Fakultät für Maschinenwesen
Lehrstuhl für Angewandte Mechanik

Trajectory Planning for Redundant Manipulators

Optimization, Real Time Methods, Agricultural Robots

Christoph Josef Schütz

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender: Prof. dr. ir. Daniel J. Rixen

Prüfer der Dissertation:

1. Prof. Dr.-Ing. Dr.-Ing. habil. Heinz Ulbrich, i.R.
2. Prof. Prof. h.c. Dr. Dr. h.c. Ulrich Walter

Die Dissertation wurde am 08.12.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Maschinenwesen am 18.05.2017 angenommen.

Abstract

By contrast to traditional industrial production, the new generation of flexible robots has to cope with changing tasks in dynamic environments. Trends towards flexible production in industry and service robotics make new demands on the design and planning capabilities of robots. Providing a maximum of flexibility, modern robots are equipped with more degrees of freedom than are necessary for most of their usual tasks. Resolving this redundancy in an optimal way, while preserving the real time planning capabilities, is one of the key challenges in modern motion planning and constitutes the main topic of this thesis. The author presents novel trajectory planning approaches for redundant manipulators in the context of an agricultural robot system for harvesting sweet peppers in a greenhouse. Several approaches to incorporate tactile feedback in real time are presented, using a distinct nullspace and/or task space projection. When the robot avoids an obstacle, this approach enables the motion of the end effector to remain defined. Furthermore, the inverse kinematics algorithm (based on Automatic Supervisory Control) is extended by predictive optimization of nullspace movements. Applying the conjugate gradient method in a moving horizon scheme, this approach can be applied in real time. Additionally, an offline trajectory optimization method is evaluated for realistic harvesting scenarios in a greenhouse w.r.t. computation time, success rates and minimization of the cost function. The thesis is completed by introducing the required methods and tools. Amongst other topics, the dynamic model and a proprioceptive joint torque estimation method is shown and a concept for teleoperation by a haptic joystick is developed. This enables the user to control the manipulator's end effector in real time while receiving haptic feedback from the arm in situations in which an autonomous operation fails.

Zusammenfassung

Im Gegensatz zur klassischen industriellen Produktion ist die neue Generation von vielseitig einsetzbaren Robotersystemen mit wechselnden Aufgaben in dynamisch veränderlichen Umgebungen konfrontiert. Insbesondere Trends hin zu flexibler Produktion und Servicerobotik stellen neue Anforderungen an die Planung und Steuerung von Robotern. Besondere Beweglichkeit und Flexibilität werden durch redundante Manipulatoren erreicht, welche mehr Freiheitsgrade besitzen, als für eine bestimmte Aufgabe notwendig sind. Eine der Schlüsselherausforderungen moderner Bewegungsplanung sowie Hauptthematik dieser Arbeit ist, diese Redundanz in einer optimalen Art und Weise in Echtzeit aufzulösen. Dabei werden neue Ansätze zur Trajektorienplanung redundanter Manipulatoren im Umfeld der Agrarrobotik vorgestellt, eingebettet in das Umfeld der Agrarrobotik mit dem Beispiel der autonomen Ernte von Paprikas in Gewächshäusern. Verschiedene Ansätze zur Berücksichtigung taktiler Information bei der Planung in Echtzeit werden entwickelt, wobei die dedizierte Projektion der Reaktion in den Null- bzw. Arbeitsraum im Vordergrund steht. Weiterhin wird die auf dem Konzept der Automatic Supervisory Control basierten Lösung der inversen Kinematik um eine vorausschauende Optimierung von Nullraumbewegungen erweitert. Dadurch bleibt die Bewegung des Endeffektors bei dem Ausweichen vor einem Hindernis definiert. Der Einsatz des konjugierten Gradientenverfahrens in einem mitbewegten Vorausschaufenster erlaubt dabei eine Anwendung in Echtzeit. Zusätzlich wird eine Methode zur Offlinetrajektorienoptimierung anhand eines realistischen Ernteszenarios im Gewächshaus hinsichtlich Rechenzeit, Erfolgsrate und Minimierung der Kostenfunktion bewertet. Die Vorstellung notwendiger Methoden und Werkzeuge rundet die Arbeit ab. Unter anderem wird das Dynamikmodell des Manipulators ebenso entwickelt wie eine Methode zur Schätzung von Gelenkmomenten ohne zusätzliche Sensorik. Für Situationen, in welchen der autonome Ablauf fehlschlägt, wurde ein Konzept zur Teleoperation durch einen haptischen Joystick entwickelt. Dieser erlaubt dem Nutzer, den End-Effektor des Roboters in Echtzeit zu steuern und dabei haptisches Feedback vom Arm zu erhalten.

Danksagung

Die vorliegende Dissertation entstand im Rahmen meiner Tätigkeit am Lehrstuhl für Angewandte Mechanik der Technischen Universität München zwischen April 2012 und Mai 2016. Ohne die Hilfe zahlreicher Leute, die mich in dieser Zeit unterstützt haben, wäre die vorliegende Arbeit nicht möglich gewesen.

Mein besonderer Dank gilt meinem Doktorvater Prof. Heinz Ulbrich. Er gab mir die Möglichkeit, in einem spannenden wie vielseitigen Forschungsprojekt zu promovieren. Seine stetige Förderung und Unterstützung, sein Vertrauen sowie die Freiräume zur Entwicklung und Umsetzung eigener Ansätze und Ideen schätze ich sehr. Mein ebenso großer Dank gilt seinem Nachfolger Professor Daniel Rixen, der mir die Weiterführung meiner Forschungsarbeit ermöglichte. Besonders dankbar bin ich ihm für die hervorragende kollegiale Arbeitsatmosphäre, die er förderte sowie die Teilnahme unseres Teams an der European Robotics Challenge 2014, die er von Beginn an unterstützte. Durch seinen unermüdlichen Einsatz für die Administration und Infrastruktur des Lehrstuhls hat ebenfalls Dr. Thomas Thümmel einen maßgeblichen Anteil an den hervorragenden Arbeitsbedingungen am Lehrstuhl, die ich erleben durfte. Viel gelernt habe ich auch bei der Zusammenarbeit mit Georg Mayr, ohne seine Hilfe wären die meisten mechatronischen Projekte im Labor nicht realisierbar gewesen. Ebenso gilt mein Dank der Werkstatt und dem Sekretariat, die mich in vielen Anliegen unterstützten.

In meiner Zeit am Lehrstuhl habe ich mich sehr wohl gefühlt, was nicht zuletzt an der freundschaftlichen, aufgeschlossenen, motivierten und stets hilfsbereiten Atmosphäre lag. Jedem einzelnen Kollegen gebührt hier höchster Dank. Hervorheben möchte ich hier meine Bürokollegen Jörg Baur und Julian Pfaff sowie später Felix Sygulla und Daniel Wahrmann. Ohne ihre Hilfe wären viele Projekte nicht denkbar gewesen, ich durfte vieles von ihnen lernen und bin sehr dankbar für die schöne gemeinsame Zeit. Ein wichtiger Bestandteil meiner Promotion war das gemeinsame Arbeiten mit Studenten. Inbesondere die Arbeiten von Jan Berger, Peter Fagerer, Caner Hazirbas, Manuel Kiener, Florian Lochner, Carina Miller, Thomas Ponn, Felix Reuß, Sebastian Roder, Raphael Schaller, Timothy Smith, Felix Sygulla und Sahand Yousefpour haben maßgeblich zum wissenschaftlichen Erfolg der Arbeit beigetragen. Ihre Betreuung war für mich stets eine Bereicherung. Immer wieder entstanden in spannenden und fachlich fordernden Diskussionen neue Anregungen und Ideen. Die hohe Motivation, Leistungsbereitschaft und die hervorragenden Arbeitsergebnisse weiß ich sehr zu schätzen.

Ebenso gilt mein Dank der Europäischen Union für die finanzielle Unterstützung des Forschungsprojektes CROPS. Der gegenseitige Austausch und die Zusammenarbeit mit den internationalen Projektpartnern waren für mich sehr wertvoll.

Weiterhin bedanke ich mich bei Jörg Baur, Thomas Buschmann, Volker Petersen, Anja Sievers, Felix Sygulla und Daniel Wahrmann für die wertvollen Hinweise

bei der Korrektur meiner Arbeit. Zuletzt danke ich herzlich meinen Eltern und Anja, dass sie mich auf meinem Lebensweg begleiten, unterstützen und mir stets den notwendigen Rückhalt geben.

München, 28. August 2017

Table of Contents

Abbreviations

ASC	automatic supervisory control
BLDC	brushless direct current
CAN	controller area network
CG	conjugate gradient
COG	center-of-gravity
CROPS	Clever Robots for Crops
DDP	differential dynamic programming
DOF	degree of freedom
IK	inverse kinematics
MPC	model predictive control
OCP	optimal control problem
ODE	ordinary differential equation
PMO	plant maintainance operation
PMP	Pontryagin's minimum principle
QP	quadratic programming
RAC	resolved acceleration control
RMRC	resolved motion rate control
RMSE	root-mean square error
ROS	Robot Operating System
RRT	rapidly-exploring random trees
SD	steepest descent
SQP	sequential quadratic programming
SSV	swept-sphere volume
TCP	tool center point
TOF	time-of-flight

TPBVP two-point boundary value problem

TUM Technische Universität München

Chapter 1

Introduction

Beginning in the late sixties, large parts of industrial production were affected by the success of industrial manipulators and automation technology. Production steps such as welding or assembly operations were converted from manual manufacturing to fully automated processes. Some modern production lines have not changed their fundamental characteristics for many years. Manipulators execute predominantly repetitive operations in a well known, carefully arranged and controllable environment. Workpieces are arranged in precise fixtures and no unforeseen obstacles enter the workspace of the robot. Thus, the robot is often equipped with minimal or even no sensing capabilities at all, its paths and trajectories are pre-planned and usually do not change during operation. The kinematic and hardware design of the robot is chosen according to its specific task, which may not change much during the manipulator's lifetime. This classical concept of the use of robots is quite efficient in static production scenarios. However, more advanced robot technologies are gradually being introduced in industrial production. For example, vision systems are used to locate workpieces precisely or robots work side-by-side with human workers, e.g. in a car door assembly line (Knight 2014). Driven by the recent advances in electronics and computational power as well as the dropping prices for robots' hardware, more and more new fields of application are considered. Applications range from robots in household, nursing and health care to agricultural tasks. These are often summarized under the topic of service robotics, a subject that has become very popular in modern society. The IFR (2015b) defines a service robot as "a robot that performs useful tasks for humans or equipment excluding industrial automation application". Compared to classical industrial production, demands on service robot's abilities are higher: Based on its perception, the robot has to cope with obstacles in its workspace. Paths and trajectories have to be replanned for every new action since tasks and the environment vary from one scenario to another. The kinematic design is no longer adapted to *one* specific operation, but flexibility is essential. In order to meet these enhanced requirements, robots are designed in a more flexible and multipurpose way. Aimed at providing sufficient flexibility for a wide range of applications and tasks, manipulators can be equipped with more

degrees of freedom. At the same time, tasks are given only as precisely as necessary in order to prevent a too narrow limitation of the robot's path and trajectory. The gap between a minimal task definition and a high kinematic flexibility of a robot is often referred as redundancy. Redundant robots have more degrees of freedom (DOFs) than it would need to fulfill a task. This redundancy can be used for pursuing additional objectives: Contact forces between the environment and the robot arm as well as joint velocities can be reduced or kinematic limits and self-collisions can be avoided better. The key issue now becomes, how the manipulator should move its joints in an optimal way. This thesis aims at providing tools and methods in order to solve this problem.

1.1 Contributions

The main contribution of this thesis is to develop trajectory generation algorithms for redundant manipulators, to extend available methods to new applications and to provide novel approaches for a more intelligent planning. In more detail, the contributions can be summarized as follows:

- **Application to an Agricultural Manipulator** The methods and approaches developed in this thesis are formulated in a general manner for arbitrary redundant manipulators. However, in order to show their potential in real world experiments, they are applied to a nine DOF agricultural manipulator. This manipulator was developed at the author's institute within the European research project CROPS[1]. The agricultural applications of this project and their requirements and challenges are the major motivation for the methods and algorithms of this thesis.
- **Modeling and Interfaces** For planning the motion of a manipulator, knowledge about kinematic and dynamic quantities is required. Thus, this thesis first introduces the kinematic and dynamic models. Parameters are identified for the considered agricultural manipulator. Additionally, a haptic interface is shown that enables the user to control the manipulator while receiving force feedback.
- **Planning based on Tactile Feedback** In many scenarios, information about the environment based only on visual data is not sufficient. Thus, two concepts are presented in order to estimate external forces acting on the manipulator: The first one uses an additional tactile sensor, while the second one requires only information from proprioceptive sensors. This thesis develops a novel approach for considering tactile information within the motion planning algorithm of redundant robots. The redundant DOFs are used to reduce contact forces between the manipulator arm and its environment,

[1] European Research Project Clever Robots for Crops (CROPS), www.crops-robots.eu. Further information can be found in section 2.1.3.

while using a clear distinction between the influences to the nullspace and the task space.

- **Predictive Inverse Kinematics** Instantaneous inverse kinematics (IK) calculation schemes such as automatic supervisory control (ASC) only consider information about one next timestep. This thesis develops an approach to preemptively optimize a manipulator's motion, i.e. to take into account several future timesteps. This enables a manipulator to adapt its motion in the presence of pending collisions or other constraints. Hence, high joint velocities can be avoided that would otherwise occur as instantaneous reactions. Since this approach shall be applied in real time, the computational effort of the numerical optimization has to be minimized. Thus, the thesis examines different implementations of the conjugate gradient (CG) method which enables a real time application of the proposed method.
- **Trajectory Optimization** Besides the predictive optimization of the inverse kinematics in real time, another method is presented to optimize a manipulator's trajectory for a given task in advance, i.e. offline. Based on real harvesting scenarios in a greenhouse for sweet peppers, the computational time and anticipated benefits are quantified for an agricultural manipulator.

1.2 Outline

This thesis is divided into five main parts: In chapter 2 an overview is given of related contributions in the field of agricultural automation and motion planning for redundant robot systems. The agricultural manipulator, including its hardware and interfaces, kinematic and dynamic models and concepts for tactile perception are presented in chapter 3. A solution of the inverse kinematics problem including secondary objective functions is shown in chapter 4. Furthermore, this chapter includes innovative approaches to integrate tactile sensor feedback into the motion planning. A numerically efficient method for extending the inverse kinematics calculation by a predictive approach is proposed in chapter 5. The main focus in this part is the development of a suitable system formulation and the numerical solution of the optimization problem. Furthermore, this section shows an approach to apply this method in real time. The potential of offline trajectory optimization is evaluated using a sweet pepper harvesting scenario in chapter 6. In chapter 7 the thesis concludes with a summary and discussion of the proposed methods. Additionally, it gives an outlook on future research perspectives regarding the context of this thesis.

Chapter 2

Literature Survey

By analogy to the highly automated industrial manufacturing, robot systems are considered to increase productivity in agriculture. The European research project CROPS aimed at developing new systems and tools to encounter this challenge. Among a wide variety of possible applications, the CROPS project considered selective harvesting of sweet peppers in greenhouses or apples in orchards. In this context, an agricultural manipulator system was designed at the Institute of Applied Mechanics, Technische Universität München (TUM). Since the manipulator was designed for multipurpose usage for different fruit, a highly flexible kinematic design was required. Flexible manipulators with many DOFs which operate e.g. in agricultural, natural environments have particular requirements regarding their planning algorithms. The development of suitable planning algorithms and methods is the main objective of this thesis.

This section summarizes the developments of two topics: First, an overview is given of automation in agriculture, focusing on autonomous robot systems. The research project CROPS with its main objectives and achievements is presented in detail. Second, related motion planning approaches are shown. In correlation to the concepts developed in this thesis, main aspects are trajectory optimization and the resolution of redundancy.

2.1 Automation in Agriculture

Today, robot systems play a major role in industrial manufacturing. However, regarding agricultural tasks, robots are still barely used. Notwithstanding, economical and political aspects such as minimum wage or shortage of acreage promote automation processes in agriculture. The worldwide operational stock of industrial robots in the field of agriculture counted only 1136 units in 2014, corresponding to a share of 0.1% of all operational robots worldwide (IFR 2015a). Facing the large amount of promising applications, from livestock farming over broad acre

Figure 2.1: Munckhof Pluk-O-Trak harvesting aid pulled by a tractor through the orchard. Human pickers place the fruit on the conveyor belt system on the left, which automatically transports them into a storage container. Image taken from Munckhof (2016).

to field crop farming and weeding, this amount seems quite low. Especially in the field of fruit care and harvesting there is as yet no significant automation. The demanding technical requirements have impeded the usage of robot systems for agricultural tasks in the past. While automation in industry is characterized by repetitive tasks in well-structured, controlled and segregated areas, the situation in greenhouses and orchards is far more challenging: Environmental conditions in outdoor cultivation vary from bright sunlight to cloudy days and rain. The climate in greenhouses is mostly hot and humid during the day while temperatures drop significantly at night which leads to condensation on the plants and the machinery. Furthermore, the foliage is dense and cluttered, obstacles and fruit may be hidden from sight. Fruit is non-regularly distributed on the plants. Hence, every single fruit needs to be localized individually and the motion of the harvesting robot has to be adapted accordingly. Many researchers and companies developed prototypes and concepts for the automation of selective harvesting by robot systems in the past decades. Some of them are presented in the following.

One of the earliest documented articles was published in the late 60s by Schertz and Brown (1968) considering the mechanization of citrus fruit harvesting. Besides bulk harvesting methods, they proposed several approaches for selective fruit harvesting. Nevertheless, to this day "there still remains no viable, cost-effective approach towards robotic mechanization", almost 50 years later (IFR 2015b, p. 39). Prototype systems have barely been commercialized because of some unresolved drawbacks: amongst others, slow operation speeds, high costs, low reliability or the need for adaption of the cultivation system were reported (Bac et al. 2014; IFR 2015b). By contrast, systems for partial automation are rather successful: the Munckhof Pluk-o-Trak, shown in fig. 2.1, supports human pickers with a conveyor belt system collecting the picked fruit and transporting them carefully to a large container (Van den Munckhof and Van den Munckhof 1994). Almost 1800 systems were in operation in Europe in 2002 (Ward 2003).

In the past, a large variety of robot systems were developed in the field of selective harvesting and plant maintainance operations (PMOs). In the following sections, some systems were selected which represent the current state-of-the-art of robot systems in agriculture. A comprehensive overview of the developments within the past decades is beyond the scope of this thesis. For this purpose, the reader is referred to further reading: A textbook covering most relevant topics and systems for agricultural or bioproduction robots was published by Kondo and Ting (1998). Other comprehensive overviews can be found in Bac et al. (2014), Henten (2004), Sarig (1993), and Tillett (1993).

Applications considered in this thesis comprise the harvesting and spraying of medium and large size crops such as sweet peppers, apples, grapes, etc.. The overview is categorized by kinematics and flexibility of the respective robots: First, bulk harvesting systems are presented which rely mostly on tree-shaking mechanisms and bulk fruit collection devices. Second, customized robot systems with a low number of DOFs or manipulators offering a higher flexibility for harvesting high quality crops are shown. Lastly, the EU-project CROPS is introduced, providing the context and background of this thesis.

2.1.1 Bulk Harvesting

Bulk harvesting systems for grapes, citrus fruit, olives or cherries have been available on the market for years. Although they provide a high harvesting rate, they are mostly not suited for high quality crops: the fruit's surface is likely to get damaged by vibrating and shaking mechanisms. Furthermore, no selection is made regarding the fruit's maturity state (Sarig 1993; Tillett 1993). The largest field of application is grape harvesting: Several companies (ERO Gerätebau GmbH 2016; New Holland Agriculture 2016; Oxbo International Corporation 2014; Pellenc SA 2015) offer self-propelled or towed harvesting machines moving continuously along the rows for juice and wine production. Oxbo adapts its shaking devices for a large variety of crops: Harvesting machines for citrus fruit, berries, coffee or olives are available. For cherry trees, Munckhof developed a single-tree shaker and collecting device. A more sophisticated shaking system for apple trees was presented by Peterson et al. (1999): a customized shaking end effector is moved to a bunch of fruit. Therefore, they adapted also the cultivation system. The branches were inclined in order to prevent drop damage.

2.1.2 Selective Harvesting

By contrast to bulk harvesting systems, selective harvesting systems localize and detach each fruit separately. Thus, a selection of harvested fruit regarding their maturity state, size or color can be made. Additionally, the detachment as well as the further transportation of the fruit can be done carefully. In the following,

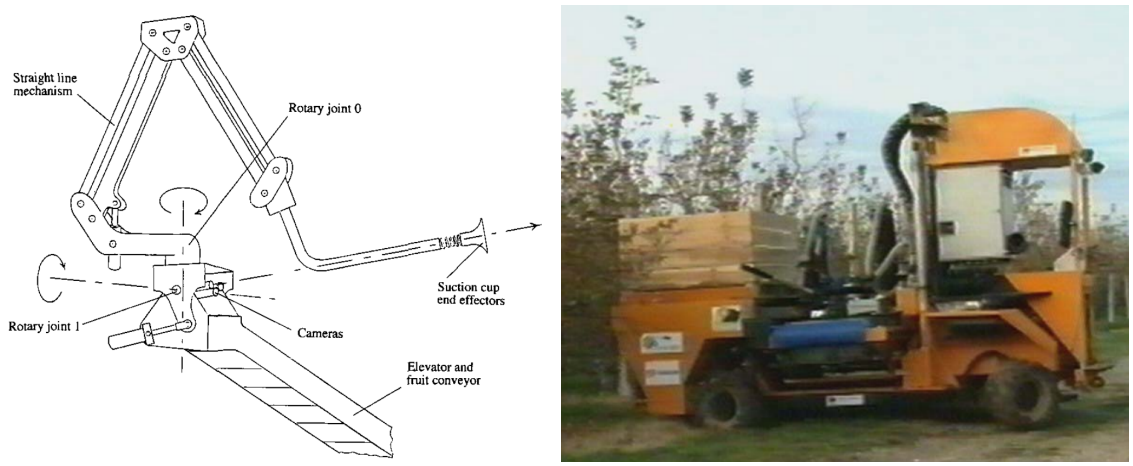
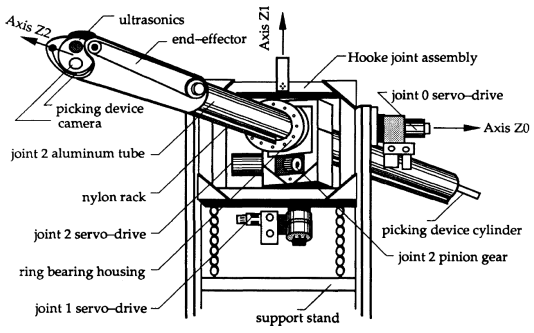


Figure 2.2: MAGALI apple harvesting robot prototype, 1994. Two of the manipulator arms (*left*) are mounted on the backside of a tractor (*right*). Image taken from Ward (2003).

systems with customized kinematics and low DOFs are presented first, followed by systems using articulated arms.

Customized Low-DOF Robot Systems

Beginning in 1986, a French research consortium started the development of an autonomous apple picking robot, the MAGALI robot (Grand d'Esnon et al. 1987). Aiming for the harvesting rate of a two-handed human picker (3-4 s per fruit), they designed an elbow mounted on a spherical manipulator with a 2D camera at its origin. The pantographic mechanism of the elbow assures a straight-line movement of the vacuum cup end effector along the line of sight of the camera. Two of these manipulators are mounted on a mobile platform. One of the robot prototypes is shown in fig. 2.2. The system was developed to a status close to commercialization. However, it was discontinued in 1996/1997 for economical reasons (Sarig 1993; Ward 2003). A spherical manipulator extended by a prismatic joint was introduced by Harrell et al. (1990, 1989) for the harvesting of citrus fruit. The system, named FLORIDA CITRUS PICKING ROBOT and depicted in fig. 2.3a, uses a camera mounted in the end effector and avoids self-occlusion by the arm, as it occurred at MAGALI. Muscato et al. (2005) applied a custom designed robot system with two 3-DOF telescopic manipulators mounted on an inclined prismatic joint for harvesting oranges. A well adapted system for picking kiwifruit was presented by Scarfe et al. (2009). Each of the four manipulator arms with 3-DOF positions a tube with a cutting end effector for collecting vertically hanging fruit (cf. fig. 2.3b). Cycle times of only 1 s per fruit were achieved. The arms are mounted on a mobile cart which navigates autonomously under the kiwifruit canopy. A 4-DOF manipulator is presented by Irie et al. (2009) for harvesting asparagus. Two revolute and one prismatic joint are used to position a telescopic arm with a cutting and gripping end effector.



(a) Florida Citrus Picking Robot.

(b) Kiwifruit picking robot.

Figure 2.3: Simple robot systems for selective harvesting of fruit. The manipulator *(a)* is designed for picking citrus fruit while the mobile cart *(b)* with up to 4 robot arms is able to pick kiwifruit autonomously. Images taken from Harrell et al. (1990) (a) and Scarfe et al. (2009) (b).

By designing customized robot systems for different crops, the aforementioned groups achieved promising results regarding the automation of selective harvesting. With the low number of DOFs, complexity was reduced and fast cycle times were reported. However, none these systems appears to have been commercialized. This may be explained by the more demanding requirements for a market-ready product regarding reliability and harvesting success rate. Both key factors have barely been reported by the research groups (Bac et al. 2014). Large efforts and investments may be necessary for a further development of these prototype systems for a cost efficient implementation in orchards or greenhouses. However, facing the increasing costs for human labor and the progress regarding intelligent algorithms in robotics, these systems may again become important. Recently, the Spanish company Agrobot presented a strawberry harvesting system based on the patent by Trinidad (2011). A mobile gantry-like tractor moves along the raised bed-rows while up to 60 2-DOF manipulators pick the strawberries and release them on a conveyor belt (IFR 2015b). Although this system is reported to operate "impressively", problems regarding the fruit detection remain. Furthermore, the grower is forced to adapt his cultivation method: plant beds have to be raised and the plants have to be distributed more sparsely, decreasing the yield per acre (Fruit Growers News 2012).

Articulated Arm Systems

Instead of using low-DOF customized positioning systems as presented in the previous section, articulated manipulators as known from industrial production can also be applied for harvesting. A first system for individual tomato harvesting was presented by Kawamura et al. (1985). It consisted of a 5-DOF manipulator arm with a harvesting end effector, mounted on a battery powered cart. Kondo et al. (1994) developed a redundant 7-DOF manipulator system with a 5-DOF

articulated arm mounted on two prismatic joints to harvest tomatoes. By using a different end effector, the same system was able to harvest cherry tomatoes as well (Kondo et al. 1996). Van Henten et al. (2002) reported a cucumber collecting system for greenhouses. A standard 6-DOF industrial manipulator was mounted on a linear guide and equipped with a customized end effector for vertically hanging cucumbers. Baeten et al. (2008) used an industrial manipulator, mounted on a vertical linear axis on the back of a tractor, for apple picking. The end effector was designed as a suction cup with a camera at its center. Compared to Grand d'Esnon et al. (1987), the system achieved a slower cycle time of 9 s.

Discussion and the Multipurpose Idea

Although the reliability of the manipulator system is increased by using off-the-shelf industrial manipulators instead of customized prototypes, the same drawbacks remain regarding the fruit detection and planning system as described in the previous section for customized robot systems. Additionally, the manipulator itself becomes more expensive with an increased number of DOFs. Cost efficiency can be achieved by high utilization rates. One approach is to design a robot system for multipurpose usage. Parts of the system such as the manipulator can be used for other agricultural tasks or crops as well due to its flexible kinematics. In grapevine production for example, the harvesting period only lasts a few weeks per year. An efficient approach in order to increase the utilization is to use the robot system not only for harvesting tasks but also for other PMOs. Based on the manipulator system developed by Kondo (1995), Monta et al. (1995) showed its multipurpose application for grapevine production. The polar coordinate 5-DOF manipulator was equipped with different end effectors for berry-thinning, bagging and spraying. This approach, i.e. the development of a multipurpose agricultural robot system for different PMOs *and* crops, was the main motivation for the European research project CROPS. By following a modular concept, the same manipulator has been used for the selective harvesting of sweet peppers, apples, grapes and the precision spraying of grapes. Using a kit of customized end effectors, sensors and platforms, the manipulator was able to operate in greenhouses and orchards. The project, including its motivation, the components and some of the results are summarized in the following section.

2.1.3 The CROPS Project

In 2010 the collaborative four-year research project CROPS was launched. Funded by the European Commission[1] with 7.64 million EUR it involved fourteen partner institutes and companies from Europe, Israel and Chile. Its main objective

[1] Project Title: "Intelligent sensing and manipulation for sustainable production and harvesting of high value crops, clever robots for crops", Duration: Oct. 2010 – Sep. 2014, Grant Agreement №246252, Call ID FP7-NMP-2009-LARGE-3.

was the development of a modular, multipurpose autonomous robot system for selective harvesting and spraying of fruit. It comprised research on suitable end effectors, sensing and control algorithms and the manipulator system itself. Research was also done into forestry operations and canopy spraying. An overview of the project and its main results was given by Bontsema et al. (2014) and Baur (2015).

This project provides the background and motivation of the methods and concepts developed in this thesis. In the following, the multipurpose robot system and its applications are shown.

Multipurpose System

The main hardware components of the multipurpose robot system are the manipulator, end effectors, vision-based sensing systems and carrier platforms. While the manipulator is designed in modules to fit all applications, other hardware components remain application specific. Within the project, experiments using the robot system are conducted for the harvest of sweet peppers in greenhouses, picking apples and grapes in orchards and precision spraying of grapes. In the following sections the manipulator system and the applications - except for harvesting grapes[2] - are summarized. Fig. 2.4 shows the application specific configurations.

Manipulator System

Two generations of manipulator prototypes were developed at the Institute of Applied Mechanics, TUM. Both generations have up to 9-DOF and are designed modularly, enabling customized kinematics. They can be mounted on various platforms and be equipped with different types of end effectors. More details about the hardware design, software framework and controls are given in chapter 3, the doctoral theses of Baur (2015) and Pfaff (2015) and were presented at several international conferences (Baur et al. 2012; Schütz, Pfaff, Baur, et al. 2014; Schütz et al. 2014; Ulbrich et al. 2015).

Applications

Sweet Pepper Harvesting The final integrated robot system for sweet pepper harvesting in greenhouses is shown in fig. 2.5. The system consists of the manipulator prototype 2 mounted on a mobile carrier platform, an end effector and a

[2] Although developed and tested under lab conditions, an autonomous grape harvesting system has not been tested successfully in orchards. Main problems were a lack of time for integration, insufficient sensor information as well as the need for a more powerful cutting end effector (Wouters et al. 2014).

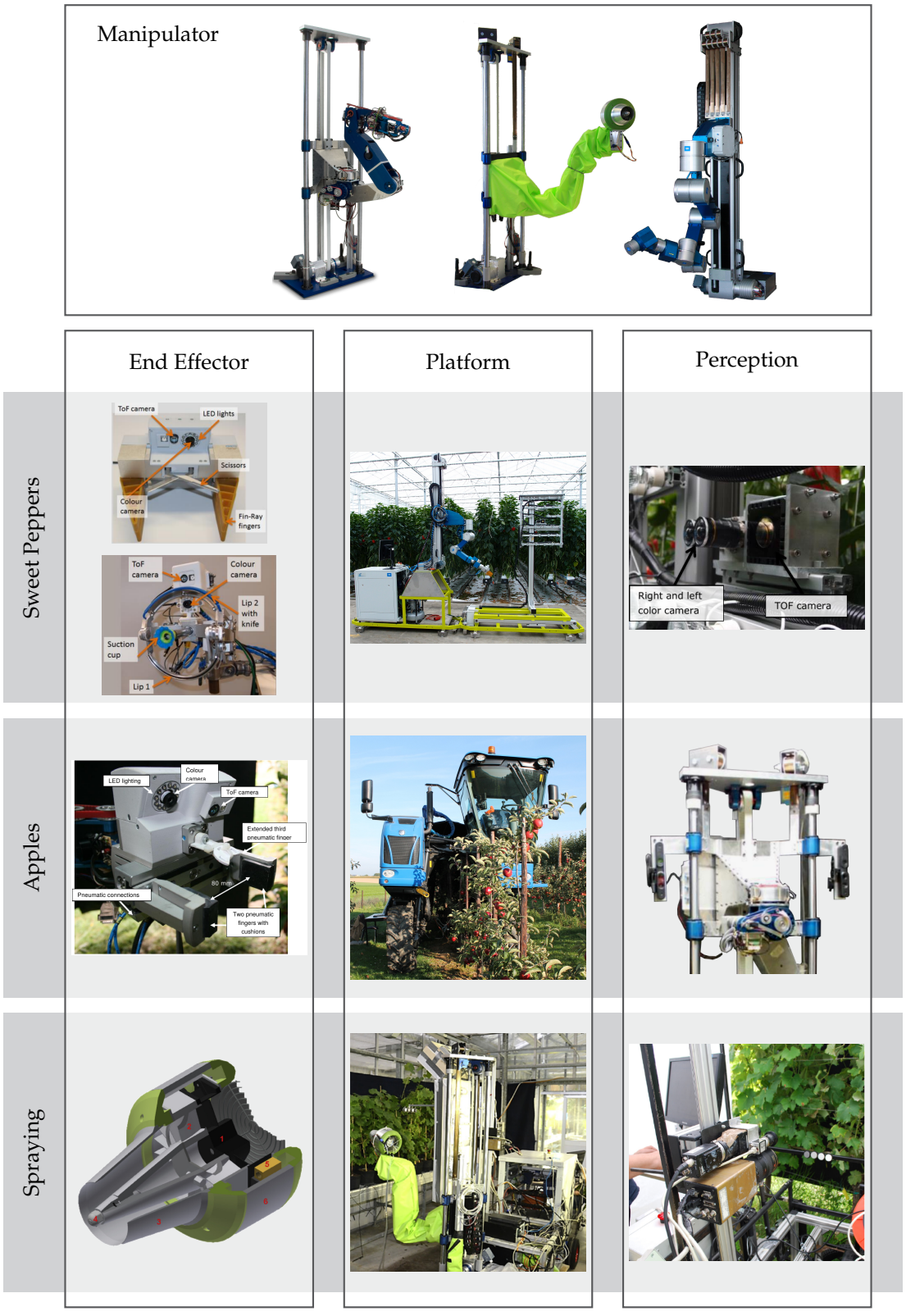


Figure 2.4: Modular robot system components (columns) for the respective applications (rows) developed within the CROPS project (Schütz 2014). Images taken from Hemming, Bac, and Edan (2014), Oberti et al. (2014), Oberti et al. (2016), and Wouters et al. (2014).

(a) Robot system in greenhouse. **(b)** Fin-Ray gripper.

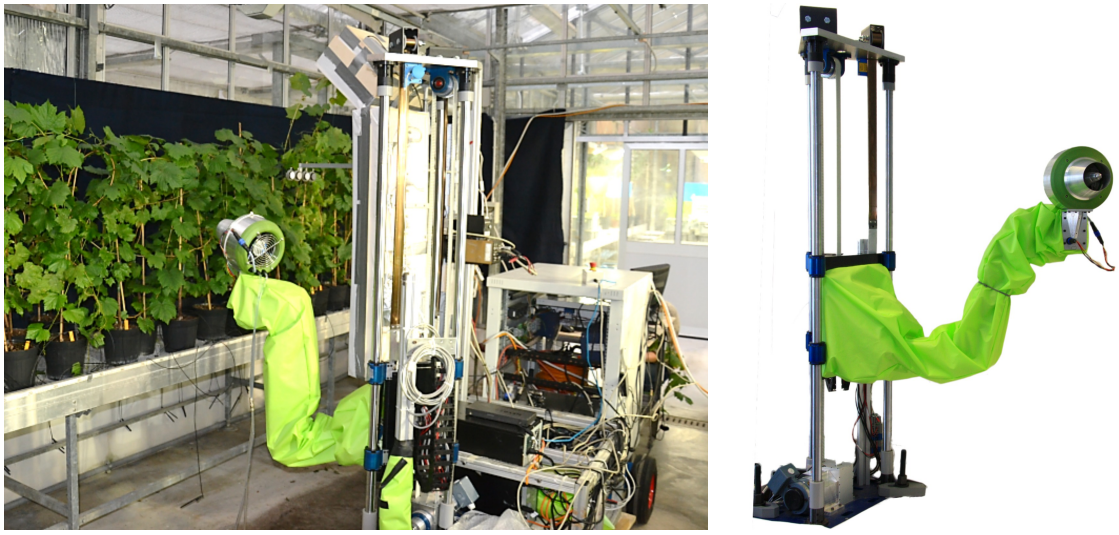
Figure 2.5: Final prototype of the CROPS sweet pepper harvesting robot system in a greenhouse at Wageningen, Netherlands (Hemming, Bac, Tuijl, et al. 2014).

sensor rig (fig. 2.5a). Integration and application specific adaption as well as experiments were done by Wageningen UR. The manipulator is used in its 9-DOF configuration, aiming for a maximum of flexibility. Two different harvesting end effectors were developed within the project: The *Fin-Ray* and *Lip-Type* gripper. The Fin-Ray gripper has four to six fingers taking advantage of the Fin-Ray principle (Gauchel and Saller 2012). Furthermore, scissors for cutting the peduncle and RGB and time-of-flight (TOF) cameras are integrated. Fig. 2.5b shows this end effector while approaching a sweet pepper in the greenhouse. The *Lip-Type* end effector was developed by Wageningen UR using a suction cup and two closing cutting rings (Hemming, Bac, Tuijl, et al. 2014). This approach allows a wide range of orientations for cutting the peduncle. The sensor rig next to the manipulator can be moved on a horizontal slide and includes an illumination system, two RGB cameras and one TOF camera. Experiments were conducted in a commercial greenhouse in the Netherlands. As known to the author, this was the first autonomous robot system to harvest sweet pepper in greenhouses. Details on cycle times and success rates were reported by Hemming, Bontsema, et al. (2014). A more detailed performance evaluation was given by Bac (2015).

