} \sum_{j \in \chi} f_{\mathrm{c},j}(\mathbf{q}_{d,k}, \mathbf{H}_{\mathcal{P},k}^{\mathcal{M}}) \ , \quad d = 1, \ldots, e_{\mathrm{f,g}} \tag{5.37}$$

for each feasible joint-space path. By comparing the individual costs $f_{\mathrm{c}}^{\mathrm{a}}$ in (5.37) for each feasible joint-space path $P_d = \{\mathbf{q}_{d,1}, \mathbf{q}_{d,2}, \cdots, \mathbf{q}_{d,n_{\mathrm{m}}}\}$ with $d = 1, \ldots, e_{\mathrm{f,g}}$, the optimal joint-space path $Q^* = \{\mathbf{q}_1^*, \cdots, \mathbf{q}_{n_{\mathrm{m}}}^*\}$ is found from the lowest costs $f_{\mathrm{c}}^{\mathrm{a}}(Q^*, H_{\mathcal{P}}^{\mathcal{M}})$ for the desired manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$. Only a particular subset $\chi \subset \{1, \ldots, n_{\mathrm{c}}\}$ of objective function terms may be used in (5.37) instead of all terms $f_j, j = 1, \ldots, n_{\mathrm{c}}$, from (5.9). For example, the optimization problem (5.9) might consider all process properties of the manufacturing process, collision avoidance, and the joint limits. Then, evaluating the optimal joint-space path in (5.37) might use a reduced subset χ that only contains the process properties and, hence, only considers the achieved manufacturing result. Consequently, the best joint-space path Q^* may be optimal regarding the manufacturing quality and is not penalized by challenging but feasible robot movements.

The weighting matrices $\mathbf{A}_{\mathrm{p}}$, $\mathbf{A}_{\mathrm{o}}$, $\mathbf{A}_{\mathrm{v}}$, and $\mathbf{A}_{\mathrm{s}}$ and the scalar weights a_{o} and a_{l} introduced in the previous section have a strong impact on the convergence behavior and the shape of the resulting joint-space solution, particularly if the considered process exhibits a large number of (redundant) process DoF, tolerances, and windows.

5.3.5 Trajectory Generation

The result of the planning algorithm is the optimal joint-space path Q^* corresponding to the desired manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$. As a final step, the joint-space path Q^* is time parametrized to compute a piecewise trajectory $\mathbf{q}^*(t)$ with the sample points $(t_k, \mathbf{q}_k^*), k = 1, \ldots, n_{\mathrm{m}}$. The time stamps t_k are derived from the Cartesian distance between two consecutive sample points $\mathbf{q}_{k-1}^*$ and $\mathbf{q}_k^*$ with the augmented forward kinematics (2.11) from Section 2.1.2 in the form

$$t_k = t_{k-1} + \frac{\left\| \mathbf{p}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}_k^*) - \mathbf{p}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}_{k-1}^*) \right\|_2}{\dot{p}_k} \ , \quad k = 1, \ldots, n_{\mathrm{m}} \ . \tag{5.38}$$

The desired path velocity is specified by $\dot{p}_k > 0, k = 1, \ldots, n_\mathrm{m}$, according to the manufacturing process.

The optimal smooth joint-space trajectory $\mathbf{q}^*(t)$ is generated by computing piecewise cubic Hermite interpolating polynomials [143] for the sample points $(t_k, \mathbf{q}_k^*), k = 1, \ldots, n_\mathrm{m}$. The obtained trajectory $\mathbf{q}^*(t)$ is then executable on the robot to perform the manufacturing process considering the specified process properties.

5.4 Robot Base Placement

In Section 4.2, a fast kinematic joint-space path planner was introduced to find the optimal TCP pose or robot base placement for a set of manufacturing paths. Based on this planner, in Section 4.3, the optimization problem for the optimal placement was formulated, considering different goals in the objective function (4.4), e.g., maximizing the number of continuous joint-space paths or minimizing the joint movements. In Section 5.3, an advanced optimization-based path planner was introduced, which can also be used to optimize the robotic work cell. This planner is superior to the previous one by including a collision avoidance scheme and systematically incorporating process properties. Therefore, the number of feasible joint-space path solutions is expected to rise compared to the original planner from Section 4.2.

The optimization-based path planner presented in Section 5.3 computes a set of multiple continuous joint-space paths P_d for a given relative position $\mathbf{p}_\mathcal{W}^\mathcal{B}$ of the robot base frame $\mathcal{B}$ w.r.t. the world frame $\mathcal{W}$. Additionally, the individual costs of the objective functions are computed for all paths P_d, and the optimal joint-space path Q^* is found by the lowest costs $f_\mathrm{c}^\mathrm{a}(Q^*(\mathbf{p}_\mathcal{W}^\mathcal{B}), H_\mathcal{P}^\mathcal{M})$, cf. (5.37). The optimal position $\mathbf{p}_\mathcal{W}^{\mathcal{B}*}$ is computed by a superordinate optimization problem

$$\mathbf{p}_\mathcal{W}^{\mathcal{B}*} = \underset{\mathbf{p}_\mathcal{W}^\mathcal{B} \in \mathbb{R}^3}{\arg\min} \; f_\mathrm{c}^\mathrm{a}\left(Q^*(\mathbf{p}_\mathcal{W}^\mathcal{B}), H_\mathcal{P}^\mathcal{M}\right) \; , \tag{5.39a}$$

$$\text{s.t. } \mathbf{p}_{\mathcal{W},\min}^\mathcal{B} \le \mathbf{p}_\mathcal{W}^\mathcal{B} \le \mathbf{p}_{\mathcal{W},\max}^\mathcal{B} \; . \tag{5.39b}$$

Similar to (4.12), the objective function is evaluated with the optimal joint-space path Q^* of the corresponding position $\mathbf{p}_\mathcal{W}^\mathcal{B}$ and minimized to find the optimal position $\mathbf{p}_\mathcal{W}^{\mathcal{B}*}$. For every evaluation of the cost function $f_\mathrm{c}^\mathrm{a}\left(Q^*(\mathbf{p}_\mathcal{W}^\mathcal{B}), H_\mathcal{P}^\mathcal{M}\right)$, the advanced joint-space path planner searches the optimal joint-space path $Q^*(\mathbf{p}_\mathcal{W}^\mathcal{B})$, which needs a longer computation time compared to the fast planner in Section 4.2. Similar to Section 5.3.4, a reduced set of objective functions $\chi \subset \{1, \ldots, n_c\}$ can be used to evaluate the cost of the optimal joint-space path, which allows us to optimize the robot base placement for different criteria than the underlying path planner. The already evaluated objective function terms needed to solve the path planning problem can be reused to find the optimal robot base placement.

5.5 Experimental Results

In this section, the proposed path-planning framework is applied to two example applications with significantly different process properties. To this end, only the weighting matrices of the objective function terms must be adapted, and the equality and inequality constraints must be adjusted to describe the respective manufacturing process, see Section 5.3.

Major parts of this section have been published in the author's work [10] and are adapted for this thesis.

The first application is a drawing process, in which a marker with rectangular nib is utilized to draw thin and thick lines on the surface of a workpiece. The thickness of the line depends on the orientation of the marker w.r.t. the desired drawing path. In the second application, a spraying process is demonstrated. The rotationally symmetric spray nozzle is considered a redundant DoF in this process. The rabbit-shaped workpiece and the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ are shown in Fig. 5.3, in which the first manufacturing frame $\mathbf{H}_{\mathcal{P},1}^{\mathcal{M}}$ is shown. The total length of the meander-shaped path is approximately 2.5 m with 1024 path poses. The z-axes of the manufacturing frames (blue) are normal to the workpiece surface and the frames rotate w.r.t. the surface normal vector along the manufacturing path.

5.5.1 Drawing Process

In this process, a line specified by the desired manufacturing path has to be drawn on a 3D-printed rabbit with a marker mounted on the end-effector of an industrial robot. This experiment demonstrates an industrial process with an end-effector mounted tool, see Fig 2.1a, and a complex continuous manufacturing path. This process is representative for similar industrial processes, e.g., welding, grinding,

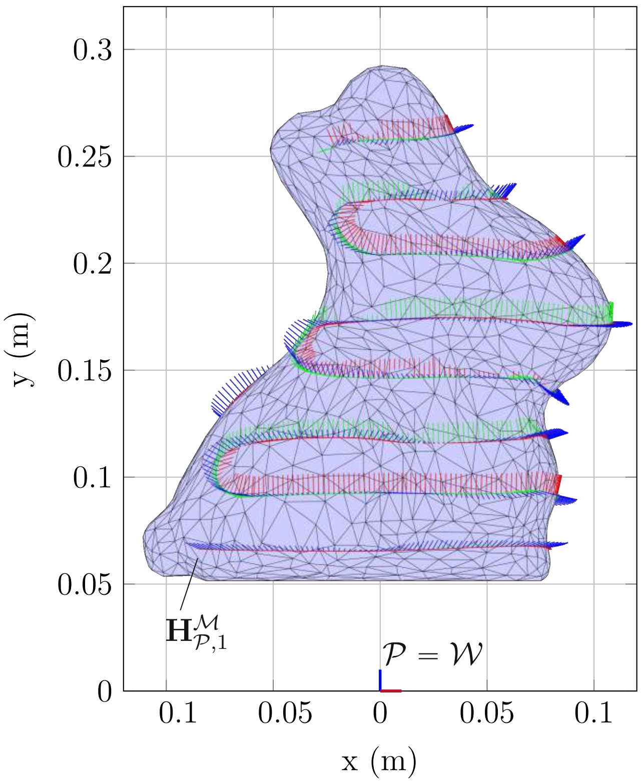


Figure 5.3: Manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ on the rabbit-shaped workpiece in the workpiece frame $\mathcal{P}$. The first frame is denoted by $\mathbf{H}_{\mathcal{P},1}^{\mathcal{M}}$; adapted from [10].

or cutting. The drawing process was executed and validated in a laboratory environment, see Fig. 5.4. Planning long continuous robot motions with multiple constraints is challenging due to the restrictive mechanical axes limits and the limited workspace of the robot.

Drawing Process Properties

The experimental setup of the drawing process is shown in Fig. 5.4. The robot Kuka LBR iiwa 14 R820, see Appendix A.1, draws a continuous line on the surface of a stationary rabbit-shaped workpiece with a marker with rectangular nib mounted on the end-effector. In order to account for the kinematic inaccuracies of the real robot, the drawing tool is equipped with a passive compliance mechanism comprising two linear springs. The rectangular nib allows us to draw different line thicknesses by rotating the marker around the surface normal vector. In order to demonstrate the capabilities of the path-planning framework including

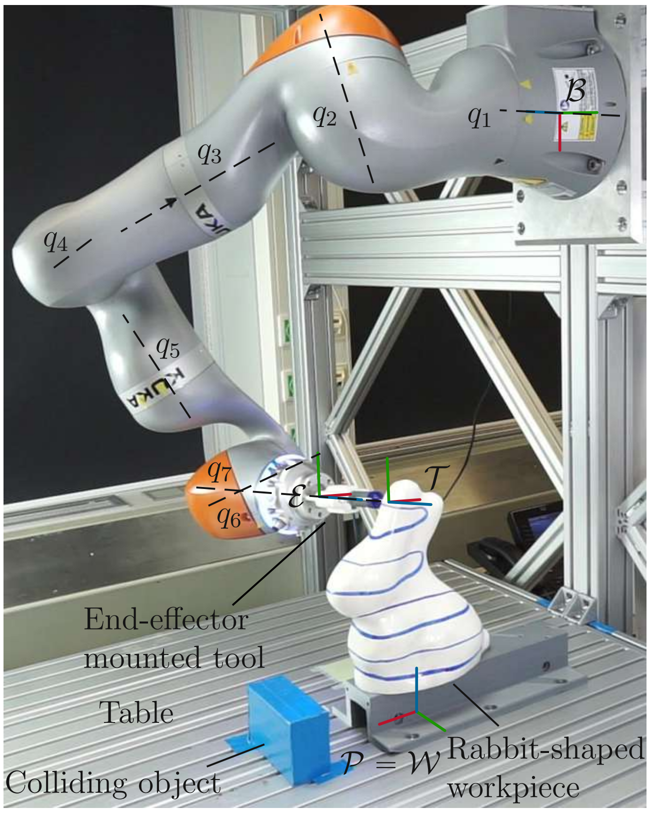


Figure 5.4: Experimental setup for the drawing process after completing the process; adapted from [10].

collision avoidance, the blue collision object in Fig. 5.4 is placed in front of the workpiece. Due to this blue object, the robot cannot reach certain sections of the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ precisely without collision, see ①-③ in Fig. 5.6.

The position of the nib must precisely follow the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ of Fig. 5.3 to perform the drawing process on the surface of the workpiece. In contrast, orientation deviations of the marker from the surface normal in a certain range only marginally degrade the quality of the drawn line. Since the nib of the marker is rectangular, a deviation from the desired rotation around the surface normal changes the line thickness, which should be avoided. Therefore, the tolerance band for the rotation around the z-axis is chosen more restrictively. The process properties for the position and orientation are represented by tolerance

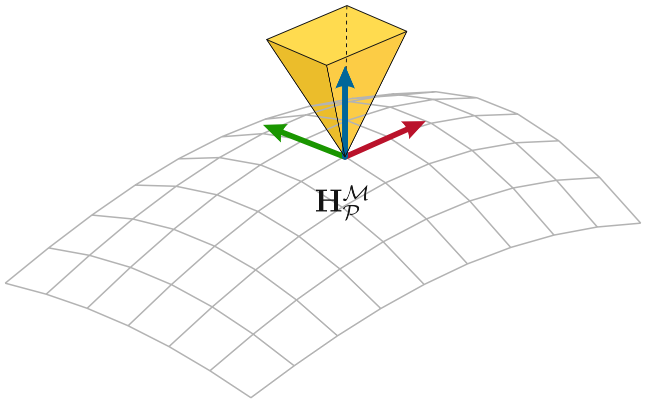


Figure 5.5: Tolerances for the drawing process; adapted from [10].

bands in the form (see Fig. 2.3)

$$\mathbf{d}_{\mathcal{M},\max}^{\mathcal{T}} = -\mathbf{d}_{\mathcal{M},\min}^{\mathcal{T}} = \begin{bmatrix} 0\,\mathrm{m} \\ 0\,\mathrm{m} \\ 0\,\mathrm{m} \end{bmatrix}, \tag{5.40a}$$

$$\boldsymbol{\varphi}_{\mathcal{M},\max}^{\mathcal{T}} = -\boldsymbol{\varphi}_{\mathcal{M},\min}^{\mathcal{T}} = \begin{bmatrix} 40^\circ \\ 40^\circ \\ 20^\circ \end{bmatrix}, \tag{5.40b}$$

and are visualized in Fig. 5.5, cf. Fig. 2.2a.