Apple Harvesting The autonomous apple harvesting system was developed and integrated by MeBios Group, KU Leuven. The manipulator prototype 1 is mounted on the back of a gantry-like tractor (fig. 2.6c). For better adjustment of its distance to the canopy, a horizontal prismatic joint is added. The manipulator is equipped with a membrane jaws gripper (principle presented by Gauchel and Saller (2012)) with an upper third finger for better detachment of the stalk. Since fruit are detected using Kinect-like RGB-D sensors, the manipulator operates in

(a) Harvesting tractor in orchard. **(b)** Rear view. **(c)** Manipulator.

Figure 2.6: Apple harvesting robot system developed in the CROPS project. Application specific integration and development done by KU Leuven. Images taken from Wouters et al. (2014).



(a) Spraying system in greenhouse. **(b)** Manipulator Prototype 1 in 6-DOF configuration.

Figure 2.7: Autonomous precision spraying of grapes in a greenhouse at the University of Milano. Treating only diseased regions on the grapes, the amount of pesticides can be significantly reduced.

a tunnel under controlled illumination conditions (see fig. 2.6b). Task and motion planning methods are reported by Nguyen et al. (2013) while fruit detection algorithms can be found in Nguyen and Keresztes (2014). Final results of the application are summarized in Wouters et al. (2014).

Precision Spraying The autonomous precision spraying system aims at selectively spraying only the diseased areas of grapes. The system shown in fig. 2.7 was developed at University of Milano. It consists of a mobile platform, the manipulator prototype 1 in its 6-DOF configuration, the spraying end effector and a multispectral and RGB camera system. Experiments were carried out in a greenhouse at the University of Milano. Detailed results as well as the robot system can be found in Oberti and Hocevar (2014) and Oberti et al. (2016).

Summary and Future Directions

The main result of the CROPS project was the development of a multipurpose agricultural robot system for the selective harvesting of sweet peppers, apples, grapes and the precision spraying of grapes. From a hardware perspective, the modular manipulator concept with up to 9-DOF is the main component which is used for all applications, while end effectors, vision sensors and platforms were developed specific to the application. It could be shown by experiments in greenhouses and orchards under nearly real-world conditions that the system is capable to perform the respective tasks autonomously. However, no commercialization of the system has been made nor is this planned. The main drawbacks of the final prototype system were its comparatively low reliability, low operational speed and low success rate. Especially the multipurpose idea of the system increases the system's complexity. Thus, the integration of all hard- and software modules still has a large potential for increasing the overall system performance.

In order to facilitate the development towards a market-ready implementation, one first approach is to focus on only one application. The follow-up research project SWEEPER[3] follows this idea. Using the technology developed within the CROPS project, it concentrates on the automation of sweet pepper harvesting. Another lesson learned within the CROPS project is that for a highly efficient autonomous harvesting system, cultivation methods *and* the robot have to be developed hand in hand. Dense foliage and narrow corridors make the automation more difficult. New trends such as *Vertical Farming* approaches (Cho 2011) may encourage the automation in agriculture. Scenarios, where the plant itself is moved to a "harvesting station" and not vice versa, may be efficient solutions as well.

[3]SWEEPER - Sweet Pepper Harvesting Robot, Duration Feb. 2015 – Jan. 2018, EU H2020 program, Grant Agreement №644313. http://www.sweeper-robot.eu.

2.2 Motion Planning

Robot systems require a motion planning module, which translates user commands into suitable actuator movements. This principle is exemplified in fig. 2.8: A *user* wants the *robot* to move to a desired location B. He tells the system "Move to Location B." The robot's end effector is currently at the location A, thus, the *motion planning* module calculates a trajectory $\boldsymbol{q}(t)$ for the robot's joints to move the end effector from A to B. This trajectory is passed to the *motion control* module of the robot, which controls the motor currents in such a way that the robot moves its end effector to B.

While the motion control module enables the single joints to follow desired positions, velocities or forces/torques, the motion planning module can be considered as the interface between the user, giving the command, and the robot, performing the actual task. It is responsible for providing joint trajectories that fulfill the actual task while respecting actuator limitations and other constraints. For example, constraints may apply for kinematics (position/ velocity/ acceleration limits) and dynamics (force/torque limits) of joints or arbitrary parts of a robot as well as the handling of obstacles. Furthermore, a desired behavior may include the optimization of time, velocities, efforts or smoothness. In consequence, most motion planning approaches rely on optimization methods. In the last decades, various concepts have been developed for many robot systems and applications. Some of the most relevant approaches w.r.t. the scope of this thesis are presented in the following. For a thorough survey, refer to the textbooks of Nakamura (1991), Craig (2005), LaValle (2006), Choset et al. (2005) and Siciliano et al. (2009).

After introducing related terms and definitions, this section divides motion planning strategies into three categories. Each of them relies on one or several planning algorithms. Subsequently, trajectory optimization concepts for redundant manipulators related to the methods developed in this thesis are presented in more detail. Finally, methods to deal with cluttered environments are shown.

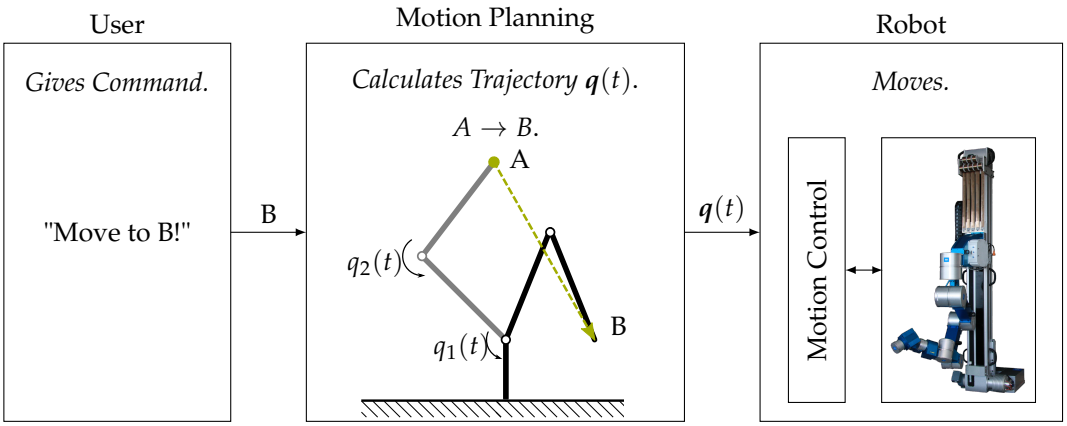


Figure 2.8: Motion planning as the interface between the user and the robot system.

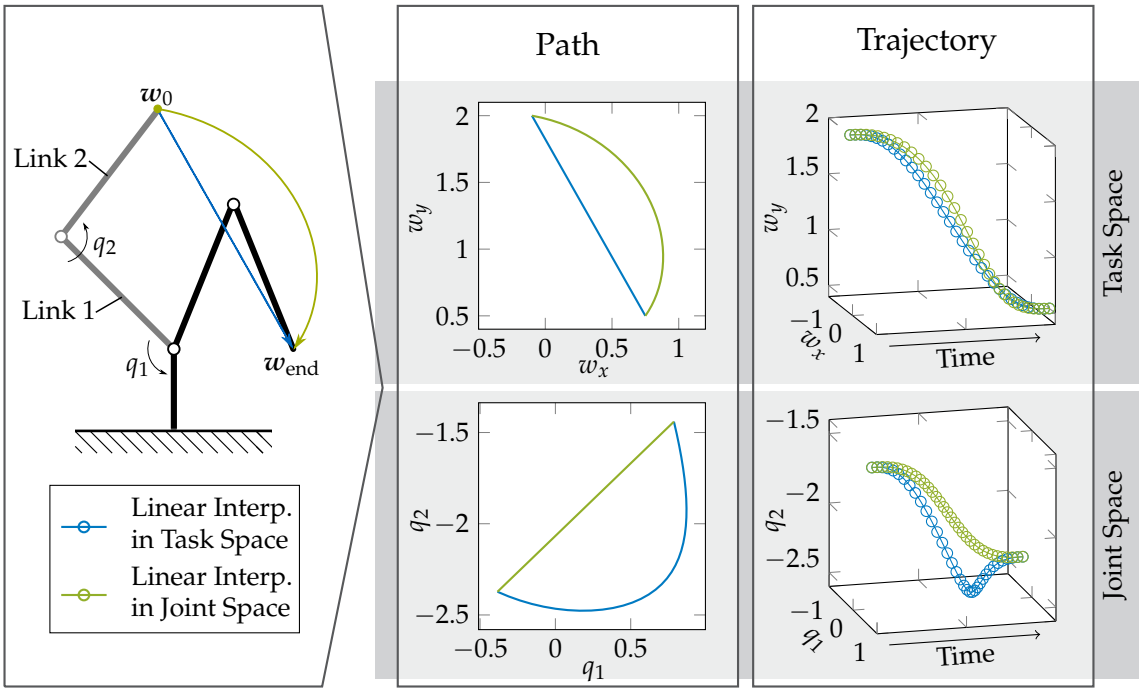


Figure 2.9: *Left:* Two-DOF manipulator moving from the start position w_0 to the goal location w_{end}. The task space is chosen as the tool center point (TCP) of the manipulator at the tip of link 2. In *blue*, the manipulator follows a linear interpolation in task space, while the *green* path indicates a linear interpolation in joint space.
Middle: The paths are pictured in both, *task* and joint space.
Right: The paths are parametrized w.r.t. time using a 5^{th} order polynomial with zero acceleration and velocity at start t_0 and end t_{end}. Thus, using a 3^{rd} dimension, the trajectory can be pictured. Paths and trajectories are pictured in task and joint-(or: *configuration-*) space.

2.2.1 Overview

Terminology

This section introduces relevant terms from the field of motion planning as they are used within this thesis.

Position and Pose In context of this thesis, *position* indicates either a scalar joint position/angle or a position in 2D or 3D Cartesian coordinates. A *pose* contains position *and* the respective orientations of a body in Cartesian space. Thus, in $\mathbb{R}^3$, a position has *three* and a pose *six* dimensions.

Tool-Center-Point (TCP) The *tool-center-point* is usually chosen as body-fixed location on the manipulator's end effector, e.g. a central point between the gripper jaws. In fig. 2.9, the TCP is defined at the end-tip of the 2-DOF manipulator.

Joint Space and Task Space Motion planning can be performed in arbitrary coordinate spaces. A common distinction for robots is the *joint space*, often referred as *configuration space*, and its *task space*. The configuration space $\mathcal{C} \in \mathbb{R}^n$ is typically the description of n joint positions $\boldsymbol{q}$ according to the robot's n DOFs. The task space $\mathcal{W}$ depends on the actual task of the robot. A common choice for manipulators is the pose of its TCP.

Inverse Kinematics The calculation of a configuration $\boldsymbol{q}$ for a given task $\boldsymbol{w}$ is noted as the inverse kinematics (IK) problem. The opposite term, i.e. the calculation of the task space representation (e.g. position of the TCP) for a given configuration, is referred as forward kinematics.

Redundancy The definition of *redundancy* is closely connected to the definition of configuration and task space. A system is denoted as redundant, when its task space has a lower dimension than its configuration space, i.e. it has more DOFs than required by the actual task. From a mathematical perspective, the set of equations to calculate the IK is under-determined. Thus, an infinite number of admissible solutions may exist ($\rightarrow$ *nullspace*).

Nullspace The *nullspace* $\mathcal{N}$ of a robot can be illustrated by the variety of joint velocities $\dot{\boldsymbol{q}}$ and accelerations $\ddot{\boldsymbol{q}}$, the robot can move without changing its task space constraint $\dot{\boldsymbol{w}}$ or $\ddot{\boldsymbol{w}}$, respectively. Thus, a motion in the nullspace results in a motion in joint space but not in task space. In non-singular configurations, the dimension of the nullspace is the difference between the joint and the task space, i.e. $\mathcal{N} \in \mathbb{R}^{(n-m)}$.

Paths and Trajectories One has to distinguish between both terms *path* and *trajectory*: A path is a sequence of states without information of time, while a trajectory comprises system states as a function of time. Regarding a robot manipulator, a path may be a geometric description of joint angles from start to goal configuration, while the joint trajectory also has information about joint velocities and accelerations. The concept of path and trajectory is pictured in fig. 2.9.
Motion planning handles both, path as well as trajectory planning. However, the planning process can be decoupled to plan a path first and determine the trajectory afterwards.

Trajectory Optimization The solution for many motion planning problems is not unique, i.e. there exists more than one trajectory that fulfills all constraints. To choose the best admissible trajectory, an additional desired behavior can be formulated in terms of a cost function. Minimizing this function while respecting constraints is named *trajectory optimization*, which leads to the optimal trajectory. This method is particularly important regarding the resolution of redundancy.

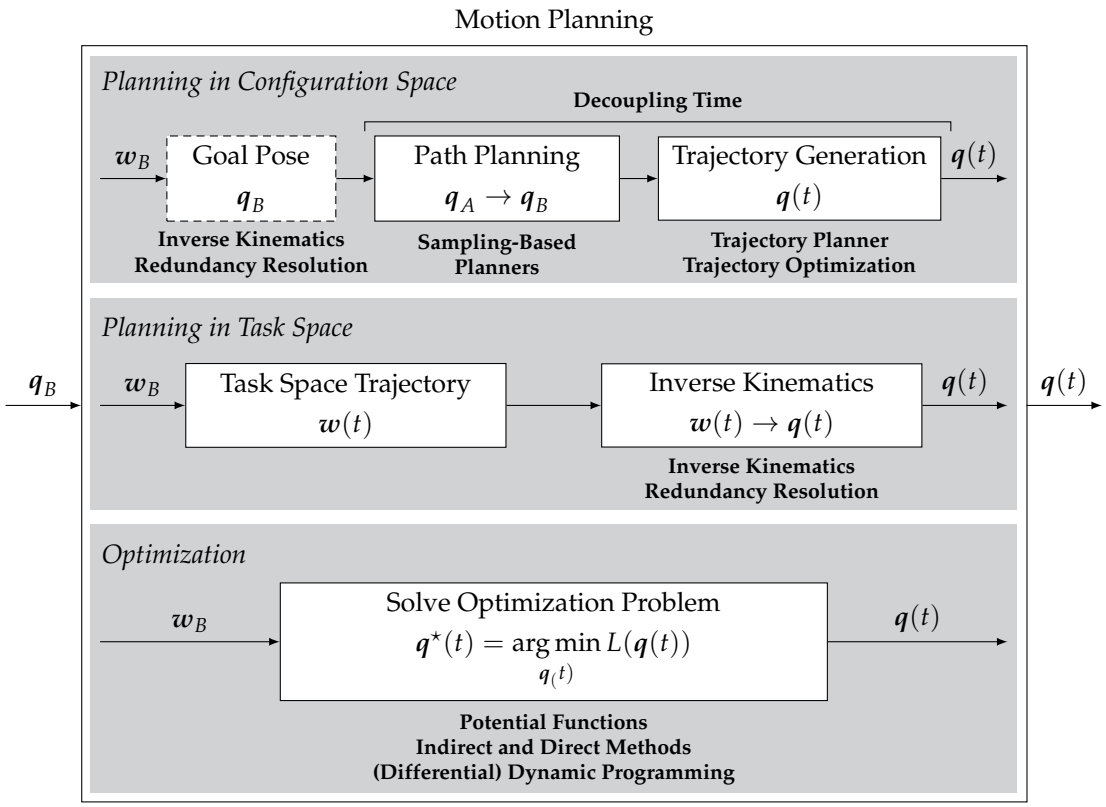


Figure 2.10: Motion planning according to fig. 2.8 in detail. The motion planning concepts are classified into three approaches. All approaches aim for a joint space trajectory $q(t)$ that fulfills the desired task space goal w_B. Concepts presented in this thesis (**bold**) are arranged according to the respective approaches and explained in further detail in section 2.2.2.

Motion Planning Approaches

A categorization of different motion planning approaches and methods is difficult. Often, the problem is solved using a combination of different approaches. Furthermore, the methods overlap regarding their planning spaces and applications. This thesis proposes a classification into three basic categories for redundant robots as follows: The *Planning in Configuration Space*, *Planning in Task Space* and *Optimization*. Although these categories are not applicable in general, they shall provide the reader with a rough guideline for the motion planning topics presented in this thesis.

Planning in Configuration Space Based on the early works of Lozano-Pérez (1983), a fundamental concept is to plan the path in the configuration space $\mathcal{C}$ of a manipulator. First, the goal configuration q_B has to be found that fulfills the task space goal w_B. This is done by applying *Inverse Kinematics* and *Redundancy Resolution* approaches. Second, a transfer path from the initial to the goal configuration is searched for in the collision-free part of $\mathcal{C}$, $\mathcal{C}_{\text{free}}$. According to the number

of joints of a robot n, its configuration space $\mathcal{C}$ has dimension $\mathbb{R}^n$. *Sampling-Based Planners* allow an efficient exploration of the search space. Some of their concepts are presented in section 2.2.2. The motion of the manipulator in task space along the path is usually not taken into account, e.g. a linear movement in $\mathcal{C}$ results in a curved one in $\mathcal{W}$-space for a manipulator with revolute joints (cf. fig. 2.9).

Once a collision free path has been found, the joint trajectory $\boldsymbol{q}(t)$ has to be determined using geometric planning approaches (*Trajectory Planner*) or *Trajectory Optimization* techniques. Calculating the path and trajectory subsequently is denoted as the *Decoupling Time* concept. This simplifies the planning problem enormously, since velocities and accelerations are not taken into account while finding a collision free path. However, capabilities of computed trajectories are limited and dynamic effects (manipulator accelerations, moving obstacles) cannot be taken into account during the path planning stage.

Planning in Task Space The second option is to plan a task space trajectory $\boldsymbol{w}(t)$ first and to then map it to the robot's joint space. This approach is advantageous for applications which require a particular task space trajectory, e.g. in the case of welding, gluing or assembly operations. The mapping from task space to configuration space is done by solving the *inverse kinematics (IK)* and *Redundancy Resolution* problem. State-of-the-art concepts are summarized in section 2.2.2. This approach requires a low computational effort. In many cases, simple assumptions for the task space trajectory are sufficient (cf. Baur, Schütz, Pfaff, Buschmann, et al. 2014) while the solution of the IK can be done in real time (cf. chapter 4).

Optimization Optimization methods can be applied to both aforementioned categories. For example, determining the shortest path or the trajectory with the minimum time or minimum effort may be desirable. Tasks and other constraints such as kinematic and dynamic limits as well as obstacles can be taken into account. Several approaches related to *Potential Functions*, *Indirect and Direct Methods* and *(Differential) Dynamic Programming* are presented in section 2.2.2.

2.2.2 Planning Concepts

This section provides an overview of related approaches and methods introduced and classified in the previous section.

Path Planning

A path can be defined either by waypoints or a continuous curve in configuration space or a task space. Advanced methods for finding suitable paths in the presence of obstacles and other constraints can be found in textbooks of Choset

et al. (2005), Latombe (1991), and LaValle (2006). One modern approach for finding collision-free paths even in highly cluttered environments is the category of sampling based planners: Operating in configuration space, the dimension of the planning space grows according to the number of DOFs of a system and the complexity of the problem increases exponentially. Within the past decade, single query sampling based planners were developed which cope efficiently with these high-dimensional planning problems. Starting from an initial configuration, they explore the search space by random samples towards a given goal configuration. By contrast to classic planning approaches, no explicit description of the collision-free $\mathcal{C}$-space is needed, but random samples are checked individually for collisions. The path results as a connection of collision free samples. One of the first reports of this approach, known as rapidly-exploring random trees (RRT), was published by LaValle (1998). A comprehensive survey of related methods is given in the textbooks of Choset et al. (2005) and LaValle (2006). In static environments it may be beneficial to reuse explored regions of the search space for multiple queries. Based on this principle, probabilistic roadmaps were proposed by Kavraki et al. (1996).

Paths that are generated by sampling based planners can be jerky and long-winded. Thus, they can be post-processed by heuristic based shortcutting methods (Chen and Hwang 1998) or be used as a feasible initial guess for trajectory optimization (El Khoury et al. 2013). Furthermore, there exists a wide variety of related planners that aim at obtaining higher quality paths by asymptotic optimization (Karaman and Frazzoli 2011; Perez et al. 2012) or that integrate differential constraints (LaValle 2006).

The availability of open-source implementations of various sampling based algorithms, e.g. within the OPEN MOTION PLANNING LIBRARY (Sucan et al. 2014) still has a large impact on their success. Sampling based planners are used for industrial applications as well, e.g. for planning and validation checking of complex assembling processes (Siemens PLM Software Inc. 2015).

Trajectory Generation

The problem of determining the sequence of states as a function of time is known as trajectory generation. When a (geometrical) path is given in advance, the problem is reduced to a time parametrization of the path. In the following, geometrical trajectory planning as well as trajectory optimization approaches for the transition "path to trajectory" are presented.

Position, velocity or acceleration joint constraints on the kinematic level can be taken into account using trapezoidal velocity profiles. Waypoints can be exactly met by stopping at each of them. In order to achieve a continuous motion of the robot, intermediate waypoints can be turned into *via-points*. By limiting the maximal accelerations, these via-points are only approximately met. This approach

is often referred as *parabolic blends*. Pseudo via-points can be inserted additionally in order to meet the via-points exactly. Other options are interpolation based schemes using polynomials or splines. An overview of these standard methods can be found in Craig (2005), Khalil and Dombre (2004), and Vukobratovic and Kircanski (1986). Although simple and robust, the aforementioned approaches have several drawbacks. Usually, they are not optimal regarding time and some do not follow the path exactly or show overshooting. A novel framework which uses a sequence of motion profiles is the REFLEXXES MOTION LIBRARIES as presented by Kröger (2010). It is designed to be able to quickly react to sensor information while executing the motion. A robust method of a time parametrization scheme providing synchronized joint trajectories respecting kinematic limits was developed by the author in collaboration with Lochner (2015).

In the late 80s, time optimal approaches were developed in order to determine dynamically feasible velocity profiles by parameterizing the path (Bobrow et al. 1985; Pfeiffer and Johanni 1986; Shin and McKay 1986). These methods follow the idea that at least one joint is in saturation regarding its velocity or acceleration limit. Pfeiffer and Johanni (1986) take advantage of the problem's topology by projecting the manipulator dynamics along the path. This yields a one-dimensional optimization problem which can be solved by dynamic programming approaches. All of these methods are obviously limited by the geometric path itself, which may not be optimal regarding system dynamics.

Redundancy Resolution

Kinematic redundancy offers additional freedom for motion planning. Secondary objectives can be pursued within the nullspace of the robot's motion, i.e. without interfering with the primary task. Additionally, a formulation of the IK problem on velocity level yields a linear relation between $\mathcal{C}$- and $\mathcal{W}$-space, which turns out to be beneficial for the following methods:

The resolved motion rate control (RMRC) algorithm as reported by Whitney (1969) uses the Moore-Penrose pseudoinverse of the Jacobian, achieving a local minimization of joint velocities. Presenting a scheme named automatic supervisory control (ASC) Liégeois (1977) extended the method, taking into account secondary objectives by a nullspace projection. Both approaches are explained in detail in section 4.1. Nakamura and Hanafusa (1987a) assigned priorities to multiple tasks in their related *task priority approach*. An overview of hierarchical approaches resolving redundancy on the velocity level was given by Siciliano (1990) and Vukobratovic and Kircanski (1986). Formulating the algorithm on the acceleration level is straight-forward and allows a calculation of the required actuator forces/torques directly using an inverse dynamic robot model (Luh et al. 1980). This is exploited by the *operational space approach* as presented by Khatib (1983) and the work of Hollerbach and Suh (1985) for solving redundancy by a local optimization of actuator torques. Secondary objectives are mostly taken

into account using cost functions which are minimized using gradient descent methods. Limits expressed by penalty functions are therefore *soft*. This may generate infeasible solutions in case e.g. joint limits can no longer be respected in nullspace. One approach is to saturate controls first in nullspace and, if needed, to alleviate the task space constraint. Recent works of Flacco et al. (2015) present real time capable efficient algorithms and a review of related approaches.

Feasible trajectories e.g. w.r.t. joint limitations can be calculated using inequality constraints. Kinematic and dynamic limits are respected in Cheng et al. (1994) solving the constraint optimization problem using a quadratic programming (QP) approach. Kanoun et al. (2011) presented a planning framework extending the QP to multiple equality and inequality constraints able to respect a hierarchical priorization. Improvements to this approach as well as its real time implementation were reported by Escande et al. (2010, 2014). Sampling based planners usually operate in configuration space, i.e. a goal configuration which fulfills the respective task has to be determined first. For planning with redundant manipulators, this is a limitation since one goal configuration out of an infinite number of admissible solutions has to be chosen. This problem has been addressed in several publications (Berenson et al. 2009; Bertram et al. 2006; Vande Weghe et al. 2008).

The term of redundant systems always depends on the respective task space definition. Using only the necessary set of constraints adds flexibility to the system and extends the range for secondary objectives. Thus, the view of motion planning as a constrained dynamic optimization problem is a logical consequence. An overview is presented in the following section.

Optimization Based Planning

Motion planning can be formulated as a dynamic optimization problem under equality and inequality constraints. Although its complexity is high, a growing number of approaches is reported solving this optimal control problem (OCP) numerically. Driven by steadily increasing computational power these strategies become more and more relevant.

Potential Functions Controlling a manipulator by a local descent along gradients of artificial *potential fields* was reported by Khatib (1986). This approach enables real time collision avoidance as well as joint limit avoidance. The local optimization of the potential field can likewise be considered as a cost function. One drawback is the presence of local minima where the solution may get stuck. Therefore, Rimon and Koditschek (1988) proposed a specially shaped cost function (*navigation function*) which guarantees the existence of only one (global) minimum at the goal. Under the assumption that an initial, collision-free path exists, Quinlan and Khatib (1993) connected the current configuration and the goal with

an elastic band. Similar to artificial potential fields, obstacles affect the elastic band by repulsive forces while the band itself is contracted by internal forces. By physical analogy to a mechanical spring, the path tends to its energy minimum. Dynamically appearing obstacles deform the path incrementally in real time, while preserving its topological properties. Thus, the path is not affected by local minima. This framework was also presented in an extended form as *elastic strips* by Brock and Khatib (2002). Related to these approaches, but releasing the prerequisite of the initial path being collision-free, the planning algorithm *CHOMP* was introduced by Ratliff et al. (2009) and Zucker et al. (2013). A cost function taking into account the path's smoothness and penalizing collisions is minimized by covariant gradient descent. Integrating obstacle distance penalties along the arc instead of w.r.t. time assures that safety distances are respected even at higher velocities. Computational efficiency is furthermore increased by using precomputed voxel-based distance fields. While the aforementioned algorithms consider obstacles in task space, Shiller and Dubowsky (1991) proposed a global optimization scheme using $\mathcal{C}$-space obstacle representations.

Indirect and Direct Methods In optimization theory, two categories can be distinguished regarding the formulation of dynamic optimization problems:

- *Indirect Methods* start by formulating the hamiltonian and deriving analytically the adherent differential optimality conditions. These result in a two-point boundary value problem which is usually solved numerically and yields the optimal trajectory (*"first optimize, then discretize"*.)
- *Direct Methods* transform the dynamic optimization problem to a static one by discretizing w.r.t. time. The static problem can be solved using nonlinear programming methods (*"first discretize, then optimize"*.)

Both approaches can be used to optimize a robot's trajectory. Their usage depends on the general set-up of the actual application.

Indirect Methods provide highly accurate solutions, but require a good initial guess since their convergence ratio is comparatively small (Papageorgiou et al. 2012). Optimality conditions with a limited system input are given by Pontryagin's minimum principle (PMP) (Pontryagin et al. 1962). Based on the formulation of the Hamiltonian, several differential constraints result from the calculus of variations. Dealing in particular with redundant manipulators, Nakamura and Hanafusa (1987b) developed a two-point boundary value problem (TPBVP) formulation based on automatic supervisory control (ASC) (Liégeois 1977), using a projected nullspace input. This formulation has been used in the author's work on predictive inverse kinematics (Schütz, Buschmann, et al. 2014). Related to Nakamura and Hanafusa (1987b), Martin et al. (1989) proposed a similar approach but formulated the task space constraint in terms of position. Minimum time and energy trajectories for an industrial manipulator were found by Stryk

(1994), solving the optimality conditions by a multiple shooting approach. Since this method is sensitive w.r.t. the initial guess, an approximate, suboptimal solution is generated first using a more robust direct collocation algorithm.

Direct Methods have a larger convergence ratio. Equality or inequality constraints can be taken into account more simply. However, the computational effort may be higher and the solution depends strongly on the chosen representation of the control function. Since the system's ordinary differential equation (ODE) is taken into account as an equality constraint, a solution may not be physically valid in case equality constraints are not met. An early example can be found at Singh and Leu (1991), who investigated trajectory optimization of serial manipulators in offline simulations. Thereby, they took obstacles into account by inequality constraints. Wang and Hamam (1992) reported a similar approach for a 3-DOF manipulator. Schlemmer (1996) divided the overall planning problem into smaller sub-problems over time. Solving these by sequential quadratic programming (SQP), he generated optimal trajectories for a 10-DOF manipulator following a moving goal in 3D task space .

Task space trajectories modeled as cubic splines for stepping over obstacles by a quadruped robot were found by Kolter and Ng (2009) using convex optimization. Convexity is achieved by limiting the joints' workspace to a convex hull. A hierarchical optimization approach is presented, named TRAJOPT by Schulman et al. (2013). Infeasible constraints of the initial guess are first handled using penalty functions in order to "push" the trajectory to feasible regions. The problem is solved using sequential convex optimization while using efficient formulations for collision avoidance. A feasible solution for a subsequent optimization of the discretized problem was found by El Khoury et al. (2013) using sampling based methods. The author presented the evaluation of a similar approach but using heuristic task space paths for initialization (Schütz et al. 2015).

(Differential) Dynamic Programming Another approach for solving dynamic optimization problems was introduced by Bellman (1954) known as *dynamic programming*. Although this method has been developed independently from Pontryagin's minimum principle, it can be shown that the resulting optimality conditions can be expressed in terms of Bellman's equations and vice versa. The dynamic programming approach yields a discrete problem. Its complexity grows exponentially with its dimension ("curse of dimensionality") and even nowadays it remains impossible to solve higher dimensional problems using this method. In consequence, Jacobson and Mayne (1970) presented a computationally efficient method known as differential dynamic programming (DDP), which solves the problem iteratively by second order approximations around an initial guess, using a feedback pass. This method is proposed by Todorov and Li (2003) for optimal control of manipulators. In order to increase robustness, Tassa et al. (2012) present several (numerical) extensions and show its computational efficiency. Control limits are further integrated by adding an additional quadratic problem (Tassa et al. 2014).

2.2.3 Planning in Cluttered Environments

Obstacles can be handled in two ways: The classic approach is to *avoid* obstacles, regardless of their mechanical properties and assuming their geometry is known. These concepts work well in structured environments with a sparsely obstructed workspace. By contrast, taking into account biological, cluttered and unstructured environments, collision free paths may not even exist. A motion planner has to respect the mechanical properties of objects and to adapt its plan for obstacles being either compliant or stiff, fixed or movable. Instead of avoiding collisions, the robot *interacts* with other objects. An overview of both approaches is given in the following.

Obstacle Avoidance

The robot's $\mathcal{C}$-space is divided into the free space $\mathcal{C}_{\text{free}}$ and the space obstructed by obstacles' geometry $\mathcal{C}_{\text{obs}}$. Trajectories are only searched for in $\mathcal{C}_{\text{free}}$. In the case of potential field approaches, repellent potentials are assigned to obstacles (Khatib 1986). Most optimization methods are based on this principle, they only differ in terms of the cost function formulation (which depends on the minimum distance between two bodies) and obstacle representation. In order to ensure fast collision checking by minimum distance calculation, simple geometries such as capsules or spheres are used to model obstacles as shown by Schwienbacher et al. (2011). Distances are calculated either on demand or can be determined by precomputed distance fields (Zucker et al. 2013). Motion planning algorithms based on sampling perform a boolean test for collision of single configurations (LaValle 2006). Since these tests have to be executed for every new explored state, efficient environment representations like *octomaps* are available (Hornung et al. 2013).

Interaction and Tactile Feedback

Planners which allow the robot to displace obstacles were presented by Dogar and Srinivasa (2011) and Stilman et al. (2007). They assume to have a suitable model of the environment as well as of obstacles' properties and do not consider feedback. The deformation of objects is taken into account in the work of Frank et al. (2011) and Patil et al. (2011) using deformation models. However, these aforementioned approaches assume either knowledge about the environment or make simplifying assumptions, which may not be admissible in cluttered and unknown environments.

The mechanical properties of the actual contact of the robot with the environment can be determined using force feedback. A typical application is an industrial assembly process where even small position inaccuracies can lead to

high contact forces. Concepts are extensively studied in literature for controlling the interaction forces of the manipulator (force control) or achieving a compliant behavior (stiffness or impedance control) while pursuing a task. An overview of related approaches was given by Villani and De Schutter (2008). Typically, external forces at the end effector are taken into account. However, contacts may also occur at the robot arm itself in cluttered environments. Park and Khatib (2008) reported a concept which deals with multiple contacts along a robot arm. While the primary task of the robot is to control these forces, the actual motion task is projected to the manipulator's nullspace. Based on the estimation of external joint torques, safety reaction strategies were developed for a 7-DOF lightweight manipulator by De Luca et al. (2006) and Haddadin et al. (2008). The manipulator switches its control strategy when collisions are detected and tries to reduce external torques. Another concept for whole arm tactile planning using a tactile skin was presented by Jain and Killpack (2013). Contact forces are minimized by solving a QP while taking into account kinematic and contact force limits as well as the desired task space demand. A further improvement of this implementation was reported by Killpack and Kemp (2013) by extending the moving horizon as well as considering dynamics of the arm. Related to these concepts, the author presented a planning approach based on the automatic supervisory control (ASC) (Liégeois 1977) scheme (see section 4.1.2).

2.2.4 Summary

This section summarizes important aspects and concepts for planning the motion of redundant robot manipulators. Many different approaches can be distinguished and the ideal choice strongly depends on the robot structure itself and its respective application. In many cases, several concepts are applied in parallel and some new motion planning approaches rely on a combination of known ones. The core requirement for redundant manipulators is to solve the IK problem and to resolve its redundancy. For example, while moving in cluttered environments, it may be necessary to adapt the motion plan in real time. Unforeseen contacts with other objects require a reaction of the robot.

In this thesis, the redundancy formulations and concepts are presented in detail and extended to a tactile planning framework in chapter 4. Furthermore, in chapter 5 these redundancy formulations are combined with indirect optimization methods in order to achieve real time optimization within a certain prediction horizon. Depending on the particular task, the solution of the motion planning problem as an offline optimization problem is applicable. Thus, chapter 6 evaluates the benefit of a direct optimization method for finding suitable trajectories for a harvesting manipulator picking peppers.

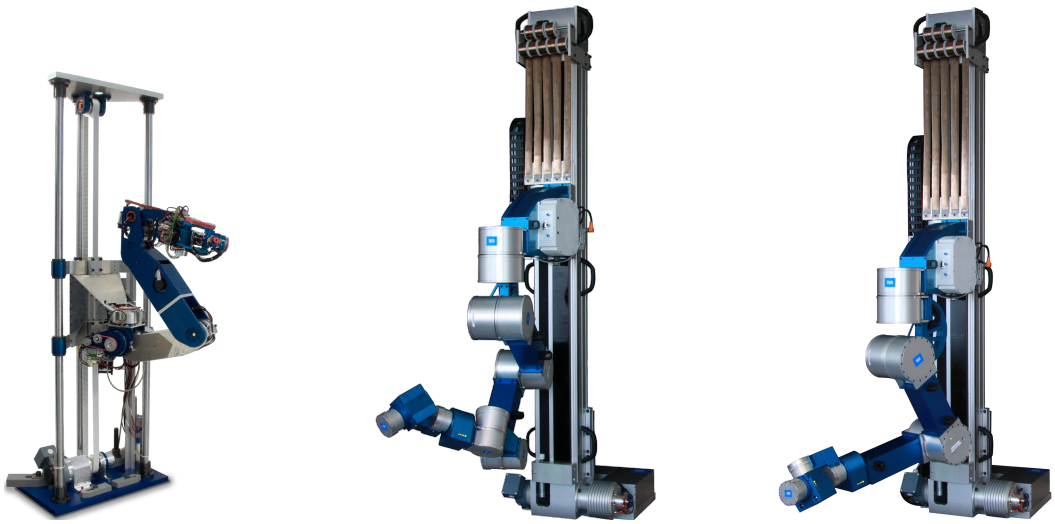
Chapter 3

Manipulator System

The motion planning methods and algorithms of this thesis were applied and tested using the CROPS manipulator. Hence, particular challenges and requirements regarding their usability in real world experiments could be evaluated. This section presents this manipulator system. The robot was developed within the EU-project CROPS (cf. section 2.1.3) at the author's institute and is the subject of the PhD theses of Baur (2015) and Pfaff (2015). Overviews of the system were given at international conferences by the author (Schütz, Pfaff, Baur, et al. 2014; Schütz et al. 2014; Ulbrich et al. 2015).

The main objective of the CROPS manipulator is to pick fruit autonomously in uncertain environments. In order to be able to solve these tasks, the manipulator has to be equipped with several tools and abilities. First, the robot system requires software and hardware interfaces which enable the user to guide its gripper towards the fruit. The integration of a 3D joystick with haptic feedback allows the user to teleoperate the arm. Second, for dealing with obstacles in the workspace of the arm, the manipulator needs information about its environment. Since the capabilities of vision sensors are limited in cluttered environments like greenhouses, tactile information is advantageous. This can be done using a tactile sensor that directly measures the contact forces with the environment. Another option is to use the signals from current and position sensors of the robots' joints to estimate these forces.

This section provides these tools and abilities: First, a brief introduction recapitulates the CROPS manipulator system including its hardware, communication, control and software architecture. For controlling the manipulator, several interfaces were implemented, including the teleoperation by a haptic device, which is shown in section 3.1. Section 3.2 presents appropriate kinematic and dynamic models required by the proposed motion planning algorithms. Tactile sensors provide information about contacts of the robot with the environment. Related concepts are shown in section 3.3.



(a) Prototype 1 (9-DOF) (b) Prototype 2 (9-DOF) (c) Prototype 2 (7-DOF)

Figure 3.1: CROPS agricultural manipulator prototypes of generation 1 (a) from 2012 and generation 2 (b,c) from 2014. The generation 2 prototype is shown in a 9-DOF (b) and 7-DOF (c) configuration emphasizing its modular design.

3.1 Hardware and Interfaces

3.1.1 Manipulator Prototypes

In the CROPS project, two different versions of an agricultural manipulator were developed. Generally, a robot manipulator is "assumed to be a system of rigid bodies connected by ideal joints and powered by ideal force generators" (Featherstone 1987, p. 5). The design of the manipulator prototypes follows the paradigm of a multipurpose usage. Different applications, such as the selective harvesting of different field crops (e.g. sweet peppers, apples, grapes) or the precision spraying of locally diseased fruit require different manipulator kinematics. Thus, kinematics of the CROPS manipulators can be changed due to their modular design. A thorough overview of the developed hardware system can be found in Pfaff (2015) while Baur (2015) presents its software architecture and motion planning concepts.

Both manipulator prototype generations are depicted in fig. 3.1: The first generation, prototype 1[1], (left) has one prismatic and eight revolute joints in its most flexible configuration as it has been used for harvesting sweet peppers and apples. A 6-DOF version of the prototype 1 has been employed for precision spraying of grapes. The second generation, prototype 2 was developed in 2014 (Pfaff

[1]Two prototypes of generation 1 were build within the project. Both have the same kinematics and a similar hardware design and are denoted as "prototype 1" in this thesis. While the first one was used for field experiments at Wageningen UR, KU Leuven and University of Milano, the second one remained at the author's institute and is used for laboratory experiments.

et al. 2014; Schütz, Pfaff, Baur, et al. 2014). It follows a modular design allowing various kinematic configurations using integrated drive modules. Fig. 3.1 exemplifies a 7-DOF and a 9-DOF configuration of the manipulator. In this thesis, the trajectory optimization approaches (cf. chapters 5 and 6) are applied to the prototype 1. A brief overview of its applications is further given in section 2.1.3. This section, however, focuses on the prototype 2 in detail. While for the trajectory optimization methods only kinematic models were taken into account, an accurate dynamic model of prototype 2 is used for the proprioceptive sensing of external torques (section 3.3.3).

3.1.2 Manipulator Hardware

The CROPS manipulator prototype 2 has up to 8 revolute and one prismatic joint. The revolute joints are actuated by integrated drive modules. Each drive module comprises a brushless direct current (BLDC) motor, HARMONIC DRIVE gear, power electronics and position measurement sensors at the motor shaft and output shaft. The prismatic joint consists of the linear guide FESTO[2] EGC-TB which is actuated via belt transmission by a BLDC motor without additional gears. It is equipped with position sensors on the motor shaft as well as on the linear guide. The robot's joints are controlled by ELMO[3] Gold motor drivers. The motor drivers implement a cascaded P-PI controller scheme. The outer loop (position control) runs at 50 μs and can either be closed by the absolute encoder or by the position sensor on motor side. The weight of the manipulator arm is compensated on the prismatic joint by constant force springs. Except for joint 2, all drive modules include a brake. For detailed information about the hardware configuration refer to Pfaff (2015).

3.1.3 Communication

The system architecture of the CROPS manipulator is pictured in fig. 3.2. All joints are serially connected by an ETHERCAT[4] bus to the real time control unit. Its high bandwidth allows a sampling rate of 1 kHz. Typical sensor data being transmitted at every cycle are

- Active motor current
- Position & velocity on the motor side
- Position on the output shaft (absolute encoder)
- Status of motor drivers

[2] Festo AG & Co. KG, http://www.festo.com.
[3] Elmo Motion Control Ltd., http://www.elmomc.com.
[4] Beckhoff Automation GmbH & Co. KG, http://www.beckhoff.com.

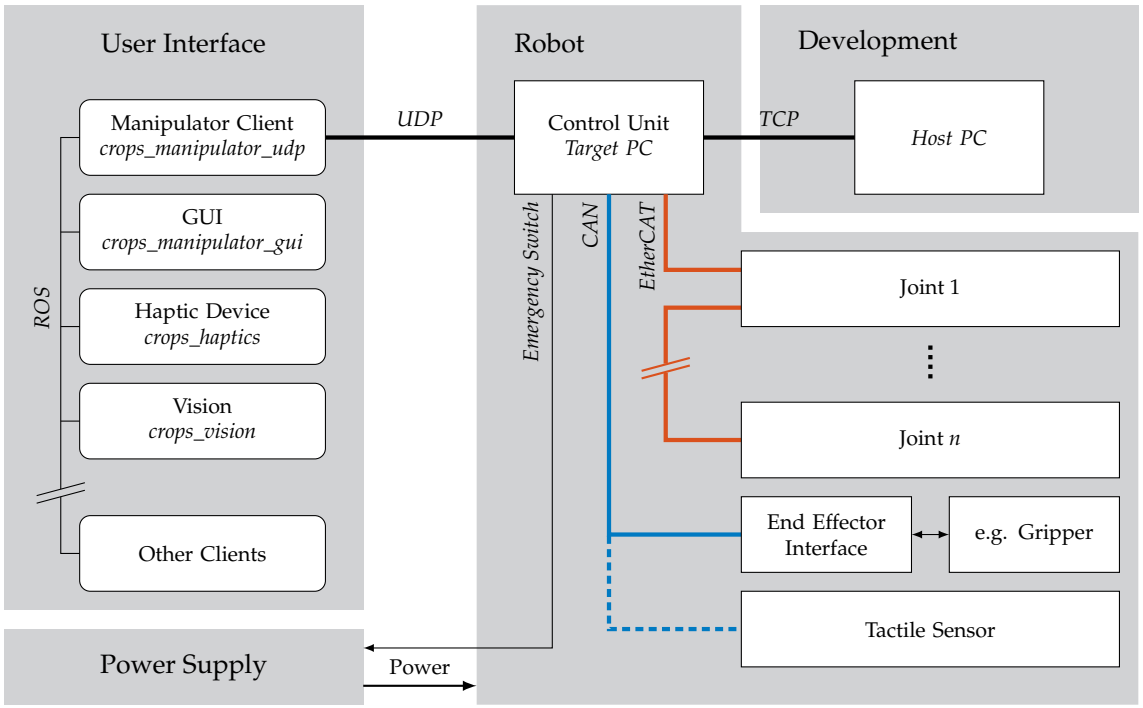


Figure 3.2: Communication Architecture of the CROPS manipulator prototype 2. Optionally, the tactile sensor can be connected to the CAN bus.

Other peripheral devices such as the end effector or the tactile sensor are connected by a CAN bus. Furthermore, the control unit is able to switch off the main motor power in case an error occurs. The real time control unit (*target PC*) runs a SIMULINK REAL TIME TARGET (Matlab R2014b)[5] with a base rate of 1 kHz. This applies to the calculation of time critical components such as direct and inverse kinematics, including collision and limit checking. Other tasks such as the precalculation of the task space trajectory run at lower sampling rates. A thorough overview of the software architecture is given in Baur (2015). The user communicates with the control unit via the *manipulator client* which maps ROS[6] messages to UDP signals. The user interfaces are presented in the following section. Furthermore, a *host PC* can be connected to the target PC for development purposes.

[5] The MathWorks, Inc., http://www.mathworks.com.

[6] Robot Operating System (ROS) "is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms." (ROS.org 2016) Basically, the ROS middleware is structured as a decentralized network of *nodes* that communicate with each other via *messages*, supervised and coordinated by a *master*. Each node is a software module that provides its own functionality. For further information, refer to Quigley et al. (2009).

3.1.4 User Interfaces

Several interfaces for controlling the manipulator are implemented to allow the user a maximum flexible usage of the system. The user interfaces connect to the control unit via UDP using an interface ROS node. Furthermore, the user is able to connect to the control unit directly via MATLAB/SIMULINK on the host PC to the target PC (cf. fig. 3.2).

This section introduces the general ROS interfaces to the manipulator. These enable the connection of other peripheral devices such as cameras or joysticks to the robot. Here, the teleoperation of the robot by a haptic joystick will be presented in greater detail.

ROS User Interfaces

The ROS software network for controlling the CROPS manipulator with its nodes is depicted in fig. 3.3. The manipulator client node is the gateway between the ROS network and the manipulator. A list of manipulator commands is given in table 3.1. Within the ROS network, the nodes are able to communicate with each other using ROS communication interfaces such as messages or actions. As presented by Baur (2015), a GUI is implemented to control and monitor the manipulator. Furthermore, the ROS communication can be tested without the real manipulator using a simulator node.

Furthermore, additional peripheral devices can be connected to the ROS network using suitable drivers. Aiming for autonomous harvesting of sweet peppers, a RGB-D camera was integrated for fruit and stem localization (Schütz, Pfaff, Baur, et al. 2014). Suitable image processing algorithms for laboratory demonstrations were developed in the student projects of Hazirbas (2013) and Yousefpour (2015). While this approach rendered possible the autonomous harvesting of fruit in the laboratory, its reliability and accuracy turned out to be insufficient. Therefore, a teleoperation system was developed in order to enable the user to control the manipulator's end effector online e.g. for gripping fruit. A haptic input device allows the control of the manipulator's end effector online while providing feedback to the user. This teleoperation system is explained in the following section.

Teleoperation with Haptic Device

In order to teleoperate the CROPS manipulator, an interface using a 3D joystick with force feedback (haptic device) was implemented. In the following, two control strategies are presented, a *position* and a *rate* controller. This section is based on the author's collaboration with D. Wahrmann and the student project

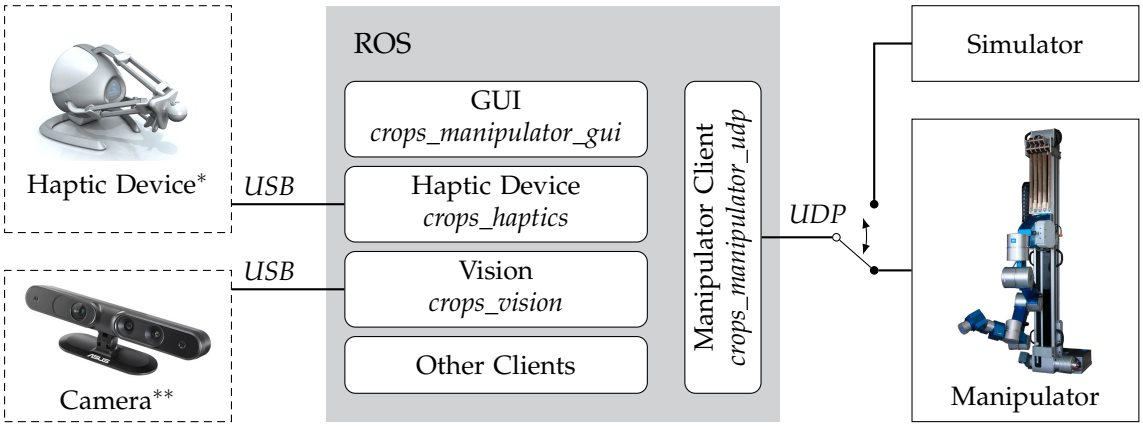


Figure 3.3: User Interface structure of the CROPS manipulator. The central part is the ROS framework, which includes several nodes for communication with the manipulator and other peripheral devices. The nodes are able to communicate with each other. Using a simulator, the framework can be tested without the real manipulator. Images taken from *Novint Technologies Inc. (2012) and **ASUSTeK Computer Inc. (2016).

of Schaller (2016). According to Niemeyer et al. (2008), the principle of teleoperation can be described by seven components as depicted in fig. 3.4. In this work, the *master* consists of the 3D joystick, which measures the user input. These commands are interpreted by a *controller* which is implemented as the ROS node *crops_haptics*. Passing the ROS framework and the UDP interface to the manipulator, the manipulator control unit receives the commands. The CROPS manipulator (*slave*) executes the motion command and interacts with the *environment*. Information about the interaction manipulator - environment can be gathered e.g. by a tactile sensor. In addition to the current status (positions, velocities) of the manipulator, these are fed back to the user in the opposite direction.

Interface	**Target Variable**	**ROS topic**
Point-to-Point End Effector Movement on straight-line task space path	w_{des}	Action
Point-to-Point End Effector Movement on a heuristic task space path accounting for the stem and fruit position (Baur, Schütz, Pfaff, Buschmann, et al. 2014)	w_{fruit}	Action
Online Joint Velocity Control	$\dot{q}_{\text{des}}$	Message
Online End Effector Velocity Control	$\dot{w}_{\text{des}}$	Message
Offline Joint Trajectories	$\left[(t^0, q^0); (t^1, q^1); \ldots; (t^N, q^N)\right]$	Message

Table 3.1: Manipulator interfaces provided by the ROS node manipulator client *(crops_manipulator_udp)*.

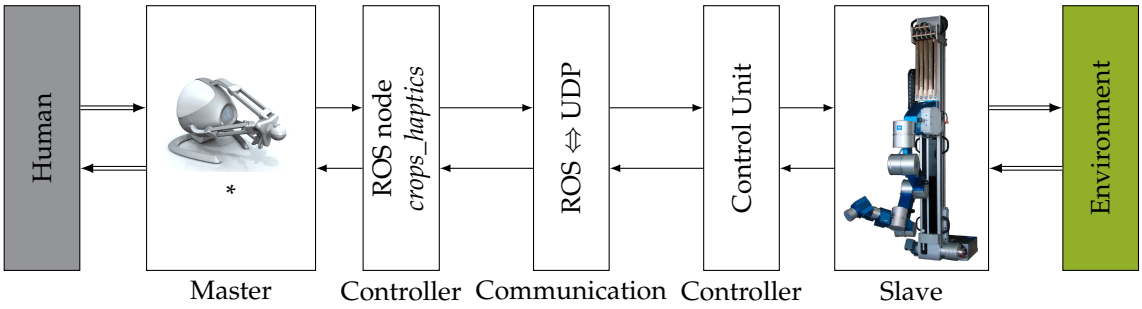
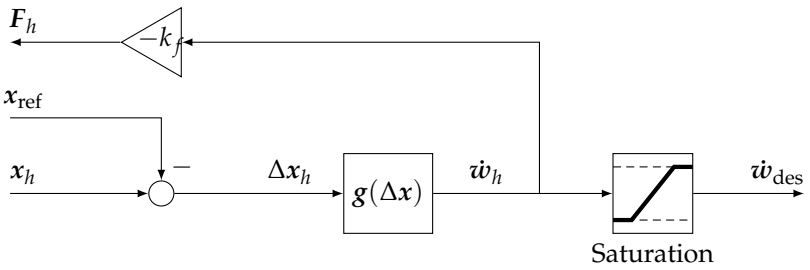


Figure 3.4: The work-flow of teleoperation of the CROPS manipulator according to the scheme presented by Niemeyer et al. (2008), depicted as a chain of two-port elements connecting the one-port user with the one-port environment. Signals are drawn as single lines while interactions are marked as double lines. Image taken from *Novint Technologies Inc. (2012).

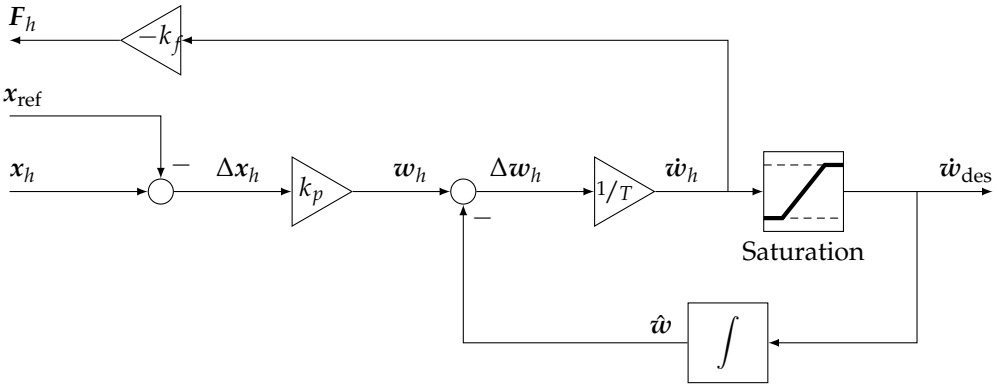
Hardware Setup For the application, the haptic device FALCON from Novint Technologies Inc. (2012) is used for the 3D information input. The device has a parallel kinematics design with 3-DOF. Every joint comprises a position encoder and an actuator enabling force feedback. It is connected to the user PC by USB and runs at 1 kHz. Its hardware driver and firmware are provided by the open-source project LIBNIFALCON (Machulis 2010) which computes the direct and inverse kinematics of the device. More information about its workspace and dynamic properties can be found in Martin and Hillier (2009). Additionally, the FALCON provides four buttons which are used for activating control options or to operate the end effector (e.g. close/open the gripper).

Rate and Position Control of the Manipulator In order to control the manipulator in an intuitive manner, the user directly controls the end effector. Thereby, the ROS interface *online end effector velocity control* (cf. table 3.1) is used. Hence, the manipulator executes a task space velocity command $\dot{\boldsymbol{w}}_{\text{des}}$ which includes the translational and angular velocity of the end effector. Inverse kinematics are calculated on the robot's side using the instantaneous ASC scheme as shown in section 4.1.2. Two approaches are implemented for controlling the manipulator's end effector by the haptic device:

- *Rate Control:* A deviation of a predefined reference position is interpreted as a task space velocity of the end effector. This is analogous to a joystick control in computer games or named *jogging* for industrial robots (either in task space or joint space).
- *Position Control:* A translational movement of the haptic device is mapped to a translational movement of the end effector. This scheme corresponds to the control of the cursor by a computer mouse.



(a) Rate controller.



(b) Position controller.

Figure 3.5: Rate and position controller for teleoperation of the CROPS manipulator.

Rate Controller Implementation Since the workspace of the manipulator is very large compared to the one of the haptic device ($\approx 15:1$), the concept of the rate controller is considered to be beneficial (Kim et al. 1987). Starting from a reference position $\boldsymbol{x}_{\text{ref}}$ of the haptic device, the user moves the device to $\boldsymbol{x}_h$. Scaled by an arbitrary function $\boldsymbol{g}(\Delta\boldsymbol{x}_h)$, the relative position $\Delta\boldsymbol{x}_h := \boldsymbol{x}_h - \boldsymbol{x}_{\text{ref}}$ can be directly interpreted as a velocity command for the manipulator. Experiments have shown that an exponential form with the scalar shaping factors $c_{1,2}$ yields an efficient and intuitive control for the user:

$$\dot{\boldsymbol{w}}_h = \boldsymbol{g}(\Delta\boldsymbol{x}_h) = \frac{1}{c_1} e^{c_2 \|\Delta\boldsymbol{x}_h\|} \frac{\Delta\boldsymbol{x}_h}{\|\Delta\boldsymbol{x}_h\|} \tag{3.1}$$

The resulting task space velocity $\dot{\boldsymbol{w}}_h$ is saturated by a lower/upper bound in order to ensure admissible manipulator commands $\dot{\boldsymbol{w}}_{\text{des}}$. Furthermore, the stability in the presence of delays is increased by lowering limits. The user receives a feedback force $\boldsymbol{F}_h$ which is proportional to the commanded velocity $\dot{\boldsymbol{w}}_h$ by a factor of $-k_f$. Hence, analogously to an exponential spring the haptic device tends towards its reference position. The scheme of the rate controller is shown in fig. 3.5a.

Position Controller Implementation The position controller maps a position user command at the haptic device to a corresponding end effector position of the manipulator. Since the manipulator provides no online task space position

interface, a position user command has to be transposed first into a suitable task space velocity $\dot{\boldsymbol{w}}_h$. This is realized by adding an extra control loop. The resulting end effector position $\hat{\boldsymbol{w}}$ is estimated internally by integrating the commanded task space velocity $\dot{\boldsymbol{w}}_{\text{des}}$. The current end effector position $\boldsymbol{w}$ provided by the manipulator itself cannot be used due to the high round trip delay ($\approx 50 - 100\,\text{ms}$) which leads to instabilities. Experiments showed that the numerical drift can be neglected since the user compensates deviations by suitable commands. Thus, the position difference $\Delta\boldsymbol{w}_h := k_p \Delta\boldsymbol{x}_h - \hat{\boldsymbol{w}}$ is converted to a task space velocity command $\dot{\boldsymbol{w}}_h$ by the reciprocal sampling time $1/T$. Additionally, the user position command can be scaled by k_p to adapt the covered workspace of the position controller. The resulting task space velocity $\dot{\boldsymbol{w}}_h$ is again saturated and sent to the manipulator as $\dot{\boldsymbol{w}}_{\text{des}}$, the desired task space velocity. Similarly to the rate controller, the user receives a force feedback on $\dot{\boldsymbol{w}}_h$. By analogy to a computer mouse, the user can move the end effector to a goal outside the haptic device's workspace by *indexing*, i.e. by resetting the reference position $\boldsymbol{x}_{\text{ref}}$ at the border of its workspace. A block diagram of the position controller is shown in fig. 3.5b.

Since the FALCON has 3-DOF, it is used only for gathering of translational user commands. The translational velocity of the end effector is expressed in world coordinates since experiments showed that a co-rotated system of the end effector is less intuitive for the user. Rotation commands can be given by keyboard inputs that are mapped to constant angular velocities.

Force Feedback Besides feedback of the motion control commands, other information can be passed to the user by superposing feedback forces on the 3D joystick. Two feedback options are implemented:

1. Contact forces of the manipulator arm with its environment are measured by a tactile sensor (cf. section 3.3.2) and transmitted to the user on the haptic device.
2. An estimate of the quality of the manipulator's configuration is given by the secondary objectives cost function of the inverse kinematics algorithm (cf. section 4.2). This information can also be mapped to the haptic device.

In the following, both variants are briefly introduced.

The tactile sensor measures a resulting external force $\boldsymbol{F}_{\text{sen,ct}}$ (cf. section 3.3.2). One approach is to project the external force to the TCP of the manipulator and to apply it to the haptic device. At least for the direction, this seems to be an obvious choice, since user commands and feedback forces are both in task space. Thus, $\boldsymbol{F}_{\text{sen,ct}}$ is first projected to the joints using the transposed translational Jacobian $\boldsymbol{J}_{\text{poc}}^T$ of the contact point $\boldsymbol{r}_{\text{poc}}$ resulting in the generalized external torque $\boldsymbol{T}_{\text{ext}}$. In a subsequent projection, $\boldsymbol{T}_{\text{ext}}$ is mapped to the manipulator's task space by its

translational Jacobian of the TCP, $\boldsymbol{J}_{\text{TCP}}$. Thus, the resulting force mapped to the haptic device $\boldsymbol{F}_{h,t}$ leads to

$$\boldsymbol{F}_{h,t} = \boldsymbol{J}_{\text{TCP}} \underbrace{\boldsymbol{J}_{\text{poc}}^{T} \boldsymbol{F}_{\text{sen,ct}}}_{\boldsymbol{T}_{\text{ext}}}. \tag{3.2}$$

However, this approach does not appear intuitive to the user, since the absolute value of the force applied to the haptic device varies with the respective lever arm. The intensity of the force is lower for the arm being stretched out than for a folded configuration. This can be compensated by using the direction of the external force while keeping its magnitude:

$$\boldsymbol{F}'_{h,t} = \|\boldsymbol{F}_{\text{sen,ct}}\| \frac{\boldsymbol{F}_{h,t}}{\|\boldsymbol{F}_{h,t}\|}. \tag{3.3}$$

As shown in section 4.2, secondary objectives can be expressed in terms of cost functions H. Examples for secondary objectives are joint limit avoidance, a comfort pose or self collision avoidance. The gradient $\nabla_{\boldsymbol{q}} H$ yields a descending direction for each joint, i.e. how a joint has to move in order to minimize H. By projecting this information to the task space and applying it to the haptic device, the user can feel which directions are more convenient, i.e. help minimizing these costs. Thus, the force feedback $\boldsymbol{F}_{h,s}$ resulting from the secondary objectives can be calculated as

$$\boldsymbol{F}_{h,s} = \boldsymbol{J}_{\text{TCP}} \left(\frac{\partial H}{\partial \boldsymbol{q}} \right)^{T}. \tag{3.4}$$

Experiments showed that, by contrast to the mapping of the tactile forces, secondary objectives $\boldsymbol{F}_{h,s}$ do not appear intuitive to the user.

All force feedback approaches that are based on sensor information from the robot are heavily influenced by delays. In the current implementation, the commanded task space velocities are limited to comparatively slow dynamics in order to ensure stability. Faster dynamics could be achieved by significantly reducing the delays. A quantitative evaluation of achievable minimal round-trip delays and their influence on the system stabilities could be a topic of further research.

Remarks

The user interfaces presented in this section allow the manipulator either to operate autonomously or to be teleoperated by the user. The ROS interfaces were successfully used for picking sweet peppers in greenhouses (Bac et al. 2015), harvesting apples in orchards (Wouters et al. 2014) or the precision spraying of grapes (Oberti et al. 2016) in an autonomous operation mode. However, this mode relies

on specific agricultural scenarios and requires many peripherals, such as sensors, illumination rigs, etc., which are not applicable for modified scenarios or laboratory experiments. Furthermore, the robot may get stuck in complicated setups where user interaction is required. Thus, the teleoperation mode is introduced in this section. It enables even non-expert users to control the manipulator to "hand-pick" fruit precisely as was shown in many laboratory demonstrations.

3.2 Kinematic and Dynamic Models

Motion planning and trajectory optimization methods require knowledge of the forward kinematics and dynamics of the manipulator system. Especially for real time applications, an efficient and well structured calculation of those is essential. Thus, this section presents the kinematic and dynamic models of the CROPS manipulator. First, the basic concepts for the (recursive) calculation of both are shown for robots with open chain (tree like) kinematic structures. Second, empirically identified parameter sets for the CROPS manipulator prototype 2 are presented in section 3.2.3, followed by a validation of the inverse dynamics model. The models developed in this section for the CROPS manipulator prototype 2 are based on the work of Baur (2015).

3.2.1 Relative Kinematics

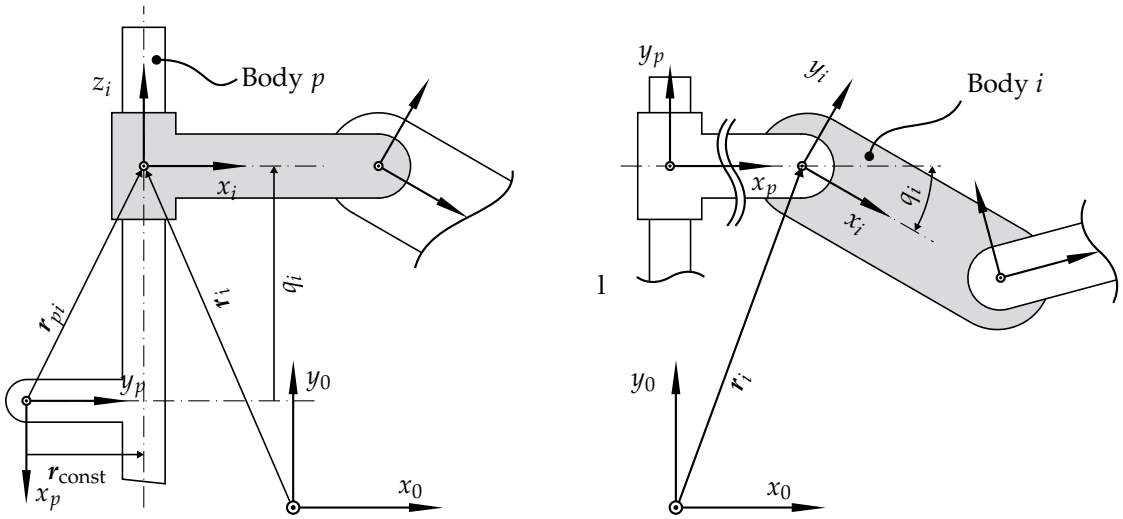
In the following, the computation of the required kinematic quantities for the inverse dynamics model is summarized briefly. Further detailed descriptions on relative kinematics and recursive computation can be found in Featherstone (1987) and Ulbrich (1996).

A serial manipulator consists of links that are connected by n joints, which correspond to its DOFs. Their current positions are concatenated in the vector $\boldsymbol{q} \in \mathbb{R}^n$ while each single position is denoted as q_i. Thus, $\boldsymbol{q}$ is considered to be the set of minimal or generalized coordinates of the system. To each link, a body-fixed frame is assigned with the z-axis aligned to the respective joint axis following the convention of Craig (2005). A scheme of a prismatic and a revolute joint including the kinematic quantities (cf. table 3.2) is depicted in fig. 3.6.

For open chain kinematic structures each body's kinematic quantities can be calculated recursively, i.e. relative to its parents' forward kinematics. Thus, only the relative forward kinematics have to be determined for each body. The translational and rotational kinematic quantities (cf. table 3.2) of the body i can be calculated by

$$
\begin{aligned}
{}_i\boldsymbol{r}_i &= \boldsymbol{A}_{ip}\ {}_p\boldsymbol{r}_p + {}_i\boldsymbol{r}_{pi} \\
{}_i\dot{\boldsymbol{r}}_i &= \boldsymbol{A}_{ip}\ {}_p\dot{\boldsymbol{r}}_p + {}_i\dot{\boldsymbol{r}}_{pi} \\
{}_i\ddot{\boldsymbol{r}}_i &= \boldsymbol{A}_{ip}\ {}_p\ddot{\boldsymbol{r}}_p + {}_i\ddot{\boldsymbol{r}}_{pi} \\
{}_i\boldsymbol{\omega}_i &= \boldsymbol{A}_{ip}\ {}_p\boldsymbol{\omega}_p + {}_i\boldsymbol{\omega}_{pi} \\
{}_i\dot{\boldsymbol{\omega}}_i &= \boldsymbol{A}_{ip}\ {}_p\dot{\boldsymbol{\omega}}_p + {}_i\dot{\boldsymbol{\omega}}_{pi}
\end{aligned}
\tag{3.5}
$$

using the relative kinematics given for prismatic and revolute joints in table 3.3.



(a) Prismatic joint. **(b)** Revolute joint.

Figure 3.6: Prismatic and revolute joint kinematics of a serial manipulator. Reference bodies are depicted in *grey* color. A list of the kinematic quantities is given in table 3.2.

General	
i	Index of the current body of interest
p	Index of the parent of body i
q_i	Position of joint i
${}_i\boldsymbol{v}$	Vector $\boldsymbol{v}$ is given in frame i
$\{x_0, y_0, z_0\}$	Base frame 0
$\{x_i, y_i, z_i\}$	Body-fixed frame of the i-th body
$\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\}$	Base vectors
Translational	
${}_0\boldsymbol{r}_i$	Absolute position of i-th body-fixed frame, denoted in frame 0
${}_i\boldsymbol{r}_{pi}$	Relative position of frame i to its parent p, denoted in frame i
Rotational	
$\boldsymbol{A}_{ip}$	Rotation matrix transforming a vector from a representation in frame p to a representation in frame i
${}_k\boldsymbol{\omega}_k$	Angular velocity of frame k, denoted in frame k
${}_k\boldsymbol{\omega}_{pi}$	Relative angular velocity between frame i and its parent p, denoted in frame i
Jacobians	
${}_i\boldsymbol{J}_{R,i}$	Rotational Jacobian of body i, denoted in frame i
${}_i\boldsymbol{J}_{TO,i}$	Translational Jacobian of the origin O_i of body i, denoted in frame i
${}_i\boldsymbol{J}_{T,pi}$	Translational Jacobian of the relative position vector ${}_i\boldsymbol{r}_{pi}$, denoted in frame i
${}_i\boldsymbol{J}_{R,pi}$	Rotational Jacobian of the relative position vector ${}_i\boldsymbol{r}_{pi}$, denoted in frame i

Table 3.2: Notation of relative kinematics.

	Prismatic Joint	Revolute Joint
Translational		
Position	${}_i\boldsymbol{r}_{pi} = {}_i\boldsymbol{r}_{\text{const}} + \boldsymbol{A}_{ip}^T\, \boldsymbol{e}_z\, q_i$	${}_i\boldsymbol{r}_{pi} = {}_i\boldsymbol{r}_{\text{const}}$
Velocity	${}_i\dot{\boldsymbol{r}}_{pi} = \boldsymbol{A}_{ip}^T\, \boldsymbol{e}_z\, \dot{q}_i + {}_p\tilde{\boldsymbol{\omega}}_p\, {}_i\boldsymbol{r}_{pi}$	${}_i\dot{\boldsymbol{r}}_{pi} = {}_p\boldsymbol{\omega}_p \times {}_i\boldsymbol{r}_{pi}$
Acceleration	${}_i\ddot{\boldsymbol{r}}_{pi} = \boldsymbol{A}_{ip}^T\, \boldsymbol{e}_z\, \ddot{q}_i + {}_p\dot{\tilde{\boldsymbol{\omega}}}_p\, {}_i\boldsymbol{r}_{pi} + 2\, {}_p\tilde{\boldsymbol{\omega}}_p\, {}_i\dot{\boldsymbol{r}}_{pi}$	${}_i\ddot{\boldsymbol{r}}_{pi} = {}_p\dot{\boldsymbol{\omega}}_p \times {}_i\boldsymbol{r}_{pi} + {}_p\boldsymbol{\omega}_p \times \left({}_p\boldsymbol{\omega}_p \times {}_i\dot{\boldsymbol{r}}_{pi}\right)$
Rotational		
Velocity	${}_i\boldsymbol{\omega}_{pi} = \boldsymbol{0}$	${}_i\boldsymbol{\omega}_{pi} = \boldsymbol{e}_z\, \dot{q}_i$
Acceleration	${}_i\dot{\boldsymbol{\omega}}_{pi} = \boldsymbol{0}$	${}_i\dot{\boldsymbol{\omega}}_{pi} = {}_i\boldsymbol{\omega}_i \times {}_i\boldsymbol{\omega}_{pi} + \boldsymbol{e}_z\, \ddot{q}_i$
Jacobians		
	${}_i\boldsymbol{J}_{T,pi} = \boldsymbol{A}_{ip}\left({}_p\boldsymbol{J}_{TO,p} + {}_p\tilde{\boldsymbol{r}}_{ip}\,{}_p\boldsymbol{J}_{R,p}\right) + \boldsymbol{J}_{\text{rel}}$	${}_i\boldsymbol{J}_{T,pi} = {}_p\tilde{\boldsymbol{r}}_{ip}\,{}_p\boldsymbol{J}_{R,p}$
	${}_i\boldsymbol{J}_{R,pi} = \boldsymbol{0}$	${}_i\boldsymbol{J}_{R,pi} = \boldsymbol{J}_{\text{rel}}$
	$J_{\text{rel},j,k} := \begin{cases} 1 & \text{for } j = 3 \text{ and } k = i \\ 0 & \text{else.} \end{cases}$	cf. *Prismatic Joint*

Table 3.3: Relative Kinematics equations for a body i with its parent p of a tree-like kinematic structure connected to a prismatic or revolute joint.

The tilde operator $\tilde{(\cdot)}$ is defined for a vector $\boldsymbol{a} = [a_x\ a_y\ a_z]^T$ by

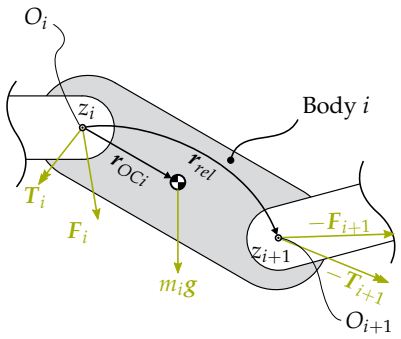
$$\tilde{\boldsymbol{a}} := \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}. \tag{3.6}$$

The translational and rotational Jacobian ${}_i\boldsymbol{J}_{T,i}$ and ${}_i\boldsymbol{J}_{R,i}$, respectively, w.r.t. the ith's body's origin are calculated by

$$\begin{aligned} {}_i\boldsymbol{J}_{TO,i} &:= \frac{\partial_i \dot{\boldsymbol{r}}_i}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{A}_{ip}\ {}_p\boldsymbol{J}_{TO,p} + {}_i\boldsymbol{J}_{T,pi} \\ {}_i\boldsymbol{J}_{R,i} &:= \frac{\partial_i \boldsymbol{\omega}_i}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{A}_{ip}\ {}_p\boldsymbol{J}_{R,p} + {}_i\boldsymbol{J}_{R,pi}. \end{aligned} \tag{3.7}$$

3.2.2 Inverse Dynamics

Considering forces and torques that are "required to cause motion" (Craig 2005, p.187), is commonly known as the dynamics of a multibody system. Generalized forces and torques corresponding to a generalized coordinate q_i are denoted by Q_i. The calculation of the robot's motion $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ based on applied forces and torques $\boldsymbol{Q}$ are denoted as forward dynamics while computing the needed forces based on a given joint trajectory $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$ is named inverse dynamics. In the following, the dynamic equations regarding the mechanical model of the manipulator is explained, while hardware specific parameters and empirical models of friction, inertia parameters and the springs for gravity compensation can be found in section 3.2.3.