By parametrizing the objective function and selecting the corresponding equality and inequality constraints, the process properties are systematically considered in the proposed optimization-based path planner. Since no position deviation is allowed, the equality constraint of the position deviation (5.13) is used, and the corresponding objective term (5.12) and inequality constraint (5.14) are omitted. To allow for orientation deviations according to (5.40), the equality constraints of the marker orientation (5.20) and (5.21) are disabled. The deviation from the desired orientation of the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ is penalized with the objective term (5.18) to minimize the utilized tolerances. Additionally, with the soft constraint (5.19), the deviations of the different orientation coordinates are weighted according to the allowed tolerances (5.40). Also, the inequality constraint (5.22) is enabled to guarantee that the tolerance bands from (5.40) are adhered to. Moreover, it is checked that there are no collisions of the marker and the last robot link with the table and the blue collision object, see Fig. 5.1. Therefore, the corresponding objective function (5.28) together with the inequality constraint (5.29) are used to ensure a collision-free robot movement. Finally, to obtain continuous joint movements, the objective terms (5.32) and (5.34) are employed with the robot joint limits $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ from Tab. A.1. The inverse kinematics is solved by numerical optimization, except for the starting configurations presented in Section 5.3.1.

The weights and parameters of the individual terms of the objective function and constraints are chosen empirically and summarized in Tab. 5.1. The matrix $\mathbf{A}_\mathrm{p}$ weighs the position deviation (5.12), the scalar a_o and the matrix $\mathbf{A}_\mathrm{o}$ account for the orientation deviation, see (5.18) and (5.19), the matrix $\mathbf{A}_\mathrm{v}$ penalizes the joint configurations near obstacles, see (5.28), the matrix $\mathbf{A}_\mathrm{s}$ weighs the joint movements according to (5.32), and the weight a_l in (5.36) penalizes joint positions near their axis limits in (5.34). In general, diagonal matrices are advantageous, as the number of parameters is greatly reduced, and couplings between the individual DoF are avoided. In general, larger matrix entries lead to smaller errors in the respective DoF. Depending on which term in the objective function is crucial for the specific process, the individual weights are chosen larger or smaller.

Experimental Results of the Drawing Process

Using the set of parameters given in Tab. 5.1, the optimal robot base placement $\mathbf{p}_\mathcal{W}^{\mathcal{B}*}$ is computed according to Section 5.4 with the optimization-based path planner from Section 5.3. While executing the optimization-based path planner, all feasible joint-space solutions Q_g^f for the first frame $\mathbf{H}_{\mathcal{P},1}^\mathcal{M}$ of the manufacturing path $H_\mathcal{P}^\mathcal{M}$, see Fig. 5.3, are computed using (3.64) with $q_\mathrm{dist} = 0.5$. Subsequently, the series of optimization problems (5.6) is solved for every starting configuration Q_g^f. The optimal joint-space path Q^* is found with (5.37), where the same objective function terms as in the optimization (5.6) are used. The subset χ in (5.37) for finding the optimal robot base placement $\mathbf{p}_\mathcal{W}^{\mathcal{B}*}$ with (5.39) contains only the objective function term of the orientation deviation (5.19) to maximize the manufacturing quality. The robot is placed as accurately as possible at the optimal position, and the exact location is determined with a similar calibration method as presented in Fig. 3.12, for which the optimal joint-space path Q^* is recomputed. Although the robot base is placed optimally w.r.t. the manufacturing quality, more than 15 % of the manufacturing poses in $H_\mathcal{P}^\mathcal{M}$ from Fig. 5.3 are only reachable by exploiting

Table 5.1: Objective function terms, constraints, and weights used for the drawing process; adapted from [10].

Variables	Equations	Weights
Position Deviation		
$f_\mathrm{c}(\tilde{\mathbf{p}}_\mathcal{M})$	(5.12)	$\mathbf{A}_\mathrm{p} = \mathbf{0}$
$\mathbf{c}_\mathrm{eq}(\tilde{\mathbf{p}}_\mathcal{M})$	(5.13)	enabled
$\mathbf{c}_\mathrm{ineq}(\tilde{\mathbf{p}}_\mathcal{M})$	(5.14)	disabled
Orientation Deviation		
$f_\mathrm{c}(\eta_\mathcal{T}^\mathcal{M})$	(5.18)	$a_\mathrm{o} = 80$
$f_\mathrm{c}(\boldsymbol{\varepsilon}_\mathcal{T}^\mathcal{M})$	(5.19)	$\mathbf{A}_\mathrm{o} = \mathrm{diag}\,(1, 1, 30)$
$c_\mathrm{eq}(\eta_\mathcal{T}^\mathcal{M})$	(5.20)	disabled
$\mathbf{c}_\mathrm{eq}(\boldsymbol{\varepsilon}_\mathcal{T}^\mathcal{M})$	(5.21)	disabled
$\mathbf{c}_\mathrm{ineq}(\boldsymbol{\varepsilon}_\mathcal{T}^\mathcal{M})$	(5.22)	enabled
Collision Avoidance		
$f_\mathrm{c}(l_i)$	(5.28)	$\mathbf{A}_\mathrm{v} = 10^3\mathrm{diag}\,(1, 1, 4)$
$\mathbf{c}_\mathrm{ineq}(l_i)$	(5.29)	enabled
Joint Limits and Path Continuity		
$f_\mathrm{c}(\mathbf{q})$	(5.32)	$\mathbf{A}_\mathrm{s} = \mathrm{diag}\,(10, 10, 5, 5, 5, 5, 5)$
$f_\mathrm{c}(\mathbf{q})$	(5.34), (5.36)	$a_\mathrm{l} = 0.05$
Robot KUKA LBR iiwa 14 R820, see Tab. A.1		

the process properties. These points cannot be reached by the robot exactly in position and orientation due to mechanical axes limits, the limited workspace of the robot, and colliding objects, i.e., the blue box and the table. These poses are shown in Fig. 5.6 as black dots.

After finding the optimal robot position and the corresponding joint-space path, the time parametrization (5.38) is applied to obtain the optimal joint-space trajectory $\mathbf{q}^*(t)$. The trajectories of the individual joints $q_h^*(t), h = 1, \ldots, n$, are normalized to the respective axes limits $\mathbf{q}_\mathrm{min}$ and $\mathbf{q}_\mathrm{max}$, see (3.66), and depicted in Fig. 5.7 as $\bar{q}_h^*$. The available axes ranges of the robot joints must be utilized to a large extent in order to perform the drawing process, which emphasizes the complexity of the task. In particular, the trajectories of axes 4, 6, and 7 occasionally reach their mechanical axes limits. The path planner can also compute joint-space trajectories

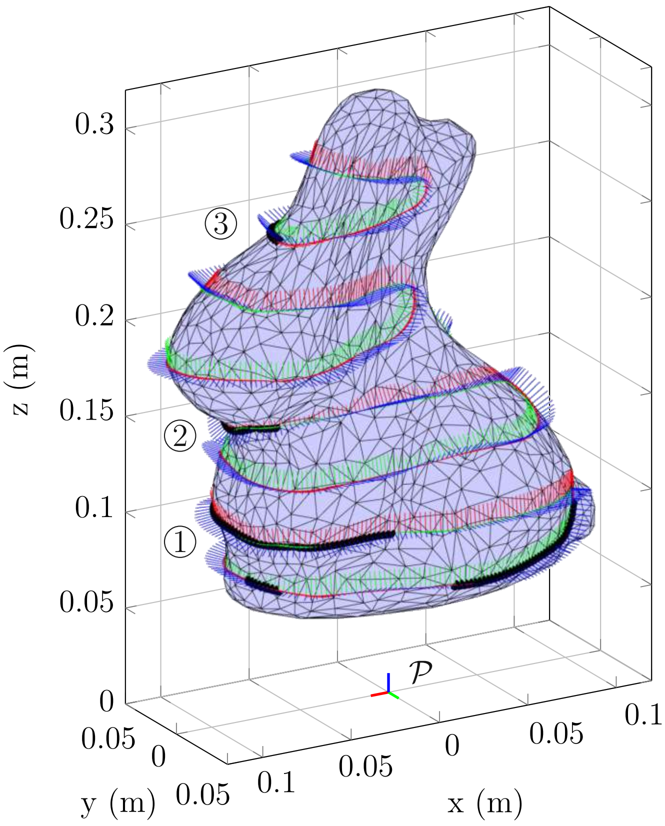


Figure 5.6: Side view of the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ on the rabbit-shaped workpiece in the workpiece frame $\mathcal{P}$. The manufacturing poses marked with black dots can only be reached by exploiting the process properties; adapted from [10].

moving through singular configurations. For the demonstrated drawing process, the robot moves three times close to the singular configuration $q_2 = 0° \wedge q_3 = -90°$, see (A.1b), to execute the given manufacturing path, see Fig. 5.7. The joint-space path is executed using the robot manufacturer's joint-space controller. To this end, all sample points $(\mathbf{q}_k^*), k = 1, \ldots, n_\mathrm{m}$, are transmitted to the controller before executing the manufacturing process.

Remark 4 *The robot controller of the* K*UKA LBR iiwa 14 R820 does not allow the definition of the sample points, including their desired time parametrization* $(t_k, (\mathbf{q}_k^*)), k = 1, \ldots, n_\mathrm{m}$, *as computed with the trajectory generation of Section 5.3.5, see [144]. In order to execute the joint-space path in the desired time, the software-defined velocity limit for the fastest joint is set accordingly for each trajectory segment to achieve the desired Cartesian velocity specified in* (5.38). *If an*

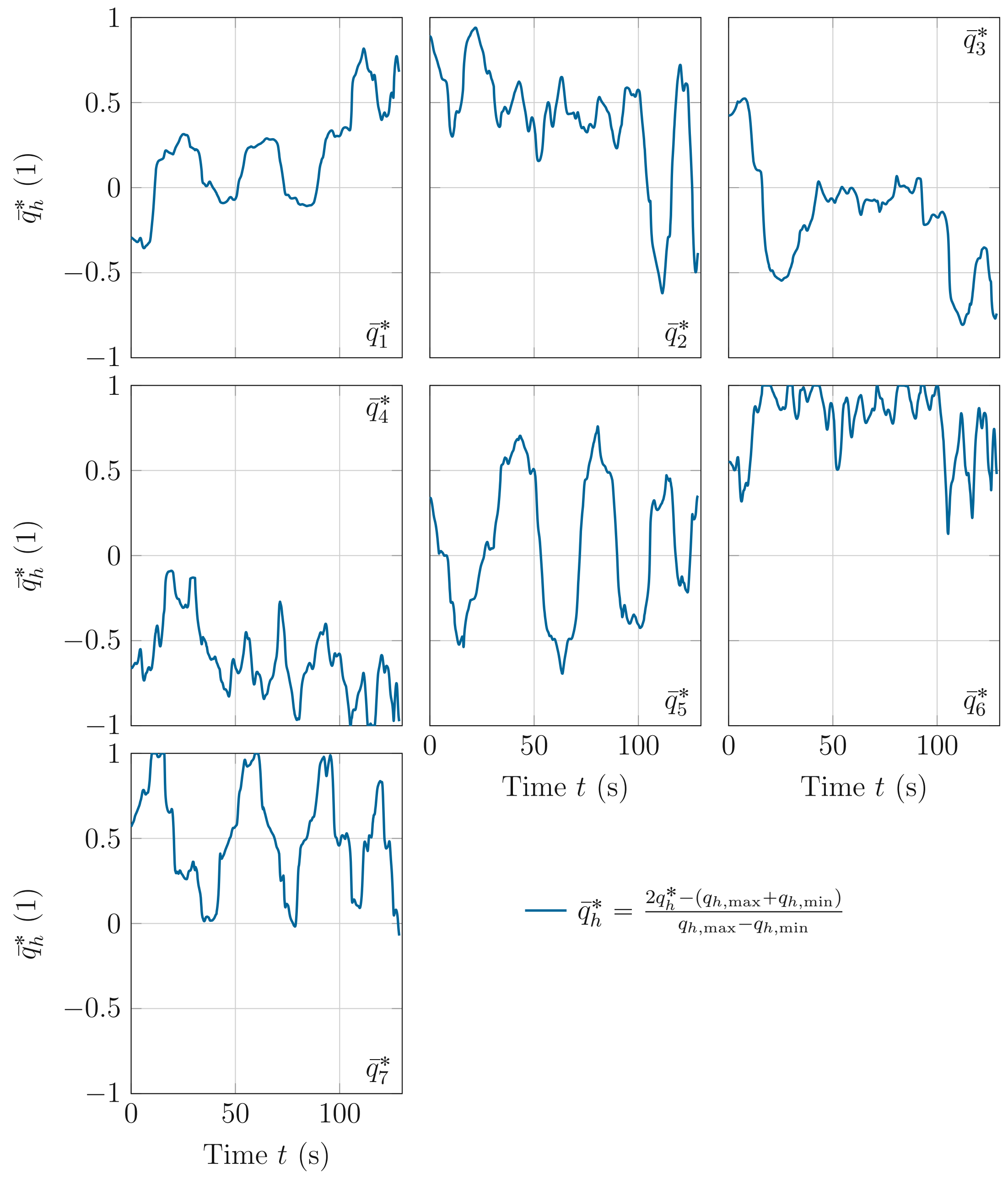


Figure 5.7: Optimal joint-space trajectory $\bar{\mathbf{q}}^*(t)$ for the drawing process. The trajectories of the individual joints are normalized to their respective axes limits $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$; adapted from [10].

appropriate torque interface is available to control the robot, the motion controller presented in Section 3.3.1 or the hybrid force/motion controller of Section 3.3.3 could also be used to execute the manufacturing process.

The mean calculation time of a single optimization problem from the series (5.6) including collision checks is approximately 70 ms on an INTEL Core i7-8700K in a single-core implementation. If all CPU cores are utilized, the mean calculation time reduces to around 18 ms, since multiple optimization problems are solved in parallel. The total calculation time of the path-planning problem with $e_{\mathrm{f,g}} = 6$ starting configurations is approximately 90 s. The total calculation time depends on the number of path poses n_{m} in the given manufacturing path, the number of different starting configurations $e_{\mathrm{f,g}}$, the CPU, and the number of available CPU cores. Without collision checks using the *V-Clip* algorithm, see Section 5.3.3, the mean optimization time further reduces by a factor of 3. The path-planning problem for the drawing process in this scenario is highly constrained, and many poses along the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ are only reachable by exploiting the process tolerances. Therefore, most joint-space solutions in $Q_{\mathrm{g}}^{\mathrm{f}}$ do not lead to a complete and feasible joint-space path.

A video of the experimental result of the drawing process is shown in `www.acin.tuwien.ac.at/4adf` . The result of the drawing process, depicted in Fig. 5.8, shows a good agreement with the desired manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ in Fig. 5.3. The path sections with the two different desired line thicknesses can be easily distinguished. The experiment is executed without absolute calibration, without feedback of the actual Cartesian end-effector position, and without estimation and control of the contact force due to the limitations of the available robot interface.