O_i	Origin of body i
m_i	Mass of body i
$\boldsymbol{r}_{Ci}$	Absolute position of the i-th body's COG
$\boldsymbol{r}_{OCi}$	Relative position of the i-th body's COG w.r.t. O_i
$\ddot{\boldsymbol{r}}_{Ci}$	Absolute acceleration of COG of body i
$\boldsymbol{\omega}_i$	Angular velocity of body i
$\boldsymbol{L}_i^{\langle O_i \rangle}$	Angular momentum of body i w.r.t. O_i
$\boldsymbol{I}_i^{\langle O_i \rangle}$	Inertia tensor of body i w.r.t. O_i
$\boldsymbol{F}_{i;i+1}$	Force on body $i; i+1$ in $O_{i;i+1}$
$\boldsymbol{T}_{i;i+1}$	Torque on body $i; i+1$ in $O_{i;i+1}$

Figure 3.7: Forces and torques on body i with its COG $\boldsymbol{r}_{Ci}$ and mass m_i.

In his doctoral thesis Baur (2015) presented the algorithms for a dynamic model of the CROPS manipulator 1. In addition, this section shows parameters and models adapted and verified for the CROPS manipulator 2. In the case of tree like manipulator kinematics, i.e. structures without kinematic loops, a recursive computation of the dynamics typically reduces the computational complexity from $\mathrm{O}(n^2)$ to $\mathrm{O}(n)$ due to the required inversion of the mass matrix. Forward kinematics are calculated first for obtaining the velocities and accelerations of the respective bodies' center-of-gravity (COG) based on the given joint trajectory $\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}$. In the same way that forward kinematics start with the first body, the inverse dynamics start with the last one. This can be explained by the assumption that all torques and forces are known for the last body of the manipulator, since it has no successor.

In the following, the recursive dynamics equations for body i (parent) dependent on its child $i+1$ are presented. Fig. 3.7 shows the respective quantities for the body i with mass m_i. For rigid bodies, the COG $\boldsymbol{r}_{OCi}$ is fixed w.r.t. to the body-fixed frame O_i. The vector from O_i to the origin of its successor $i+1$ is given by $\boldsymbol{r}_{\mathrm{rel}}$. Forces and torques in O_i are denoted as $\boldsymbol{F}_i$ and $\boldsymbol{T}_i$, respectively. The inertia of body i w.r.t. O_i is denoted by $\boldsymbol{I}_i^{\langle O_i \rangle}$. The equilibria of linear ($\boldsymbol{p}$) and angular momentum ($\boldsymbol{L}_i$) result in

$$\dot{\boldsymbol{p}}_i = m_i \ddot{\boldsymbol{r}}_{Ci} = \boldsymbol{F}_i - \boldsymbol{F}_{i+1} + m_i\, \boldsymbol{g} \tag{3.8}$$

$$\dot{\boldsymbol{L}}_i^{\langle O_i \rangle} = \boldsymbol{I}_i^{\langle O_i \rangle} \boldsymbol{\omega}_i + \tilde{\omega}_i \boldsymbol{I}_i^{\langle O_i \rangle} \boldsymbol{\omega}_i = \boldsymbol{T}_i - \boldsymbol{T}_{i+1} + \underbrace{\boldsymbol{T}_{G_i}^{\langle O_i \rangle}}_{m_i \tilde{\boldsymbol{r}}_{OCi} \boldsymbol{g}} - \underbrace{\boldsymbol{T}_{\boldsymbol{F}_{i+1}}^{\langle O_i \rangle}}_{\tilde{\boldsymbol{r}}_{rel} \boldsymbol{F}_{i+1}} - m_i \tilde{\boldsymbol{r}}_{OCi} \ddot{\boldsymbol{r}}_i \tag{3.9}$$

Forces and torques of body $i+1$ ($\boldsymbol{F}_{i+1}, \boldsymbol{T}_{i+1}$) are assumed to be known due to the recursive calculation. By evaluating eq. (3.8) and eq. (3.9), forces and torques on body i ($\boldsymbol{F}_i, \boldsymbol{T}_i$) are determined. $\boldsymbol{F}_i$ and $\boldsymbol{T}_i$ can be separated into the generalized

forces and torques $\boldsymbol{Q}$ and the constrained forces/torques $\boldsymbol{Z}$.

$$\begin{bmatrix} F_{x,i} \\ F_{y,i} \\ F_{z,i} \\ T_{x,i} \\ T_{y,i} \\ T_{z,i} \end{bmatrix} = \begin{cases} \begin{bmatrix} Z_{x,i} & Z_{y,i} & Q_i & Z_{\alpha,i} & Z_{\beta,i} & Z_{\gamma,i} \end{bmatrix}^T & \text{for prismatic joints} \\ \begin{bmatrix} Z_{x,i} & Z_{y,i} & Z_{z,i} & Z_{\alpha,i} & Z_{\beta,i} & Q_i \end{bmatrix}^T & \text{for revolute joints} \end{cases} \tag{3.10}$$

While constrained forces $\boldsymbol{Z}$ are needed for dimensioning link structures, bearings or fittings, generalized forces/torques $\boldsymbol{Q}$ give information about motor torques $T_{\text{mot},i}$ taking into account friction torques (forces) $T_{f,i}$ ($F_{f,i}$) and the respective gear reduction ratio N_i. According to the motor manufacturer, the motor torque $T_{\text{mot},i}$ is proportional to its active current $I_{\text{mot},i}$ by the torque constant $k_{\text{mot},i}$. Taking into account possible forces F_{sp} from springs for gravity compensation, the following relation can be given.

$$Q_i = \begin{cases} \underbrace{N_i \, k_{\text{mot},i} I_{\text{mot},i}}_{T_{\text{mot},i}} + F_{f,i} + F_{sp} & \text{for prismatic joints and} \\ N_i \, (\underbrace{k_{\text{mot},i} I_{\text{mot},i}}_{T_{\text{mot},i}/N_i} + T_{f,i}) & \text{for revolute joints.} \end{cases} \tag{3.11}$$

Eqs. (3.8) and (3.9) form a system of linear equations in triangular form. By multiplying the system with the translational and rotational Jacobians of the body i (Newton-Euler Method), the constrained forces/torques $\boldsymbol{Z}_i$ are eliminated. Thus, the equations can be directly solved for the generalized forces and torques $\boldsymbol{Q}$.

3.2.3 Parameter Identification

This section presents empirical models for the friction, stiffness of the manipulator joints and the springs for gravity compensation of the CROPS manipulator prototype 2. Kinematic and dynamic parameters are given in appendix A.

Friction

Revolute Joints For the large and medium sized drive modules (joint 2-6) of the CROPS manipulator 2, the following friction model for revolute joints is proposed by Baur, Dendorfer, et al. (2014):

$$T_{f,i,\text{Baur}} = -\operatorname{sgn}(\dot{q}_{\text{inc},i})(\bar{T}_{f,i,0} + \bar{\mu}\,|T_{l,i}|) - \bar{b}\dot{q}_{\text{inc},i} - \frac{T_{f,S} - \bar{T}_{f,i,0}}{1 + (\frac{\dot{q}_{\text{inc},i}}{\dot{q}_S})^2} \tag{3.12}$$

Joint	$\bar{T}_{fi,0}[Nm]$	$\bar{\mu}[-]$	$\bar{b}[\frac{Nms}{rad}]$	$T_{f,S}[Nm]$	$\dot{q}_S[\frac{rad}{s}]$
2	$2.76 \cdot 10^{-1}$	$2.10 \cdot 10^{-3}$	$5.24 \cdot 10^{-4}$	$1.68 \cdot 10^{-1}$	$5.33 \cdot 10^{1}$
3-4	$2.79 \cdot 10^{-1}$	$3.75 \cdot 10^{-4}$	$4.68 \cdot 10^{-4}$	$1.28 \cdot 10^{-1}$	$8.17 \cdot 10^{1}$
5	$1.04 \cdot 10^{-1}$	$5.20 \cdot 10^{-3}$	$1.42 \cdot 10^{-4}$	$6.03 \cdot 10^{-2}$	$5.45 \cdot 10^{1}$
6	$1.36 \cdot 10^{-1}$	$4.40 \cdot 10^{-3}$	$1.95 \cdot 10^{-5}$	$4.77 \cdot 10^{-2}$	$6.52 \cdot 10^{1}$

Table 3.4: Parameters for the friction model eq. (3.13) according to Baur (2015, p.77).

Thus, the friction torque depends on the joint velocity on the motor shaft $\dot{q}_{\text{inc},i}$ and the external load $T_{l,i}$. The parameters $[\bar{T}_{f,0}, \bar{\mu}, \bar{b}, T_{f,S}, \dot{q}_S]$ were experimentally determined on a drive module testbed (cf. table 3.4). However, this model assumes the motion direction to be non-negative, i.e. $\dot{q} \geq 0$. This is corrected for arbitrary directions of $\dot{q}$ by introducing the coefficient $\text{sgn}(\dot{q}_{\text{inc},i})$. Furthermore, it is adjusted by a piecewise defined correction term χ_i for the assembled manipulator. Hence, the friction law results in:

$$\begin{aligned} T_{f,i} = &- \text{sgn}(\dot{q}_{\text{inc},i})(\bar{T}_{f,i,0} + \bar{\mu}\,|T_{l,i}|) - \bar{b}\dot{q}_{\text{inc},i} - \\ &\text{sgn}(\dot{q}_{\text{inc},i})\frac{T_{f,S} - \bar{T}_{f,i,0}}{1 + (\frac{\dot{q}_{\text{inc},i}}{\dot{q}_S})^2} - \chi_i \qquad \text{for joint 2 – 6.} \end{aligned} \tag{3.13}$$

For the small drive modules on the manipulator's wrist (joint 7-9), a model based on the gear manufacturer's catalog was developed by Buschmann (2010). Parameters $[T_{fi,0}, \mu, b_g, \gamma_f]$ of the original model were identified for the CROPS manipulator by Baur (2015) (cf. table 3.5). Furthermore, the correction term χ_i for the assembled manipulator is added as well:

$$\begin{aligned} T_{f,i} = &- \text{sgn}(\dot{q}_{\text{inc},i})(T_{fi,0} + \mu|T_{l,i}|) \\ &- (b_g + \gamma_f|T_{l,i}|)\dot{q}_{\text{inc},i} - \chi_i \end{aligned} \qquad \text{for joint 7 – 9} \tag{3.14}$$

The piecewise defined correction term χ_i with its parameters $\chi_{i,(\cdot)}$ (cf. table 3.6) is defined by

$$\chi_i = \begin{cases} \chi_{i,\dot{q}_i<0} \cdot \tanh(N_i \cdot \dot{q}_i) & \text{if } \dot{q}_i < 0 \\ \chi_{i,\dot{q}_i>0} \cdot \tanh(N_i \cdot \dot{q}_i) & \text{else.} \end{cases} \tag{3.15}$$

Joint	$T_{f,i,0}[Nm]$	$\mu[-]$	$b_g[\frac{Nms}{rad}]$	$\gamma_f[\frac{s}{rad}]$
7-9	$1.88 \cdot 10^{-2}$	$1.04 \cdot 10^{-4}$	$2.85 \cdot 10^{-5}$	$1.50 \cdot 10^{-5}$

Table 3.5: Parameters for the friction model eq. (3.14) according to Baur (2015, p. 125)

Joint	2	3	4	5	6	7	8	9
$\chi_{i,\dot{q}_i<0}$ [Nm]	+1.00	+7.00	+11.0	−5.00	−2.00	+1.75	+0.60	+0.70
$\chi_{i,\dot{q}_i<0}$ [Nm]	+1.25	+7.00	+11.0	−4.50	−2.50	+1.50	+1.00	+0.80

Table 3.6: Correction Term of the friction model for the assembled manipulator.

Prismatic Joint The CROPS manipulator 2 uses the linear axis FESTO EGC-TB for its prismatic joint. Using the model presented by Baur (2015), the friction force F_f is calculated by

$$F_f = -\operatorname{sgn}(\dot{q}_1) F_{f,0} - b\dot{q}_1 \tag{3.16}$$

with the Coulomb friction constant $F_{f,0}$ and the viscous friction gain b. Baur (2015) identified the parameters of the prismatic joint oriented horizontally without any force. This setup does not correlate to its normal operation. Therefore, these parameters are recalculated by setting the joint vertically and adding a constant mass (with a comparable weight to the manipulator arm) to joint 1. As depicted in fig. 3.8 the orientation shows a major influence on the friction parameters (cf. table 3.7).

Gravitational Force Compensation

In order to compensate for the gravitational load of the arm, eight constant force springs are attached to the prismatic joint of the CROPS manipulator 2. Two of these springs were tested individually by measuring the force while the joint 1 moves up and down. Fig. 3.9 shows typical results for two measurements. Both springs show a similar behavior, due to Coulomb friction. Their respective forces F_{sp} depend on the direction of the movement. Furthermore, the right spring shows an abnormality, which is explained by a buckling located in the spring. The average force for all measurements is $-50.8\,\mathrm{N}$ per spring.

Joint Stiffness

In practice, each joint has a limited stiffness. Hence, the difference Δq_i between the joint position on the output shaft $q_{i,\mathrm{abs}}$ and on the motor shaft $q_{i,\mathrm{inc}}$ gives information about the total acting torque T_i. While the positions on the motor and output shaft can be measured, a model for the stiffness $\tilde{c}_i$ has to be determined. This section introduces an empirical model for the joint stiffness $\tilde{c}_i$. By analogy to a torsional spring, the quantities can be written as

$$\begin{aligned} T_i &= \tilde{c}_i \, \Delta\varphi_i \\ \Delta\varphi_i &:= q_{\mathrm{abs}} - q_{i,\mathrm{inc}}. \end{aligned} \tag{3.17}$$

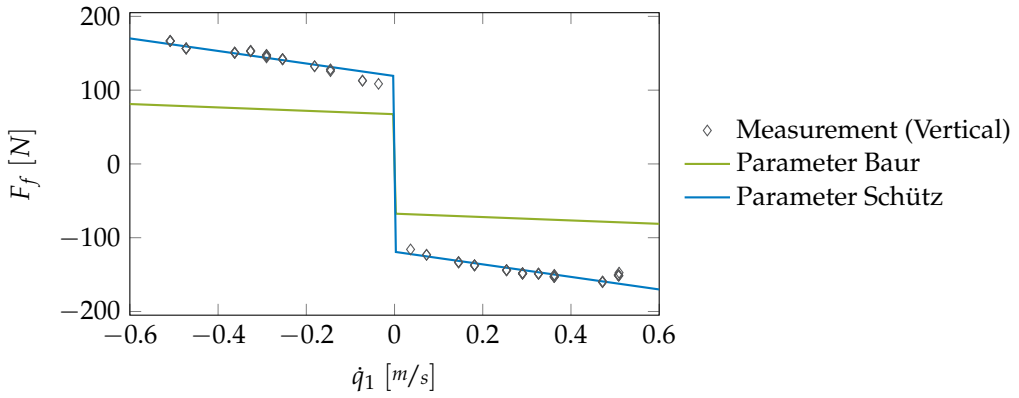


Figure 3.8: Friction parameters for the prismatic joint of the CROPS manipulator 2

Parameter	$F_{f,0}$ [N]	b [Ns/m]
Horizontal Setup (Baur)	67.41	22.97
Vertical Setup (Schütz)	119.1	85.01

Table 3.7: Friction parameter sets for prismatic joint.

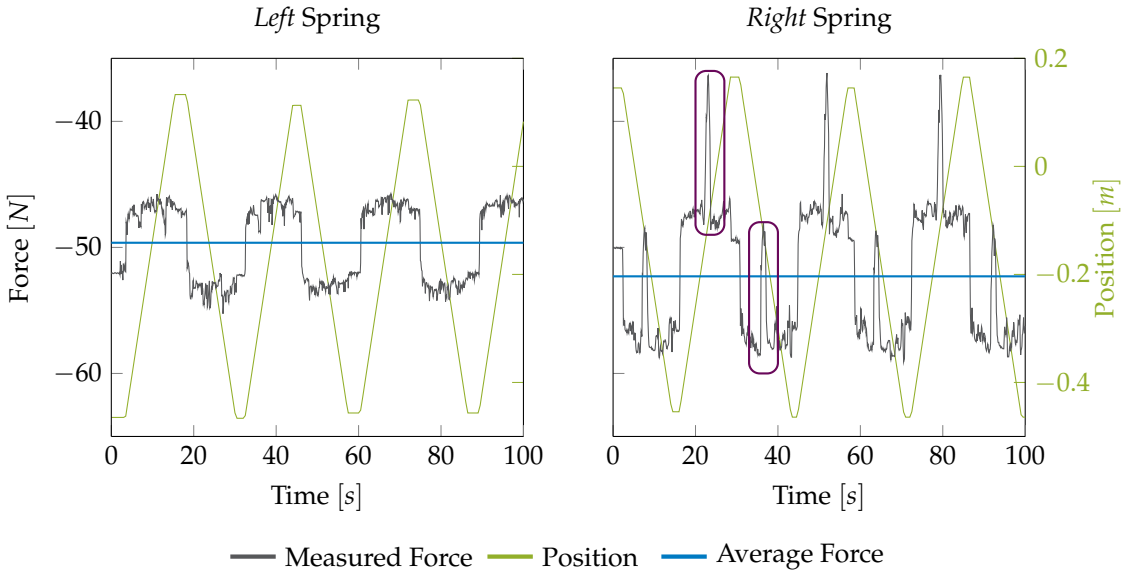


Figure 3.9: Forces of two gravitation compensation springs attached to the prismatic joint of the CROPS manipulator 2. The force F_{sp} is measured (gray) while the joint 1 moves up and down. The position is indicated in *green*. The average force of the measurement is given in *blue*. Both springs are assumed to show a similar behavior. The right spring shows an abnormality (*purple*) at one position due to a buckling in the spring.

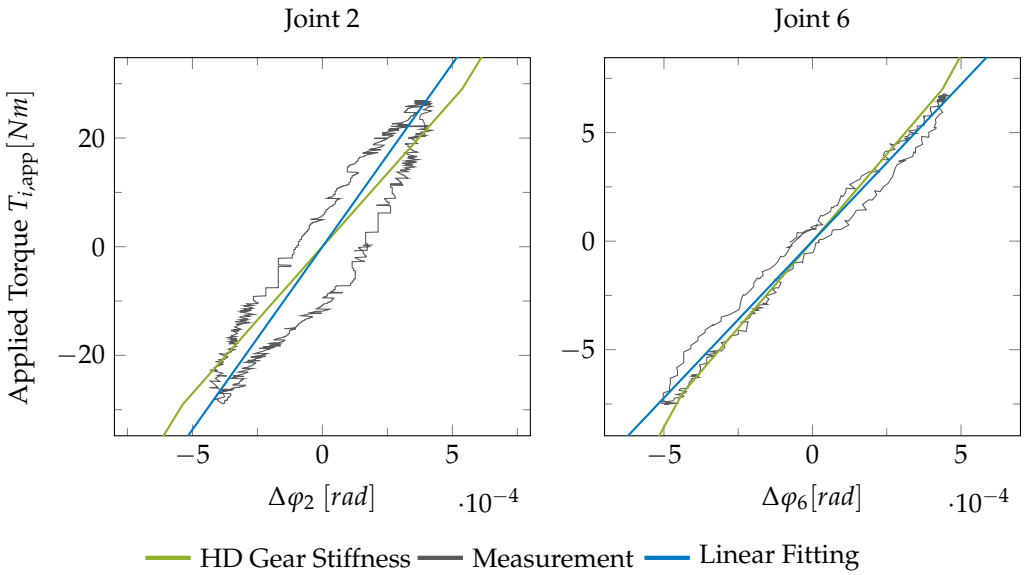


Figure 3.10: Comparison of measured stiffness (gray) from quasi-static experiment and HarmonicDrive catalog data (green). Stiffness measurements for joint 2 (large) and 6 (medium size). Maximal load in the measurements corresponds to 40% of the nominal torque of joint 2 and 30% of joint 6.

In the following, $\tilde{c}_i$ is identified in quasi-static experiments. Known forces are applied to the manipulator, which can be mapped to the single joints as generalized torques $Q_{i,\text{app}}$. Fig. 3.10 shows the results for a large drive module (joint 2) with a gear reduction of $N_2 = 50$ and for a medium size drive module (joint 6) with $N_6 = 100$. In this context, the tangent stiffness $\tilde{c}_i$ is defined as the gradient $\frac{\partial T_{i,\text{app}}}{\partial \Delta\varphi_i}$. A linear fit c_i of the measurement data is shown which is considered to be an appropriate approximation for $\tilde{c}_i$, i.e. $\tilde{c} \approx c$. Stiffness values for the gears given by the manufacturer are plotted for comparison. The measurements show a hysteresis, which is larger in joint 2 than in joint 6. Experimental data for joint 2-8 are given in appendix B. Results varied among the different measurements, their boundary values $[\bar{c}_{i,\text{min}}, \bar{c}_{i,\text{max}}]$ as well as the total average value c_i of all measurements are listed in table 3.8. Additionally, the stiffness values of the gears' manufacturer are specified, as well as the boundaries for the piecewise definition for low, medium and high loads. Note that the measured stiffness of the large modules coincides with the middle to high range of the catalog stiffnesses. The measured stiffness of medium sized modules is significantly lower compared to the catalog data.

3.2.4 Model Validation

The dynamic model is validated for each revolute joint without external loads. Fig. 3.11 compares the computed torque $T_{6,\text{int}}$ of the inverse dynamics model with the estimated torque $T_{6,\text{mot}}$ resulting from the measured motor current $I_{6,\text{mot}}$ for

		Measured Stiffness			Catalog Stiffness			Catalog Boundaries $[Nm]$	
Joint i	N_{meas}	c_i	$\bar{c}_{i,\min}$	$\bar{c}_{i,\max}$	*low*	*medium*	*high*	*low/medium*	*medium/high*
2	9	67.3	59.7	79.9	54	78	98	29	108
3	6	104	103	105	67	110	120	29	108
4	6	120	119	121	67	110	120	29	108
5	3	11.9	11.7	12.0	16	25	29	7	25
6	6	14.4	13.2	14.9	16	25	29	7	25

Table 3.8: Results of stiffness measurements for joint 2 to 6. All stiffness values are given in $[10^3 \frac{\text{Nm}}{\text{rad}}]$.

joint 6 while tracking a desired trajectory $\boldsymbol{q}_{6,\text{des},t}$. Similiar evaluations of the other revolute joints are given in appendix B.2. Evidently, the model follows the estimated motor torque except for sections with nearly zero velocity (highlighted in light gray) where a large discrepancy can be observed. Due to the high influence of stick-slip effects of the drive modules, the model gives no valid information. The bandwidth of the static friction is indicated in dark gray in fig. 3.11. Furthermore, oscillations are induced in the measured motor current by the joint controller. The root-mean square error (RMSE) for several measurements is shown for the revolute joints of the manipulator in fig. 3.12. It is calculated for joint i over the time $t^k \in [t^0, t^{\tilde{N}_i}]$ by

$$\text{RMSE}_{\text{dyn},i} = \sqrt{\frac{1}{\tilde{N}_i} \sum_{k=0}^{\tilde{N}_i} \left(T_{i,\text{int}}^k - T_{i,\text{mot}}^k \right)^2} \tag{3.18}$$

It can be seen that the absolute error for the large drive modules (joint 2-4) is higher than for the medium size (joint 5,6) and the manipulator wrist (joint 7-9). However, normalizing the RMSE to the respective nominal torque (drawn as *gray* bars in fig. 3.12), the error in the model for the large and medium sized modules ($<\approx 5\%$) is lower than for the wrist ($\approx 10\%$).

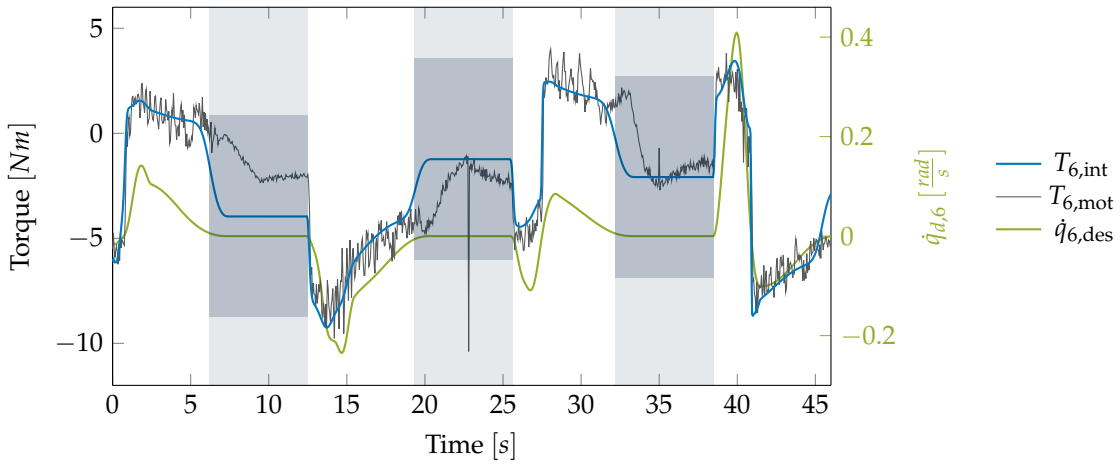


Figure 3.11: Comparison of the computed torque of the inverse dynamics model T_{int} (*blue*) and the torque T_{mot} (*black*) resulting from the measured motor current I_{mot}. Sections with a joint velocity $\dot{q}$ (*green*) close to zero are marked as *gray*. The sticking friction band is shown in *dark gray*. For comparison, the nominal torque of joint 6 is 28 Nm (Pfaff et al. 2014).

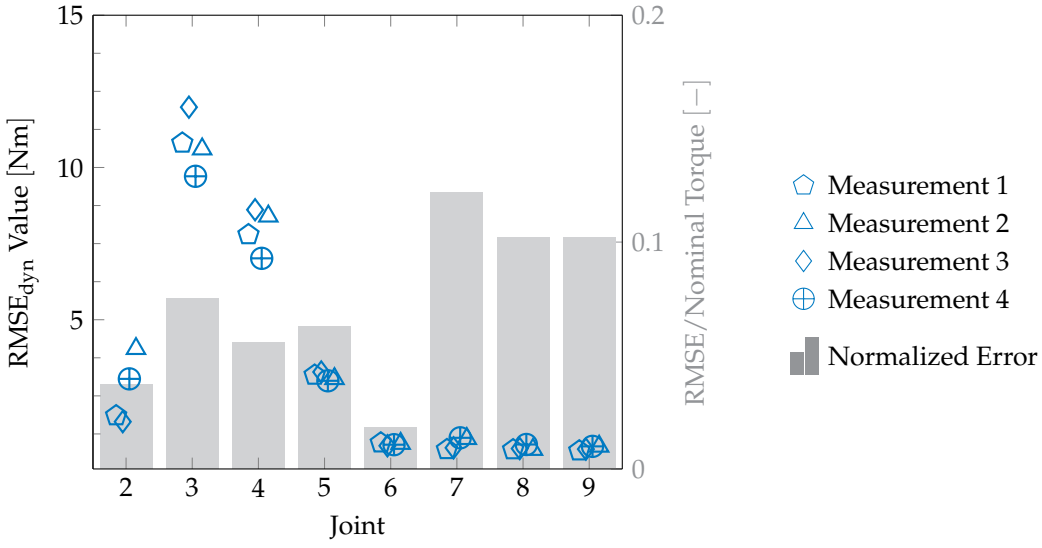


Figure 3.12: Root-mean square error of the inverse dynamics model for several tests: The computed joint torque is compared to the torque calculated based on the measured motor currents and the proportional motor constant eq. (3.11). Only intervals where the respective joints are moving are taken into account.

3.3 Tactile Perception

In order to safely plan the motion of a robot, knowledge is required about the planning scenario, i.e. the robot and its environment. Information about the robot system itself is given by its joint encoders and dynamic as well as geometrical models. By contrast, the environment can either be modeled offline or by using additional perception. Tactile sensing capabilities extend the field of applications for robot systems significantly. Obstacles may be soft or hard, movable or fixed, elastic or stiff. Vision sensors give only information about geometries. Furthermore, their field of view may be occluded in cluttered and dense environments. Mechanical properties or contact states can only be measured by force sensing capabilities. This is especially important for a robot system which interacts with its environment or other objects. Concepts for manipulation and exploration using tactile sensing in occluded areas are shown by Jain and Killpack (2013) or in the author's previous work (Schütz et al. 2015). An approach that uses the external force information within the inverse kinematics algorithm is introduced in section 4.3.

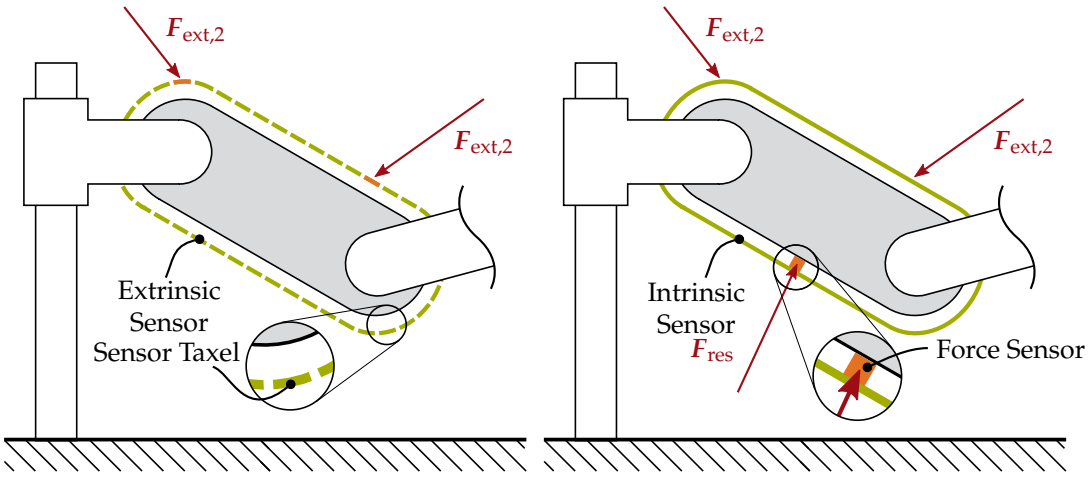
After a brief classification and overview for tactile sensing approaches, two different concepts for the measurement of external forces and torques are presented in this section which are applied to the CROPS robot: The first concept uses an additional tactile sensor attached to the manipulator arm while the second estimates external torques by proprioceptive sensor information and models.

3.3.1 Basic Concepts

This section briefly recapitulates the main concepts of existing measurement devices and sensing concepts. Detailed reviews about tactile sensors, technologies and applications are given in Cutkosky et al. (2008), Dahiya et al. (2013), Howe (1993), Lee (2000), and Lee and Nicholls (1999).

Sensing Principles & Transductions

Forces are measured by considering the deformations they induce. Most of the sensing devices available rely on the piezoresistive effect, where materials such as semiconductors or polymers change electrical resistance when mechanical stresses are applied. Using appropriate electrical circuits and materials, drift free and static measurements with high accuracies can be realized. By contrast, the piezoelectric effect causes voltages due to unbalanced charges. This effect characteristically shows drift, which is why it is mostly suited for dynamic measurements. Deformations or displacements in conductive fluids or materials cause changes of electrical impedance and/or capacity which can be measured between electrodes. Another principle is the optical detection of deformations. Hence, the



(a) Extrinsic Sensing **(b)** Intrinsic Sensing

Figure 3.13: Comparison of extrinsic and intrinsic tactile sensors.

surface of soft and easily deformable materials of known shapes is monitored e.g. by cameras.

In the following, examples for different transduction types are given. Forces can either be applied directly to the sensor by contacts with solid materials or by fluids. Zillich and Feiten (2012) presented a device using an air chamber covered with a viscoelastic material and a barometric sensor. Syntouch, LLC. offers a tactile fingertip that consists of several electrodes that measure the electrical impedance and are embedded in a conductive fluid covered with a viscoelastic skin, equipped with a pressure sensor as well as temperature sensors (Fishel et al. 2014; Fishel et al. 2008).

Design Types

Tactile sensors differ in their structural design, i.e. how they are applied to a robot system. In the literature they are commonly grouped into intrinsic and extrinsic ones (cf. fig. 3.13), an overview of both is given in the following. An early comparison of expected accuracies regarding an estimation of the contact location is presented by Son et al. (1996).

Extrinsic Sensors With extrinsic devices, force is directly applied to the sensor or sensor array, i.e. the sensor is in contact with the object (cf. fig. 3.13a). Typical examples comprise tactile skins or sensor arrays covering the surface of a robot system. A summary of methods and concepts was given by Schmitz et al. (2011). Duchaine et al. (2009) presented a pressure sensitive resistive rubber layer where resistance is measured by taxel-wise arranged plane electrodes. Bhattacharjee et

al. (2013) extended these concepts by using stretchable fabrics. Their skin was used as a tactile sleeve covering a manipulator arm in Jain and Killpack (2013) for manipulation in unstructured environments. A similar concept was shown by Strohmayr and Schneider (2013) and Strohmayr et al. (2013): Two orthogonally arranged layers with parallel conductive polymer wires define a matrix structure where mechanical stresses can be located at the respective crossing points. Classical unidirectional force as well as temperature and proximity sensors were mounted on hexagonal taxels by Mittendorfer and Cheng (2011). In Tenzer et al. (2014) barometric sensors covered by a rubber material formed a low-cost sensing array.

Intrinsic Sensors By contrast, intrinsic designs measure the resulting force (cf. fig. 3.13b): another structure (rigid or viscoelastic covering) of known geometry is in contact with the object and the force is transmitted to a sensor. The contact location is estimated by balancing forces/torques and the structure's geometry. They offer a high bandwidth and highly accurate sensor signal that can be used directly for force control. A common example are force/torques (F/T) sensors at the end effector of robot arms. This concept was first introduced by Nevins and Whitney (1979) and is widely used in industrial applications like grinding, tightening of screws, etc.. Jamisola et al. (2014) and Jain and Killpack (2013) used wrist-mounted F/T sensors for exploring unknown environments. An early application of intrinsic sensing is grasping: Bicchi et al. (1989) and Salisbury (1984) used small-sized F/T sensors with a viscoelastic fingertip for robot hands. Larger sensors were applied in the ankle of a humanoid robot for determining contact forces with the ground (Löffler et al. 2004; Lohmeier 2010). Rigid covers mounted on force sensors were presented for humanoid limbs (Iwata et al. 2001) or mobile robots (Tsuji et al. 2009). Recently, a similar concept was developed by Pfaff (2015) covering one link of the CROPS manipulator (cf. section 3.3.2).

Joint Torques Another approach related to intrinsic sensing is the calculation of external forces by measuring joint torques (Eberman and Salisbury 1990). Since geometry, kinematics and dynamics of a manipulator are assumed to be known, torques resulting from external forces can be identified. This method has the disadvantage that in some kinematic configurations, an external force may not be detectable. Joint torques can be measured directly by torque sensors within the joints (Eberman and Salisbury 1990), by considering motor currents (De Luca and Mattone 2005) or relative angular position differences (Zhang et al. 2015). An estimator for the CROPS manipulator is developed in section 3.3.3.

3.3.2 Tactile Sensor

Pfaff (2015) developed an intrinsic tactile sensor for the CROPS manipulator in collaboration with Roder (2015) and the author. This sensor is also presented in

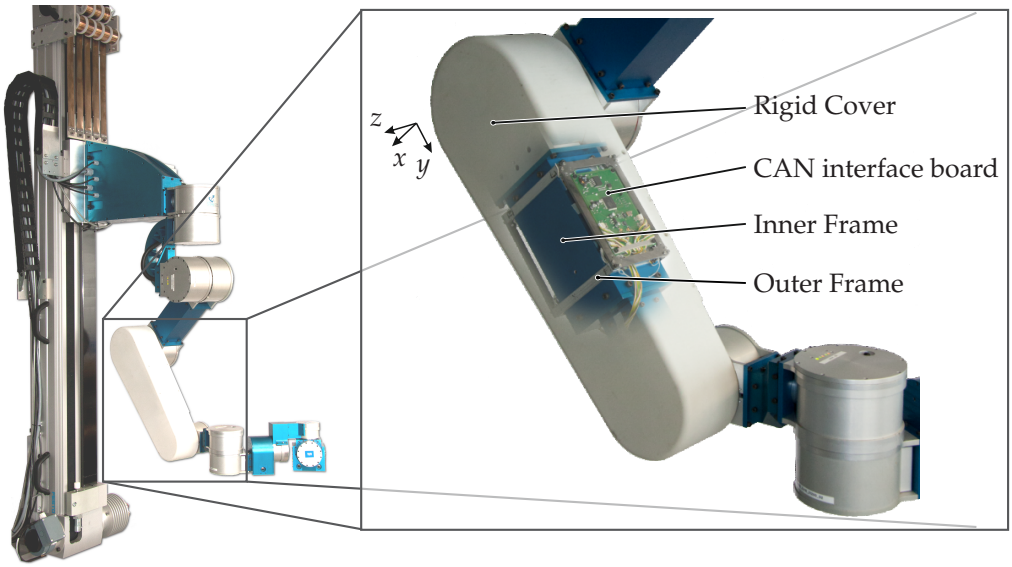


Figure 3.14: *Left:* CROPS manipulator with the tactile sensor on link 4. *Right:* Tactile sensor showing its interior design. Images taken from Schütz, Pfaff, Sygulla, et al. (2015).

a publication by the author (Schütz, Pfaff, Sygulla, et al. 2015). Its design is pictured in fig. 3.14: The rigid cover of link 4 of the manipulator houses a sensor frame, itself consisting of two rigid frames. Eight unidirectional force sensors are mounted between the outer and the inner frame. The forces/torques equilibria yield the resulting external force $\boldsymbol{F}_{\text{sen}}$. Furthermore, its theoretical application point can be estimated by knowledge of the rigid cover's geometry. The sensor signals of the force sensors are collected by the CAN interface board and are sent to the control unit via CAN bus (cf. section 3.1.3). The sensor is able to detect external forces between 1.3 N and 20 N. It provides reliable and accurate information about external forces which even enables force control based approaches as shown in section 4.3.