As shown in Fig. 5.6, a significant portion of the manufacturing path poses in $H_{\mathcal{P}}^{\mathcal{M}}$ is not exactly reachable with the tool mounted on the end-effector. Hence, process tolerances must be utilized to solve the path-planning problem and compute the optimal trajectory $\mathbf{q}^*(t)$. In Fig. 5.9, the orientation deviations of the TCP frame poses $\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}^*)$ from the desired manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ are depicted. The maximum values of the utilized process tolerances are below 30° for φ_x and φ_y and below 10° for φ_z, which results from the higher weighting of φ_z in (5.19), see Tab. 5.1. The process tolerances are within the allowed tolerance bands defined in (5.40).

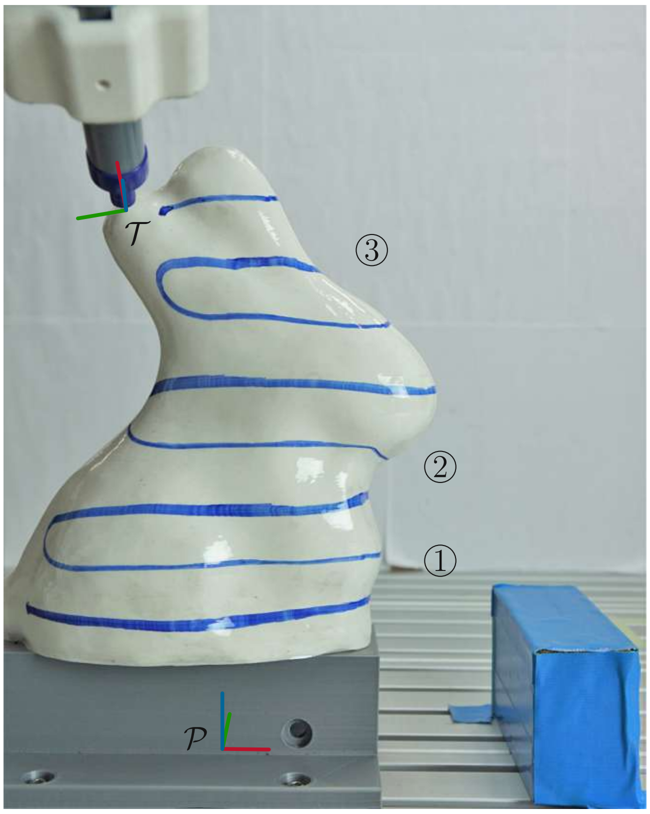


Figure 5.8: Result of the drawing process using the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ shown in Fig. 5.3 with the segments from Fig. 5.6; adapted from [10].

The first path pose of the manufacturing path $\mathbf{H}_{\mathcal{P},1}^{\mathcal{T}}$ is located near the bottom of the workpiece, see Fig. 5.3. Moving with the marker from the first path pose $\mathbf{H}_{\mathcal{P},1}^{\mathcal{T}}$ towards ① in Fig. 5.6, the proposed path planner exploits the tolerances to avoid collisions with the blue collision object and the table, see Fig. 5.4. High usage of orientation tolerances is also necessary at $t = 71\,\mathrm{s}$ located at ② in Fig. 5.6. At ②, the robot motion cannot be solved without the additional freedom provided by the tolerances due to the blue collision object. The surface normals at ② directly point towards the blue collision object, see Fig. 5.3. High orientation tolerances are used again at $t = 105\,\mathrm{s}$, located at ③. These orientation tolerances result from moving multiple times near the singular configuration $q_2 = 0° \wedge q_3 = -90°$, see (A.1b), which is required to solve the path-planning problem in one continuous robot motion, see also Fig. 5.7. Because the Cartesian position of the TCP frame $\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}$ w.r.t. the manufacturing frame $H_{\mathcal{P}}^{\mathcal{M}}$ is implemented as equality constraint, see Tab. 5.1, the resulting deviations of the position coordinates are below the numer-

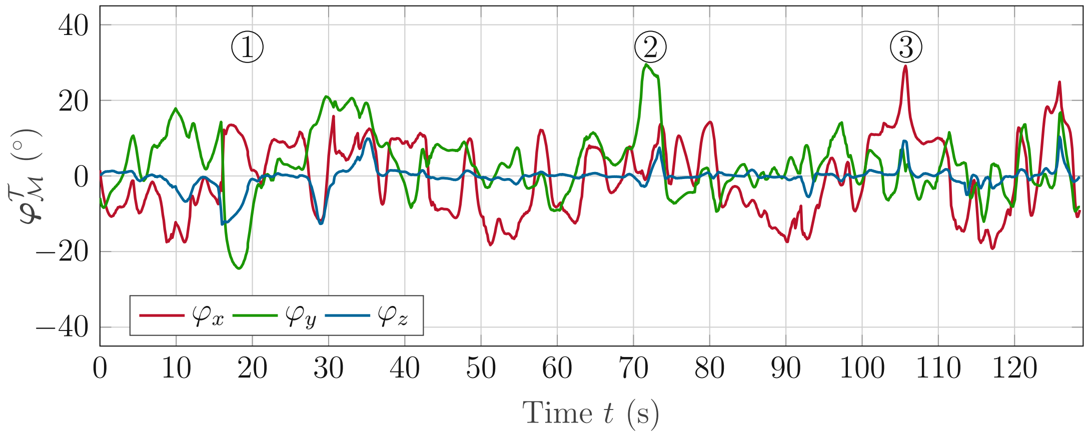


Figure 5.9: Utilized orientation tolerances of the TCP frame poses $\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}^*(t))$ w.r.t. the desired manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ at the drawing process; adapted from [10].

ical tolerance of the used optimization solver. The experiment shows an accurate execution of the drawing process with the demanded process quality based on the specified process properties. The continuous joint-space trajectory $\mathbf{q}^*(t)$ could be computed only with the help of the available tolerance bands, demonstrating the proposed optimization-based path-planning framework.

5.5.2 Spraying Process

In this section, the proposed optimization-based path-planning framework is applied to a spraying process. In this process, the workpiece is mounted on the end-effector of a robot, and a stationary spray nozzle has to be moved along the manufacturing path to perform the process. The spray jet is considered rotationally symmetric, i.e., a rotation of the spray jet around the spray direction does not affect the process quality and is, therefore, considered a redundant process DoF. This property also appears in other industrial processes, e.g., robotic milling, drilling, or laser cutting. Although this spraying process differs significantly from the drawing process in Section 5.5.1, the proposed path planner can be applied to the new process properties. Additionally, long and complex spray paths are challenging path-planning problems, where the available redundancy has to be exploited to compute continuous joint-space paths for the process execution.

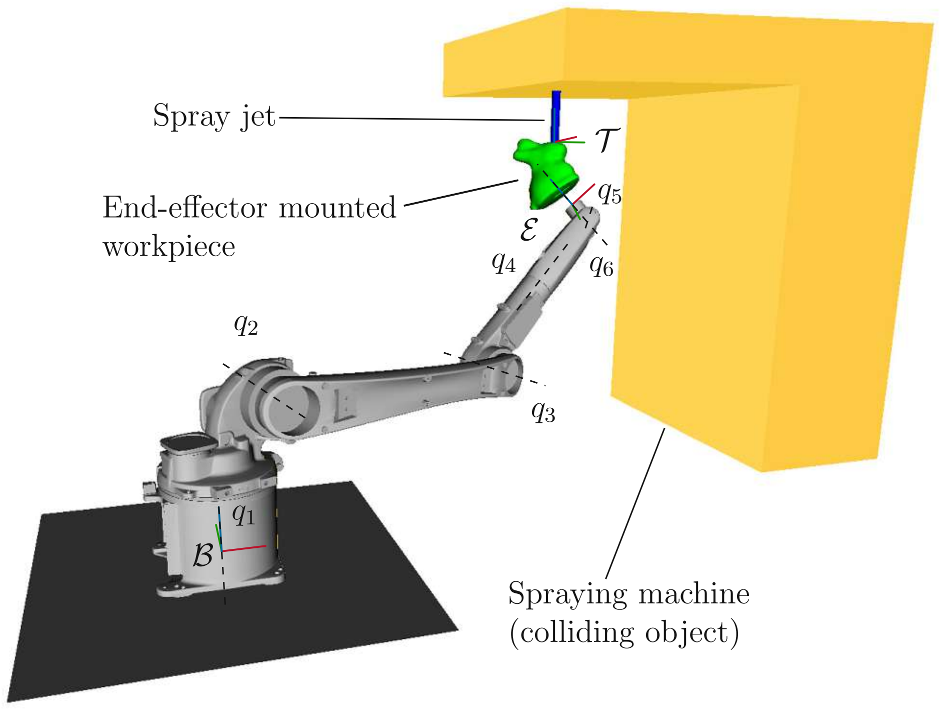


Figure 5.10: Setup for the spraying process in simulation; adapted from [10].

Spraying Process Properties

The simulation environment SIMULINK 3D Animation of the spraying process is shown in Fig. 5.10 with the robot KUKA Cybertech KR8 R1620, see Appendix A.2. The rabbit-shaped workpiece mounted on the robot's end-effector is shown in green, and the spray machine is implemented as a stationary tool and depicted as a yellow object, see also Fig. 2.1b. The rotationally symmetric spray jet is blue and considered a redundant DoF in this process. In order to implement collision avoidance from Section 5.2 in the path planning, the spraying machine is embedded in two box objects, and collisions with the workpiece are checked. Collisions between the robot and the spraying machine do not occur.

The desired manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ from Fig. 5.3 has to be followed with the spray jet to perform the spray process on the rabbit-shaped workpiece. Hence, the lateral position (x- and y-direction of the manufacturing frame) of the spray jet must follow the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ exactly, while small deviations along the surface normal vector (z-direction of the manufacturing frame) are allowed. The latter only slightly degrades the process quality and is considered a process tolerance. Due to the rotationally symmetric spray nozzle, this process has a redundant process DoF, i.e., the orientation around the surface normal vector. In contrast, no rotational deviations around the x- and y-axis are permitted to maintain a circular spray deposit. The process DoF of the spraying process are visualized in Fig. 2.2b. These process properties are represented by the tolerance bands and process windows given by

$$\mathbf{d}^{\mathcal{T}}_{\mathcal{M},\max} = \begin{bmatrix} 0\,\mathrm{m} \\ 0\,\mathrm{m} \\ 0.045\,\mathrm{m} \end{bmatrix} , \quad \mathbf{d}^{\mathcal{T}}_{\mathcal{M},\min} = \begin{bmatrix} 0\,\mathrm{m} \\ 0\,\mathrm{m} \\ -0.055\,\mathrm{m} \end{bmatrix} , \tag{5.41a}$$

$$\boldsymbol{\varphi}^{\mathcal{T}}_{\mathcal{M},\max} = -\boldsymbol{\varphi}^{\mathcal{T}}_{\mathcal{M},\min} = \begin{bmatrix} 0^\circ \\ 0^\circ \\ \infty \end{bmatrix} . \tag{5.41b}$$

Based on (5.41), the optimization problem (5.6) is tailored to the spraying process by parametrizing the objective function terms and enabling the appropriate constraints. The equality constraint of the x- and y-position deviation in (5.13) is enabled to precisely follow the given manufacturing path $H^{\mathcal{M}}_{\mathcal{P}}$. Hence, the corresponding position inequality constraints in (5.14) are disabled and the respective weights of the objective function term (5.12) are zero. To use the process tolerances along the z-direction of the manufacturing path $H^{\mathcal{M}}_{\mathcal{P}}$, the equality constraint of the z-position deviation (5.13) is disabled. Instead, the z-position deviation is used in the objective function term (5.12), and the corresponding inequality constraint (5.14) ensures compliance with the defined tolerance band (5.41a). Because no orientation deviation is allowed around the x- and y-axis w.r.t. the manufacturing path $H^{\mathcal{M}}_{\mathcal{P}}$, the corresponding equality constraints in (5.21) are enabled and the inequality constraints (5.22) and objective function terms (5.18) and (5.19) are omitted. The redundant process DoF of the spray process, i.e., the orientation around the z-axis, is implemented by setting the respective weight in (5.19) to zero and disabling the corresponding equality (5.20), (5.21) and inequality constraints (5.22). The orientation deviations around the z-axis are neither penalized in the optimization problem (5.6) nor constrained in any way. Collision avoidance is considered between the rabbit-shaped workpiece and the spraying machine with the objective function term (5.28) and the inequality constraint (5.29). The objective function terms (5.32) and (5.34) are used to ensure continuous joint movements of the robot. The objective function terms, equality and inequality constraints for the spraying process are summarized in Tab. 5.2. The individual weights are empirically chosen as diagonal matrices according to the specific process and the robot used.

Simulation Results of the Spraying Process

The path planner is parametrized for the spraying process with the objective functions and constraints from Tab. 5.2. Similar to the drawing process, the optimization (5.39) computes the optimal relative position of the robot base frame $\mathcal{B}$ w.r.t. the world frame $\mathcal{W}$, see Fig. 2.1b, together with the optimal joint-space path Q^*. The series of optimization problems of the path planner is solved based on the joint-space solutions Q_g^f for the first manufacturing path pose $\mathbf{H}_{\mathcal{P},1}^{\mathcal{M}}$, see Fig. 5.3, using $q_\mathrm{dist} = 0.5$. For this process, the objective function terms in Tab. 5.2 are reused to evaluate the optimal joint-space path Q^* with (5.37), where the objective function term of the position deviation in z-direction (5.12) is used to find the optimal position $\mathbf{p}_{\mathcal{W}}^{\mathcal{B}*}$. The joint-space trajectory $\mathbf{q}^*(t)$ is computed with (5.38) with a constant path velocity to obtain a uniform spray coating on the workpiece. The joint trajectory $\mathbf{q}^*(t)$ is shown in Fig. 5.11, where the individual joints $q_h^*(t), h = 1, \dots, n$, are normalized to the respective axes limits $\mathbf{q}_\mathrm{min}$ and $\mathbf{q}_\mathrm{max}$, see (3.66). The robot does not reach the mechanical axes limits during the motion and continuously follows the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$. This is possible since the redundant DoF is considered and the robot can move through a singular configuration at $t = 63\,\mathrm{s}$ where $q_5 = 0°$ and at $t = 74\,\mathrm{s}$ and $t = 88\,\mathrm{s}$ where $q_3 \approx 1.7°$; see the mathematical description of the singular configurations in (A.3). For this application, the mean optimization time for a single optimization in (5.6) on an Intel Core i7-8700K is approximately 36 ms in a single-core implementation and around 9 ms using all cores. The total calculation time is around 60 s with $e_\mathrm{f,g} = 6$ starting configurations. These optimization times are lower compared to the drawing process, which originates from the robot's lower axis count and, hence, the reduced number of optimization variables, see (5.6). Furthermore, only two collision checks in each optimization are required for the two boxes surrounding the spraying machine. Without collision avoidance checks, an additional 1.5 times reduction of the computing time is possible. A video of the simulation result of the spraying process is shown in `www.acin.tuwien.ac.at/4adf` .

Table 5.2: Objective function terms, constraints, and weights used for the spraying process; adapted from [10].