3.3.3 Joint Torque Estimation

This section presents a concept for estimating external torques by using only proprioceptive sensors. The approach and results are based on a former publication of the author (Schütz et al. 2016).

General Approach

In section 3.2.2, eq. (3.11), a linear relation between motor torque $T_{\text{mot},i}$ and current $I_{\text{mot},i}$ with the proportionality constant $k_{\text{mot},i}$ (motor constant) was introduced. A second relationship regarding the actual joint torque was given by eq. (3.17): by measuring the relative torsional angle $\Delta\varphi_i$ between motor ($q_{\text{inc},i}$)

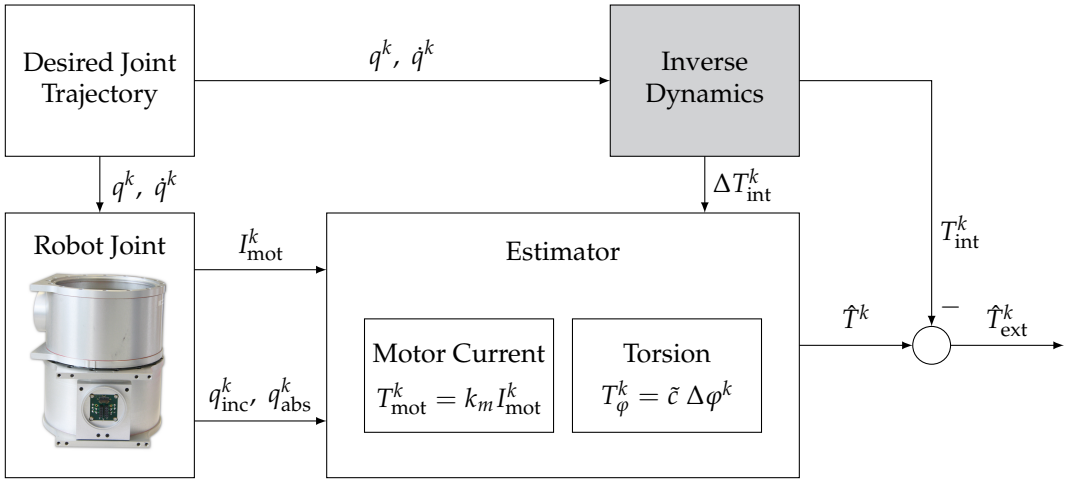


Figure 3.15: Scheme for estimating external torques of one exemplary joint module based on motor currents and relative torsional angles at time t^k.

and output shaft ($q_{\text{abs},i}$) and using the joint stiffness $\tilde{c}_i$[7], the joint torque $T_{\varphi,i}$ can be determined as well. In this section, concepts for one revolute joint i are developed. For convenience, its respective index i is omitted.

Both measurable torques T_{mot} and T_{φ} are estimates of the real joint torque T. Thus, both measurements are combined in the proposed approach aiming to achieve a more accurate estimate of the actual joint torque T. Uncertainties and simplifications originate in the modeling of the motor constant k_{mot} and the joint stiffness $\tilde{c}$. While k_{mot} is given by the motor manufacturer, the joint stiffness $\tilde{c}$ is measured and compared to the data of the gears' manufacturer in section 3.2.3. The joint torque T is the sum of an internal T_{int} and an external torque T_{ext}:

$$T := T_{\text{int}} + T_{\text{ext}} \tag{3.19}$$

The internal torque T_{int} comprises all efforts induced by the robot's motion itself. Based on the robot's trajectory $\boldsymbol{q}(t)$, it can be computed using an inverse dynamics model, which is presented in section 3.2.2. External torques T_{ext} result from the difference between joint torque T and the internal torque T_{int}. Estimated quantities are noted with circumflex as $\hat{(\cdot)}$. The concept is shown in fig. 3.15 and can be summarized as follows:

- For every time instant k, the motion planner computes a smooth desired joint trajectory $[\boldsymbol{q}^k, \dot{\boldsymbol{q}}^k]$. The trajectory is communicated to the robot joints as well as to the inverse dynamics algorithm. Accelerations are calculated by finite differences.
- The inverse dynamics module computes the internal torque T^k_{int} and its rate of change ΔT^k_{int}. These dynamics of the internal torque represent the maximum information available for the prediction of the joint torque, since the dynamics of the external torques are assumed to be arbitrary.

[7] Note that $\tilde{c}$ is assumed to be an arbitrary function describing the stiffness of the joint.

- The controllers of the drive modules control the motor current I_{mot} according to the desired trajectory. The motor current I_{mot} and the positions $q_{\text{inc}}, q_{\text{abs}}$ are measured and sent to the estimator.
- The estimator computes the joint torque $\hat{T} \approx T$ by sensor data fusion of the motor current $T_{\text{mot}}(I_{\text{mot}})$ and the relative torsional angle $T_{\varphi}(\Delta\varphi)$.
- The difference between internal torque T_{int} and estimated total torque $\hat{T}$ yields the external torque $\hat{T}_{\text{ext}}$.

Estimator Design

Based on the concept shown in fig. 3.15, an estimator for the external joint torque T_{ext} is designed as follows.

System Formulation The system equations for the estimator design are given in their discrete form at the time t^k with the time increment index k. The state space has dimension n and its system equation can be written as

$$\begin{aligned} \boldsymbol{x}^k &= \boldsymbol{A}\boldsymbol{x}^{k-1} + \boldsymbol{B}\boldsymbol{u}^{k-1} + \boldsymbol{w}^{k-1} \\ \boldsymbol{y}^k &= \boldsymbol{C}\boldsymbol{x}^k + \boldsymbol{v}^k \end{aligned} \tag{3.20}$$

using the state $\boldsymbol{x}^k \in \mathbb{R}^n$, the output (or measurement) $\boldsymbol{y}^k \in \mathbb{R}^{2n}$ and the system input $\boldsymbol{u}^k \in \mathbb{R}^n$ at t^k. The system matrix is denoted as $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, the input mapping as $\boldsymbol{B} \in \mathbb{R}^{n\times n}$ and the output matrix as $\boldsymbol{C} \in \mathbb{R}^{2n\times n}$. The vectors $\boldsymbol{w}^k \in \mathbb{R}^n$ and $\boldsymbol{v}^k \in \mathbb{R}^{2n}$ represent the process and measurement noise, respectively. $\boldsymbol{w}^k$ and $\boldsymbol{v}^k$ are assumed to behave as white noise, hence being independent of each other.

The inputs $\boldsymbol{u}$, outputs $\boldsymbol{y}$ and states $\boldsymbol{x}$ are chosen as

$$\boldsymbol{x} = \boldsymbol{T},\ \boldsymbol{u} = \Delta\boldsymbol{T}_{\text{int}},\ \boldsymbol{y} = \begin{bmatrix} \boldsymbol{I}_{\text{mot}} \\ \Delta\boldsymbol{\varphi} \end{bmatrix}. \tag{3.21}$$

The state $\boldsymbol{x}$ is is chosen as the joint torque $\boldsymbol{T}$. The system input $\boldsymbol{u}$ is defined as the change of the internal torque $\boldsymbol{T}_{\text{int}}$ between two consecutive time increments, i.e. $\Delta\boldsymbol{T}_{\text{int}}^k := \boldsymbol{T}_{\text{int}}^k - \boldsymbol{T}_{\text{int}}^{k-1}$. The system output is given by the measurement of the motor current $\boldsymbol{I}_{\text{mot}}$ and the relative torsion angle $\Delta\boldsymbol{\varphi}$. Thus, the matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ can be written as:

$$\boldsymbol{A} = \boldsymbol{I} \in \mathbb{R}^{n\times n}, \quad \boldsymbol{B} = \boldsymbol{I} \in \mathbb{R}^{n\times n}, \quad \boldsymbol{C} = \begin{bmatrix} 1/(N_1 k_{m,1}) & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & 1/(N_n k_{m,n}) \\ 1/c_1 & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & 1/c_n \end{bmatrix} \in \mathbb{R}^{2n\times n}. \tag{3.22}$$

with the gear ratios N_i, the motor constants $k_{m,i}$ and the linear approximated joint stiffness values c_i.

Kalman Filter The estimator is designed as a Kalman filter (Kálmán 1960). The Kalman filter consists of two consecutive steps in every control cycle: the prediction (noted as $(\cdot)^-$) and the correction step.

1. *Prediction:* The time update leads to the a priori estimate $\hat{\boldsymbol{x}}^{k-}$ and the a priori error covariance matrix $\boldsymbol{P}^{k-}$ using estimates from the previous time increment t^{k-1} and the process noise covariance matrix $\boldsymbol{Q}$:

$$\begin{aligned} \hat{\boldsymbol{x}}^{k-} &= \boldsymbol{A}\hat{\boldsymbol{x}}^{k-1} + \boldsymbol{B}\boldsymbol{u}^{k-1} \\ \boldsymbol{P}^{k-} &= \boldsymbol{A}\boldsymbol{P}^{k-1}\boldsymbol{A}^T + \boldsymbol{Q}. \end{aligned} \tag{3.23}$$

2. *Correction:* In the correction step, the estimate is updated to the a posteriori guess $\hat{\boldsymbol{x}}^k$ by the measurement $\boldsymbol{y}^k$ using the Kalman filter gain $\boldsymbol{K}^k$ and the measurement noise covariance $\boldsymbol{R}$. Furthermore, the a posteriori error covariance matrix $\boldsymbol{P}^k$ is calculated.

$$\begin{aligned} \hat{\boldsymbol{x}}^k &= \hat{\boldsymbol{x}}^{k-} + \boldsymbol{K}^k(\boldsymbol{y}^k - \boldsymbol{C}\hat{\boldsymbol{x}}^{k-}) \\ \boldsymbol{K}^k &= \boldsymbol{P}^{k-}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}^{k-}\boldsymbol{C}^T + \boldsymbol{R})^{-1} \\ \boldsymbol{P}^k &= \boldsymbol{P}^{k-} - \boldsymbol{K}^k\boldsymbol{C}\boldsymbol{P}^{k-} = (\boldsymbol{I} - \boldsymbol{K}^k\boldsymbol{C})\boldsymbol{P}^{k-} \end{aligned} \tag{3.24}$$

The characteristics of the Kalman filter are set up by designing the process noise covariance matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and the measurement noise covariance $\boldsymbol{R} \in \mathbb{R}^{2n\times 2n}$. The a posteriori state estimate $\hat{\boldsymbol{x}}^k$ is calculated as a combination of the a priori state estimate $\hat{\boldsymbol{x}}^{k-}$ and a weighted difference between the actual measurement $\boldsymbol{y}^k$ and the predicted measurement $\boldsymbol{C}\hat{\boldsymbol{x}}^{k-}$. The weighting matrix $\boldsymbol{K}^k \in \mathbb{R}^{n\times 2n}$ is called Kalman filter gain and $\boldsymbol{P}^k \in \mathbb{R}^{n\times n}$ is the a posteriori estimate error covariance matrix. In order to reduce computing time for the real time application, a steady state Kalman filter is used. This is motivated by the fact that both matrices $\boldsymbol{K}^k$ and $\boldsymbol{P}^k$ converge to a constant value.

The output of the Kalman filter is an estimate of the effective torque $\hat{\boldsymbol{T}} := \hat{\boldsymbol{x}}$ in the joints. In a subsequent step, the external share is calculated by the difference to the internal torques given by the inverse dynamics model:

$$\hat{\boldsymbol{T}}_{\text{ext}} = \hat{\boldsymbol{T}} - \boldsymbol{T}_{\text{int}} \tag{3.25}$$

Reference Torque For an evaluation and parametrization of the filter coefficients, knowledge about the actual external torque $\boldsymbol{T}_{\text{ref}}$ is advantageous. Thus, a variable external force $\boldsymbol{F}_{\text{ext}}$ is applied to the TCP of the CROPS manipulator, which can be mapped to the joint torques by the transposed translational Jacobian of the TCP $\boldsymbol{J}_{T,\text{tcp}}^T$ yielding the reference torque $\boldsymbol{T}_{\text{ref}}$:

$$\boldsymbol{T}_{\text{ref}} = \boldsymbol{J}_{T,\text{tcp}}^T\boldsymbol{F}_{\text{ext}} \tag{3.26}$$

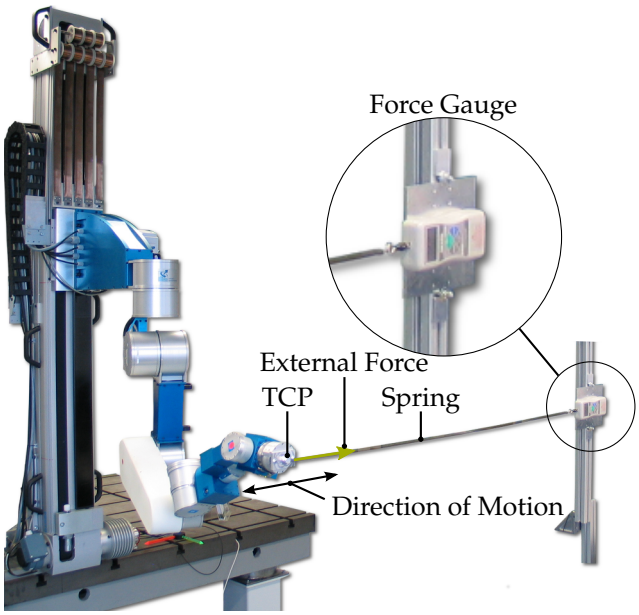


Figure 3.16: Test setup of the validation experiments of the external torque estimator. The force is applied in horizontal direction and measured by a force gauge.

The external force F_{ext} is measured by a unidirectional force gauge IMADA DS2[8], which is connected to the real time control unit for logging the measurement data. The manipulator's TCP moves on a horizontal or vertical straight line according to the measurement direction. For the horizontal measurement, the force gauge is mounted horizontally as shown in fig. 3.16. The TCP is connected to the force gauge with a spring. Thus, variable loads are applied to the manipulator's TCP while moving. By comparing the reference torque $T_{\text{ref},i}$ of the force measurement to the estimated external torque $\hat{T}_{\text{ext},i}$, the estimation error can be quantified.

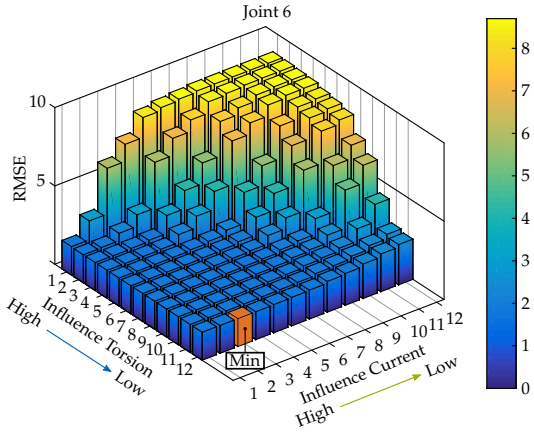
Parametrization The filter is parametrized by adjusting the process noise covariance matrix $\boldsymbol{Q}$ and the measurement noise covariance matrix $\boldsymbol{R}$. Since the model and measurement are independent for each joint, both are chosen in diagonal form. Their diagonal entries are constant and parametrized for each joint individually. The values of $\boldsymbol{Q}$ remain fixed, while the entries of $\boldsymbol{R}$ are varied. $\boldsymbol{R}$ contains separated diagonal entries for the current $\boldsymbol{R}_I$ and torsion measurement $\boldsymbol{R}_\varphi$:

$$\boldsymbol{R} := \begin{bmatrix} \boldsymbol{R}_I & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_\varphi \end{bmatrix} \tag{3.27}$$

The RMSE of the actual torque $\boldsymbol{T}_{\text{ref}}$ () and the estimated one $\hat{\boldsymbol{T}}_{\text{ext}}$ over the time $t^k \in [t^0, t^{\tilde{N}}]$ is chosen as evaluation criterion:

$$\text{RMSE}_i = \sqrt{\frac{1}{\tilde{N}} \sum_{k=0}^{\tilde{N}} (T_{\text{ref},i}^k - \hat{T}_{\text{ext},i}^k)^2} \tag{3.28}$$

[8] IMADA Inc., imada.com/products/ds2-digital-force-gauge/.



Number	Coefficient
1	0.001
2	0.01
3	0.05
4	0.1
5	0.5
6	1
7	2
8	10
9	20
10	100
11	1000
12	5000

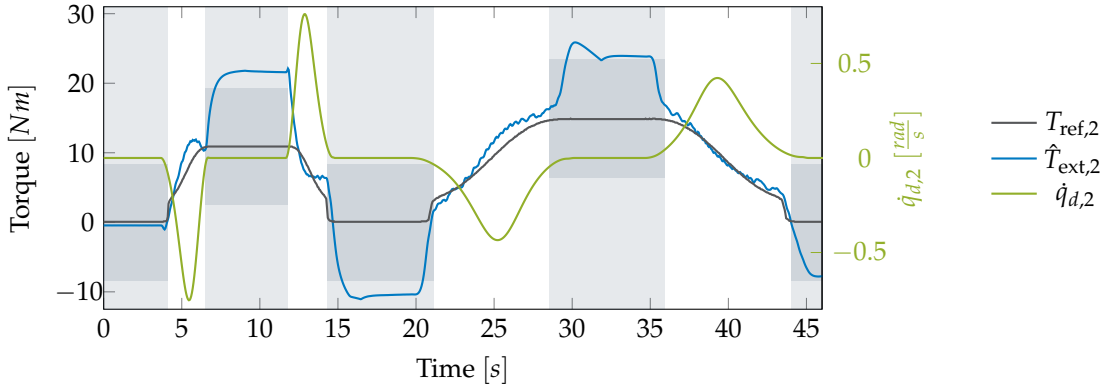
Figure 3.17: Determination of the covariance entries for $\boldsymbol{R}$ at joint 6. Measurement noise including information from the torsion and motor current is weighted relatively to the process noise ($\boldsymbol{Q}$ is kept constant). A low RMSE value signifies a low estimation error. The minimum is marked in *orange*.

Fig. 3.17 exemplifies this evaluation depicting the RMSE value for several variations of $\boldsymbol{R}$ weighting information from the relative torsion angle $\Delta\varphi$ and the motor current I_{mot} for joint 6. The results show that the measurement of I_{mot} yields more reliable information about the torque compared to the information from $\Delta\varphi$. The minimum and therefore best relation is marked in fig. 3.17.

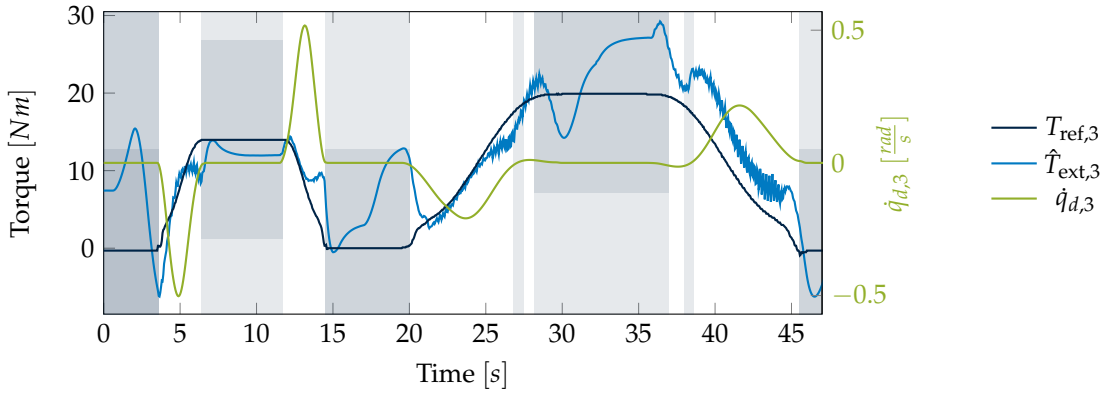
Results

Typical Result This section presents the torque estimation of three joints. The measured external forces at the TCP are mapped using eq. (3.26) to generalized reference torques $T_{\text{ref},i}$. The manipulator moves to two different goal positions on a straight line and back to its initial position, respectively. Fig. 3.18 shows results for the horizontal and vertical test setup of joint 2, 3 and 6. For results of the other joints refer to appendix B.3. Joint velocities are drawn for comparison on the right hand axis. While the estimate of the external torque shows good results during motion of the joints (white sections), the estimate differs from the applied torque at very low or zero joint velocities (gray sections). This discrepancy is mainly caused by the static friction of the drive modules, whose bandwidth is given in table 3.9 and depicted as dark gray.

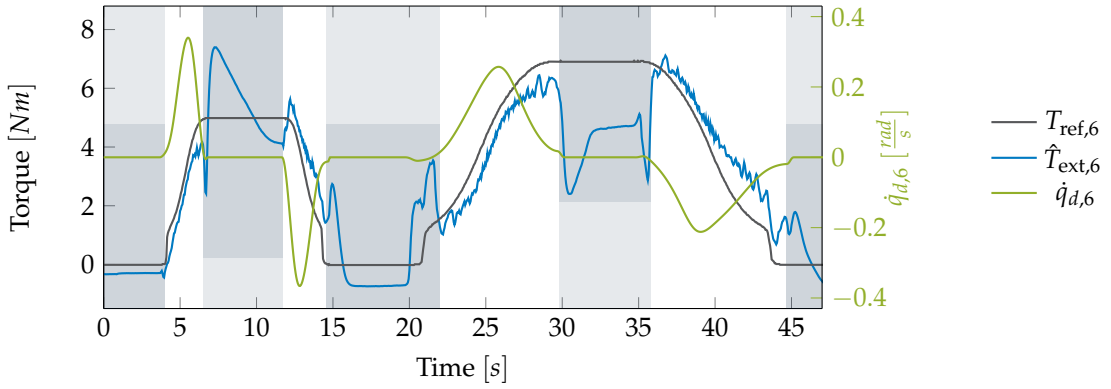
Performance Evaluation Four different experiments with different velocities and maximum forces were conducted in both horizontal and vertical directions for evaluating the developed estimation concept. Fig. 3.19 shows in *blue* the RMSE value of external torque estimation. For the evaluation, only sections with non-zero velocity of the manipulator joints (white areas) are considered. The RMSE for joint 9 is not evaluated, since no considerable loads could be applied



(a) Joint 2.



(b) Joint 3.



(c) Joint 6.

Figure 3.18: Typical results based on the test setup shown in fig. 3.16. Joint 2 and 6 are evaluated in the horizontal setup and joint 3 in the vertical setup. The signal of the force gauge mapped to the corresponding joint torques by eq. (3.26) (*black*) serves as a reference $T_{\text{ref},i}$ while the estimated torque $\hat{T}_{\text{est},i}$ is depicted in *blue*. Additionally, the respective desired joint velocity $\dot{q}_{d,i}$ is shown in *green*. Sections with joint velocities close to zero are highlighted in *gray* and the corresponding static friction band in *dark gray*.

Joint	2	3-4	5	6	7-9
Static Friction Band $[Nm]$	± 8.4	± 12.8	± 6.03	± 4.77	± 1.88

Table 3.9: Friction band of the joints.

using the chosen experimental setup. For comparison, the RMSE of the inverse dynamics model is depicted as *black* markers. This RMSE of the inverse dynamics model is calculated comparing the torque of the measured motor currents and the estimated ones while moving without external loads.The error in the estimation of $\hat{\boldsymbol{T}}_{\text{ext}}$ shows a correlation to the error of the inverse dynamics model. This can be explained by the fact that, according to eq. (3.25), the inverse dynamics model determines the ratio of internal and external torques. The normalized average torque estimation error (w.r.t. the nominal joint torques) of joint 2-6 and 8 is below 5%, only joint 7 shows a larger error (8%).

3.3.4 Comparison

In order to compare the tactile sensor (Pfaff 2015) with the proposed proprioceptive torque estimation, the measured force of the tactile sensor is mapped to generalized torques in the joints. For the comparison, the following assumptions were made:

- The threshold for the minimum detectable external force of the tactile sensor is 2 N (Schütz, Pfaff, Sygulla, et al. 2015).
- The force application point of the tactile sensor is fixed at the middle of link 4 (cf. fig. 3.20).
- For the proprioceptive torque estimation approach, the most favorable configuration of the manipulator is considered (i.e. with the maximum lever arm). This assumption yields the highest resolution of the proprioceptive sensing w.r.t. to the considered force application point at link 4. The configuration is shown in fig. 3.20a.

An external force that is applied to link 4 (cf. fig. 3.20a) results (depending on the kinematic configuration and direction of the force) in an external torque in the joints 1 to 4. In table 3.10, the minimum force threshold of the tactile sensor of $F_{th} = 2\,\text{N}$ is mapped to joint 2-4 with the maximum lever arm $d_{i,ts,\text{max}}$ and compared to the performance evaluation of the proprioceptive estimation of external torques.

By comparison to the tactile sensor presented in Schütz, Pfaff, Sygulla, et al. (2015), the detection by the proprioceptive approach is significantly poorer. One limitation is the detection of external loads while the manipulator is not moving due to the high static friction. Furthermore, the performance of the approach is

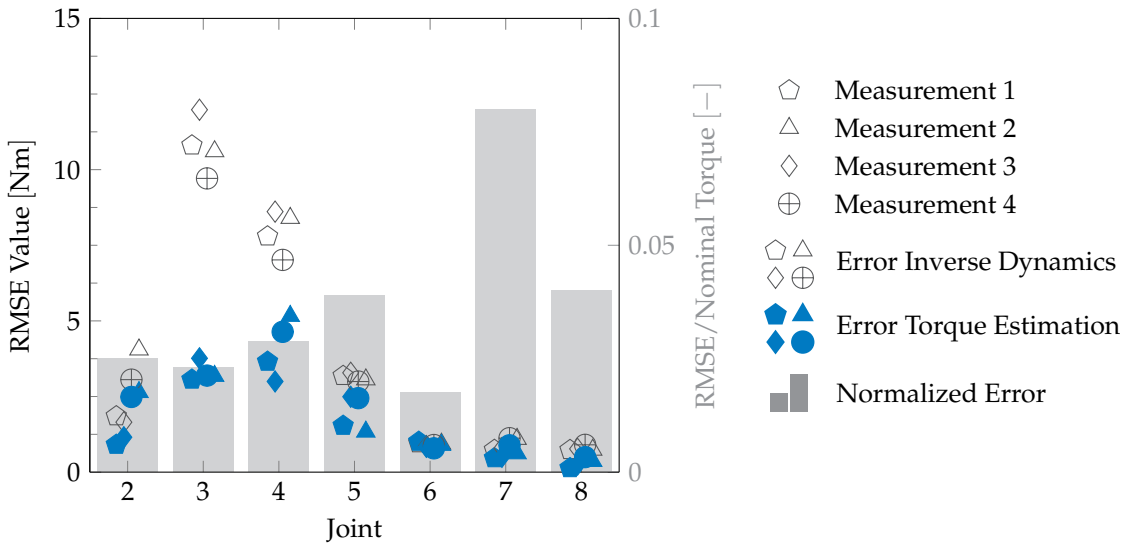
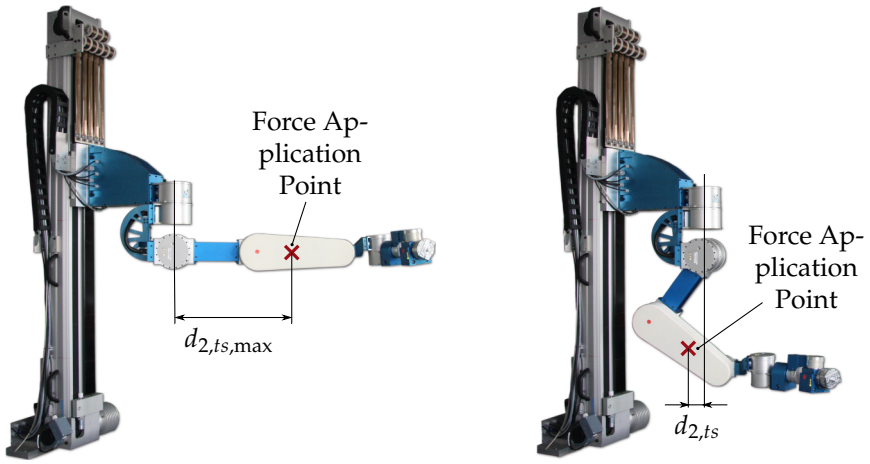


Figure 3.19: Evaluation of the proposed approach for estimating external torques. The RMSE of several experiments with external loads for joint 2-8 is plotted in *blue*. By comparison, the error of the inverse dynamics model for the same experiments without external loads are shown as *black* markers. The estimation error normalized to the nominal torque of the respective joints is given as *gray* bars.



(a) Maximum lever arm. **(b)** Small lever arm.

Figure 3.20: Lever Arm of joint 2 for the comparison of the minimum detectable torque by the tactile sensor and the proprioceptive torque estimation approach.

Joint i	$d_{i,ts,\max}$	Tactile Sensor	Proprioceptive Torque Estimation
2	0.51 m	$\leq$ 1.02 Nm	3.0 Nm
3	0.51 m	$\leq$ 1.02 Nm	7.0 Nm
4	0.16 m	$\leq$ 0.32 Nm	8.0 Nm

Table 3.10: Comparison of the minimal measurable external torque of the tactile sensor and the torque estimation approach.

intrinsically limited by the accuracy of the inverse dynamics model. Its further investigation and an improved parametrization may yield a higher detection accuracy. By contrast, it is shown in Schütz, Pfaff, Sygulla, et al. (2015) that the signal quality of the tactile sensor is sufficient for the implementation of force control related approaches.

However, external torques above a certain threshold can be detected reliably and may still be used e.g. for collision detection. The respective threshold is quantified in several experiments. Since the approach uses only sensor information that is already available, its implementation is very simple and inexpensive. By contrast to the tactile sensor, which covers only one link of the manipulator in the current setup, the torque estimator provides information about all joints without additional hardware.

Chapter 4

Inverse Kinematics

The mapping of a task space coordinate to the configuration space is referred to as the inverse kinematics (IK) problem. An introduction of the related terms can be found in previous sections of this thesis (section 2.2.1). For a thorough overview of related approaches, their theoretical background and further references refer to Buschmann (2014), Nakamura (1991), and Siciliano et al. (2009). This chapter presents common approaches and methods for solving in real time the IK with focus on redundant systems. Additionally, extensions for planning based on tactile feedback are developed.

4.1 Basic Concepts

For tree like kinematic structures, the forward kinematics $\boldsymbol{w} := \boldsymbol{\Phi}(\boldsymbol{q})$[1] can be calculated directly, e.g. by employing a recursive scheme. While the mapping $\boldsymbol{\Phi}$ is unique, its inverse $\boldsymbol{\Phi}^{-1}$ can have either no, exactly one, more than one or an infinite number of solutions. The inverse kinematics problem can be summarized as the solution of the forward kinematics equation for the configuration $\boldsymbol{q}$ at a given desired task space coordinate $\boldsymbol{w}_{\text{des}}$:

$$\boldsymbol{w}_{\text{des}} = \boldsymbol{\Phi}\left(\boldsymbol{q}\right) \tag{4.1}$$

4.1.1 Non-Redundant Systems

First, the non-redundant case, i.e. $m = n$ is considered. Since $\boldsymbol{\Phi}(\boldsymbol{q})$ is nonlinear, an analytical solution can only be found for simple or special kinematics. For the general kinematic structure, the system of nonlinear equations can be solved

[1] In this chapter, the terms *forward* and *inverse kinematics* always refer to the definition of a task space $\mathcal{W} \in \mathbb{R}^m$ as the TCP at the end effector of a manipulator with n-DOF, i.e. $\mathcal{C} \in \mathbb{R}^n$.

using numerical methods. The Newton-Raphson method provides an efficient solution on the position level. It approximates (4.1) linearly at $\boldsymbol{q}_0$ with $\boldsymbol{q} = \boldsymbol{q}_0 + \Delta\boldsymbol{q}$:

$$\boldsymbol{w}_{\text{des}} = \boldsymbol{\Phi}(\boldsymbol{q}_0 + \Delta\boldsymbol{q}) \approx \boldsymbol{\Phi}(\boldsymbol{q}_0) + \left.\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{q}}\right|_{\boldsymbol{q}_0} \Delta\boldsymbol{q} = \boldsymbol{\Phi}(\boldsymbol{q}_0) + \boldsymbol{J}_w(\boldsymbol{q}_0)\Delta\boldsymbol{q} \tag{4.2}$$

The solution is found iteratively, starting at the initial guess (or current configuration) $\boldsymbol{q}_0$ with the task space Jacobian $\boldsymbol{J}_w$ until the error $|\Delta\boldsymbol{w}|$ is smaller than ϵ. Thus, its convergence relies on a well chosen initial guess $\boldsymbol{q}_0$. An implementation is given in alg. 1.

Algorithm 1 *Newton-Raphson* IK algorithm, position level

$k \leftarrow 0$, $\boldsymbol{q}^k \leftarrow \boldsymbol{q}_0$
repeat
  $\Delta\boldsymbol{w} \leftarrow \boldsymbol{w}_{\text{des}} - \boldsymbol{\Phi}(\boldsymbol{q}^k)$
  Solve $\boldsymbol{J}_w \Delta\boldsymbol{q} = \Delta\boldsymbol{w}$
  $\boldsymbol{q}^{k+1} \leftarrow \boldsymbol{q}^k + \Delta\boldsymbol{q}$
  $k \leftarrow k + 1$
until $\|\Delta\boldsymbol{w}\| < \epsilon$
return $\boldsymbol{q}^k$

Related to the solution of the linearized IK problem on the position level, the IK problem can be solved efficiently by differentiating eq. (4.1) w.r.t. time, i.e. calculating the IK on the velocity level.

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{w}_{\text{des}}\right) = \frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{\Phi}(\boldsymbol{q})\right) = \underbrace{\frac{\partial \boldsymbol{\Phi}}{\partial \boldsymbol{q}}}_{:=\boldsymbol{J}_w} \frac{\mathrm{d}\boldsymbol{q}}{\mathrm{d}t} \tag{4.3}$$

$$\dot{\boldsymbol{w}}_{\text{des}} = \boldsymbol{J}_w \dot{\boldsymbol{q}} \tag{4.4}$$

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_w^{-1} \dot{\boldsymbol{w}}_{\text{des}} \tag{4.5}$$

Thus, a linear mapping from the $\mathcal{C}$ to the $\mathcal{W}$ space can be performed using the task space Jacobian $\boldsymbol{J}_w$. Providing the fact, that there exists a solution $\dot{\boldsymbol{q}}$ for $\boldsymbol{J}_w\dot{\boldsymbol{q}} = \dot{\boldsymbol{w}}$, the IK can be solved. Joint positions are obtained by numerical integration, e.g. the explicit Euler method.

4.1.2 Redundant Systems

For redundant systems, i.e. $n > m$, the system of equations of the IK is underdetermined and the inverse $\boldsymbol{J}_w^{-1}$ does not exist. Thus, there is in general an infinite number of solutions $\boldsymbol{q}$ fulfilling $\boldsymbol{w} \leftarrow \boldsymbol{\Phi}(\boldsymbol{q})$. Whitney (1969) proposed an optimization based scheme, denoted as resolved motion rate control (RMRC). The

IK is written as an optimization problem with the cost function l, a (diagonal) weighting matrix $\boldsymbol{W}$ and the equality constraints $\boldsymbol{g}$:

$$\min_{\dot{\boldsymbol{q}}} l_{RMRC}, \text{ s.t. } \boldsymbol{g}_{RMRC} \tag{4.6}$$

$$l_{RMRC} = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}} \tag{4.7}$$

$$\boldsymbol{g}_{RMRC} = \dot{\boldsymbol{w}}_{\text{des}} - \boldsymbol{J}_w \dot{\boldsymbol{q}} = \boldsymbol{0} \tag{4.8}$$

This problem is solved by the weighted *Moore-Penrose* pseudoinverse $\boldsymbol{J}_{\boldsymbol{W}}^{\#}$ of $\boldsymbol{J}_w$:

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{J}_{\boldsymbol{W}}^{\#} \dot{\boldsymbol{w}}_{\text{des}} \\ \boldsymbol{J}_{\boldsymbol{W}}^{\#} &= \boldsymbol{W}^{-1} \boldsymbol{J}_{\boldsymbol{w}}^T \left(\boldsymbol{J}_w \boldsymbol{W}^{-1} \boldsymbol{J}_w^T \right)^{-1} \end{aligned} \tag{4.9}$$

This scheme minimizes the (weighted) squared joint velocities locally, i.e. for every time increment, while fulfilling the task space constraint $\boldsymbol{w}_{\text{des}}$. Liégeois (1977) proposed the extension of eq. (4.6) by an additional term H for secondary objectives. Thus, the cost function l_{ASC} is extended by the change ΔH of the objective function $H(\boldsymbol{q})$ in the next time increment Δt and its linear approximation $\Delta H = \frac{\partial H}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} \Delta t$:

$$\min_{\dot{\boldsymbol{q}}} l_{ASC}, \text{ s.t. } \boldsymbol{g}_{ASC} \tag{4.10}$$

$$l_{ASC} = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{W} \dot{\boldsymbol{q}} + \alpha_N \frac{\partial H}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} \tag{4.11}$$

$$\boldsymbol{g}_{ASC} = \dot{\boldsymbol{w}}_{\text{des}} - \boldsymbol{J}_w \dot{\boldsymbol{q}} = \boldsymbol{0}$$

The solution introduced by Liégeois (1977) is known as automatic supervisory control (ASC) and can be written using the weighted nullspace projection matrix $\boldsymbol{N}_{\boldsymbol{W}}$:

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{J}_{\boldsymbol{W}}^{\#} \dot{\boldsymbol{w}}_{\text{des}} + \boldsymbol{N}_{\boldsymbol{W}} \left(\frac{\partial H}{\partial \boldsymbol{q}} \right)^T \\ \boldsymbol{N}_{\boldsymbol{W}} &:= \alpha_N \left(\boldsymbol{I} - \boldsymbol{J}_{\boldsymbol{W}}^{\#} \boldsymbol{J}_w \right) \end{aligned} \tag{4.12}$$

This scheme can be explained as follows: Since the system of equations of the IK is under-determined, its kernel or nullspace $\mathbb{N} \in \mathbb{R}^{n-m}$ can be used for secondary objectives l while respecting the primary objective, i.e. the task space constraint. Examples for secondary objective functions are given in section 4.2. While joint velocities are locally minimized by the weighted pseudoinverse $\boldsymbol{J}_{\boldsymbol{W}}^{\#}$, $\boldsymbol{N}_{\boldsymbol{W}}$ projects the gradient $\nabla_{\boldsymbol{q}} H$ into the nullspace of the task space velocity.

Another approach for solving the IK for redundant systems is the augmentation of the task space. Secondary objectives are taken into account by additional task space variables leading to a determined systems of equations with $n = m$. Details on this approach and an overview of related algorithms can be found in Siciliano (1990).

4.1.3 Acceleration Level Inverse Kinematics

By differentiating the aforementioned algorithms w.r.t. time, they can be formulated on the acceleration level as well. For some applications, this formulation is beneficial since forces and torques can be considered in a more natural way due to Newton's second law of motion. Related to the approaches presented in section 4.1.2, Luh et al. (1980) present a scheme known as resolved acceleration control (RAC). Differentiating eq. (4.4) w.r.t. time leads to

$$\ddot{\boldsymbol{w}}_{\text{des}} = \boldsymbol{J}_w \ddot{\boldsymbol{q}} + \dot{\boldsymbol{J}}_w \dot{\boldsymbol{q}} \tag{4.13}$$

By analogy to eq. (4.10) the correspondent optimal constraint problem in terms of accelerations for one time increment Δt at t^k, using the linear approximations $\Delta H = \frac{\partial H}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} \Delta t$ and $\dot{\boldsymbol{q}} = \ddot{\boldsymbol{q}} \Delta t$, is formulated as:

$$\min_{\ddot{\boldsymbol{q}}} l^k_{RAC}, \text{ s.t. } \boldsymbol{g}^k_{RAC} \tag{4.14}$$

$$\begin{aligned} l^k_{RAC} &= \frac{1}{2} \ddot{\boldsymbol{q}}^{k,T} \boldsymbol{W} \ddot{\boldsymbol{q}}^k + \alpha_N \Delta t \left. \frac{\partial H}{\partial \boldsymbol{q}} \right|_{t^k} \ddot{\boldsymbol{q}}^k \\ \boldsymbol{g}^k_{RAC} &= \ddot{\boldsymbol{w}}^k_{\text{des}} - \boldsymbol{J}_w \ddot{\boldsymbol{q}}^k - \dot{\boldsymbol{J}}_w \dot{\boldsymbol{q}}^k = \boldsymbol{0} \end{aligned} \tag{4.15}$$

is solved by the RAC algorithm:

$$\ddot{\boldsymbol{q}}^k = \boldsymbol{J}^{\#}_{\boldsymbol{W}} \left(\ddot{\boldsymbol{w}}^k_{\text{des}} - \dot{\boldsymbol{J}}_w \dot{\boldsymbol{q}}^k \right) + \boldsymbol{N}_{\boldsymbol{W}} \Delta t \left(\nabla_q H^k \right)^T. \tag{4.16}$$

This solution eq. (4.16) locally minimizes joint accelerations $\ddot{\boldsymbol{q}}^k$ at $t = t^k$ for the next time increment Δt and the secondary objectives H while respecting the task space constraint eq. (4.13).

4.1.4 Remarks on the Implementation

Numerical Drift

The IK calculation on the velocity or acceleration levels requires the numerical integration of $\dot{\boldsymbol{q}}$ or $\ddot{\boldsymbol{q}}$, respectively. A simple and well established scheme is the explicit Euler method. Joint positions $\boldsymbol{q}^{k+1}$ of a consecutive time increment t^{k+1} are calculated by

$$\begin{aligned} \boldsymbol{q}^{k+1} &= \boldsymbol{q}^k + \dot{\boldsymbol{q}}^k \, \Delta t \\ \dot{\boldsymbol{q}}^k &= \dot{\boldsymbol{q}}^{k-1} + \ddot{\boldsymbol{q}}^{k-1} \, \Delta t \end{aligned} \tag{4.17}$$

This open loop scheme shows a numerical drift, i.e. an integration error accumulates over time. This problem can be addressed by closing the loop, introducing a proportional control gain K for the integration error $\boldsymbol{w}_{\text{err}}$ in task space. K is

chosen as a diagonal matrix with scalar gains for each task space dimension. The error is calculated by the desired task space position $\boldsymbol{w}_{\text{des}}$ and the resulting position from the calculated joint positions $\boldsymbol{w}(\boldsymbol{q})$. The loop is closed by adjusting the input, the task space velocity $\dot{\boldsymbol{w}}_{\text{des}}$ or task space acceleration $\ddot{\boldsymbol{w}}_{\text{des}}$, respectively:

$$\begin{aligned} \boldsymbol{w}_{\text{err}} &:= \boldsymbol{w}_{\text{des}} - \boldsymbol{\Phi}(\boldsymbol{q}) = \boldsymbol{w}_{\text{des}} - \boldsymbol{w}(\boldsymbol{q}) \\ \dot{\boldsymbol{w}}_{\text{err}} &:= \dot{\boldsymbol{w}}_{\text{des}} - \dot{\boldsymbol{\Phi}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{\boldsymbol{w}}_{\text{des}} - \dot{\boldsymbol{w}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \dot{\boldsymbol{w}}_{\text{des,eff}} &= \dot{\boldsymbol{w}}_{\text{des}} + K\boldsymbol{w}_{\text{err}} \qquad (4.18) \\ \ddot{\boldsymbol{w}}_{\text{des,eff}} &= \ddot{\boldsymbol{w}}_{\text{des}} + K'\dot{\boldsymbol{w}}_{\text{err}} + K\boldsymbol{w}_{\text{err}} \qquad (4.19) \end{aligned}$$

This applies to eqs. (4.9), (4.12) and (4.16), i.e. the desired task space velocity $\dot{\boldsymbol{w}}_{\text{des}}$ (or acceleration $\ddot{\boldsymbol{w}}_{\text{des}}$) is replaced by the effective term $\dot{\boldsymbol{w}}_{\text{des,eff}}$ ($\ddot{\boldsymbol{w}}_{\text{des,eff}}$):

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_W^{\#}\dot{\boldsymbol{w}}_{\text{des,eff}} \qquad \text{Resolved Motion Rate Control} \qquad (4.20)$$

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_W^{\#}\dot{\boldsymbol{w}}_{\text{des,eff}} + \boldsymbol{N}_W \left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T \qquad \text{Automatic Supervisory Control} \qquad (4.21)$$

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}_W^{\#}\left(\ddot{\boldsymbol{w}}_{\text{des,eff}} - \dot{\boldsymbol{J}}_w\dot{\boldsymbol{q}}\right) + \boldsymbol{N}_W \left(\frac{\partial H}{\partial \dot{\boldsymbol{q}}}\right)^T \qquad \text{Resolved Acceleration Control} \qquad (4.22)$$

Damping Term

Joint velocities $\dot{\boldsymbol{q}}$ are not considered in the objective function eq. (4.14). Thus, in order to minimize joint velocities, an additional damping term H_{dmp} with a scalar weight $\zeta_{\text{dmp}} > 0$ has to be introduced as a secondary objective function in eq. (4.24):

$$H_{\text{dmp}} = \zeta_{\text{dmp}}\, \dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}}. \qquad (4.23)$$

Efficient Calculation of the Pseudoinverse

Another aspect relates to the calculation of the pseudoinverse $\boldsymbol{J}^{\#}$ of the Jacobian: Instead of calculating the inverse at eq. (4.9) explicitly, a system of linear equations is solved as proposed by Klein and Huang (1983). The application of this algorithm as well as the implementation of the instantaneous schemes for computing the IK regarding the CROPS manipulators is described by Baur (2015).

4.2 Secondary Objective Functions

The pseudoinverse $\boldsymbol{J}^{\#}$ minimizes joint velocities for the next timestep in an instantaneous scheme. According to the ASC scheme, other objective functions depending on $\boldsymbol{q}$, summarized as $H(\boldsymbol{q})$, are minimized by a gradient descent along

$\nabla_{\boldsymbol{q}} H$ and projected to the nullspace by the projection matrix $\boldsymbol{N}$. Arbitrary terms can be included in H: Common terms penalize joint positions that are near kinematic limits or may lead to (self-)collisions. Other examples are application specific: Schwienbacher et al. (2011) add a term for minimizing the vertical angular momentum of the biped robot LOLA. A function for increasing the manipulability in direction of desired task space motions is proposed by Baur et al. (2012) for the CROPS manipulator.

This section recapitulates the objective functions used for the CROPS manipulator and furthermore presents an extension of the joint limit avoidance by a comfort pose. The cost function H is a sum of h sub-functions H_i weighted by ζ_i:

$$H = \sum_{i=1}^{h} \zeta_i H_i(\boldsymbol{q}) \tag{4.24}$$

The following subsections present suitable functions for joint limit avoidance H_{jla} (in combination with a comfort configuration H_{cmf}) and (self-)collision avoidance H_{coll}. Note that H only depends on the configuration of the robot $\boldsymbol{q}^k$ at the current timestep t^k. For a more convenient presentation of the instantaneous calculation schemes, the notation of the current timestep is omitted, i.e. $\boldsymbol{q} := \boldsymbol{q}^k$.

4.2.1 Kinematic Configuration

In general, joint positions of a manipulator are limited by a minimum (maximum) value $\boldsymbol{q}_{\text{min,(max)}}$. To preserve the manipulator's flexibility, it is favorable to avoid configurations near these limits. This section introduces the term H_{jla} for avoiding explicitly these near-limit regions and a term H_{cmf} for a preferred comfort pose.

Comfort Pose

Many robot systems have a favorable kinematic configuration, i.e. a *comfort pose*. This pose may be defined by large kinematic reserves for each joint, e.g. in the middle position between the respective limits. Regarding a humanoid robot, the comfort pose may be specified by the definition of a natural, human-like appearance (Schwienbacher et al. 2011). Additionally, the comfort pose encounters in some way the problem with repeatability of redundant kinematics: After moving from one task space coordinate to another and back, the robot finishes in a different configuration compared to its initial one. The comfort pose always directs the robot to its favorable configuration. The comfort pose can be implemented by using a simple convex penalty term (e.g. of order 2). Weighted by the respective position range between $\boldsymbol{q}_{\text{min}}$ and $\boldsymbol{q}_{\text{max}}$, the cost function H_{cmf} increases by order 2

departing from a predefined comfort position $\boldsymbol{q}_{\text{cmf}}$. H_{cmf} and its gradient $\nabla_{\boldsymbol{q}} H_{\text{cmf}}$ for descending in the nullspace can be defined as:

$$H_{\text{cmf}} = \frac{1}{2} \sum_{i=1}^{n} \frac{(q_i - q_{\text{cmf}})^2}{(q_{i,\max} - q_{i,\min})^2} \tag{4.25}$$

$$\nabla_{\boldsymbol{q}} H_{\text{cmf}} = \left[\frac{\partial H_{\text{cmf}}}{\partial q_1} \; \frac{\partial H_{\text{cmf}}}{\partial q_2} \cdots \frac{\partial H_{\text{cmf}}}{\partial q_n} \right] ; \; \frac{\partial H_{\text{cmf}}}{\partial q_i} = \frac{q_i - q_{\text{cmf}}}{(q_{\max,i} - q_{\min,i})^2} \tag{4.26}$$

Joint Limit Avoidance

H_{jla} penalizes positions near the kinematic joint limits $\boldsymbol{q}_{\text{min(max)}}$ if a certain threshold (or *soft limit*) $\boldsymbol{q}_{\text{min(max),soft}}$ is exceeded. For the penalty function, normalized polynomial terms are considered that increase from 0 at the soft limit to 1 by the order σ:

$$H_{\text{jla}} = \sum_{i=1}^{n} H_{\text{jla},i} \tag{4.27}$$

$$H_{\text{jla},i} = \begin{cases} \frac{(q_i - q_{\min,\text{soft},i})^{\sigma}}{(q_{\min,i} - q_{\min,\text{soft},i})^{\sigma}} & \text{if } (q_i < q_{\min,\text{soft},i}) \\ \frac{(q_i - q_{\max,\text{soft},i})^{\sigma}}{(q_{\max,i} - q_{\max,\text{soft},i})^{\sigma}} & \text{else if } (q_i > q_{\max,\text{soft},i}) \\ 0 & \text{else} \end{cases} \tag{4.28}$$

The gradient $\nabla_{\boldsymbol{q}} H_{\text{jla}}$ is calculated by

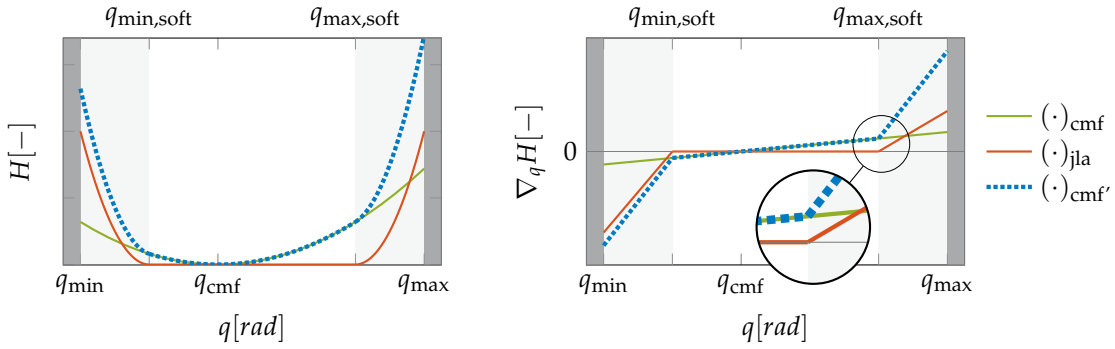
$$\nabla_{\boldsymbol{q}} H_{\text{jla}} = \left[\frac{\partial H_{\text{jla}}}{\partial q_1} \; \frac{\partial H_{\text{jla}}}{\partial q_2} \cdots \frac{\partial H_{\text{jla}}}{\partial q_n} \right]$$

$$\frac{\partial H_{\text{jla}}}{\partial q_i} = \begin{cases} \sigma \frac{(q_i - q_{\min,\text{soft},i})^{(\sigma-1)}}{(q_{\min,i} - q_{\min,\text{soft},i})^{\sigma}} & \text{if } (q_i < q_{\min,\text{soft},i}) \\ \sigma \frac{(q_i - q_{\max,\text{soft},i})^{(\sigma-1)}}{(q_{\max,i} - q_{\max,\text{soft},i})^{\sigma}} & \text{else if } (q_i > q_{\max,\text{soft},i}) \\ 0 & \text{else} \end{cases} \tag{4.29}$$

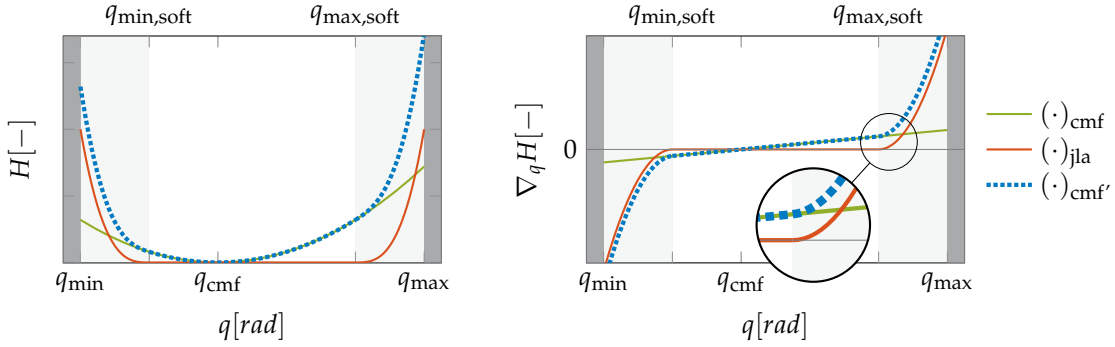
Since the gradient $\nabla_{\boldsymbol{q}} H$ is applied directly on the velocity level (cf. eq. (4.12)), a smooth transition (order 2, $\sigma = 3$) at the threshold $\boldsymbol{q}_{\text{soft}}$ is favorable since it prevents discontinuous accelerations. A comparison of a transition with order 1 ($\sigma = 2$) and order 2 ($\sigma = 3$) is shown in fig. 4.1.

Combined Term

Both aforementioned penalty terms can be combined to formulate a convex potential function $H_{\text{cmf}'}$ for each joint. While the maximum value of the joint limit



(a) H_{jla} with $\sigma = 2$.



(b) H_{jla} with $\sigma = 3$.

Figure 4.1: Cost function *(left)* and its gradient *(right)* for joint limit avoidance (H_{jla}), comfort pose (H_{cmf}) and both combined ($H_{\text{cmf'}}$) depending on the position q of a single joint. In dark gray, positions outside the limits $q_{\text{min(max)}}$ are shown while light gray areas depict the activation threshold for the soft limits $q_{\text{min(max),soft}}$. The transition at q_{soft} is highlighted: for $\sigma = 3$, the transition at the gradient is smooth by order 2.

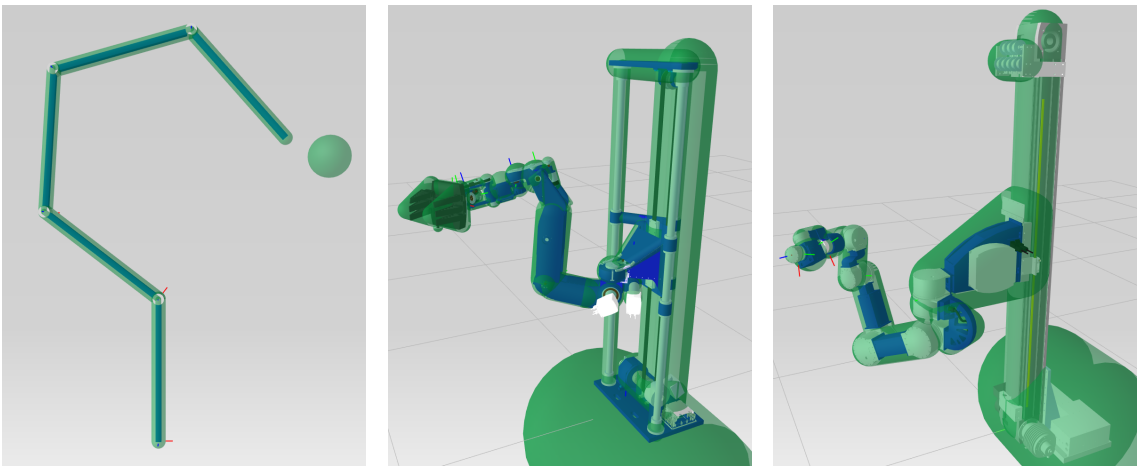
avoidance term H_{jla} is set to 1, the share of the comfort pose term can be weighted by ζ_{cmf}. Fig. 4.1 shows the objective function for different σ values.

$$H_{\text{cmf}'} = H_{\text{jla}} + \zeta_{\text{cmf}} H_{\text{cmf}} \tag{4.30}$$

4.2.2 Collision Avoidance

The secondary objective function for (self-)collision avoidance H_{coll} is based on the work of Schwienbacher (2013) and Schwienbacher et al. (2011) and was applied by Baur (2015) to the CROPS manipulators.

In order to simplify the collision model, the robot's geometry is approximated by swept-sphere volumes (SSVs). The three elementary SSV types are a point, a line and a triangular segment. Those are "inflated" by a certain radius to the respective 3D object. At every time increment, forward kinematics of the SSV bodies according to the current configuration $\boldsymbol{q}$ are calculated and the smallest



(a) 4-DOF pendulum. **(b)** CROPS prototype 1. **(c)** CROPS prototype 2.

Figure 4.2: Collision models of different robots appearing in this thesis based on the work of Baur (2015) and Schwienbacher (2013). Robots' geometries from CAD data are modeled by swept sphere volumes (transparent green capsules).

distance d between pre-defined collision pairs is determined. The approximation of both CROPS manipulators and the simplified example of a 4-DOF pendulum as used in this thesis are depicted in fig. 4.2.

If $d(\boldsymbol{q})$ exceeds the activation threshold d_a, the cost function H_{coll} increases by order 3 up to a maximum value $H_{\text{coll,max}}$ at zero distance $d(\boldsymbol{q}) = 0$. $H_{\text{coll}}\left[d(\boldsymbol{q})\right]$ is shown in fig. 4.3[2].

$$
\begin{aligned}
H_{\text{coll}} &= \begin{cases} \frac{s_0}{3}\left(d_a - d(\boldsymbol{q})\right)^3 & \text{if } d(\boldsymbol{q}) < d_a \\ 0 & \text{else} \end{cases} \\
s_0 &= 3\frac{H_{\text{coll,max}}}{d_a^3}
\end{aligned}
\tag{4.31}
$$

[2] A similar cost function using a piecewise defined function was presented by Schwienbacher (2013) and used by Baur (2015).

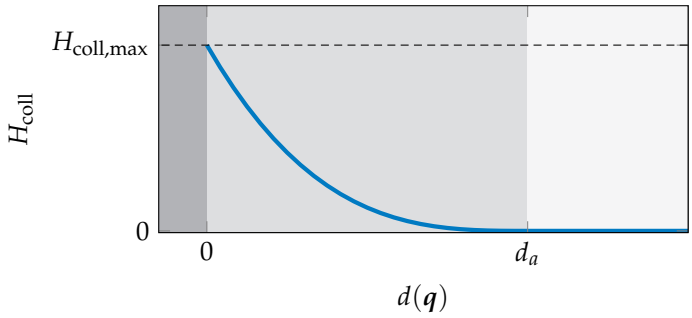
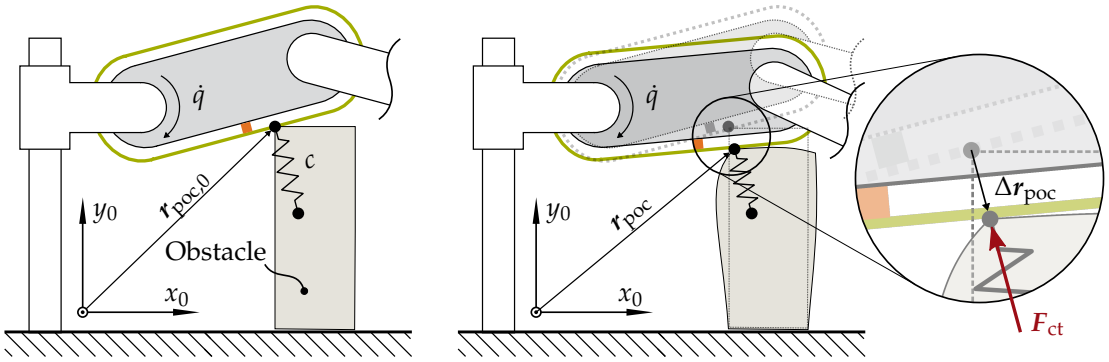


Figure 4.3: Cost function H_{coll} over the minimum distance of the nearest collision pair $d(\boldsymbol{q})$.

4.3 Incorporating Tactile Feedback

The aforementioned concepts provide efficient solutions for the inverse kinematics problem. Knowledge about the planning problem itself, such as kinematics, limits or collision geometries, is considered to be static and available in advance. This section extends these methods by also considering tactile sensor information in real time. The approaches and results shown in this section were presented by the author (Schütz, Pfaff, Sygulla, et al. 2015; Sygulla et al. 2016). The planning concepts were developed, implemented and investigated by Felix Sygulla in his master's thesis (Sygulla 2015).

In scenarios where the robot acts in uncertain environments, e.g. natural surroundings such as greenhouses or orchards, information about the planning scene is not sufficient: obstacles are occluded or all paths to the goal may be obstructed, i.e. collision-free paths may not even exist. Humans and animals use their sense of touch in addition to visual information: By getting in contact with the environment, they explore and characterize its varying mechanical properties. Using this information, they are able to reach even obstructed goal regions by pushing other objects away. Transferring this concept to robot systems seems obvious. Equipped with a tactile sensor, the robot measures contact forces with the environment and adapts its motion plan based on the tactile information. Examples of different tactile sensor concepts are given in section 3.3.1. An incentive application is shown in fig. 2.5b. A manipulator system harvests sweet peppers in a greenhouse while dense foliage occludes fruit and obstacles. Thus, visual information is limited. The manipulator has to push leaves away while it must not damage stems or collide with cultivation fixtures. The adaption of the plan has to be done in real time, since information from tactile sensors is not predictable. Typically, the contact force as well as the application point are measured by a tactile sensor as presented in section 3.3. While small external forces are admissible, the robot has to counteract increasing forces with an increasing effort. High loads have to be avoided in order to prevent damage to the plant.



(a) In contact: No deformation. **(b)** In contact: Obstacle is deformed.

Figure 4.4: The robot touches an obstacle. The contact between robot and environment is modeled as a linear elastic spring with a finite stiffness c. Fig. 4.4a shows the moment when the robot touches the obstacle and the contact force is zero ($\boldsymbol{F}_{ct} \approx 0$). At this point $\boldsymbol{r}_{poc,0}$, the spring is unstressed. Fig. 4.4b depicts the deformed state: The spring is compressed which results in a contact force $\boldsymbol{F}_{ct} > 0$. The small displacement $\Delta\boldsymbol{r}_{poc}$ of the contact point $\boldsymbol{r}_{poc}$ is assumed to be linear. Note that this drawing depicts a very low stiffness of the environment compared to the robot: In general, the stiffness c describes the contact between robot and environment, which is influenced by the robot *and* the environment.

4.3.1 Gradient Based Input

The ASC scheme presented in section 4.1.2 is suited to take into account additional sensor inputs in terms of secondary objectives. By introducing a new potential H_e for the contact of the robot with the environment, its gradient can be used for its instantaneous minimization. A simple assumption is for H_e to be a linear elastic potential of a spring according to Hooke's law while neglecting all tangential forces in the contact point $\boldsymbol{r}_{poc}$[3]. The contact force $\boldsymbol{F}_{ct}$ applied at $\boldsymbol{r}_{poc}$ increases linearly with the constant stiffness c by translational displacement of the contact point $\Delta\boldsymbol{r}_{poc}$ w.r.t. to the initial (unstressed) position $\boldsymbol{r}_{poc,0}$. The considered quantities are depicted in fig. 4.4.

$$\Delta\boldsymbol{r}_{poc} = \boldsymbol{r}_{poc} - \boldsymbol{r}_{poc,0} \tag{4.32}$$

$$H_e = \frac{1}{2} c\, \Delta\boldsymbol{r}_{poc}^T \Delta\boldsymbol{r}_{poc} \tag{4.33}$$

Without contact, the potential H_e and $\boldsymbol{F}_{ct}$ vanish. In contact, H_e can be interpreted as a repellent potential causing a force $\boldsymbol{F}_{ct}$ to the robot. The gradient $\frac{\partial H_e}{\partial \boldsymbol{q}}$ is calcu-

[3]The point of contact $\boldsymbol{r}_{poc}$ is assumed to be fixed w.r.t. the surface of the deformed environment. This simplification seems applicable, since only small deformations are expected to occur for each time increment Δt.

lated as follows:

$$\frac{\partial H_e}{\partial \boldsymbol{q}} = \begin{cases} \underbrace{c\,\Delta \boldsymbol{r}_{\text{poc}}}_{-\boldsymbol{F}_{\text{ct}}^T} \cdot \underbrace{\frac{\partial \boldsymbol{r}_{\text{poc}}}{\partial \boldsymbol{q}}}_{\boldsymbol{J}_{\text{poc}}} = -\boldsymbol{F}_{\text{ct}}^T \boldsymbol{J}_{\text{poc}}, & \forall \boldsymbol{F}_{\text{ct}} \geq \boldsymbol{0} \\ \boldsymbol{0} & \text{else.} \end{cases} \tag{4.34}$$

The gradient $\nabla_{\boldsymbol{q}} H_e$ of the linear elastic potential corresponds to the contact (spring-like) force $\boldsymbol{F}_{\text{ct}}$ multiplied by its effective direction $\frac{\partial \boldsymbol{r}_{\text{poc}}}{\partial \boldsymbol{q}}$ in $\mathcal{C}-$space. This direction is determined by the robot's Jacobian $\boldsymbol{J}_{\text{poc}}$ at $\boldsymbol{r}_{\text{poc}}$. One can now take advantage of the measured contact force signal $\boldsymbol{F}_{\text{ct,sen}}$ assuming $\boldsymbol{F}_{\text{ct,sen}} \approx \boldsymbol{F}_{\text{ct}}$ and include $\boldsymbol{F}_{\text{ct,sen}}$ in the control equation by

$$\nabla_{\boldsymbol{q}} H_e \approx -\boldsymbol{F}_{\text{ct,sen}}^T \boldsymbol{J}_{\text{poc}} \tag{4.35}$$

$\nabla_{\boldsymbol{q}} H_e$ can be included in the ASC scheme eq. (4.12) or to RAC eq. (4.16) analogously to other secondary objective functions H with the scalar weight ζ_e:

$$\left(\frac{\partial H'}{\partial \boldsymbol{q}}\right)^T \approx \left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T + \zeta_e \boldsymbol{J}_{\text{poc}}^T \boldsymbol{F}_{\text{ct,sen}}. \tag{4.36}$$

For multiple contact scenarios at the robot arm, sub-potential functions $H_{e,i}$ can be defined for each tactile sensing module or measurable resulting force, respectively. The overall potential for multiple contacts $H_{e,\text{mult}}$ is calculated as the sum of $H_{e,i}$ with a total number of considered contact forces k_{ct}:

$$H_{e,\text{mult}} = \sum_{i=1}^{k_{\text{ct}}} H_{e,i}. \tag{4.37}$$

This approach provides a simple and intuitive reduction of contact forces by their projection to $\mathcal{C}-$space that corresponds to external torques in the joints. While this approach is comparable to an admittance control scheme, the proposed concept differs by the projection of these torques to the nullspace. The robot's motion leads to a minimization of a Hooke-like contact potential similar to other secondary objective functions (cf. section 4.2) without interfering the task space constraint.

4.3.2 Controller Based Approach

Besides the simple gradient based approach, control theory techniques can also be used to reduce the contact forces. These approaches were developed and extensively studied in the master's thesis of Sygulla (2015). The following section recapitulates briefly the main ideas for the adaption in nullspace. After adding a new input to the system, a projection of the input vector along the effective force direction is introduced, yielding a scalar control equation. While the resulting controller can only influence the nullspace due to the system formulation, an outlook to extended formulations is given in section 4.3.3.

Scalar Contact Force Dynamics

First, the system input $\boldsymbol{u}_{n,j}$ is added to the system equation based on the ASC scheme eq. (4.12):

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_W^{\#}\dot{\boldsymbol{w}}_{\text{des,eff}} + \frac{1}{\alpha_N}\boldsymbol{N}_W\left[\alpha_N\nabla_{\boldsymbol{q}}H^T + \boldsymbol{u}_{n,j}\right] \tag{4.38}$$

Note that the input $\boldsymbol{u}_{n,j}$ has $\dim(\mathcal{C})$. In order to reduce the complexity of the controller design, the system equation is projected to the one-dimensional space along the contact force (denoted as *contact space*). First, the direction of the contact force applied at the point of contact $\boldsymbol{r}_{\text{poc}}$ is normalized by

$$\boldsymbol{f}_{\text{ct}} := \frac{\boldsymbol{F}_{\text{ct}}}{\|\boldsymbol{F}_{\text{ct}}\|}. \tag{4.39}$$

The n-dimensional input $\boldsymbol{u}_{n,j}$ can be projected in the effective direction of the contact force by the Jacobian $\boldsymbol{J}_{\text{poc}}$ of contact point $\boldsymbol{r}_{\text{poc}}$ and furthermore to its scalar quantity u_n by the effective force direction $\boldsymbol{f}_{\text{ct}}$:

$$\begin{aligned}\boldsymbol{u}_n &:= \boldsymbol{J}_{\text{poc}}\boldsymbol{u}_{n,j}\\ u_n &:= \boldsymbol{f}_{\text{ct}}^T\boldsymbol{u}_n = \boldsymbol{f}_{\text{ct}}^T\boldsymbol{J}_{\text{poc}}\boldsymbol{u}_{n,j}\end{aligned} \tag{4.40}$$

The velocity of the contact point $\boldsymbol{r}_{\text{poc}}$ with the input $\boldsymbol{u}_{n,j}$ are derived by

$$\dot{\boldsymbol{r}}_{\text{poc}} = \frac{\partial \boldsymbol{r}_{\text{poc}}}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}} = \boldsymbol{J}_{\text{poc}}\dot{\boldsymbol{q}} = \boldsymbol{J}_{\text{poc}}\left[\boldsymbol{J}_W^{\#}\dot{\boldsymbol{w}}_{\text{des,eff}} + \frac{1}{\alpha_N}\boldsymbol{N}_W\left(\alpha_N\nabla_{\boldsymbol{q}}H^T + \boldsymbol{u}_{n,j}\right)\right] \tag{4.41}$$

using the robot's Jacobian $\boldsymbol{J}_{\text{poc}}$. The secondary objective gradients $\nabla_{\boldsymbol{q}}H$ are considered as superposed dynamics and therefore omitted for the controller design. This assumption, together with the projection of the system input (4.40) and the multiplication of eq. (4.41) with the transpose of the normalized effective force direction $\boldsymbol{f}_{\text{ct}}^T$ leads to

$$\begin{aligned}\underbrace{\boldsymbol{f}_{\text{ct}}^T\dot{\boldsymbol{r}}_{\text{poc}}}_{\dot{x}_{\text{poc}}} &= \boldsymbol{f}_{\text{ct}}^T\boldsymbol{J}_{\text{poc}}\boldsymbol{J}_W^{\#}\dot{\boldsymbol{w}}_{\text{des,eff}} + \boldsymbol{f}_{\text{ct}}^T\boldsymbol{J}_{\text{poc}}\boldsymbol{N}_W\boldsymbol{u}_{n,j}\\ &= \underbrace{\boldsymbol{f}_{\text{ct}}^T\boldsymbol{J}_{\text{poc}}\boldsymbol{J}_W^{\#}\dot{\boldsymbol{w}}_{\text{des,eff}}}_{\dot{x}_{\text{poc,des,eff}}} - \underbrace{\boldsymbol{f}_{\text{ct}}^T\boldsymbol{J}_{\text{poc}}\boldsymbol{N}_W\boldsymbol{J}_{\text{poc}}^T\boldsymbol{f}_{\text{ct}}}_{k_n(t)}u_n\end{aligned} \tag{4.42}$$

and can be written as the scalar equation of the contact point dynamics in contact space

$$\begin{aligned}\dot{x}_{\text{poc}} &= \dot{x}_{\text{poc,des,eff}} - k_n(t)u_n\\ k_n(t) &= \boldsymbol{f}_{\text{ct}}^T\boldsymbol{J}_{\text{poc}}\boldsymbol{J}_W^{\#}\dot{\boldsymbol{w}}_{\text{des,eff}}.\end{aligned} \tag{4.43}$$

The projected dynamics of the contact point to a scalar equation in contact space $\dot{x}_{\text{poc}}$ are used to formulate the dynamics of the scalar contact force F_{ct} according to Hooke's law. Assuming a discrete displacement Δx_{poc} in contact space of the contact point $\boldsymbol{r}_{\text{poc}}$ and a constant stiffness c, the force changes[4] by

$$\Delta F_{\text{ct}} = -c \cdot \Delta x_{\text{poc}}. \tag{4.44}$$

For a finite timestep Δt, the discrete approximation $\frac{\mathrm{d}}{\mathrm{d}t}(\cdot) \approx \Delta(\cdot)$ holds. Thus, the dynamics of the contact force F_{poc} in contact space are described by

$$\dot{F}_{\text{poc}} := -c\dot{x}_{\text{poc,des,eff}} + c\, k_n(t) u_n. \tag{4.45}$$

Controller Design

By applying the concept of feedback linearization, a control input u_n can be designed such that a control variable follows specified target dynamics $\Xi(t)$. Linear dynamics with a second order decrease, parametrized by the time constant T_Ξ and the damping factor d_Ξ, are a possible choice:

$$\ddot{\Xi} = -\frac{2d_\Xi}{T_\Xi}\dot{\Xi} - \frac{1}{T_\Xi^2}\Xi \tag{4.46}$$

Equalizing eq. (4.45) with the integrated term of eq. (4.46) and setting $\dot{\Xi} \stackrel{!}{=} \dot{F}_{\text{poc}}$ determines the control equation

$$u_n = \frac{1}{k_n(t)}\dot{x}_{\text{poc,des,eff}} - \frac{2d_\Xi}{T_\Xi\, c\, k_n(t)}F_{\text{ct}} - \frac{1}{T_\Xi^2\, c\, k_n(t)}\int_0^t F_{\text{ct}}\,\mathrm{d}\tau. \tag{4.47}$$

Inserting eq. (4.47) into eq. (4.38) and using the measured contact force $\boldsymbol{F}_{\text{sen}}$ as an estimate of $\boldsymbol{F}_{\text{ct}}$ yields the control law for following the target dynamics $\Xi(t)$. The controller itself consists of a proportional and an integral part which depend on the current force $\boldsymbol{F}_{\text{ct}}$. The first part includes the desired task space velocity of the contact point $\dot{x}_{\text{poc,des,eff}}$, trying to avoid movements of the robot against the obstacle as feed forward term.