Variables	Equations	Weights
Position Deviation		
$f_\text{c}(\tilde{\mathbf{p}}_\mathcal{M})$	(5.12)	$\mathbf{A}_\text{p} = \text{diag}\,(0, 0, 90)$
$c_{\text{eq},x}(\tilde{x}_\mathcal{M})$	(5.13)	enabled
$c_{\text{eq},y}(\tilde{y}_\mathcal{M})$	(5.13)	enabled
$c_{\text{eq},z}(\tilde{z}_\mathcal{M})$	(5.13)	disabled
$c_{\text{ineq},x}(\tilde{x}_\mathcal{M})$	(5.14)	disabled
$c_{\text{ineq},y}(\tilde{y}_\mathcal{M})$	(5.14)	disabled
$c_{\text{ineq},z}(\tilde{z}_\mathcal{M})$	(5.14)	enabled
Orientation Deviation		
$f_\text{c}(\eta_\mathcal{T}^\mathcal{M})$	(5.18)	$a_\text{o} = 0$
$f_\text{c}(\boldsymbol{\varepsilon}_\mathcal{T}^\mathcal{M})$	(5.19)	$\mathbf{A}_\text{o} = \text{diag}\,(0, 0, 0)$
$c_\text{eq}(\eta_\mathcal{T}^\mathcal{M})$	(5.20)	disabled
$c_{\text{eq},x}(\varepsilon_{\mathcal{T},x}^\mathcal{M})$	(5.21)	enabled
$c_{\text{eq},y}(\varepsilon_{\mathcal{T},y}^\mathcal{M})$	(5.21)	enabled
$c_{\text{eq},z}(\varepsilon_{\mathcal{T},z}^\mathcal{M})$	(5.21)	disabled
$\mathbf{c}_\text{ineq}(\boldsymbol{\varepsilon}_\mathcal{T}^\mathcal{M})$	(5.22)	disabled
Collision Avoidance		
$f_\text{c}(l_i)$	(5.28)	$\mathbf{A}_\text{v} = 10^4\text{diag}\,(1, 1)$
$\mathbf{c}_\text{ineq}(l_i)$	(5.29)	enabled
Joint Limits and Path Continuity		
$f_\text{c}(\mathbf{q})$	(5.32)	$\mathbf{A}_\text{s} = \text{diag}\,(5, 5, 5, 5, 5, 5)$
$f_\text{c}(\mathbf{q})$	(5.34), (5.36)	$a_\text{l} = 0.05$
Robot KUKA Cybertech KR8 R1620, see Tab. A.2		

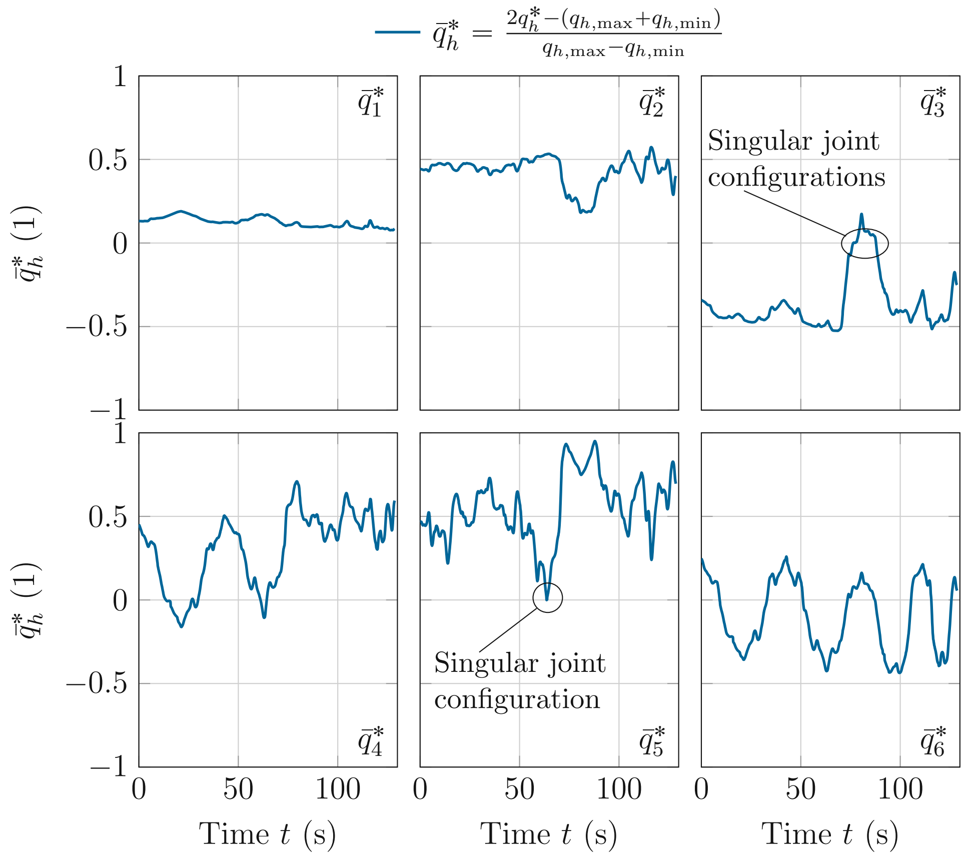


Figure 5.11: Optimal joint-space trajectory $\bar{\mathbf{q}}^*(t)$ for the spraying process. The trajectories of the individual joints are normalized to their respective axes limits $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$; adapted from [10].

In Fig. 5.12, the position and orientation deviations from the TCP frame poses $\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}^*)$ w.r.t. the desired manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ are presented. Deviations of the x- and y-position are below the numerical tolerance of the used optimization solver. The tolerance band of the z-position is utilized only to a small extent, cf. (5.41a). The orientation deviation of the TCP frame poses $\mathbf{H}_{\mathcal{P}}^{\mathcal{T}}(\mathbf{q}^*)$ w.r.t. the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ around the x- and y-axis are also constrained and are consistently below the optimizer tolerance. In contrast, rotations around the z-axis of the spray direction, i.e., the redundant DoF, are used extensively to generate a continuous joint-space path Q^*. This experiment shows that the proposed path planner can solve complex path-planning problems for various manufacturing processes with long paths by only adapting the objective function terms and constraints based on the required process properties. As a consequence, the path planner signifi-

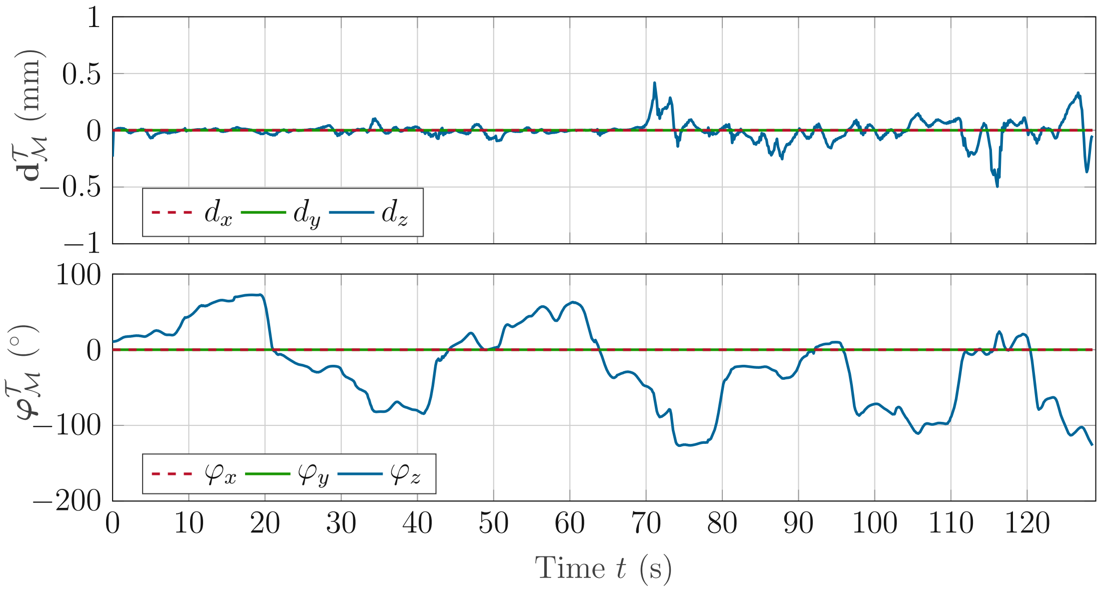


Figure 5.12: Position and orientation tolerances of the TCP frame poses $\mathbf{H}^{\mathcal{T}}_{\mathcal{P}}(\mathbf{q}^*(t))$ w.r.t. the desired manufacturing path $H^{\mathcal{M}}_{\mathcal{P}}$ for the spraying process; adapted from [10].

cantly exploits the redundant DoF of the spraying process to compute feasible and continuous robot trajectories.

5.6 Conclusions

In this chapter, a novel optimization-based path-planning framework for robots is proposed, which systematically accounts for all process properties (process tolerances, process windows, constraints, redundant DoF) of the specific manufacturing process and includes a collision avoidance scheme. This allows us to find solutions to path planning problems for complex manufacturing paths that cannot be solved with state-of-the-art concepts. By parametrizing the objective function terms and selecting the appropriate equality and inequality constraints, the underlying optimization problem can be easily tailored to the specific needs of the considered manufacturing process.

In the first step, the path planner computes multiple joint configurations for the robot based on the pose of the first manufacturing frame. Based on these joint-space solutions, series of optimization problems are solved that are all independent.

This makes it possible to perform a parallel execution of the path-planning algorithm on a multi-core CPU, significantly reducing the computing time. Moreover, analytical gradients of the objective function and of the constraints are provided to improve the convergence further and curb the computing time. The proposed path-planning framework also allows the robot to move through kinematic singularities, often avoided in state-of-the-art algorithms that rely on task-space controllers.

In order to demonstrate the feasibility of the proposed approach, a drawing process and a spraying process are considered. For the drawing process, the robot is equipped with an end-effector that holds the marker to draw a predefined line on the surface of a workpiece. The marker has a rectangular nib, with which two different line thicknesses can be realized by rotating the marker around the surface's normal direction by 90°. The orientation of the end-effector is allowed to deviate from the surface's normal direction within a specified tolerance band without deteriorating the final drawing quality. Similar process properties can be defined accordingly for many other industrial processes, like welding or cutting. The drawing process was implemented experimentally on a KUKA LBR iiwa 14 R820. Interestingly, 15 % of the poses along the drawing path cannot be reached for the nominal path. However, by exploiting the process tolerances, the drawing task could be successfully performed with high quality and the desired line thickness in the corresponding segments. For the second application, a joint-space path for a simulated spraying process was computed. In this process, the rotation of the spray jet around the surface's normal vector is considered a redundant DoF, and the distance of the spray nozzle to the surface can vary within a predefined tolerance band. Again, by taking advantage of the redundant DoF, a feasible complex robot path could be planned with the proposed framework. These two applications demonstrate the versatility of the proposed collision-free robotic path-planning framework, which can be easily adjusted to different manufacturing processes and systematically account for process tolerances, process windows, and redundant DoF.

Major parts of this section have been published in the author's work [10] and are adapted for this thesis.

Chapter 6 Conclusions and Outlook

This thesis deals with the fast-paced increase in the demand for flexible robotic production cells. The trend of individualized products and decreasing lot sizes drive this demand. To this end, flexible offline path-planning algorithms were developed to solve planning problems in fully automated manufacturing, which cannot be solved with state-of-the-art concepts. In each chapter of this thesis, conclusions of the presented concepts are given. The following focuses on summarizing the entire work, the commonalities and possible combinations of the proposed concepts, and an outlook.

Major parts of this chapter have been published in the author's works [7, 8, 9, 10, 11] and are adapted for this thesis.

6.1 Conclusions

This work investigates processes that require continuous contact between tool and workpiece which are common in industrial manufacturing, e.g., welding, trimming, milling, polishing, cutting, and sewing. In order to execute such manufacturing processes automatically, an industrial robot must perform a continuous motion with a tool or workpiece mounted on the end-effector. The industrial processes exemplified in this work are trimming and spraying in simulation and drawing processes performed in laboratory experiments.

First, an automatic production process is presented in Chapter 3 that automatically executes manufacturing paths on 3D workpieces based on user-generated 2D input patterns. For this, a manufacturing path has to be computed based on this input pattern to generate a program for execution on an industrial robot. The accuracy of this automatic manufacturing pipeline is demonstrated in an experimental drawing process with an industrial robot. For the experimental setup, a sophisticated way to compute the optimal placement of the robot in the work cell is developed. This workflow provides an easy way to manufacture 3D workpieces with robots based on 2D input patterns.

The robotic work cell is productive as long as the industrial robot can execute the required manufacturing paths. In some cases, new manufacturing paths may exceed the possibilities of the robotic work cell and are not executable with the current setup due to, e.g., the limited workspace of the robot. Chapter 4 shows that small adaptions of the tool's end-effector mounting, i.e. the tool center point (TCP), are sometimes sufficient to execute given manufacturing paths, which were not executable before. Therefore, a fast joint-space path planner was developed as a basis for an optimization problem to find the optimal TCP or robot base placement. This algorithm is a natural extension of finding the optimal robot base placement of the drawing process in Chapter 3. It proved its potential in an industrial trimming process performed in semi-automated shoe manufacturing.

In Chapters 3 and 4, the computed manufacturing paths are followed exactly with an industrial robot. Besides the limited workspace of a robot, this exact execution of continuous manufacturing paths can be impeded by collisions or mechanical joint limits. Adaptions of the TCP, as presented in Chapter 4, could be applied if manufacturing paths are not executable. However, this can only be used in some situations due to, e.g., geometrical or dynamical limits of the TCP on the robot. Nevertheless, many manufacturing processes allow deviations of the tool from the desired manufacturing path in certain process degrees of freedom (DoF). Therefore, the surface processes are modeled in the form of process tolerances, windows, constraints, and redundant DoF, as described in Chapter 2. Based on these process properties, a path-planning algorithm can provide maximum flexibility without further modification of the robotic setup or desired manufacturing path. Therefore, in Chapter 5, an advanced joint-space path planner that systematically incorporates all process properties of a manufacturing path was developed. This optimization-based approach can solve even more challenging path-planning problems compared to the simple joint-space planner presented in Chapter 4. This is shown for two different manufacturing processes, i.e., a drawing process and a spray-painting process, where the available process DoF, tolerances, and redundancies are beneficially utilized. By using this planner, the robot is more flexible in executing complex manufacturing paths without changes to the robotic work cell or manufacturing path.

This thesis presents several algorithms for flexible, individualized manufacturing in surface processes, i.e., the automatic user-based generation of robot programs, the optimal placement of the TCP and robot base, and the advanced path planning using all available process properties. Employing these algorithms in practical applications allows us to increase the flexibility of a robotic work cell already in the design stage or without significant hardware modifications during operation. Additionally, these algorithms can be combined appropriately to fit the application at hand. One example of such a combination is presented in Chapter 5 to find the optimal robot base placement, where the advanced path planner with process properties from Chapter 5 was used instead of the fast joint-space planner of Chapter 4. This way, the robot base placement algorithm is extended to find the optimal placement for the specific manufacturing process.