4.3.3 Modification of the Task

The system formulation of the previous sections restricts the adaption of a robot's motion plan to its nullspace, i.e. the task space constraint has to be fulfilled. This limitation may yield forces which exceed the maximum allowed load in case a reduction within the nullspace is no longer feasible. Extensions to the ASC algorithm by further inputs for the task space $\boldsymbol{u}_{t,t}$ or joint space $\boldsymbol{u}_{j,c}$, as presented

[4] The *minus* in the equation results from the definition that movement x_{poc} causes a force F_{poc} in its opposite direction.

in Schütz, Pfaff, Sygulla, et al. (2015), allow a deformation of the task space constraints:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_W^{\#} \left(\dot{\boldsymbol{w}}_{\text{des,eff}} - \boldsymbol{u}_{t,t} \right) - \boldsymbol{u}_{j,c} - \boldsymbol{N}_W \left[\boldsymbol{u}_{n,c} + \left(\frac{\partial H^T}{\partial \boldsymbol{q}} \right) \right] \tag{4.48}$$

Using the inputs $\boldsymbol{u}_{t,t}$ or $\boldsymbol{u}_{j,c}$, the motion planner is no longer restricted to the nullspace and may override the task space constraint. However, one should note that the compensation of the numerical drift as presented in section 4.1.4 "pulls" the end effector back to the task space trajectory. Control laws for $\boldsymbol{u}_{t,t}$ and $\boldsymbol{u}_{j,c}$ can be derived according to $\boldsymbol{u}_{n,j}$ as presented in the previous section. Results and further remarks regarding these controllers are presented in Sygulla (2015) and Sygulla et al. (2016).

Based on the idea to adapt the task space trajectory only in scenarios where the nullspace is exploited, two controller inputs can be used in the same control scheme. This approach is denoted as the *multi-space tactile feedback* and is reported in Sygulla (2015) and Sygulla et al. (2016). It combines e.g. the task space controller with the nullspace control input (*multi-space/task*) in a hierarchical manner: Only above a certain force threshold F_{act} is the task space controller activated and adapts $\boldsymbol{w}_{\text{des}}(t)$. Notably, the combination nullspace/task space by contrast to nullspace/joint space leads to a more comprehensible behavior since both spaces are orthogonal.

4.3.4 Results

The proposed approaches were extensively tested in several variations and evaluated in Sygulla (2015). Some significant results are presented in this section to show their capabilities and benefit.

Empirically Motivated Extensions

The implementation of the aforementioned algorithms on a real system requires empirical extensions that are presented in the following.

Closing a control loop usually needs signal processing on the sensor outputs. The tactile sensor module presented in section 3.3.2 provides a force signal $\boldsymbol{F}_{\text{sen}} \in \mathbb{R}^3$ and an estimate of the contact point with the environment $\boldsymbol{r}_{\text{poc,sen}}$. Especially the feedback linearization controller is highly sensitive to the force direction $\boldsymbol{f}_{\text{ct}}$ and the point of contact $\boldsymbol{r}_{\text{poc}}$. Both directly influence the projection of the equations to the contact space and the control law. Since experiments showed that the measured force direction $\boldsymbol{f}_{sen}$ as well as $\boldsymbol{r}_{\text{poc,sen}}$ are affected by tangential (friction induced) forces and other disturbances, three simplifying assumptions had to be made for robust behavior:

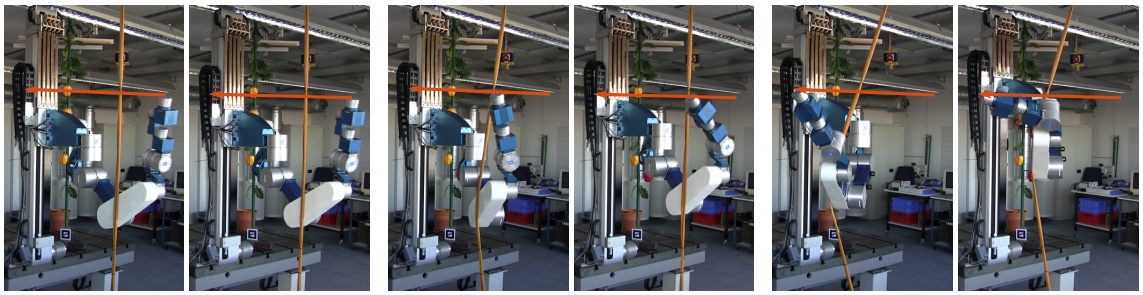


Figure 4.5: Test Setup for the proposed tactile feedback control approaches. The 9-DOF CROPS manipulator hits a vertical latex band at link 4 following a straight-line task space path. The *baseline* approach, i.e. without using tactile feedback and the RMRC approach considering tactile feedback are compared.

- The contact point $\boldsymbol{r}_{\text{poc}}$ is approximated by the equidistant body-fixed point between revolute joints 4 and 5 on link 4.
- Due to the assumption of a frictionless contact, only components of the force $\boldsymbol{F}_{\text{ct}}$ normal to the surface of the tactile sensor are taken into account.
- Pretensions in the tactile sensor lead to an offset. Thus, only contact forces above a certain force threshold $F_{\text{ct}} > F_{\text{ct,min}}$ are considered.

Furthermore, fast and jerking movements of the manipulator induce dynamic forces to the rigid hull of the tactile sensor. Feeding back these forces in a closed loop yields an unstable behavior of the planning module. A simple compensation based on measured velocities and accelerations of the joints showed no improvement due to time lags and unmodeled link elasticities. Therefore, the force signal is filtered by a low-pass filter (Butterworth, 8th order) with a cutoff frequency of 10 Hz.

In addition to the signal processing, transition strategies for the contact/no-contact states regarding the integral terms of the feedback controller are required. To ensure a safe operation even in case the control loop becomes unstable, the control inputs $\boldsymbol{u}_{(.)}$ as well as the gradients $\nabla_{\boldsymbol{q}} H_e$ are saturated. Furthermore, the environment stiffness c is assumed to be known in order to achieve the given target dynamics. It is shown that the controller performs worse with a given constant stiffness c_0 than using the real c. Therefore, a simple but efficient online estimator for the contact stiffness is proposed, based on the measured changes of the contact force w.r.t. the distance travelled by the contact point. More detailed information can be found in Sygulla (2015) and Sygulla et al. (2016).

Test Setup

The proposed algorithms were tested in laboratory setup with the CROPS manipulator prototype 2 in its 9-DOF configuration and equipped with the tactile sensor at link 4. The manipulator moves along a straight-line trajectory defined

in a three-dimensional task space ($\boldsymbol{w} := [x, y, z]^T \in \mathbb{R}^3$). The movement of the arm is obstructed by a vertical band of elastic rubber. Parameters of the scene are given in table 4.1. Two different tasks are presented:

- *Avoidable* Scenario: While following the task space path, the manipulator is able to avoid the obstacle in its nullspace.
- *Inevitable* Scenario: Due to an extended task space path the manipulator is no longer capable of avoiding the obstacle in its nullspace: The task space constraint must be abandoned in order to reduce the contact force to zero.

A typical experiment is shown in fig. 4.5. The manipulator starts moving along a given task space path (orange line). During the motion, link 4 touches the vertical, golden latex band. Depending on the chosen controller, the manipulator reduces the contact force by appropriate nullspace movements. The following controller implementations are evaluated for the aforementioned scenarios:

Baseline The baseline controller is the implementation of the automatic supervisory control scheme (eq. 4.12) *without* tactile feedback.

RMRC The gradient based RMRC controller is the extension of the baseline controller but with tactile feedback as a secondary objective function (eq. 4.35). This controller is limited to the nullspace of the manipulator.

RAC The gradient based RAC controller is analogous to the RMRC controller, but uses the acceleration level RAC scheme.

Feedback The feedback controller implements the approach proposed in section 4.3.2 with a feedback linearization nullspace controller and second order target dynamics.

Multi-Space/Task This controller is a hierarchical combination of a feedback controller in nullspace and task space. When the contact force exceeds a threshold F_{act}, the task space controller is activated and reduces the contact force by an adaption of the task space constraint.

Measurements

This section evaluates the performance of the different controllers measuring the normal contact force $F_{\text{ct,sen}}$ and the joint velocities $\dot{\boldsymbol{q}}$. For better comparability, the square root of the pseudo energy $\dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}}$ (equal to the sum of joint velocities per timestep $\sum_{i=1}^{n} \dot{q}_i^k$) is considered over time. The *baseline* controller, i.e. the manipulator moving without tactile feedback, is used as the reference.

Fig. 4.6 shows a comparison of both gradient based controllers in the *avoidable* scenario. While the baseline controller does not adapt its motion plan to the obstacle and the contact force increases to $> 9\,\text{N}$, the gradient based controllers

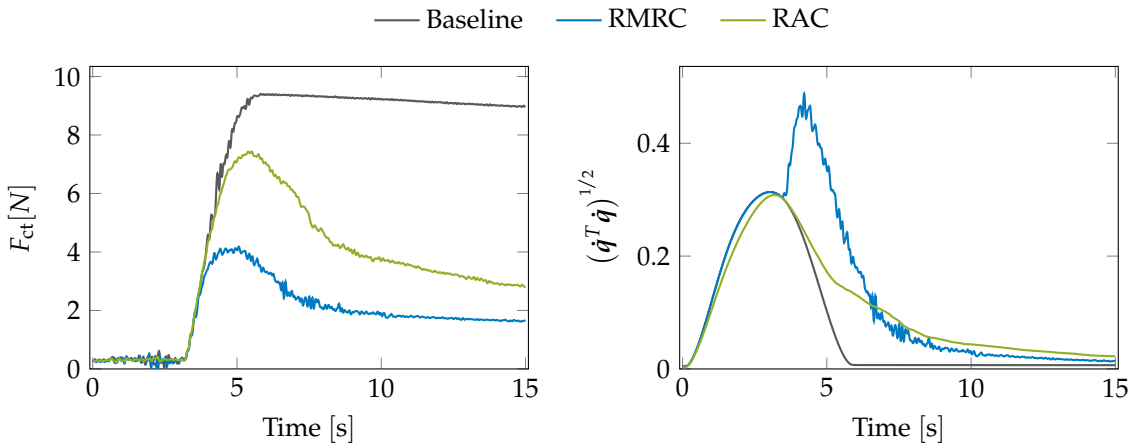


Figure 4.6: Scalar normal force F_{ct} (*left*) and pseudo energy $\dot{q}^T\dot{q}$ (*right*) for the *avoidable* scenario using the baseline controller (gray, no tactile feedback) and the gradient based controllers RMRC (blue, velocity level) and RAC (green, acceleration level).

induce a nullspace movement to reduce $\boldsymbol{F}_{ct}$. The RMRC controller shows a faster adaption and therefore faster reduction of the contact force compared to the RAC controller, at the cost of higher joint velocities.

The same scenario is tested using the feedback controller (cf. fig. 4.7). This controller reacts faster than the RMRC controller, trying to control the contact force to the threshold $F_{ct,min}$. At the beginning (magnification in fig. 4.7), a fast reduction to $F_{ct} \approx 0$ can be observed. This can be explained by the feed-forward term in eq. (4.47) trying to avoid all manipulator movements in direction of the contact force. Since the reaction of the feedback controller is faster, higher joint velocities are required. Notably, by contrast to the feedback controller, the RMRC controller does not achieve a reduction of $F_{ct} \to F_{ct,min}$. This is explained by the character of the external force being the gradient of a potential function, which is low for small differences $|F_{ct} - F_{ct,min}|$.

In the *inevitable* scenario, none of the nullspace control approaches is able to reduce the contact force when the nullspace is exploited. Thus, after a short initial reduction, the contact force increases further. In fig. 4.8, this can be observed for the RMRC as well as for the feedback controller. This scenario is well suited for showing the idea of the multi-space (task) controller: First, only the nullspace controller reduces the contact force. Reaching its limits, the force increases again. Having reached the activation threshold $F_{act} = 5\,\mathrm{N}$, the task space controller is activated and the task space path is abandoned in favor of a further reduction of F_{ct}. Until F_{act} is reached, it can be seen that the multi-space/task and the feedback controller act the same. The initial fast contact force reduction by the feed-forward term can also be found in this scenario (magnification at fig. 4.8).

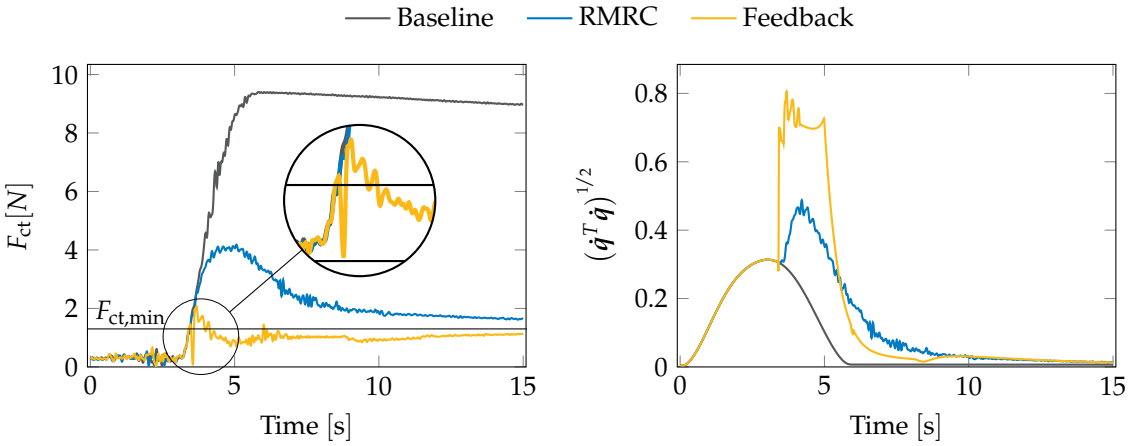


Figure 4.7: Comparison of scalar normal force F_{ct} (*left*) and pseudo energy $\dot{q}^T\dot{q}$ (*right*) for the *avoidable* scenario using the gradient based RMRC (blue, velocity level) and the *feedback linearization* controller (yellow, second order dynamics) controller. Motion without tactile feedback drawn for comparison (gray).

4.3.5 Discussion

In this section, several approaches are presented to integrate tactile sensor feedback into the online inverse kinematics planning module for a redundant manipulator. Based on the RMRC or RAC inverse kinematics solution, external forces on the manipulator arm can be reduced efficiently by adapting the motion plan. Dependent on the system formulation, the reaction is either limited

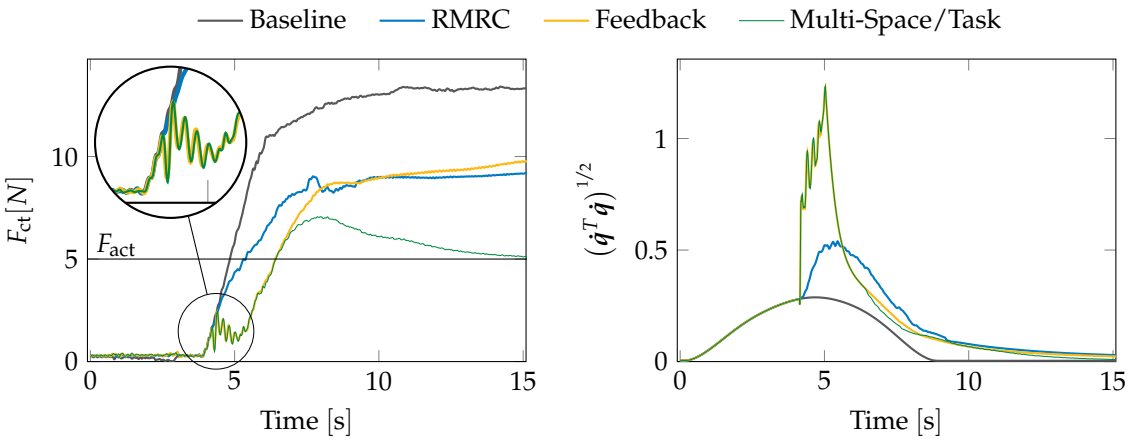


Figure 4.8: Comparison of scalar normal force F_{ct} (*left*) and pseudo-energy $\dot{q}^T\dot{q}$ (*right*) for the *inevitable* scenario using the nullspace bounded gradient based RMRC (blue, velocity level) and *feedback linearization* controller (yellow, second order dynamics) as well as the *multi-space/task* controller (green, $F_{act} = 5\,\mathrm{N}$). Motion without tactile feedback drawn for comparison (gray).

to the robot's nullspace or is even able to adapt the task space using a hierarchical scheme. Apart from the application of a simple gradient-based scheme, a feedback controller law is proposed. In the following, results are discussed w.r.t. the implemented approach and compared to the well-known impedance control approaches. Furthermore, current limitations are summarized and an outlook to multi-contact scenarios is given.

Gradient-Based Approach vs. Feedback Control The gradient-based approach directly uses the measured external force in a RMRC or RAC scheme and projects it to the robot's nullspace. Thus, this approach can be efficiently implemented and applied. Besides the filtering of the measured force, only the scalar weight needs to be tuned. Many public laboratory demonstrations showed that this approach is especially robust and smooth for human-robot interaction, even for non-expert users. However, a chosen parametrization is always a trade-off between smoothness and a fast reaction. This makes it difficult for the gradient-based approaches to cope with obstacles of different stiffnesses. By comparison, the feedback controller approach allows the tuning of its behavior by adapting its target system dynamics and shows a more "intelligent" behavior. This approach directly tracks the contact force and is able to achieve a fast reduction, if necessary. Experiments showed that it was even capable to reduce contact forces sufficiently while colliding with a stiff aluminum bar (Sygulla et al. 2016). For the user, the application of this approach is more time consuming to achieve robust behavior. Furthermore, the online estimation of the contact stiffness has a high impact on the controller performance as was shown by Sygulla et al. (2016).

Relation to Force Control Approaches The presented approaches are closely related to classic force control methods. However, there are significant differences regarding performance and application. Indirect force control methods, such as admittance or impedance control, do not intend a distinct priorization of tasks and contact forces. Thus, even low external forces may lead to position errors of the end effector. Classic direct force control approaches mainly consider only forces on the end effector while the proposed methods aim for a reduction of contact forces on the arm itself. Generally, these approaches do not distinguish between nullspace or task space, nor do they provide an adaptive priorization as shown with the multi-space approach. An appropriate classification of the proposed approach related for force control was given by Sygulla et al. (2016): "[It] can be compared with a hybrid force/motion control scheme with adaptive selection of the spaces for force and motion control."

Current Limitations The presented approaches behave sensitively to the signal quality of the measured contact force. Therefore, some simplifying assumptions (body fixed force application point, disregard of tangential forces, minimum force threshold) had to be taken. With higher signal quality, these simplification may

be lifted gradually. This sensitivity is mainly caused by the projection of the force to the contact space. While this approach facilitates the controller design tremendously, deviations in the force direction are directly transmitted to the motion plan. Thus, noise, friction at the contact and other disturbances of the force measurement deteriorate the control performance. Some of the previously mentioned simplifications may be alleviated by the design of a controller in Cartesian space instead of the contact space and including friction, i.e. tangential contact forces in the model. Most important, improvements of the tactile sensor hardware itself would further enhance the force control related approaches.

Extension to Multi-Contact Scenarios In this thesis, the tactile sensor only covers one link of the manipulator. Thus, only one-contact scenarios were examined in this section. The application of the proposed approaches to multi-contact scenarios are one important direction of future research. For gradient related methods, a suitable formulation is already given in eq. (4.37). Using the controller based approaches, a controller for each measured resulting contact force needs to be implemented. Probably, these may be required to be coordinated on a higher level.

Summary The approaches developed in this section allow an efficient reduction of contact forces in unknown cluttered environments. The proposed framework enables the resolution of the conflict between low contact forces and sufficient positional tracking. Contact forces are either reduced by movements in the nullspace of the manipulator, i.e. precisely keeping track of the desired task, or by adapting the task, if required.

Parameter	Symbol	Value
Inverse Kinematics		
Definition task space	$\boldsymbol{w}$	$[x, y, z]^T \in \mathbb{R}^3$
Sample time	Δt	0.001 s
Measurement duration	t_{meas}	
for the avoidable *scenario*		15 s
for the inevitable *scenario*		25 s
Nullspace weighing matrix	$\boldsymbol{W}$	$\mathrm{diag}(10, 1, 2, 2, 1, 1, 1, 1.5, 1)$
Numerical drift compensation factor	K	
for joint space controllers		$0.5/t_s$
for all other controllers		$0.005/t_s$
Scaling factor for self-collision avoidance	s_{sca}	
on velocity level		8
on acceleration level		23
Scaling factor for joint limit avoidance	s_{jla}	
on velocity level		5
on acceleration level		20
Soft joint limits	$q_{i,s,min/max}$	
for prismatic joints		$\{q_{i,h,min} + 0.3\,\mathrm{m}; q_{i,h,max} - 0.3\,\mathrm{m}\}$
for revolute joints		$\{q_{i,h,min} + 30\,^\circ; q_{i,h,max} - 30\,^\circ\}$
Scaling factor for acceleration level damping	s_{dmp}	7
Scaling factor for the elastic potential	s_e	1
Tactile Feedback		
Force threshold for controller activation	F_{th}	1.3 N
Nullspace limitation threshold		
on velocity level	$\dot{q}_{n,max}$	$5\,\mathrm{rad/s}$ or $5\,\mathrm{m/s}$
on acceleration level	$\ddot{q}_{n,max}$	$20\,\mathrm{rad/s^2}$ or $20\,\mathrm{m/s^2}$
Sample time for the tactile sensor module	$t_{s,t}$	0.004 s
Time constant for controller target dynamics	T	0.5 s
Damping factor for controller target dynamics	d	0.5
Minimum force for task space relaxation	F_{min}	5 N
Transition handling distance	d_t	0.05 m
Max. transition handling time constant	$T_{t,\mathrm{max}}$	1 s
Minimum penetration depth for stiffness estimation	$\Delta x_{est,min}$	5.0×10^{-4} m
Trajectory and Environment		
Relative position of the tool center point	$\boldsymbol{r}_{tcp}$	$(\,0.0\ 0.0\ 0.013\,)^T$ m
Task space trajectory, *Avoidable* scenario		
Start position		$(\,-0.30\ 1.30\ 1.1\,)^T$ m
End position		$(\,0.40\ 1.0\ 1.1\,)^T$ m
Execution time	t_{traj}	6 s
Task space trajectory, *Inevitable* scenario		
Start position		$(\,-0.30\ 1.30\ 1.1\,)^T$ m
End position		$(\,0.75\ 1.0\ 1.1\,)^T$ m
Execution time	t_{traj}	9 s
Environment stiffness Thera-band Gold	c_e	$\approx 60\,\mathrm{N/m}$
Assumed environment stiffness	c	$60\,\mathrm{N/m}$

Table 4.1: Parameters at the experimental evaluation of the tactile planning approaches. See also (Sygulla 2015).

Chapter 5

Predictive Inverse Kinematics

The inverse kinematics problem for redundant systems can be solved efficiently using the resolved motion rate control (RMRC) and automatic supervisory control (ASC) algorithms by Whitney (1969) and Liégeois (1977). Joint velocities as well as secondary objective functions are minimized by the pseudoinverse $\boldsymbol{J}^{\#}$ and a local descent along the gradient $\nabla_q H$ of the secondary objective function H. This method provides a computationally efficient and reliable solution, however, it only takes into account costs of the next discrete time increment. A more advanced approach would also take into account constraints of future time increments such as workspace trajectories, limits, collisions, etc. while planning the joint space trajectory. Such an approach is developed in this chapter.

Basically, this approach is a computational extension to the ASC scheme. Taking advantage of its system formulation, an indirect optimization scheme is applied to the inverse kinematics problem. Thus, future events can be taken into account while optimizing the nullspace movement of the manipulator. Following the idea of model predictive control (MPC), this method can be applied in real time with a finite prediction horizon. Parts of this chapter, including the problem formulation, the solution by the conjugate gradient method and real time implementation are presented in a former publication by the author (Schütz, Buschmann, et al. 2014). Furthermore, this chapter shows the solution of the problem as a two-point boundary value problem (TPBVP) and extends the numerical background of the gradient method by evaluating further algorithms for the calculation of the conjugate gradient in combination with several line search approaches.

5.1 The Optimization Problem in Nullspace

This chapter presents a suitable problem formulation and the corresponding optimality conditions according to Pontryagin's minimum principle (PMP) based on

the system formulation originally presented by Nakamura and Hanafusa (1987b). Furthermore, two additions are shown that assure continuous behavior on the position and velocity levels.

5.1.1 Nakamura's System Equation

Based on the automatic supervisory control scheme shown in section 4.1.2 Nakamura (1991) proposed to replace the gradient descent in nullspace $\frac{\partial H}{\partial q}$ by a general input $\boldsymbol{u}$ and to use it as the system dynamics' equation $\boldsymbol{f}_w(\cdot)$:

$$\begin{aligned} \boldsymbol{f}_W(\boldsymbol{q},\boldsymbol{u}) &:= \dot{\boldsymbol{q}} = \boldsymbol{J}_W^{\#}\dot{\boldsymbol{w}}_{\text{des}} + \boldsymbol{N}_W\boldsymbol{u} \\ \boldsymbol{N}_W &:= \alpha_N\left(\boldsymbol{I} - \boldsymbol{J}_W^{\#}\boldsymbol{J}_w\right) \end{aligned} \tag{5.1}$$

For the sake of simplicity, the Jacobian $\boldsymbol{J}$, its pseudoinverse $\boldsymbol{J}^{\#}$ and the nullspace projection $\boldsymbol{N}$ are used without weights in the further description. However, one should note that the relative weights are useful, e.g. while dealing with kinematic structures which include both prismatic and revolute joints. Without loss of generality the equations can be written as

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{q},\boldsymbol{u}) &:= \dot{\boldsymbol{q}} = \boldsymbol{J}^{\#}\dot{\boldsymbol{w}}_{\text{des}} + \boldsymbol{N}\boldsymbol{u} \\ \boldsymbol{N} &:= \alpha_N\left(\boldsymbol{I} - \boldsymbol{J}^{\#}\boldsymbol{J}\right). \end{aligned} \tag{5.2}$$

Using the system eq. (5.2), the optimization problem is formulated by Nakamura and Hanafusa (1987b) using the cost functional $L(\cdot)$ in Lagrangian form:

$$\min_{\boldsymbol{u}(\tau)} L(\boldsymbol{q},\boldsymbol{u},t) = \int_\tau l\left(\boldsymbol{q},\boldsymbol{u},t\right)\mathrm{d}t \tag{5.3a}$$

$$\boldsymbol{f}(\boldsymbol{q},\boldsymbol{u}) = \boldsymbol{J}^{\#}\dot{\boldsymbol{w}}_{\text{des}} + \boldsymbol{N}\boldsymbol{u} \tag{5.3b}$$

$$\boldsymbol{\Phi}\left[\boldsymbol{q}(t_0)\right] = \boldsymbol{w}_{\text{des},0} \tag{5.3c}$$

$$\tau \in [t_0, t_{\text{end}}]$$

While the end state $\boldsymbol{q}(T_{\text{end}})$ is free and inherently fulfills the condition $\boldsymbol{\Phi}\left[\boldsymbol{q}(t_{\text{end}})\right] = \boldsymbol{w}_{\text{des,end}}$, the condition eq. (5.3c) applies to the initial configuration $\boldsymbol{q}(t_0) = \boldsymbol{q}_0$. The considered time interval $\tau := t \in [t_0, t_{\text{end}}]$ is fixed. Optimality conditions for the optimal control problem eq. (5.3) are given by applying PMP (Pontryagin et al. 1962) using the Hamiltonian $H(\cdot)$ and the adjoint variables $\boldsymbol{\lambda}(t)$[1]:

$$H(\boldsymbol{q},\boldsymbol{u},\boldsymbol{\lambda},t) = l(\boldsymbol{q},\boldsymbol{u},t) + \boldsymbol{\lambda}^T(t)\boldsymbol{f}(\boldsymbol{q},\boldsymbol{u},t) = l + \boldsymbol{\lambda}^T\dot{\boldsymbol{q}} \tag{5.4a}$$

$$\left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T = \boldsymbol{f}(\boldsymbol{q},\boldsymbol{u}) = \dot{\boldsymbol{q}} \tag{5.4b}$$

[1] For the sake of clarity, the explicit notation of the time dependency $(\cdot)(t)$ is omitted, e.g. $\boldsymbol{\lambda} \overset{\text{def.}}{=} \boldsymbol{\lambda}(t)$

$$-\left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T = \dot{\boldsymbol{\lambda}} = -\left(\frac{\partial l}{\partial \boldsymbol{q}}\right)^T - \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^T \boldsymbol{\lambda} \tag{5.4c}$$

$$\boldsymbol{\Phi}(\boldsymbol{q})|_{t_0} = \boldsymbol{w}_0 \tag{5.4d}$$

$$\boldsymbol{\lambda}|_{t_{\text{end}}} = \mathbf{0} \tag{5.4e}$$

$$\boldsymbol{N}\boldsymbol{\lambda}|_{t_0} = \mathbf{0} \tag{5.4f}$$

Eq. (5.4) defines a TPBVP with the boundary conditions:

- Eq. (5.4d): The initial configuration $\boldsymbol{q}|_{t_0}$ has to coincide with the desired initial position $\boldsymbol{w}_0$ in task space (eq. 5.3c).
- Eq. (5.4e): The adjoint variable $\boldsymbol{\lambda}|_{t_{\text{end}}}$ vanishes since the problem has Lagrangian form with a free final state $\boldsymbol{q}|_{t_{\text{end}}}$.
- Eq. (5.4f): The transversality condition ensures that the adjoint variables $\boldsymbol{\lambda}$ are normal to the tangent cone of the constraint set defined by eq. (5.4f).

Due to the transversality condition (eq. 5.4f) the set of boundary conditions is coupled: The adjoint variable $\boldsymbol{\lambda}$ has to fulfill conditions on both sides: start t_0 (*left* side) and endpoint t_{end} (*right* side). The TPBVP has dimension $2n$, since $\boldsymbol{q} \in \mathcal{C} \in \mathbb{R}^n$ and $\dim(\boldsymbol{\lambda}) \mathrel{\widehat{=}} \dim(\boldsymbol{q})$.

5.1.2 Extension 1: Decoupling the Problem

The formulation eq. (5.4) is suitable for applications where the initial configuration $\boldsymbol{q}_0$ is only constrained by the desired task space pose $\boldsymbol{w}_0$. However, if the initial configuration is given by the current joint positions, the boundary conditions have to be adapted. In particular, the initial state has to be defined in order to ensure a continuous operation while planning in an iterative MPC-like scheme[2]. Thus, the optimality conditions presented in eq. (5.4) can be rewritten taking into account the initial condition $\boldsymbol{q}|_{t_0} \overset{!}{=} \boldsymbol{q}_0$:

$$H(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{\lambda}, t) = l(\boldsymbol{q}, \boldsymbol{u}, t) + \boldsymbol{\lambda}^T(t)\boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}, t) = l + \boldsymbol{\lambda}^T\dot{\boldsymbol{q}} \tag{5.5a}$$

$$\left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}) = \dot{\boldsymbol{q}} \tag{5.5b}$$

$$-\left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T = \dot{\boldsymbol{\lambda}} = -\left(\frac{\partial l}{\partial \boldsymbol{q}}\right)^T - \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\right)^T \boldsymbol{\lambda} \tag{5.5c}$$

$$\boldsymbol{q}|_{t_0} = \boldsymbol{q}_0 \tag{5.5d}$$

$$\boldsymbol{\lambda}|_{t_{\text{end}}} = \mathbf{0} \tag{5.5e}$$

This formulation decouples the boundary conditions of the differential equations for the system dynamics and the adjoint variables. While the system equation is bounded on the left side by the initial system state $\boldsymbol{q}|_{t_0}$, the right side of the adjoint equation is determined. The coupling by eq. (5.4f) is no longer applicable. This property enables an efficient numerical solution of the problem by the (conjugate) gradient method presented in section 5.2.

[2] Details on the MPC implementation are presented in section 5.3.

5.1.3 Extension 2: State Augmentation to Acceleration Level

While *extension 1* guarantees continuity on the position level, a state augmentation of the system equation to the acceleration level ensures continuous velocities. Therefore, the new system input $\hat{\boldsymbol{u}}$ is introduced as the time derivative of $\boldsymbol{u}$. The new system state variable $\boldsymbol{y}$ is composed of the previous state $\boldsymbol{q}$ and its derivative $\dot{\boldsymbol{q}}$. Thus, the optimization problem (eq. 5.3) using the boundaries (eq. 5.5) can be rewritten as:

$$\min_{\hat{\boldsymbol{u}}(\tau)} L = \int_{\tau} l\left(\boldsymbol{q}, \hat{\boldsymbol{u}}, t\right) \mathrm{d}t \tag{5.6}$$

$$\boldsymbol{f}_{\mathrm{acc}}(\boldsymbol{q}, \hat{\boldsymbol{u}}) =: \dot{\boldsymbol{y}} = \begin{bmatrix} \dot{\boldsymbol{q}} \\ \dot{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}^{\#} \dot{\boldsymbol{w}}_{\mathrm{des}} + \boldsymbol{N}\boldsymbol{u} \\ \hat{\boldsymbol{u}} \end{bmatrix} \tag{5.7}$$

$$\begin{aligned} \boldsymbol{q}|_{t_0} &= \boldsymbol{q}_0 \\ \boldsymbol{u}|_{t_0} &= \boldsymbol{0} \\ \tau &\in [t_0, t_{\mathrm{end}}] \end{aligned} \tag{5.8}$$

The optimality conditions according to Pontryagin's minimum principle with the augmented adjoint variable $\boldsymbol{\lambda} := [\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2]^T$ lead to

$$H(\boldsymbol{q}, \hat{\boldsymbol{u}}, \boldsymbol{\lambda}) = l(\boldsymbol{q}, \hat{\boldsymbol{u}}) + \boldsymbol{\lambda}^T \boldsymbol{f}_{\mathrm{acc}}(\boldsymbol{q}, \hat{\boldsymbol{u}}) = l + \boldsymbol{\lambda}_1^T \dot{\boldsymbol{q}} + \boldsymbol{\lambda}_2^T \hat{\boldsymbol{u}} \tag{5.9a}$$

$$\left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T = \boldsymbol{f}_{\mathrm{acc}}(\boldsymbol{q}, \hat{\boldsymbol{u}}) = \dot{\boldsymbol{y}} \tag{5.9b}$$

$$-\left(\frac{\partial H}{\partial \boldsymbol{y}}\right)^T = -\begin{bmatrix} \left(\frac{\partial H}{\partial \boldsymbol{q}}\right)^T \\ \left(\frac{\partial H}{\partial \boldsymbol{u}}\right)^T \end{bmatrix} = \begin{bmatrix} \dot{\boldsymbol{\lambda}}_1 \\ \dot{\boldsymbol{\lambda}}_2 \end{bmatrix} = \begin{bmatrix} -\left(\frac{\partial l}{\partial \boldsymbol{q}}\right)^T - \left(\frac{\partial \dot{\boldsymbol{q}}}{\partial \boldsymbol{q}}\right)^T \boldsymbol{\lambda}_1 \\ -\left(\frac{\partial l}{\partial \boldsymbol{u}}\right)^T - \left(\frac{\partial \dot{\boldsymbol{q}}}{\partial \boldsymbol{u}}\right)^T \boldsymbol{\lambda}_2 \end{bmatrix} \tag{5.9c}$$

$$\begin{aligned} \boldsymbol{y}|_{t_0} &= \begin{bmatrix} \boldsymbol{q}_0 \\ \boldsymbol{0} \end{bmatrix} \\ \boldsymbol{\lambda}|_{t_{\mathrm{end}}} &= \boldsymbol{0}. \end{aligned} \tag{5.9d}$$

The TPBVP now has the dimension $2 \cdot 2n = 4n$.

5.1.4 Cost Function

The cost function can be chosen arbitrarily, in general. This thesis uses a cost function in Lagrangian form, i.e. a function $l(\cdot)$ is integrated over time to the cost function $L(\cdot)$:

$$\begin{aligned} L(\boldsymbol{q}, \boldsymbol{u}, t) &= \int_{\tau} l\left(\boldsymbol{q}, \boldsymbol{u}, t\right) \mathrm{d}t \\ \tau &\in [t_0, t_{\mathrm{end}}] \end{aligned} \tag{5.10}$$

By analogy to section 4.2 the objective function $l(\cdot)$ can be composed of a term dependent only on the current configuration $\boldsymbol{q}$ and $[\boldsymbol{q}, \dot{\boldsymbol{q}}]$ respectively.

$$l(\boldsymbol{q}, \dot{\boldsymbol{q}}, \hat{\boldsymbol{u}}) = \zeta_q l_q(\boldsymbol{q}) + \zeta_{\dot{q}} l_{\dot{q}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \zeta_{\hat{u}} l_{\hat{u}}(\hat{\boldsymbol{u}}) \tag{5.11}$$

The term l_q includes e.g. penalty functions for joint limit avoidance (cf. section 4.2.1) and collision avoidance (cf. section 4.2.2):

$$l_q := H = \sum_{i=1}^{h} \zeta_i H_i(\boldsymbol{q}) \tag{5.12}$$

$l_{\dot{q}}$ is chosen to minimize joint velocities $\dot{\boldsymbol{q}}$:

$$l_{\dot{q}}(\boldsymbol{q}.\dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}} \tag{5.13}$$

Furthermore, the input $\hat{\boldsymbol{u}}$ is damped by the quadratic penalty term:

$$l_{\hat{u}}(\hat{\boldsymbol{u}}) = \frac{1}{2}\hat{\boldsymbol{u}}^T\hat{\boldsymbol{u}}. \tag{5.14}$$

5.2 Numerical Solution

The solution of OCPs, i.e. dynamic optimization problems, by indirect methods relies on the formulation of the OCP's Hamiltonian. Applying Pontryagin's minimum principle, the optimality conditions, i.e. the canonical equations can be derived. Various numerical approaches are capable of determining the optimal control trajectory:

- *Solving a TPBVP*: By eliminating the control input, using an expression for the optimal control input which only depends on the state and adjoint variables, the problem can be transformed into an ODE with boundary conditions on the left and/or right sides of the time interval. This problem can be solved by standard solver routines, e.g. the MATLAB function `bvp4c`.
- *Minimizing an objective function*: The optimal control input minimizes the objective function. Thus, it can be solved in an iterative scheme using first or second order gradient descent methods. In section 5.2, the conjugate gradient method as a first-order algorithm with nearly quadratic convergence is examined for a predictive solution of the inverse kinematics problem.
- *Differential Dynamic Programming*: Although originating from the dynamic programming approach as presented by Bellman (1954), this method is also related to indirect optimization methods. The cost-to-go (*value*) function is approximated quadratically at a current estimate of the optimal trajectory, yielding a locally optimal feedback law. Promising results are presented by Tassa et al. (2008, 2012, 2014). These were confirmed for the nullspace optimization problem considered in this thesis by the student theses of Berger (2015) and Smith (2014).