Another example for a valuable combination of the presented algorithms is the automatic pipeline to draw 2D input patterns on 3D objects in Chapter 3 or the trim application in Chapter 4. If a task-space path on the 3D object is not executable with the robot, the advanced joint-space path planner from Chapter 5 can generate executable joint-space trajectories by considering the process properties. This joint-space path deviates from the original path in certain process DoF, from which a modified task-space path is computed and executed using the control concepts presented in Chapter 3.

6.2 Outlook

This thesis shows several ways to improve the workflows of robot program generation to fulfill the demands for flexible production systems. The presented algorithms are versatile and can be applied to a large number of different robotic manufacturing tasks, e.g., welding, spray painting, milling, drilling, sanding, polishing, grinding, chamfering, and also textile fabrication processes like cutting, sewing, gluing, and drawing on workpieces. Consequently, robotic work cells can be used and reused for multiple manufacturing tasks without a redesign of the work cell layout. Moreover, the setup time for new manufacturing paths can be minimized with automatic path planning and execution pipelines. More intuitive human-machine interfaces for generating 2D input patterns can further improve the usability of the workflow presented in Chapter 3, e.g., with a digital pen and a tablet device.

Additional future work focuses on different aspects of the algorithms to improve their performance. Hence, the computation time of the advanced path planner in Chapter 5 of long manufacturing paths with high resolution and multiple collision objects can be improved by using compiled algorithms, different optimization solvers, e.g., [145], more advanced collision avoidance concepts, e.g., [146], and machine learning [147]. Machine learning concepts are already employed to solve online and offline path-planning problems for industrial manufacturing processes, especially for grasping, imitation, and robot vision tasks, e.g., [148, 149]. Those concepts can be included in the presented algorithms to improve the computation time, e.g., an automatic generation of databases for specific processes and reuse in the planning [150] or by learning motion primitives from imitation [151].

Additionally, by incorporating the dynamic model of the manipulator into the advanced path-planning framework from Chapter 5, time or energy-optimal joint-space trajectories can be obtained. Instead of solving individual optimization problems sequentially without a look-ahead, graph-based or predictive methods can enlarge the search space and improve the optimal joint-space paths. An example approach is a bidirectional search to solve the path planning problem simultaneously starting from the first and last manufacturing path point [152]. A particular focus is placed on identifying sections where the robot cannot use all available DoF and, therefore, is very constrained, e.g., due to possible collisions, see [153, 154, 155]. Using these concepts to improve the presented algorithms increases the solution space and reduces the computation time, which makes flexible robotic work cells more attractive for complex manufacturing tasks.

Appendix A Industrial Robots

This section describes the industrial robots used in this thesis with their kinematic parameters and singular configurations. First, the collaborative robot Kuka LBR iiwa 14 R820, and second, the industrial robot Kuka Cybertech KR8 R1620 is presented.

Major parts of this chapter have been published in the author's work [8] and are adapted for this thesis.

A.1 KUKA LBR iiwa 14 R820

The Kuka LBR iiwa 14 R820 is a lightweight collaborative industrial robot. This robot is redundant due to the $n = 7$ revolute joints. Each joint includes a torque sensor, which allows one to estimate the external torques. A schematic drawing of the industrial robot attached with a pen on the end-effector is given in Fig. A.1. Note that the manipulator shown in Fig. A.1 is in the configuration $\mathbf{q} = \mathbf{0}$.

The kinematic parameters of the Kuka LBR iiwa 14 R820 are given in Tab. A.1, and the analytical inverse kinematics is presented in [73]. The singular configurations of the Kuka LBR iiwa 14 R820 manipulator are

$$\text{(A)}\ \ q_4 = 0^\circ\ , \tag{A.1a}$$

$$\text{(B)}\ \ q_2 = 0^\circ\ \wedge\ (q_3 = 90^\circ \vee q_3 = -90^\circ)\ , \tag{A.1b}$$

$$\text{(C)}\ \ q_2 = 0^\circ\ \wedge\ q_6 = 0^\circ\ , \tag{A.1c}$$

$$\text{(D)}\ \ (q_5 = 90^\circ \vee q_5 = -90^\circ)\ \wedge\ q_6 = 0^\circ\ , \tag{A.1d}$$

see [99].

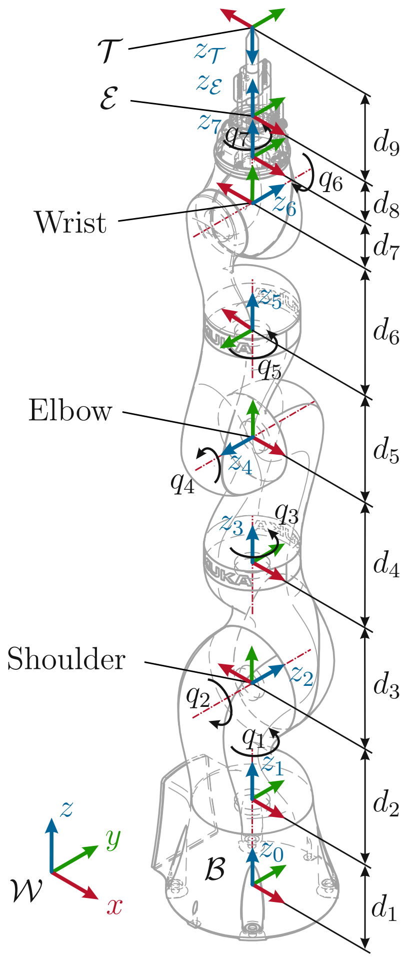


Figure A.1: KUKA LBR iiwa 14 R820 with annotated coordinate frames and revolute joints q_h turning around the z_h axes, $h = 1, \ldots, 7$, and a drawing tool mounted on the end-effector, cf. [144].

Table A.1: Kinematic parameters of the KUKA LBR iiwa 14 R820, see [144].

h	θ_h	d_h	$q_{h,\min}$	$q_{h,\max}$	$\dot{q}_{h,\max}$
-	-	mm	$^\circ$	$^\circ$	$^\circ\,\mathrm{s}^{-1}$
1	q_1	157.5	-170	170	85
2	q_2	202.5	-120	120	85
3	q_3	237.5	-170	170	100
4	q_4	182.5	-120	120	75
5	q_5	217.5	-170	170	130
6	q_6	182.5	-120	120	135
7	q_7	81	-175	175	135
8	-	71	-	-	-
9	-	150	-	-	-

A.2 KUKA Cybertech KR8 R1620

The KUKA Cybertech KR8 R1620 is an industrial robot with $n = 6$ revolute joints, of which the last three joints constitute a spherical wrist [29]. A schematic drawing is given in Fig. A.2 with a trim tool mounted on the end-effector. The configuration shown in Fig. A.2 is

$$\mathbf{q}^{\mathrm{T}} = \begin{bmatrix} 0^\circ & -90^\circ & 90^\circ & 0^\circ & 0^\circ & 0^\circ \end{bmatrix} . \tag{A.2}$$

The DENAVIT-HARTENBERG parameters, see, e.g., [29], together with the mechanical joint limits $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$ of the robot are listed in Tab. A.2. The analytical inverse kinematics of this manipulator is presented in [29]. Due to the wide axis range of joint 6, additional inverse kinematics solutions exist, see Tab. A.2.

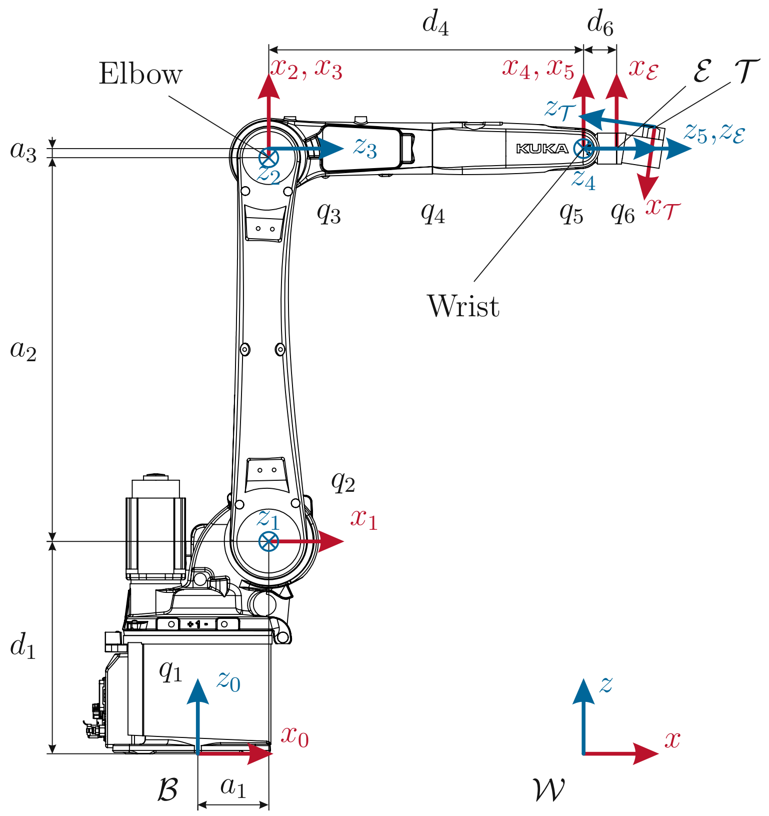


Figure A.2: KUKA Cybertech KR8 R1620 with annotated coordinate frames and revolute joints q_h turning around the z_{h-1} axes, $h = 1, \dots, 6$, and a trim tool mounted on the end-effector; adapted from [8], cf. [135].

Table A.2: DENAVIT-HARTENBERG and kinematic parameters of the KUKA Cybertech KR8 R1620; adapted from [8], see [135].

h	θ_h	d_h	ϑ_h	a_h	$q_{h,\mathrm{min}}$	$q_{h,\mathrm{max}}$	$\dot{q}_{h,\mathrm{max}}$
-	-	mm	$^\circ$	mm	$^\circ$	$^\circ$	$^\circ\,\mathrm{s}^{-1}$
1	q_1	450	-90	150	-170	170	220
2	q_2	0	0	810	-185	65	210
3	$q_3-\frac{\pi}{2}$	0	-90	20	-137	163	270
4	q_4	660	90	0	-185	185	381
5	q_5	0	-90	0	-120	120	311
6	q_6	80	0	0	-350	350	492

The singular configurations of the KUKA Cybertech KR8 R1620 manipulator are

$$\text{(A)}\quad q_5 = 0^\circ \tag{A.3a}$$

$$\text{(B)}\quad q_3 = \frac{\pi}{2} - \arctan\left(\frac{d_4}{a_3}\right) \approx 1.7^\circ \tag{A.3b}$$

$$\text{(C)}\quad a_1 + a_2\cos(q_2) + a_3\cos\left(\frac{\pi}{2} - q_2 - q_3\right) + d_4\sin\left(\frac{\pi}{2} - q_2 - q_3\right) = 0\ , \tag{A.3c}$$

see [99].

Major parts of this section have been published in the author's work [8] and are adapted for this thesis.

Appendix B Mathematics of Unit Quaternions

In this section, the utilized mathematical relations for unit quaternions are presented and derived. First, the definition of the quaternion product and the time derivative of unit quaternions are presented. Based on this, the time derivative of the LYAPUNOV function from Section 3.3 is computed. Next, the motivation of the objective function regarding the orientation deviation from Section 5.3.3 is shown, and finally, the gradients of the quaternion error are derived. Preliminary work was developed in the diploma thesis written by the author [37].

Major parts of this chapter have been published in the author's works [7, 10] and are adapted for this thesis.

B.1 Quaternion Product and Time Derivative of Unit Quaternions

The quaternion product $\otimes$ of two unit quaternions $\mathbf{o}_i^{\mathrm{T}} = \left[\eta_i \ \boldsymbol{\varepsilon}_i^{\mathrm{T}}\right], i = 1, 2$, is defined as, see, e.g., [29],

$$\mathbf{o}_1 \otimes \mathbf{o}_2 = \begin{bmatrix} \eta_1 \\ \boldsymbol{\varepsilon}_1 \end{bmatrix} \otimes \begin{bmatrix} \eta_2 \\ \boldsymbol{\varepsilon}_2 \end{bmatrix} = \begin{bmatrix} \eta_1\eta_2 - (\boldsymbol{\varepsilon}_1)^{\mathrm{T}}\boldsymbol{\varepsilon}_2 \\ \eta_1\boldsymbol{\varepsilon}_2 + \eta_2\boldsymbol{\varepsilon}_1 + \mathbf{S}(\boldsymbol{\varepsilon}_1)\boldsymbol{\varepsilon}_2 \end{bmatrix} , \tag{B.1}$$

with the skew-symmetric operator $\mathbf{S}(\cdot)$ from (2.14). The time derivative of a unit quaternion $\mathbf{o}^{\mathrm{T}} = \begin{bmatrix}\eta \ \boldsymbol{\varepsilon}^{\mathrm{T}}\end{bmatrix}$ is called *quaternion propagation* and reads as

$$\dot{\mathbf{o}} = \begin{bmatrix}\dot{\eta} \\ \dot{\boldsymbol{\varepsilon}}\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 \\ \boldsymbol{\omega}\end{bmatrix} \otimes \begin{bmatrix}\eta \\ \boldsymbol{\varepsilon}\end{bmatrix} = \frac{1}{2}\begin{bmatrix}0 & -\boldsymbol{\omega}^{\mathrm{T}} \\ \boldsymbol{\omega} & \mathbf{S}(\boldsymbol{\omega})\end{bmatrix}\begin{bmatrix}\eta \\ \boldsymbol{\varepsilon}\end{bmatrix} , \tag{B.2}$$

with the angular velocity $\boldsymbol{\omega}$, see [29]. Taking the quaternion product of $\dot{\mathbf{o}}$ with the inverse unit quaternion $(\mathbf{o}^{-1})^{\mathrm{T}} = \begin{bmatrix}\eta \ -\boldsymbol{\varepsilon}^{\mathrm{T}}\end{bmatrix}$ yields, see (B.2)

$$\boldsymbol{\omega} = 2\left(\eta\dot{\boldsymbol{\varepsilon}} - \dot{\eta}\boldsymbol{\varepsilon} - \mathbf{S}(\dot{\boldsymbol{\varepsilon}})\boldsymbol{\varepsilon}\right) , \tag{B.3}$$

see [29].

Major parts of this section have been published in the author's works [7, 37] and are adapted for this thesis.

B.2 Time Derivative of Lyapunov Function

In this section, the time derivative of the Lyapunov function (3.28) is derived. Similar to [68], the time derivative with the introduced terms a and b is

$$\dot{V} = \underbrace{2K_{\mathrm{o}}\left(\tilde{\eta}_{\mathcal{P}}(\dot{\eta}^{\mathcal{M}}_{\mathcal{P}} - \dot{\eta}^{\mathcal{T}}_{\mathcal{P}}) + (\tilde{\boldsymbol{\varepsilon}}_{\mathcal{P}})^{\mathrm{T}}(\dot{\boldsymbol{\varepsilon}}^{\mathcal{M}}_{\mathcal{P}} - \dot{\boldsymbol{\varepsilon}}^{\mathcal{T}}_{\mathcal{P}})\right)}_{a} + \underbrace{(\tilde{\boldsymbol{\omega}}_{\mathcal{P}})^{\mathrm{T}}(\dot{\boldsymbol{\omega}}^{\mathcal{M}}_{\mathcal{P}} - \dot{\boldsymbol{\omega}}^{\mathcal{T}}_{\mathcal{P}})}_{b} , \tag{B.4}$$

with the relations $\tilde{\eta}_{\mathcal{P}} = \eta^{\mathcal{M}}_{\mathcal{P}} - \eta^{\mathcal{T}}_{\mathcal{P}}$, $\tilde{\boldsymbol{\varepsilon}}_{\mathcal{P}} = \boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{P}} - \boldsymbol{\varepsilon}^{\mathcal{T}}_{\mathcal{P}}$, and $\tilde{\boldsymbol{\omega}}_{\mathcal{P}} = \boldsymbol{\omega}^{\mathcal{M}}_{\mathcal{P}} - \boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}}$. With (3.26) and (B.2), the term a simplifies to

$$\begin{aligned} a =& K_{\mathrm{o}}\tilde{\eta}_{\mathcal{P}}\left((-\boldsymbol{\omega}^{\mathcal{M}}_{\mathcal{P}})^{\mathrm{T}}\boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{P}} + (\boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}})^{\mathrm{T}}\boldsymbol{\varepsilon}^{\mathcal{T}}_{\mathcal{P}}\right) + \\ & \quad K_{\mathrm{o}}(\tilde{\boldsymbol{\varepsilon}}_{\mathcal{P}})^{\mathrm{T}}\left(\boldsymbol{\omega}^{\mathcal{M}}_{\mathcal{P}}\eta^{\mathcal{M}}_{\mathcal{P}} + \mathbf{S}(\boldsymbol{\omega}^{\mathcal{M}}_{\mathcal{P}})\boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{P}} - \boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}}\eta^{\mathcal{T}}_{\mathcal{P}} - \mathbf{S}(\boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}})\boldsymbol{\varepsilon}^{\mathcal{T}}_{\mathcal{P}}\right) \\ =& K_{\mathrm{o}}(\boldsymbol{\omega}^{\mathcal{M}}_{\mathcal{P}})^{\mathrm{T}}\left(\eta^{\mathcal{T}}_{\mathcal{P}}\boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{P}} - \eta^{\mathcal{M}}_{\mathcal{P}}\boldsymbol{\varepsilon}^{\mathcal{T}}_{\mathcal{P}}\right) - K_{\mathrm{o}}(\boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}})^{\mathrm{T}}\left(\eta^{\mathcal{T}}_{\mathcal{P}}\boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{P}} - \eta^{\mathcal{M}}_{\mathcal{P}}\boldsymbol{\varepsilon}^{\mathcal{T}}_{\mathcal{P}}\right) - \\ & \quad K_{\mathrm{o}}(\boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{P}})^{\mathrm{T}}\mathbf{S}(\boldsymbol{\omega}^{\mathcal{T}}_{\mathcal{P}})\boldsymbol{\varepsilon}^{\mathcal{T}}_{\mathcal{P}} - K_{\mathrm{o}}(\boldsymbol{\varepsilon}^{\mathcal{T}}_{\mathcal{P}})^{\mathrm{T}}\mathbf{S}(\boldsymbol{\omega}^{\mathcal{M}}_{\mathcal{P}})\boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{P}} \\ =& K_{\mathrm{o}}(\tilde{\boldsymbol{\omega}}_{\mathcal{P}})^{\mathrm{T}}\boldsymbol{\varepsilon}^{\mathcal{M}}_{\mathcal{T}} , \end{aligned} \tag{B.5}$$

where the relation

$$-(\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}})^{\mathrm{T}}\mathbf{S}(\boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{T}})\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} - (\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}})^{\mathrm{T}}\mathbf{S}(\boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{M}})\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} = -(\tilde{\boldsymbol{\omega}}_{\mathcal{P}})^{\mathrm{T}}(\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} \times \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}})$$
$$= -(\tilde{\boldsymbol{\omega}}_{\mathcal{P}})^{\mathrm{T}}\mathbf{S}(\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}})\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} \tag{B.6}$$

with the cross product $\times$ was used.

The term b in (B.4) with (3.25) takes the form

$$b = (\tilde{\boldsymbol{\omega}}_{\mathcal{P}})^{\mathrm{T}}\left(-\mathbf{K}_{\omega}\tilde{\boldsymbol{\omega}}_{\mathcal{P}} - K_{\mathrm{o}}\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}\right) \ . \tag{B.7}$$

Inserting (B.5) and (B.7) into (B.4) results in

$$\dot{V} = -(\tilde{\boldsymbol{\omega}}_{\mathcal{P}})^{\mathrm{T}}\mathbf{K}_{\omega}\tilde{\boldsymbol{\omega}}_{\mathcal{P}} \ , \tag{B.8}$$

see (3.29).

Major parts of this section have been published in the author's works [7, 37] and are adapted for this thesis.

B.3 Objective Function of Orientation Deviation

In this section, the relation between unit quaternions and the angle-axis representation with the vector $\mathbf{l}$ and the rotation θ around $\mathbf{l}$ is discussed. The mathematical relation is given by, e.g., [29],

$$\eta = \cos\left(\frac{\theta}{2}\right) \quad \text{and} \tag{B.9a}$$

$$\boldsymbol{\varepsilon} = \mathbf{l}\sin\left(\frac{\theta}{2}\right) \ . \tag{B.9b}$$

In the following, two examples with rotations around one axis in (B.10) and around two axes in (B.11) are shown, where the specific range of rotation is chosen to clarify the visual presentation in Fig. B.1. In Fig. B.1a, pure rotations around the z-axis w.r.t. the dashed coordinate frame $\mathcal{T}$ are illustrated, which correspond to

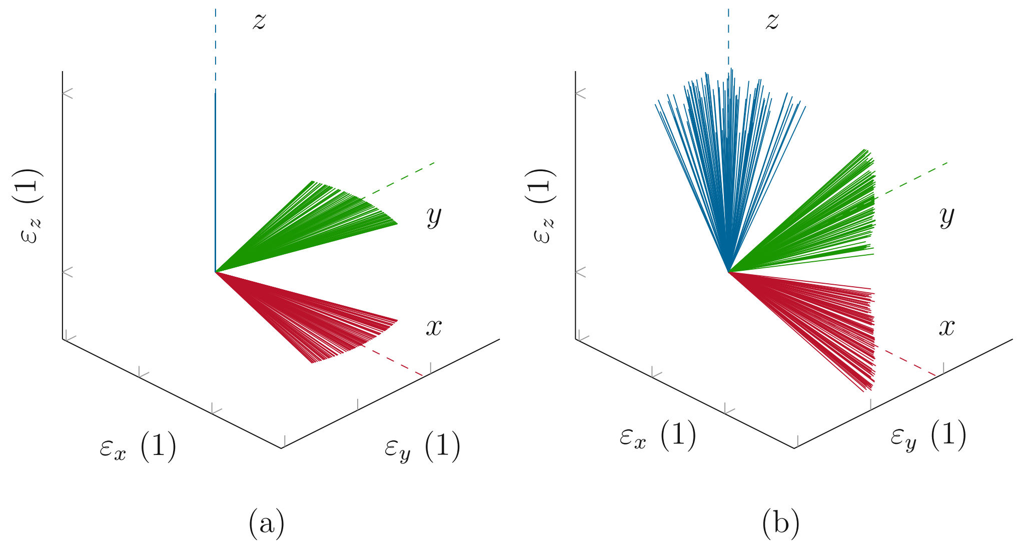


Figure B.1: Orientation deviations $\mathbf{o}_{\mathcal{T}}^{\mathcal{M}}$ w.r.t. the dashed coordinate frame $\mathcal{T}$ in the origin with rotations around (a) the z-axis, (b) the x- and y-axis; adapted from [10].

unit quaternions $\mathbf{o}_{\mathcal{T}}^{\mathcal{M}}$ in the form

$$\mathbf{o}_{\mathcal{T}}^{\mathcal{M}} = \frac{\check{\mathbf{o}}_{\mathcal{T}}^{\mathcal{M}}}{\|\check{\mathbf{o}}_{\mathcal{T}}^{\mathcal{M}}\|_2} \ , \quad \check{\mathbf{o}}_{\mathcal{T}}^{\mathcal{M}} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \varepsilon_z \end{bmatrix} \ , \quad -0.15 \leq \varepsilon_z \leq 0.15 \ . \tag{B.10}$$

In (B.10), the first two entries of the vector part of the quaternion error $\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}$ are zero, while the last entry is a small random number corresponding to $\mathbf{l}^{\mathrm{T}} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$ in (B.9b). Hence, unit quaternions in the form (B.10) represent pure rotations around the z-axis.

In contrast, quaternion errors in the form

$$\mathbf{o}_{\mathcal{T}}^{\mathcal{M}} = \frac{\check{\mathbf{o}}_{\mathcal{T}}^{\mathcal{M}}}{\|\check{\mathbf{o}}_{\mathcal{T}}^{\mathcal{M}}\|_2} \ , \quad \check{\mathbf{o}}_{\mathcal{T}}^{\mathcal{M}} = \begin{bmatrix} 1 \\ \varepsilon_x \\ \varepsilon_y \\ 0 \end{bmatrix} , \quad \begin{matrix} -0.15 \leq \varepsilon_x \leq 0.15 \\ -0.15 \leq \varepsilon_y \leq 0.15 \end{matrix} \ , \tag{B.11}$$

have negligible rotations around the z-axis, while the rotations around the x- and y-axis are significant, see Fig. B.1b. This example shows that penalizing the individual components of $\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}$ in (5.19) using $\mathbf{A}_{\mathrm{o}}$ minimizes the rotation around the respective axis in $\mathbf{l}$.

Major parts of this section have been published in the author's work [10] and are adapted for this thesis.

B.4 Gradients of Unit Quaternions

The time derivative of the quaternion error (3.26) reads as

$$\frac{\mathrm{d}\mathbf{o}_{\mathcal{T}}^{\mathcal{M}}}{\mathrm{d}t} = \begin{bmatrix} \frac{\mathrm{d}\eta_{\mathcal{T}}^{\mathcal{M}}}{\mathrm{d}t} \\ \frac{\mathrm{d}\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} \dot{\eta}_{\mathcal{P}}^{\mathcal{M}}\eta_{\mathcal{P}}^{\mathcal{T}} + \eta_{\mathcal{P}}^{\mathcal{M}}\dot{\eta}_{\mathcal{P}}^{\mathcal{T}} + (\dot{\boldsymbol{\varepsilon}}_{\mathcal{P}}^{\mathcal{M}})^{\mathrm{T}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} + (\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}})^{\mathrm{T}}\dot{\boldsymbol{\varepsilon}}_{\mathcal{P}}^{\mathcal{T}} \\ \dot{\eta}_{\mathcal{P}}^{\mathcal{T}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} + \eta_{\mathcal{P}}^{\mathcal{T}}\dot{\boldsymbol{\varepsilon}}_{\mathcal{P}}^{\mathcal{M}} - \dot{\eta}_{\mathcal{P}}^{\mathcal{M}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} - \eta_{\mathcal{P}}^{\mathcal{M}}\dot{\boldsymbol{\varepsilon}}_{\mathcal{P}}^{\mathcal{T}} - \dot{\boldsymbol{\varepsilon}}_{\mathcal{P}}^{\mathcal{M}} \times \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} - \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} \times \dot{\boldsymbol{\varepsilon}}_{\mathcal{P}}^{\mathcal{T}} \end{bmatrix} =$$
$$\frac{1}{2}\begin{bmatrix} \left(\boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{M}}\right)^{\mathrm{T}}\left(-\eta_{\mathcal{P}}^{\mathcal{T}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} + \eta_{\mathcal{P}}^{\mathcal{M}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} + \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} \times \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}}\right) + \left(\boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{T}}\right)^{\mathrm{T}}\left(\eta_{\mathcal{P}}^{\mathcal{T}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} - \eta_{\mathcal{P}}^{\mathcal{M}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} - \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} \times \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}}\right) \\ \left(\boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{M}} - \boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{T}}\right)\left(\eta_{\mathcal{P}}^{\mathcal{M}}\eta_{\mathcal{P}}^{\mathcal{T}} + (\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}})^{\mathrm{T}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}}\right) + \left(\boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{M}} + \boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{T}}\right) \times \left(\eta_{\mathcal{P}}^{\mathcal{T}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} - \eta_{\mathcal{P}}^{\mathcal{M}}\boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}} - \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{M}} \times \boldsymbol{\varepsilon}_{\mathcal{P}}^{\mathcal{T}}\right) \end{bmatrix}, \tag{B.12}$$

where the time derivative of a unit quaternion (B.2) is inserted.

Factorizing (B.12) with (3.26) results in

$$\frac{\partial \mathbf{o}_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}}\dot{\mathbf{q}} = \begin{bmatrix} \frac{\partial \eta_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \\ \frac{\partial \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \end{bmatrix}\dot{\mathbf{q}} = \frac{1}{2}\begin{bmatrix} -\left(\boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{M}} - \boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{T}}\right)^{\mathrm{T}}\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}} \\ \left(\boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{M}} - \boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{T}}\right)\eta_{\mathcal{T}}^{\mathcal{M}} + \left(\boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{M}} + \boldsymbol{\omega}_{\mathcal{P}}^{\mathcal{T}}\right) \times \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}} \end{bmatrix}, \tag{B.13}$$

and after eliminating $\dot{\mathbf{q}}$ with (2.13), the gradient yields

$$\frac{\partial \mathbf{o}_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial \eta_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \\ \frac{\partial \boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}}}{\partial \mathbf{q}} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} (\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}})^{\mathrm{T}}\mathbf{J}_{\mathcal{P},\omega}^{\mathcal{T}}(\mathbf{q}) \\ -\eta_{\mathcal{T}}^{\mathcal{M}}\mathbf{J}_{\mathcal{P},\omega}^{\mathcal{T}}(\mathbf{q}) - \mathbf{S}(\boldsymbol{\varepsilon}_{\mathcal{T}}^{\mathcal{M}})\mathbf{J}_{\mathcal{P},\omega}^{\mathcal{T}}(\mathbf{q}) \end{bmatrix}. \tag{B.14}$$

Since the manufacturing path $H_{\mathcal{P}}^{\mathcal{M}}$ is assumed to be stationary, the corresponding Jacobian $\mathbf{J}_{\mathcal{P},\omega}^{\mathcal{M}}$ vanishes.

Major parts of this section have been published in the author's work [10] and are adapted for this thesis.

Appendix C Supplementary Material

Along with the published works from the author, several videos were created and published as supplementary material. The links to those videos are listed in the following:

[7] T. Weingartshofer, M. Schwegel, C. Hartl-Nesic, T. Glück, and A. Kugi, “Collaborative Synchronization of a 7-Axis Robot”, *IFAC-PapersOnLine*, vol. 52, no. 15, pp. 507–512, 2019, © IFAC.
`www.acin.tuwien.ac.at/b2ac`

[8] T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, “Optimal TCP and Robot Base Placement for a Set of Complex Continuous Paths”, in *IEEE International Conference on Robotics and Automation*, 2021, pp. 9659-9665, © IEEE.
`www.acin.tuwien.ac.at/9c5f`

[9] T. Weingartshofer, A. Haddadi, C. Hartl-Nesic, and A. Kugi, “Flexible Robotic Drawing on 3D Objects with an Industrial Robot”, in *IEEE Conference on Control Technology and Applications*, 2022, pp. 29-36, © IEEE.
`www.acin.tuwien.ac.at/3bc0`

[10] T. Weingartshofer, B. Bischof, M. Meiringer, C. Hartl-Nesic, and A. Kugi, “Optimization-based path planning framework for industrial manufacturing processes with complex continuous paths”, *Robotics and Computer-Integrated Manufacturing*, vol. 82, no. 102516, pp. 1-16, 2023, © Authors, CC BY 4.0.
`www.acin.tuwien.ac.at/4adf`

[11] T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, “Automatic and Flexible Robotic Drawing on Complex Surfaces with an Industrial Robot”, *IEEE Transactions on Control Systems Technology*, 2023, © Authors, CC BY 4.0, in press.
`www.acin.tuwien.ac.at/c1eb`

Bibliography

[1] International Federation of Robotics, "World Robotics 2020 Report," 2020, Accessed on 1st July 2022. [Online]. Available: http://reparti.free.fr/robotics2000.pdf

[2] P. Barosz, G. Gołda, and A. Kampa, "Efficiency Analysis of Manufacturing Line with Industrial Robots and Human Operators," *Applied Sciences*, vol. 10, no. 8: 2862, pp. 1–15, 2020.

[3] M. T. Fralix, "From mass production to mass customization," *Journal of Textile and Apparel, Technology and Management*, vol. 1, no. 2, pp. 1–7, 2001.

[4] W. Terkaj, T. Tolio, and A. Valente, *Design of Flexible Production Systems*. Berlin Heidelberg: Springer, 2009, ch. A Review on Manufacturing Flexibility, pp. 41–61.

[5] Y. Liu, L. Wang, S. Makris, and J. Krüger, "Smart robotics for manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 82, no. 102535, pp. 1–2, 2023.

[6] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, "Intelligent Manufacturing in the Context of Industry 4.0: A Review," *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.

[7] T. Weingartshofer, M. Schwegel, C. Hartl-Nesic, T. Glück, and A. Kugi, "Collaborative Synchronization of a 7-Axis Robot," *IFAC-PapersOnLine*, vol. 52, no. 15, pp. 507–512, 2019, © IFAC.

[8] T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, “Optimal TCP and Robot Base Placement for a Set of Complex Continuous Paths,” in *IEEE International Conference on Robotics and Automation*, 2021, pp. 9659–9665, © IEEE.

[9] T. Weingartshofer, A. Haddadi, C. Hartl-Nesic, and A. Kugi, “Flexible Robotic Drawing on 3D Objects with an Industrial Robot,” in *IEEE Conference on Control Technology and Applications*, 2022, pp. 29–36, © IEEE.

[10] T. Weingartshofer, B. Bischof, M. Meiringer, C. Hartl-Nesic, and A. Kugi, “Optimization-based Path Planning Framework for Industrial Manufacturing Processes with Complex Continuous Paths,” *Robotics and Computer-Integrated Manufacturing*, vol. 82, no. 102516, pp. 1–16, 2023, © Authors, CC BY 4.0.

[11] T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, “Automatic and Flexible Robotic Drawing on Complex Surfaces with an Industrial Robot,” *IEEE Transactions on Control Systems Technology*, 2023, © Authors, CC BY 4.0, in press.

[12] I. Iglesias, M. Sebastián, and J. Ares, “Overview of the state of robotic machining: Current situation and future potential,” *Procedia Engineering*, vol. 132, pp. 911–917, 2015.

[13] S. H. Kim, E. Nam, T. I. Ha, S.-H. Hwang, J. H. Lee, S.-H. Park, and B.-K. Min, “Robotic Machining: A Review of Recent Progress,” *International Journal of Precision Engineering and Manufacturing*, vol. 20, no. 9, pp. 1629–1642, 2019.

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

[15] T. Wagner, “Integrated robotic gluing system,” in *VDE International Symposium on Robotics and German Conference on Robotics*, 2010, pp. 1–3.

[16] G. Biegelbauer, M. Richtsfeld, W. Wohlkinger, M. Vincze, and M. Herkt, “Optical Seam Following for Automated Robot Sewing,” in *IEEE International Conference on Robotics and Automation*, 2007, pp. 4758–4763.

[17] H. Jindal and S. Kaur, “Robotics and Automation in Textile Industry,” *International Journal of Scientific Research in Science, Engineering and Technology*, vol. 8, no. 3, pp. 40–45, 2021.

[18] G. Ye and R. Alterovitz, *Robotics Research*, ser. Springer Tracts in Advanced Robotics. Cham: Springer, 2017, vol. 100, ch. Demonstration-Guided Motion Planning, pp. 291–307.

[19] C. Connolly, “Technology and applications of ABB RobotStudio,” *Industrial Robot*, vol. 36, no. 6, pp. 540–545, 2009.

[20] Z. Pan, J. Polden, N. Larkin, S. V. Duin, and J. Norrish, “Recent progress on programming methods for industrial robots,” *Robotics and Computer-Integrated Manufacturing*, vol. 28, no. 2, pp. 87–94, 2012.

[21] Z. Liu, Q. Liu, W. Xu, L. Wang, and Z. Zhou, “Robot learning towards smart robotic manufacturing: A review,” *Robotics and Computer-Integrated Manufacturing*, vol. 77, no. 102360, pp. 1–21, 2022.

[22] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, “A Comprehensive Review of Coverage Path Planning in Robotics Using Classical and Heuristic Algorithms,” *IEEE Access*, vol. 9, pp. 119 310–119 342, 2021.

[23] A. Samuel and C. Burvill, “Tracing Surfaces with a Robot Manipulator,” in *IEEE International Conference on Advanced Robotics 'Robots in Unstructured Environments*, vol. 1, 1991, pp. 493–499.

[24] W. Sheng, H. Chen, N. Xi, and Y. Chen, “Tool Path Planning for Compound Surfaces in Spray Forming Processes,” *IEEE Transactions on Automation Science and Engineering*, vol. 2, no. 3, pp. 240–249, 2005.

[25] H. Chen, T. Fuhlbrigge, and X. Li, “Automated Industrial Robot Path Planning for Spray Painting Process: A Review,” in *IEEE International Conference on Automation Science and Engineering*, 2008, pp. 522–527.

[26] Q. Yu, G. Wang, and K. Chen, “A Robotic Spraying Path Generation Algorithm for Free-Form Surface Based on Constant Coating Overlapping Width,” in *IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems*, 2015, pp. 1045–1049.

[27] Z. Liu, R. Li, L. Zhao, Y. Xia, Z. Qin, and K. Zhu, “Automatic joint motion planning of 9-DOF robot based on redundancy optimization for wheel hub polishing,” *Robotics and Computer-Integrated Manufacturing*, vol. 81, no. 102500, pp. 1–13, 2023.

[28] A. Kharidege, D. T. Ting, and Z. Yajun, “A practical approach for automated polishing system of free-form surface path generation based on industrial arm robot,” *The International Journal of Advanced Manufacturing Technology*, vol. 93, no. 9, pp. 3921–3934, 2017.

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

[30] S. Lengagne, N. Ramdani, and P. Fraisse, “Planning and Fast Replanning Safe Motions for Humanoid Robots,” *IEEE Transactions on Robotics*, vol. 27, no. 6, pp. 1095–1106, 2011.

[31] N. C. N. Doan and W. Lin, “Optimal robot placement with consideration of redundancy problem for wrist-partitioned 6R articulated robots,” *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 233–242, 2017.

[32] H. Fang, S. Ong, and A. Nee, “Robot path planning optimization for welding complex joints,” *The International Journal of Advanced Manufacturing Technology*, vol. 90, no. 9, pp. 3829–3839, 2016.

[33] W. Gao, Q. Tang, J. Yao, and Y. Yang, “Automatic motion planning for complex welding problems by considering angular redundancy,” *Robotics and Computer-Integrated Manufacturing*, vol. 62, no. 101862, pp. 1–16, 2020.

[34] N. Asakawa, K. Toda, and Y. Takeuchi, “Automation of chamfering by an industrial robot for the case of hole on free-curved surface,” *Robotics and Computer-Integrated Manufacturing*, vol. 18, no. 5-6, pp. 379–385, 2002.

[35] M. W. Spong and M. Vidyasagar, *Robot Dynamics and Control*. Hoboken: John Wiley & Sons, 1989.

[36] R. K. Malhan, A. V. Shembekar, A. M. Kabir, P. M. Bhatt, B. Shah, S. Zanio, S. Nutt, and S. K. Gupta, “Automated planning for robotic layup of composite prepreg,” *Robotics and Computer-Integrated Manufacturing*, vol. 67, no. 102020, pp. 1–27, 2021.

[37] T. Weingartshofer, “Kollaborative Synchronisation eines 7-Achs-Roboters,” Master’s thesis, Automation & Control Institute (ACIN), TU Wien, 2018.

[38] A. Haddadi, “Robotic Drawing on a 3D Object,” Master’s thesis, Automation & Control Institute (ACIN), TU Wien, 2020.

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

[40] 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.

[41] 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.

[42] P. Tresset and F. Fol Leymarie, “Portrait Drawing by Paul the Robot,” *Computers Graphics*, vol. 37, no. 5, pp. 348–363, 2013.

[43] 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.

[44] 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.

[45] M. Pichkalev, R. Lavrenov, R. Safin, and K. Hsia, “Face drawing by KUKA 6 axis robot manipulator,” in *IEEE International Conference on Developments in eSystems Engineering*, 2019, pp. 709–714.

[46] 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.

[47] 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.

[48] D. Ding, C. Shen, Z. Pan, D. Cuiuri, H. Li, N. Larkin, and S. van Duin, "Towards an automated robotic arc-welding-based additive manufacturing system from CAD to finished part," *Computer-Aided Design*, vol. 73, pp. 66–75, 2016.

[49] M. Daneshmand, A. Helmi, E. Avots, F. Noroozi, F. Alisinanoglu, H. S. Arslan, J. Gorbova, R. E. Haamer, C. Ozcinar, and G. Anbarjafari, "3D Scanning: A Comprehensive Survey," *arXiv:1801.08863*, pp. 1–18, 2018.

[50] Wandelbots, "TracePen," Accessed on 23rd Nov 2022. [Online]. Available: https://wandelbots.com/

[51] M. Schwegel, H. Augschöll, C. Hartl-Nesic, and A. Kugi, "Teach-In for Force Sensitive Automatic Rope Winding in Three Dimensions," in *Digital-Fachtagung VDI Mechatronik*, 2021, pp. 193–198.

[52] D. Ehmann and C. Wittenberg, "The idea of Virtual Teach-In in the field of industrial robotics," in *IEEE International Conference on Control and Automation*, 2018, pp. 680–685.

[53] M. Spitzer, I. Gsellmann, M. Hebenstreit, and M. Rosenberger, "Mixed Reality Robot Teach-In Assistant," in *Conference on Learning Factories*, 2022, pp. 1–6.

[54] C. Wögerer, M. Mühlberger, M. Ikeda, J. Kastner, N. C. Chitturi, and A. Pichler, "Inkjet Printings on FFF printed curved surfaces," in *Fraunhofer Direct Digital Manufacturing Conference*, 2018, pp. 1–4.

[55] ABB, "PixelPaint," Accessed on 23rd Nov 2022. [Online]. Available: https://new.abb.com/products/robotics/de/funktionspakete/pixelpaint

[56] P. Urhal, A. Weightman, C. Diver, and P. Bartolo, "Robot assisted additive manufacturing: A review," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 335–345, 2019.

[57] B. Levy, S. Petitjean, N. Ray, and J. Maillot, "Least Squares Conformal Maps for Automatic Texture Atlas Generation," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 362–371, 2002.

[58] 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, no. 9, pp. 2469–2476, 2012.

[59] A. Hubeli and M. Gross, "Multiresolution Feature Extraction for Unstructured Meshes," in *IEEE Conference on Visualization*, 2001, pp. 287–294.

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

[61] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy, *Polygon Mesh Processing*. Natick: A K Peters, 2010.

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

[63] P. Shirley and S. Marschner, *Fundamentals of Computer Graphics*, 3rd ed. Boca Raton: Taylor & Francis, 2009.

[64] T. Möller and B. Trumbore, "Fast, Minimum Storage Ray-Triangle Intersection," *Journal of Graphics Tools*, vol. 2, no. 1, pp. 21–28, 1997.

[65] E. Lee, "Choosing nodes in parametric curve interpolation," *Computer-Aided Design*, vol. 21, no. 6, pp. 363–370, 1989.

[66] 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.

[67] 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.

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

[69] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River: Prentice Hall, 2002.

[70] C. Ott, *Cartesian Impedance Control of Redundant and Flexible-Joint Robots*, ser. Springer Tracts in Advanced Robotics. Berlin Heidelberg: Springer, 2008.

[71] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge: Cambridge University Press, 2004.

[72] J. Tuszynski, "Triangle/Ray Intersection," *MATLAB Central File Exchange*, 2018, Accessed on 23rd Nov 2022. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/33073-triangle-ray-intersection

[73] 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.

[74] G. Pamanes and S. Zeghloul, "Optimal placement of robotic manipulators using multiple kinematic criteria," in *IEEE International Conference on Robotics and Automation*, vol. 1, 1991, pp. 933–938.

[75] A. Nektarios and N. A. Aspragathos, "Optimal location of a general position and orientation end-effector's path relative to manipulator's base, considering velocity performance," *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 2, pp. 162–173, 2010.

[76] B. Nelson and M. Donath, "Optimizing the Location of Assembly Tasks in a Manipulator's Workspace," *Journal of Robotic Systems*, vol. 7, no. 6, pp. 791–811, 1990.

[77] U. Schneider, J. R. D. Posada, and A. Verl, "Automatic Pose Optimization for Robotic Processes," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 2054–2059.

[78] J. Feddema, "Kinematically Optimal Robot Placement for Minimum Time Coordinated Motion," in *IEEE International Conference on Robotics and Automation*, vol. 4, 1996, pp. 3395–3400.

[79] Y. Lin, H. Zhao, and H. Ding, "Posture optimization methodology of 6R industrial robots for machining using performance evaluation indexes," *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 59–72, 2017.

[80] C. Dumas, S. Caro, S. Garnier, and B. Furet, “Workpiece Placement Optimization of Six-Revolute Industrial Serial Robots for Machining Operations,” in *ASME Conference on Engineering Systems Design and Analysis*, 2012, pp. 419–428.

[81] S. Caro, C. Dumas, S. Garnier, and B. Furet, “Workpiece Placement Optimization for Machining Operations with a KUKA KR270-2 Robot,” in *IEEE International Conference on Robotics and Automation*, 2013, pp. 2921–2926.

[82] G.-C. Vosniakos and E. Matsas, “Improving feasibility of robotic milling through robot placement optimisation,” *Robotics and Computer-Integrated Manufacturing*, vol. 26, no. 5, pp. 517–525, 2010.

[83] R. K. Malhan, A. M. Kabir, B. Shah, and S. K. Gupta, “Identifying Feasible Workpiece Placement with Respect to Redundant Manipulator for Complex Manufacturing Tasks,” in *IEEE International Conference on Robotics and Automation*, 2019, pp. 5585–5591.

[84] D. Spensieri, J. S. Carlson, R. Bohlin, J. Kressin, and J. Shi, “Optimal robot placement for tasks execution,” *Procedia CIRP*, vol. 44, pp. 395–400, 2016.

[85] S. Mitsi, K.-D. Bouzakis, D. Sagris, and G. Mansour, “Determination of optimum robot base location considering discrete end-effector positions by means of hybrid genetic algorithm,” *Robotics and Computer-Integrated Manufacturing*, vol. 24, no. 1, pp. 50–59, 2008.

[86] W. Bu, Z. Liu, and J. Tan, “Industrial robot layout based on operation sequence optimisation,” *International Journal of Production Research*, vol. 47, no. 15, pp. 4125–4145, 2009.

[87] J. Xu, K. Harada, W. Wan, T. Ueshiba, and Y. Domae, “Planning an Efficient and Robust Base Sequence for a Mobile Manipulator Performing Multiple Pick-and-place Tasks,” in *IEEE International Conference on Robotics and Automation*, 2020, pp. 11 018–11 024.

[88] N. Vahrenkamp, T. Asfour, and R. Dillmann, “Robot Placement based on Reachability Inversion,” in *IEEE International Conference on Robotics and Automation*, 2013, pp. 1970–1975.

[89] A. Makhal and A. K. Goins, “Reuleaux: Robot Base Placement by Reachability Analysis,” in *IEEE International Conference on Robotic Computing*, 2018, pp. 137–142.

[90] F. Zacharias, C. Borst, and G. Hirzinger, “Capturing Robot Workspace Structure: Representing Robot Capabilities,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3229–3236.

[91] B. Kamrani, V. Berbyuk, D. Wäppling, U. Stickelmann, and X. Feng, “Optimal robot placement using response surface method,” *The International Journal of Advanced Manufacturing Technology*, vol. 44, no. 1, pp. 201–210, 2009.

[92] Z. Kingston, M. Moll, and L. E. Kavraki, “Sampling-Based Methods for Motion Planning with Constraints,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 159–185, 2018.

[93] J. D. Maeyer, B. Moyaers, and E. Demeester, “Cartesian Path Planning for Arc Welding Robots: Evaluation of the Descartes Algorithm,” in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2017, pp. 1–8.

[94] D. Rakita, B. Mutlu, and M. Gleicher, “RelaxedIK: Real-time Synthesis of Accurate and Feasible Robot Arm Motion,” in *Robotics: Science and Systems*, vol. XIV, 2018, pp. 26–30.

[95] K. Hauser, *Algorithmic Foundations of Robotics XII*, ser. Springer Proceedings in Advanced Robotics. Cham: Springer, 2020, vol. 13, ch. Continuous Pseudoinversion of a Multivariate Function: Application to Global Redundancy Resolution, pp. 496–511.

[96] H.-M. Gutmann, “A Radial Basis Function Method for Global Optimization,” *Journal of Global Optimization*, vol. 19, no. 3, pp. 201–227, 2001.

[97] Q. Wu, J. Lu, W. Zou, and D. Xu, “Path Planning for Surface Inspection on a Robot-Based Scanning System,” in *IEEE International Conference on Mechatronics and Automation*, 2015, pp. 2284–2289.

[98] F. Nagata, Y. Kusumoto, Y. Fujimoto, and K. Watanabe, “Robotic sanding system for new designed furniture with free-formed surface,” *Robotics and Computer-Integrated Manufacturing*, vol. 23, no. 4, pp. 371–379, 2007.

[99] F. Beck, M. N. Vu, C. Hartl-Nesic, and A. Kugi, “Singularity Avoidance with Application to Online Trajectory Optimization for Serial Manipulators,” *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 284–291, 2023.

[100] I. A. Sucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[101] K. Hauser and V. Ng-Thow-Hing, “Fast Smoothing of Manipulator Trajectories using Optimal Bounded-Acceleration Shortcuts,” in *IEEE International Conference on Robotics and Automation*, 2010, pp. 2493–2498.

[102] R. Smits, “KDL: Kinematics and Dynamics Library,” Orocos, Accessed on 1st July 2022. [Online]. Available: http://www.orocos.org/kdl

[103] R. Diankov, “Automated Construction of Robotic Manipulation Programs,” Ph.D. dissertation, Carnegie Mellon University, 2010, Accessed on 1st July 2022. [Online]. Available: http://www.programmingvision.com/rosen_diankov_thesis.pdf

[104] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, “An Integrated Approach to Inverse Kinematics and Path Planning for Redundant Manipulators,” in *IEEE International Conference on Robotics and Automation*, 2006, pp. 1874–1879.

[105] M. Elbanhawi and M. Simic, “Sampling-Based Robot Motion Planning: A Review,” *IEEE Access*, vol. 2, pp. 56–77, 2014.

[106] K. Hauser and Y. Zhou, “Asymptotically Optimal Planning by Feasible Kinodynamic Planning in a State–Cost Space,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1431–1443, 2016.

[107] S. Seereeram and J. Wen, “A Global Approach to Path Planning for Redundant Manipulators,” in *IEEE International Conference on Robotics and Automation*, vol. 2, 1993, pp. 283–288.

[108] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization," in *Robotics: Science and Systems*, vol. IX, 2013, pp. 1–10.

[109] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient Optimization Techniques for Efficient Motion Planning," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 489–494.

[110] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic Trajectory Optimization for Motion Planning," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[111] N. C. N. Doan, P. Y. Tao, and W. Lin, "Optimal Redundancy Resolution for Robotic Arc Welding Using Modified Particle Swarm Optimization," in *IEEE International Conference on Advanced Intelligent Mechatronics*, 2016, pp. 554–559.

[112] M. Gadaleta, M. Pellicciari, and G. Berselli, "Optimization of the energy consumption of industrial robots for automatic code generation," *Robotics and Computer-Integrated Manufacturing*, vol. 57, pp. 452–464, 2019.

[113] C. G. L. Bianco and A. Piazzi, "A genetic/interval approach to optimal trajectory planning of industrial robots under torque constraints," in *IEEE European Control Conference*, 1999, pp. 942–947.

[114] I. Gentilini, K. Nagamatsu, and K. Shimada, "Cycle Time based Multi-Goal Path Optimization for Redundant Robotic Systems," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1786–1792.

[115] J. Kim and E. A. Croft, "Online near time-optimal trajectory planning for industrial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 58, pp. 158–171, 2019.

[116] A. Gasparetto and V. Zanotto, "Optimal trajectory planning for industrial robots," *Advances in Engineering Software*, vol. 41, no. 4, pp. 548–556, 2010.

[117] J. Huang, P. Hu, K. Wu, and M. Zeng, "Optimal time-jerk trajectory planning for industrial robots," *Mechanism and Machine Theory*, vol. 121, pp. 530–544, 2018.

[118] F. Rubio, C. Llopis-Albert, F. Valero, and J. L. Suñer, "Industrial robot efficient trajectory generation without collision through the evolution of the optimal trajectory," *Robotics and Autonomous Systems*, vol. 86, pp. 106–112, 2016.

[119] M. Ratiu and M. A. Prichici, "Industrial robot trajectory optimization- a review," in *MATEC Web of Conferences*, vol. 126, no. 02005, 2017, pp. 1–6.

[120] T. Chettibi, H. Lehtihet, M. Haddad, and S. Hanchi, "Minimum cost trajectory planning for industrial robots," *European Journal of Mechanics - A/Solids*, vol. 23, no. 4, pp. 703–715, 2004.

[121] J. Polden, Z. Pan, N. Larkin, and S. van Duin, "Adaptive Partial Shortcuts: Path Optimization for Industrial Robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 1, pp. 35–47, 2016.

[122] S. Alatartsev, A. Belov, M. Nykolaychuk, and F. Ortmeier, "Robot Trajectory Optimization for the Relaxed End-effector Path," in *IEEE International Conference on Informatics in Control, Automation and Robotics*, 2014, pp. 385–390.

[123] F. Debrouwere, W. V. Loock, G. Pipeleers, and J. Swevers, "Optimal Tube Following for Robotic Manipulators," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 305–310, 2014.

[124] A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir, "Inverse Kinematics Techniques in Computer Graphics: A Survey," *Computer Graphics Forum*, vol. 37, no. 6, pp. 35–58, 2017.

[125] P. Beeson and B. Ames, "TRAC-IK: An Open-Source Library for Improved Solving of Generic Inverse Kinematics," in *IEEE-RAS International Conference on Humanoid Robots*, 2015, pp. 928–935.

[126] K. Hauser, "Learning the Problem-Optimum Map: Analysis and Application to Global Optimization in Robotics," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 141–152, 2017.

[127] L.-C. Wang and C. Chen, “A Combined Optimization Method for Solving the Inverse Kinematics Problems of Mechanical Manipulators,” *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 489–499, 1991.

[128] P. Ruppel, N. Hendrich, S. Starke, and J. Zhang, “Cost Functions to Specify Full-Body Motion and Multi-Goal Manipulation Tasks,” in *IEEE International Conference on Robotics and Automation*, 2018, pp. 3152–3159.

[129] M. Kang, H. Shin, D. Kim, and S.-E. Yoon, “TORM: Fast and Accurate Trajectory Optimization of Redundant Manipulator given an End-Effector Path,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 9417–9424.

[130] M. Kang and S.-E. Yoon, “Analysis and acceleration of TORM: optimization-based planning for path-wise inverse kinematics,” *Autonomous Robots*, vol. 46, no. 5, pp. 599–615, 2022.

[131] R. Holladay, O. Salzman, and S. Srinivasa, “Minimizing Task-Space Fréchet Error via Efficient Incremental Graph Search,” *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1999–2006, 2019.

[132] M. Faroni, M. Beschi, N. Pedrocchi, and A. Visioli, “Predictive Inverse Kinematics for Redundant Manipulators With Task Scaling and Kinematic Constraints,” *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 278–285, 2019.

[133] D. Rakita, B. Mutlu, and M. Gleicher, “STAMPEDE: A Discrete-Optimization Method for Solving Pathwise-Inverse Kinematics,” in *IEEE International Conference on Robotics and Automation*, 2019, pp. 3507–3513.

[134] P. Praveena, D. Rakita, B. Mutlu, and M. Gleicher, “User-Guided Offline Synthesis of Robot Arm Motion from 6-DoF Paths,” in *IEEE International Conference on Robotics and Automation*, 2019, pp. 8825–8831.

[135] KUKA Deutschland GmbH, *KR CYBERTECH nano Spezifikation*, Spez KR CYBERTECH nano V2 ed., 2018.

[136] B. Mirtich, “V-Clip: Fast and Robust Polyhedral Collision Detection,” *ACM Transactions on Graphics*, vol. 17, no. 3, pp. 177–208, 1998.

[137] H. Anton and C. Rorres, *Elementary Linear Algebra*, 11th ed. Hoboken: Wiley, 2019.

[138] M. Meiringer, A. Kugi, and W. Kemmetmüller, "Time-optimal fold out of large-scale manipulators with obstacle avoidance," *IFAC-PapersOnLine*, vol. 52, no. 16, pp. 114–119, 2019.

[139] R. A. Horn and C. R. Johnson, *Matrix Analysis*, 2nd ed. Cambridge: Cambridge University Press, 2013.

[140] J. Nocedal and S. Wright, *Numerical Optimization*, ser. Springer Series in Operations Research. New York: Springer, 1999.

[141] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming," *Mathematical Programming*, vol. 89, no. 1, pp. 149–185, 2000.

[142] R. Waltz, J. Morales, J. Nocedal, and D. Orban, "An interior algorithm for nonlinear optimization that combines line search and trust region steps," *Mathematical Programming*, vol. 107, no. 3, pp. 391–408, 2006.

[143] F. N. Fritsch and R. E. Carlson, "Monotone Piecewise Cubic Interpolation," *SIAM Journal on Numerical Analysis*, vol. 17, no. 2, pp. 238–246, 1980.

[144] KUKA Robot GmbH, *LBR iiwa 7 R800, LBR iiwa 14 R820 Betriebsanleitung*, BA LBR iiwa V5 ed., 2015.

[145] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[146] M. Montanari and N. Petrinic, "OpenGJK for C, C# and Matlab: Reliable solutions to distance queries between convex bodies in three-dimensional space," *SoftwareX*, vol. 7, pp. 352–355, 2018.

[147] S. Russell and P. Norvig, *Artificial Intelligence A Modern Approach*, 3rd ed. Upper Saddle River: Pearson, 2016.

[148] J. Arents and M. Greitans, "Smart Industrial Robot Control Trends, Challenges and Opportunities within Manufacturing," *Applied Sciences*, vol. 12, no. 2: 937, pp. 1–20, 2022.

[149] T. Osa, "Motion planning by learning the solution manifold in trajectory optimization," *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 281–311, 2022.

[150] M. Vu, A. Lobe, F. Beck, T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, "Fast trajectory planning and control of a lab-scale 3D gantry crane for a moving target in an environment with obstacles," *Control Engineering Practice*, vol. 126, no. 105255, pp. 1–13, 2022.

[151] V. Vonasek, A. Vick, and M. Saska, "Motion planning with Motion Primitives for Industrial Bin Picking," in *IEEE International Conference on Emerging Technologies and Factory Automation*, 2017, pp. 1–4.

[152] S. Nayak and M. W. Otte, "Bidirectional Sampling-Based Motion Planning Without Two-Point Boundary Value Solution," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3636–3654, 2022.

[153] S. Ruan, K. L. Poblete, H. Wu, Q. Ma, and G. S. Chirikjian, "Efficient Path Planning in Narrow Passages for Robots With Ellipsoidal Components," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 110–127, 2023.

[154] X. Zhang, R. Belfer, P. G. Kry, and E. Vouga, "C-space tunnel discovery for puzzle path planning," *ACM Transactions on Graphics*, vol. 39, no. 4, pp. 104:1–104:14, 2020.

[155] A. Orthey and M. Toussaint, "Section Patterns: Efficiently Solving Narrow Passage Problems in Multilevel Motion Planning," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1891–1905, 2021.