One has to note that the aforementioned methods in their original form apply only to unconstrained problems. Input or state constraints can be taken into account by penalty functions. However, the system formulation eq. (5.9) based on Automatic Supervisory Control assures that the task space trajectory constraint is fulfilled inherently.

5.2.1 Solution as a TPBVP

In order to solve eq. (5.9) as a two-point boundary value problem (TPBVP) by standard solver algorithms, the control variable $\hat{\boldsymbol{u}}$ has to be eliminated in the canonical equations. Thus, this section derives the optimal control input $\boldsymbol{u}^\star$ and shows the corresponding TPBVP formulation.

Optimal Control Input $u^\star$

The optimal control input $\boldsymbol{u}^\star$ minimizes the Hamiltonian $H(\cdot)$ yielding the optimal state $\boldsymbol{q}^\star$ and adjoint variable $\boldsymbol{\lambda}^\star$:

$$\boldsymbol{u}^\star = \underset{\hat{\boldsymbol{u}}}{\operatorname{argmin}}\, H(\boldsymbol{q}^\star, \boldsymbol{\lambda}^\star, \hat{\boldsymbol{u}}, t) \tag{5.15}$$

Since $\hat{\boldsymbol{u}}$ is unconstrained, the optimal input $\boldsymbol{u}^\star$ is defined by

$$\begin{aligned} \frac{\partial H}{\partial \hat{\boldsymbol{u}}} &\overset{!}{=} \mathbf{0} \\ &= \frac{\partial l}{\partial \hat{\boldsymbol{u}}} + \boldsymbol{\lambda}_1^T \frac{\partial \dot{\boldsymbol{q}}}{\partial \hat{\boldsymbol{u}}} + \boldsymbol{\lambda}_2^T \frac{\partial \dot{\boldsymbol{u}}}{\partial \hat{\boldsymbol{u}}} = \frac{\partial l}{\partial \hat{\boldsymbol{u}}} + \boldsymbol{\lambda}_2 = \zeta_{\hat{\boldsymbol{u}}} \hat{\boldsymbol{u}} + \boldsymbol{\lambda}_2 \end{aligned} \tag{5.16}$$

$$\boldsymbol{u}^\star = -\frac{\boldsymbol{\lambda}_2}{\zeta_{\hat{\boldsymbol{u}}}}. \tag{5.17}$$

The Two-Point Boundary Value Problem

Eqs. (5.9), (5.11) and (5.13) and using the optimal input $\boldsymbol{u}^\star = -\frac{\boldsymbol{\lambda}_2}{\zeta_{\hat{\boldsymbol{u}}}}$ eq. (5.17) yield the TPBVP:

$$H(\boldsymbol{q}, \hat{\boldsymbol{u}}, \boldsymbol{\lambda}) = \left[\zeta_q l_q(\boldsymbol{q}) + \zeta_{\dot{q}} l_{\dot{q}}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \zeta_{\hat{u}} l_{\hat{u}}(\hat{\boldsymbol{u}})\right] + \boldsymbol{\lambda}_1^T \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}) + \boldsymbol{\lambda}_2^T \hat{\boldsymbol{u}} \tag{5.18a}$$

$$\dot{\boldsymbol{y}} = \left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T = \begin{bmatrix} \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}) \\ \hat{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}^\# \dot{\boldsymbol{w}}_{\text{des}} + \boldsymbol{N}\boldsymbol{u} \\ \hat{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}^\# \dot{\boldsymbol{w}}_{\text{des}} + \boldsymbol{N}\boldsymbol{u} \\ -\boldsymbol{\lambda}_2/\zeta_{\hat{\boldsymbol{u}}} \end{bmatrix} \tag{5.18b}$$

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial H}{\partial \boldsymbol{y}}\right)^T = \begin{bmatrix} \dot{\boldsymbol{\lambda}}_1 \\ \dot{\boldsymbol{\lambda}}_2 \end{bmatrix} = -\begin{bmatrix} \left(\zeta_q \nabla_q l_q + \zeta_{\dot{q}} \nabla_q l_{\dot{q}} + \boldsymbol{\lambda}_1^T \nabla_q \boldsymbol{f}\right)^T \\ \left(\zeta_{\dot{q}} \nabla_u l_{\dot{q}} + \boldsymbol{\lambda}_1^T \nabla_u \boldsymbol{f}\right)^T \end{bmatrix} \tag{5.18c}$$

$$\begin{bmatrix} -\left(\nabla_q \dot{\boldsymbol{q}}\right)^T \left(\zeta_{\dot{q}} \dot{\boldsymbol{q}} + \boldsymbol{\lambda}_1\right) - \zeta_q \left(\nabla_q l_q\right)^T \\ -\boldsymbol{N}^T \left(\zeta_{\dot{q}} \dot{\boldsymbol{q}} + \boldsymbol{\lambda}_1\right) \end{bmatrix} \tag{5.18d}$$

$$\begin{aligned} \boldsymbol{y}|_{t_0} &= \begin{bmatrix} \boldsymbol{q}_0 \\ \boldsymbol{0} \end{bmatrix} \\ \boldsymbol{\lambda}|_{t_{\text{end}}} &= \boldsymbol{0} \end{aligned} \tag{5.18e}$$

Examples for the cost function gradients $\nabla_{(\cdot)} l_{(\cdot)}$ are shown in section 5.2.3. The gradients of the system equation $\nabla_q \boldsymbol{f}, \nabla_u \boldsymbol{f}$ are calculated by

$$\underset{(n,n_q)}{\nabla_q \boldsymbol{f}} = \underset{(n,m,n_q)(m,1)}{\nabla_q \boldsymbol{J}^{\#}} \; \underset{(m,1)}{\dot{\boldsymbol{w}}} - \underset{(n,m)(m,n,n_q)(n,1)}{\boldsymbol{J}^{\#}} \; \underset{(m,n,n_q)}{\nabla_q \boldsymbol{J}} \; \underset{(n,1)}{\boldsymbol{u}} + \underset{(n,m,n_q)(m,n)(n,1)}{\nabla_q \boldsymbol{J}^{\#}} \; \underset{(m,n)}{\boldsymbol{J}} \; \underset{(n,1)}{\boldsymbol{u}} \tag{5.19}$$

$$\underset{(n,n)}{\nabla_u \boldsymbol{f}} = \underset{(n,n)}{\boldsymbol{N}} \tag{5.20}$$

$$\nabla_q \boldsymbol{J}^{\#} = \boldsymbol{J}_q^T \left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1} - \boldsymbol{J}^T \left[\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1} \left(\boldsymbol{J}_q \boldsymbol{J}^T + \boldsymbol{J}\boldsymbol{J}_q^T\right) \left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1}\right]. \tag{5.21}$$

For convenience, the dimensions of the respective tensors are indicated by subscripts. Introducing the state variable $\boldsymbol{z} := [\boldsymbol{y}, \boldsymbol{\lambda}]^T$, the boundary conditions $\boldsymbol{z}_0 := \boldsymbol{z}(t_0)$ and $\boldsymbol{z}_e := \boldsymbol{z}(t_e)$ of eq. (5.18) are defined as:

$$\boldsymbol{z} := \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{u} \\ \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \end{bmatrix}, \; \boldsymbol{z}_0 = \begin{bmatrix} \boldsymbol{q}_0 \\ \boldsymbol{0} \\ \square \\ \square \end{bmatrix}, \; \boldsymbol{z}_e = \begin{bmatrix} \square \\ \square \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \tag{5.22}$$

This can be solved directly using e.g. the MATLAB routine bvp4c. The symbol $\square$ denotes a free state. In the following sections, numerically efficient gradient based methods are presented that allow solving this problem iteratively in real time as shown in Schütz (2014). The bvp4c implementation is considered to be very robust and serves as a reference to the developed gradient based methods.

5.2.2 Minimizing the Objective Function by Gradient Methods

Basic Algorithm

The gradient method is suited for decoupled dynamic optimization problems. An early description is presented by Kopp et al. (1964). It calculates the trajectory of the system state $\boldsymbol{y}(t)$ and the adjoint variable $\boldsymbol{\lambda}(t)$ iteratively by forward and backward integration, respectively. The variable j denotes the $j-$th iteration. The control input $\hat{\boldsymbol{u}}$ is updated by the gradient information $\boldsymbol{g}(t)$ using a step length α^j. Finding the optimal value for α^j, i.e. the step length that minimizes the cost function $L(\cdot)$ (cf. eq. (5.10)) is denoted as the *line search* problem. The loop is repeated until the maximum number of iterations $j_{\max}$ is reached, the reduction in the cost function is smaller than a lower barrier ϵ, or the solution diverges, i.e.

the cost function increases again. Since only the gradient $\boldsymbol{g} := \frac{\partial H}{\partial \hat{\boldsymbol{u}}}$ is used, this method is a first-order optimization scheme. In its simplest form shown in alg. 2, it is denoted as the steepest descent (SD).

Algorithm 2 Basic Algorithm of the Gradient Method, Steepest Descent (SD)

$j \leftarrow 0,\ \tau := t \in [t_0, t_{\text{end}}]$.
$\boldsymbol{q}^0(\tau) \leftarrow \int_\tau \dot{\boldsymbol{q}}(\hat{\boldsymbol{u}}^0, t)\,\mathrm{d}t$
$L^0 \leftarrow L(\boldsymbol{q}^0, \dot{\boldsymbol{q}}^0, \hat{\boldsymbol{u}}^0)$ (Evaluation of the Cost Function)
repeat
  $\boldsymbol{\lambda}^j(\tau) \leftarrow \int_\tau \dot{\boldsymbol{\lambda}}(\boldsymbol{q}^j, \dot{\boldsymbol{q}}^j, \hat{\boldsymbol{u}}^j, t)\,\mathrm{d}t$, backward in time: $t_{\text{end}} \rightarrow t_0$
  $\boldsymbol{g}^j \leftarrow \frac{\partial H}{\partial \hat{\boldsymbol{u}}}$
  $\boldsymbol{s}^j \leftarrow -\boldsymbol{g}^j$
  $\hat{\boldsymbol{u}}^{(j+1)} \leftarrow \hat{\boldsymbol{u}}^j + \alpha^j \boldsymbol{s}^j$
  $j \leftarrow j + 1$
  $\boldsymbol{q}^j \leftarrow \int_\tau \dot{\boldsymbol{q}}(\hat{\boldsymbol{u}}^j, t)\,\mathrm{d}t$, forward in time: $t_0 \rightarrow t_{\text{end}}$
  $L^j \leftarrow L(\boldsymbol{q}^j, \dot{\boldsymbol{q}}^j, \hat{\boldsymbol{u}}^j)$
  exit $\leftarrow (j > j_{\max}) \vee \left(\frac{L^j - L^{j-1}}{L^{j-1}} < \epsilon\right) \vee \left(L^j > L^{j-1}\right)$
until (exit = **true**)
return $\hat{\boldsymbol{u}}^j$

Further additions to this basic algorithm consider two aspects:

- *Search Direction:* The use of information from previous iterations accelerates the convergence of the method by using the CG.
- *Adaptive Stepsize:* Instead of choosing a fixed step size, *line search* algorithms try to approximate the optimal solution for the step size α^j.

Conjugate Gradient

Newton methods require the second derivative (Hessian) of the objective function, whose calculation is more time consuming compared to the first-order gradient. Although their rate of convergence is faster compared to first order methods, their stability with poor initial guesses is lower. Another option is to use gradient information from previous iterations to increase convergence, instead of using only the current gradient. This method is denoted as the conjugate gradient (CG), early descriptions can be found at Hestènes and Stiefel (1952). The current search direction $\boldsymbol{s}^j$ is determined using the current gradient $\boldsymbol{g}^j$ and the previous search direction $\boldsymbol{s}^{j-1}$ weighted by β^j. The algorithm of the CG method is shown in alg. 3. For the calculation of β several approaches are developed. While an early approach is presented by Hestènes and Stiefel (1952), the later adaption of Fletcher and Reeves (1964) is applied by Lasdon et al. (1967) to OCPs. They show that the convergence of the CG for OCPs is faster than the SD and

Algorithm 3 Conjugate Gradient Method (CG)

$j \leftarrow 0$
$\boldsymbol{q}^0(\tau) \leftarrow \int_\tau \dot{\boldsymbol{q}}(\hat{\boldsymbol{u}}_0, t)\mathrm{d}t$
$L^0 \leftarrow L(\boldsymbol{q}^0, \dot{\boldsymbol{q}}^0, \hat{\boldsymbol{u}}^0)$
repeat
  $\boldsymbol{\lambda}^j(\tau) \leftarrow \int_\tau \dot{\boldsymbol{\lambda}}(\boldsymbol{q}^j, \dot{\boldsymbol{q}}^j, \hat{\boldsymbol{u}}^j, t)\mathrm{d}t$, backward in time: $t_\mathrm{end} \rightarrow t_0$
  $\boldsymbol{g}^j \leftarrow \frac{\partial H}{\partial \hat{\boldsymbol{u}}}$
  if $j \neq 0$ **then**
    $\beta^j \leftarrow$ *Conjugate Gradient* algorithm by eq. (5.23)
    $\boldsymbol{s}^j \leftarrow -\boldsymbol{g}^j + \beta^j \boldsymbol{s}^{(j-1)}$
  else
    $\boldsymbol{s}^j \leftarrow -\boldsymbol{g}^j$
  end if
  $\hat{\boldsymbol{u}}^{(j+1)} \leftarrow \hat{\boldsymbol{u}}^j + \alpha^j \boldsymbol{s}^j$
  $j \leftarrow j + 1$
  $\boldsymbol{q}^j \leftarrow \int_\tau \dot{\boldsymbol{q}}(\hat{\boldsymbol{u}}^j, t)\mathrm{d}t$, forward in time: $t_0 \rightarrow t_\mathrm{end}$
  $L^j \leftarrow L(\boldsymbol{q}^j, \dot{\boldsymbol{q}}^j, \hat{\boldsymbol{u}}^j)$
  exit $\leftarrow (j > j_\mathrm{max}) \vee \left(\frac{L^j - L^{j-1}}{L^{j-1}} < \epsilon\right) \vee \left(L^j > L^{j-1}\right)$
until (exit = **true**)
return $\hat{\boldsymbol{u}}^j$

a descent in every iteration can be guaranteed. An efficient implementation of the later adaption of the Fletcher/Reeves algorithm by Polak and Ribiere (1969) is published by Klessig and Polak (1972) and is later used by Haas (1975) in the context of OCPs.

In the following, the different CG algorithms for calculating $\beta^j_{(\cdot)}$ are shown. They differ w.r.t. the scalar product of the current iteration's gradient $\nabla H^j_{\boldsymbol{u}}$ and the previous iteration's $\nabla H^{j-1}_{\boldsymbol{u}}$:

$$\beta^j_\mathrm{HS} = \frac{\left\langle \nabla H^{j,T}_{\boldsymbol{u}},\ [\nabla H^j_{\boldsymbol{u}} - \nabla H^{j-1}_{\boldsymbol{u}}] \right\rangle}{\left\langle [\nabla H^j_{\boldsymbol{u}} - \nabla H^{j-1}_{\boldsymbol{u}}]^T,\ \nabla H^{j-1}_{\boldsymbol{u}} \right\rangle} \quad \text{Hestènes and Stiefel (1952)} \quad (5.23a)$$

$$\beta^j_\mathrm{FR} = \frac{\left\langle \nabla H^{j,T}_{\boldsymbol{u}},\ \nabla H^j_{\boldsymbol{u}} \right\rangle}{\left\langle \nabla H^{j-1,T}_{\boldsymbol{u}},\ \nabla H^{j-1}_{\boldsymbol{u}} \right\rangle} \quad \text{Fletcher and Reeves (1964)} \quad (5.23b)$$

$$\beta^j_\mathrm{PR} = \frac{\left\langle \nabla H^{j,T}_{\boldsymbol{u}},\ [\nabla H^j_{\boldsymbol{u}} - \nabla H^{j-1}_{\boldsymbol{u}}] \right\rangle}{\left\langle \nabla H^{j-1,T}_{\boldsymbol{u}},\ \nabla H^{j-1}_{\boldsymbol{u}} \right\rangle} \quad \text{Polak and Ribiere (1969)} \quad (5.23c)$$

The scalar product for the vector space of real-valued functions is defined as follows:

$$\left\langle \boldsymbol{a}^j, \boldsymbol{b}^j \right\rangle = \int_\tau \left[\boldsymbol{a}^j(t)\right]^T \boldsymbol{b}^j(t)\mathrm{d}t \quad (5.24)$$

In section 5.2.3 the aforementioned approaches to determine the conjugate gradient are applied to the nullspace OCP eq. (5.9) and compared to the steepest descent method.

Line Search Algorithms

Once the search direction is found, the step size α^j for descending along this direction has to be determined. The simplest choice is to use a fixed step size, i.e. $\alpha := \text{const.}$. Thereby, α^j has to be chosen sufficiently small to ensure convergence for all iterations. However, this obviously results in slow convergence in cases where larger steps are admissible. Thus, a more sophisticated approach is to use a variable step size, i.e. to solve the one-dimensional optimization problem in order to find an optimal $\alpha^{j,\star}$ that yields the fastest descent:

$$\alpha^{j,\star} = \underset{\alpha^j}{\operatorname{argmin}}\, L(\hat{\boldsymbol{u}}^j + \alpha^j \boldsymbol{s}^j) \tag{5.25}$$

Usually, the minimum of $L(\cdot)$ is found numerically by multiple evaluations of the cost function $L(\cdot)$ while varying α^j. Thus, one has to be aware of the computational costs of these additional cost function evaluations. In the following, α instead of α^j is used for convenience.

Recent work by Käpernick and Graichen (2014) examines two line search methods for solving OCPs: A polynomial approximation ("adaptive line search") of the cost function and the secant method ("explicit line search"). Both approaches, as well as the simple backtracking method, are presented and discussed in the following.

Backtracking According to the Wolfe conditions (Nocedal and Wright 2006), a sufficient decrease in the cost function $L(\cdot)$ is obtained in case the following inequality condition (Armijo condition) is met:

$$L\left(\boldsymbol{q}^j, \boldsymbol{u}^j + \alpha \boldsymbol{s}^j\right) \leq L\left(\boldsymbol{u}^j\right) + c\, \alpha \nabla H(\boldsymbol{u}^j)^T \boldsymbol{s}^j \tag{5.26}$$

The constant c determines the expected slope of the cost function L with the search direction $\boldsymbol{s}^j$ and the gradient of the Hamiltonian ∇H. Evaluating this condition by applying a sequentially decreasing step size α leads to the following simple alg. 4. It terminates either when the Armijo condition is met or when α is smaller than a lower threshold ε_α. This approach is fairly simple: it starts with a large step size which is iteratively decreased until a sufficient descent is reached. However, the selection of the expected slope c is difficult. In this thesis it is chosen empirically as a small value to prevent a premature abortion of the iteration.

Algorithm 4 *Backtracking*

$\alpha \leftarrow \alpha_0,\ \rho \in (0,1),\ c \in (0,1)$
while $\left(L\left(\boldsymbol{q}^j, \boldsymbol{u}^j + \alpha \boldsymbol{s}^j\right) \leq L\left(\boldsymbol{q}^j, \boldsymbol{u}^j\right) + c\,\alpha \nabla H(\boldsymbol{u}^j)^T \boldsymbol{s}^j\right)$ **and** $(\alpha > \varepsilon_\alpha)$ **do**
  $\alpha \leftarrow \rho\alpha$
end while
return α

Polynomial Approximation The main idea of the adaptive line search algorithm is the polynomial approximation of the cost function L w.r.t. the step size α and choosing its minimum as the approximated optimal step size. This algorithm is presented by Graichen and Käpernick (2011) for the application of OCPs. It is included in the open-source software framework GRAMPC and further discussions can be found in Käpernick and Graichen (2014). They report that a second order polynomial provides a sufficient approximation:

$$L(\boldsymbol{q}^j, \boldsymbol{u}^j + \alpha \boldsymbol{s}^j) \approx g(\alpha) := c_0 + c_1\alpha + c_2\alpha^2 \tag{5.27}$$

In order to determine the coefficients c_i, the cost function $L(\alpha^j)$ has to be evaluated at three points: $\alpha_1, \alpha_2, \alpha_3$. Graichen and Käpernick (2011) propose an equidistant distribution of these points, i.e. $\alpha_1 < \alpha_2 < \alpha_3$.
Solving the set of equations

$$L(\boldsymbol{q}^j, \boldsymbol{u}^j + \alpha_i \boldsymbol{s}^j) =: L_i = g(\alpha_i),\ i = 1,2,3 \tag{5.28}$$

yields the coefficients c_i:

$$\begin{aligned} c_0 &= \frac{\alpha_1(\alpha_1-\alpha_2)\alpha_2 L_3 + \alpha_2\alpha_3(\alpha_2-\alpha_3)L_1 + \alpha_1\alpha_3(\alpha_3-\alpha_1)L_2}{(\alpha_1-\alpha_2)(\alpha_1-\alpha_3)(\alpha_2-\alpha_3)} \\ c_1 &= \frac{(\alpha_2^2-\alpha_1^2)L_3 + (\alpha_1^2-\alpha^3)L_2 + (\alpha_3^2-\alpha_2^2)L_1}{(\alpha_1-\alpha_2)(\alpha_1-\alpha_3)(\alpha_2-\alpha_3)} \\ c_2 &= \frac{(\alpha_1-\alpha_2)L_3 + (\alpha_2-\alpha_3)L_1 + (\alpha_3-\alpha_1)L_2}{(\alpha_1-\alpha_2)(\alpha_1-\alpha_3)(\alpha_2-\alpha_3)} \end{aligned} \tag{5.29}$$

A minimum of $g(\alpha)$ exists when the curvature condition $c_2 > 0$ is fulfilled. Hence, the optimal step size $\alpha^\star$ is approximated by

$$\underset{\alpha}{\operatorname{argmin}}\, g(\alpha) := \alpha^\star = -\frac{c_1}{2c_2}. \tag{5.30}$$

In case $\alpha^\star$ lies outside the interval, α is set to the nearest limit. The interval $[\alpha_1, \alpha_3]$ can be adapted for the next gradient iteration by the scalar factors $\kappa^{+(-)}$ when $\alpha^\star$ is outside or near the borders of the bounding box. The method is summarized in alg. 5.

Approximation of the Secant Method Barzilai and Borwein (1988) suggest a two-point approximation of the *secant method* which is reported by Käpernick and

Algorithm 5 *Polynomial Approximation*

$\alpha_2^j \leftarrow \left(\alpha_1^j + \alpha_3^j\right)/2$
$c_0, c_1, c_2 \leftarrow \{L_i - g(\alpha_i) = 0, i \in (1,2,3)\}$
if $c_2 > 0$ **then**
$\alpha^\star \leftarrow -\frac{c_1}{2c_2}$

$$\alpha^j \leftarrow \begin{cases} \alpha_1 & \text{if } \alpha^\star < \alpha_1 \\ \alpha_3 & \text{if } \alpha^\star > \alpha_3 \\ \alpha^\star & \text{else} \end{cases}$$

else

$$\alpha^j \leftarrow \begin{cases} \alpha_1 & \text{if } L_1 + \varepsilon_g \leq \min\{L_2, L_3\} \\ \alpha_3 & \text{if } L_3 + \varepsilon_g \leq \min\{L_1, L_2\} \\ \alpha_2 & \text{else} \end{cases}$$

end if

$$[\alpha_1^{(j+1)}, \alpha_3^{(j+1)}] \leftarrow \begin{cases} \kappa^+[\alpha_1^j, \alpha_3^j] & \text{if } \alpha^\star \geq \alpha_1 + \varepsilon_\alpha^+(\alpha_3 - \alpha_1) \text{ and } \alpha_3 \leq \alpha_{\max} \\ \kappa^-[\alpha_1^j, \alpha_3^j] & \text{if } \alpha^\star \leq \alpha_1 + \varepsilon_\alpha^+(\alpha_3 - \alpha_1) \text{ and } \alpha_1 \geq \alpha_{\max} \\ [\alpha_1^j, \alpha_3^j] & \text{else} \end{cases}$$

Graichen (2014) for dynamic optimization problems. The step size α^j is chosen using the differences $\Delta \boldsymbol{u}_k^j := \boldsymbol{u}_k^j - \boldsymbol{u}_k^{j-1}$ and $\Delta \boldsymbol{s}_k^j := \boldsymbol{s}_k^j - \boldsymbol{s}_k^{j-1}$ as follows:

$$\alpha^j = \underset{\alpha > 0}{\operatorname{argmin}} \left\| \Delta \boldsymbol{u}_k^j + \alpha \Delta \boldsymbol{s}_k^j \right\|_{L_m^2[0,T]}^2 \tag{5.31}$$

$$\alpha^j = \frac{\Delta t \sum_{k=0}^{T} \left(\Delta \boldsymbol{u}_k^j\right)^T \Delta \boldsymbol{s}_k^j}{\Delta t \sum_{k=0}^{T} \left(\Delta \boldsymbol{s}_k^j\right)^T \Delta \boldsymbol{s}_k^j} =: \frac{\left\langle \Delta \boldsymbol{u}_k^j, \Delta \boldsymbol{s}_k^j \right\rangle}{\left\langle \Delta \boldsymbol{s}_k^j, \Delta \boldsymbol{s}_k^j \right\rangle} \tag{5.32}$$

This method is considered to be computationally very efficient since information of the last iteration is required, instead of evaluating the cost function $L(\cdot)$.

5.2.3 Example: Collision Avoidance 4-DOF pendulum

In the following, the problem formulation is exemplified by a 4-DOF pendulum robot. Furthermore, the aforementioned numerical methods to solve the dynamic optimization problem are investigated. Its configuration space $\mathcal{C} \in \mathbb{R}^4$ is defined by the four joint positions $\boldsymbol{q}$ whereas the task space $\mathcal{W} \in \mathbb{R}^2$ is specified by the position of its TCP $\boldsymbol{w} := [w_x, w_y]^T$:

$$\begin{aligned} \boldsymbol{q} &:= [q_1, q_2, q_3, q_4]^T \in \mathcal{C} \in \mathbb{R}^4 \\ \boldsymbol{w} &:= \left[w_x, w_y\right]^T \in \mathcal{W} \in \mathbb{R}^2 \end{aligned} \tag{5.33}$$

Since $\dim \mathcal{C} > \dim \mathcal{W}$, the robot is redundant. In the example, the robot follows a given straight-line task space trajectory. A point-like obstacle obstructs the free motion of the robot but can be avoided in nullspace. Parameters of the scenario are given in table 5.1. Redundancy is resolved

- using the instantaneous approach based on automatic supervisory control (cf. section 4.1.2),
- as a TPBVP as shown in section 5.2.1 and
- by minimizing the objective function using the (conjugate) gradient method (cf. section 5.2.2).

Problem Formulation

The cost function $l(\cdot)$ is the weighted sum of eqs. (5.13) and (5.14) and a collision avoidance term eq. (4.31):

$$l(\boldsymbol{q}, \dot{\boldsymbol{q}}, \hat{\boldsymbol{u}}) = \zeta_{\text{coll}} l_{\text{coll}}(\boldsymbol{q}) + \zeta_{\dot{q}} \underbrace{l_{\dot{q}}(\boldsymbol{q}, \dot{\boldsymbol{q}})}_{\frac{1}{2}\dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}}} + \zeta_{\hat{u}} \underbrace{l_{\hat{u}}(\hat{\boldsymbol{u}})}_{\frac{1}{2}\hat{\boldsymbol{u}}^T\hat{\boldsymbol{u}}} \tag{5.34}$$

$$l_{\text{coll}}(\boldsymbol{q}) = \begin{cases} \frac{1}{3}\left(d_a - d(\boldsymbol{q})\right)^3 & \text{if } d(\boldsymbol{q}) < d_a \\ 0 & \text{else} \end{cases} \tag{5.35}$$

The canonical equations yield

$$\dot{\boldsymbol{y}} = \left(\frac{\partial H}{\partial \boldsymbol{\lambda}}\right)^T = \begin{bmatrix} \boldsymbol{J}^{\#}\dot{\boldsymbol{w}}_{\text{des}} + \boldsymbol{N}\boldsymbol{u} \\ \hat{\boldsymbol{u}} \end{bmatrix} \overset{\hat{u}\leftarrow u^\star}{=} \begin{bmatrix} \boldsymbol{J}^{\#}\dot{\boldsymbol{w}}_{\text{des}} + \boldsymbol{N}\boldsymbol{u} \\ -\lambda_2/\zeta_{\hat{u}} \end{bmatrix} \tag{5.36}$$

$$\dot{\boldsymbol{\lambda}} = -\left(\frac{\partial H}{\partial \boldsymbol{y}}\right)^T = \begin{bmatrix} -\left(\nabla_q \boldsymbol{f}\right)^T\left(\zeta_{\dot{q}}\boldsymbol{f} + \boldsymbol{\lambda}_1\right) - \zeta_{\text{coll}}\left(\nabla_q l_{\text{coll}}\right)^T \\ -\boldsymbol{N}^T\left(\zeta_{\dot{q}}\boldsymbol{f} + \boldsymbol{\lambda}_1\right) \end{bmatrix} \tag{5.37}$$

$$\boldsymbol{y}|_{t_0} = [\boldsymbol{q}_0\ \boldsymbol{0}]^T,\ \boldsymbol{\lambda}|_{t_{\text{end}}} = \boldsymbol{0}. \tag{5.38}$$

For the calculation of the collision avoidance gradient, the point on the robot $\boldsymbol{r}_{\text{cp}}$ of body i which is closest to the point-like obstacle $\boldsymbol{r}_{\text{obs}}$ has to be determined. The gradient $\nabla_q l_{\text{coll}}$ and the required quantities are calculated by

$$\begin{aligned} \nabla_q l_{\text{coll}} &= \begin{cases} \left[d_a - d(\boldsymbol{q})\right]^2 \nabla_q d(\boldsymbol{q}) & \text{if } d(\boldsymbol{q}) < d_a \\ 0 & \text{else} \end{cases} \\ \nabla_q d(\boldsymbol{q}) &= \begin{cases} 1/d\left(\boldsymbol{J}_{TO,i} - \tilde{\boldsymbol{r}}_{\text{cp},i}\boldsymbol{J}_{RO,i}\right)^T\left(\boldsymbol{r}_{\text{cp}} - \boldsymbol{r}_{\text{obs}}\right) & \text{if } d(\boldsymbol{q}) < d_a \\ 0 & \text{else} \end{cases} \\ d(\boldsymbol{q}) &= \sqrt{\left[\boldsymbol{r}_{\text{obs}} - \boldsymbol{r}_{\text{cp}}(\boldsymbol{q})\right]^2}. \end{aligned} \tag{5.39}$$

Parameter	Symbol	Value
Inverse Kinematics		
Definition task space	$\boldsymbol{w}$	$\begin{pmatrix} w_x & w_y \end{pmatrix}^T \in \mathbb{R}^2$
Sample time	Δt	0.04 s
Nullspace weighting matrix	$\boldsymbol{W}$	$\mathrm{diag}(1,1,1,1)$
Numerical drift compensation factor	K	1
Activation distance for collision avoidance	d_a	0.2
Scaling factor for collision avoidance	ζ_{coll}	$50\ \frac{1}{\mathrm{m}^3}$
Scaling factor for comfort pose	ζ_{cmf}	0
Scaling factor for joint limit avoidance	ζ_{jla}	0
Scaling factor for velocity penalty	$\zeta_{\dot{\boldsymbol{q}}}$	$2\ \frac{\mathrm{s}^2}{\mathrm{rad}^2}$
Scaling factor for input damping	$\zeta_{\hat{\boldsymbol{u}}}$	$2\ \frac{\mathrm{s}^2}{\mathrm{rad}^2}$
Robot and Trajectory		
Link length	l	1
Initial configuration	$\boldsymbol{q}^0$	$\begin{pmatrix} \pi/6, -\pi/6, -\pi/6, -\pi/3 \end{pmatrix}^T$
Initial guess	$\hat{\boldsymbol{u}}^0$	$\begin{pmatrix} 0,0,0,0 \end{pmatrix}^T$
Task space trajectory		
Start position	$\boldsymbol{w}_0$	$\begin{pmatrix} 3.73 & 1 \end{pmatrix}^T$
End position	$\boldsymbol{w}_{\mathrm{end}}$	$\begin{pmatrix} 0.5 & 0 \end{pmatrix}^T$
Execution time	t_{traj}	4.0 s
Obstacle position	$\boldsymbol{r}_{\mathrm{obs}}$	$\begin{pmatrix} 1.0 & 1.1 \end{pmatrix}^T$
TPBVP Solution		
Matlab solver algorithm		`bvp4c`
Relative tolerance	`RelTol`	10^{-2}
Nodes of initial mesh		20
Gradient Method Approach		
Maximum number of iterations	$j_{\max}$	100
Lower barrier for exit condition	ϵ	10^{-4}
Step size		
Fixed step size	α_0	0.01
Variable step size	$\alpha_{0,\mathrm{bt}}$	0.05
	ρ	0.62
	c_{armijo}	10^{-4}
	ϵ_α	$\alpha_{0,\mathrm{bt}} \cdot \rho^{10}$

Table 5.1: Parameters of the example of the 4-DOF robot following a straight-line task space path while avoiding collisions with an obstacle by suitable nullspace motion (cf. section 5.2.3).

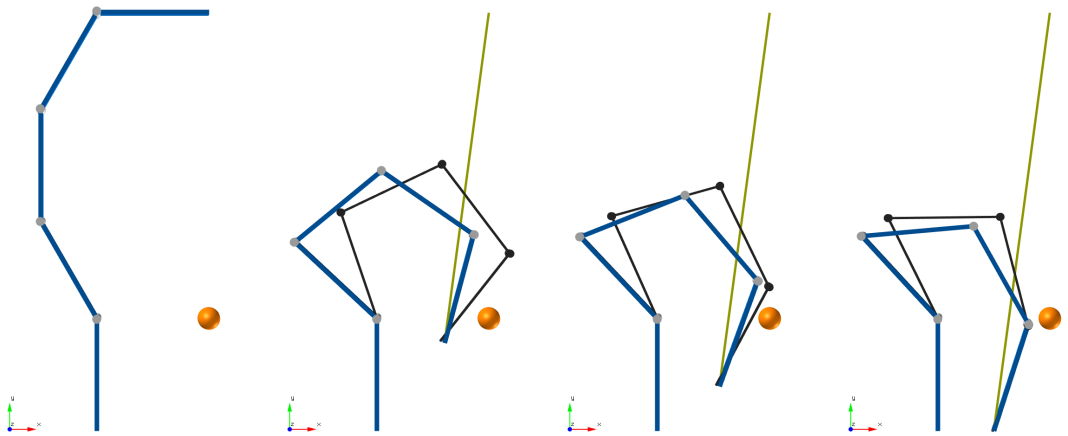
Note that the translational and rotational Jacobians of the ith body's origin, as well as the closest point $\boldsymbol{r}_{\text{cp}}$ depend on the robot configuration $\boldsymbol{q}$. The task space trajectory $\boldsymbol{w}(t),\ t \in [0, t_{\text{traj}}]$ is given by

$$\begin{aligned} \boldsymbol{w}_{\text{des}}(t) &= \boldsymbol{w}_0 + s(t)\,(\boldsymbol{w}_{\text{end}} - \boldsymbol{w}_0) & s(t) &= {}^{10}/_{t_{\text{traj}}^3} t^3 - {}^{15}/_{t_{\text{traj}}^4} t^4 + {}^{6}/_{t_{\text{traj}}^5} t^5 \\ \dot{\boldsymbol{w}}_{\text{des}}(t) &= \dot{s}(t)\,(\boldsymbol{w}_{\text{end}} - \boldsymbol{w}_0) & \dot{s}(t) &= {}^{30}/_{t_{\text{traj}}^3} t^2 - {}^{60}/_{t_{\text{traj}}^4} t^3 + {}^{30}/_{t_{\text{traj}}^5} t^4. \end{aligned} \tag{5.40}$$

The trajectory of the path parameter $s(t)$ is chosen as a 5th order polynomial which fulfills the boundary conditions

$$s(0) = 0,\ s(t_{\text{traj}}) = 1,\ \dot{s}(0) = \dot{s}(t_{\text{traj}}) = 0,\ \ddot{s}(0) = \ddot{s}(t_{\text{traj}}) = 0. \tag{5.41}$$

Numerical Solution



(a) t=0.0 s **(b)** t=2.68 s **(c)** t=3.0 s **(d)** t=4.0 s

Figure 5.1: Example with a 4-DOF pendulum showing the benefit of the optimization of the trajectory in nullspace. The robot follows a straight-line task space trajectory from top $\boldsymbol{w}_0 = (1, 3.73)^T$ to bottom $\boldsymbol{w}_{\text{end}} = (0.5, 0)^T$, its trace is drawn in *green*. At $(1, 1)$, the robot encounters an obstacle (*orange* sphere), which can be avoided by suitable nullspace motions. The optimized solution (TPBVP) is drawn in *blue* while the initial guess (instantaneous solution, cf. section 4.1.2) is depicted in *gray*.

This dynamic optimization problem (with parameters given in table 5.1) is solved as a TPBVP and by using the presented gradient approaches. First, the optimal solution is compared to the instantaneous automatic supervisory control approach based on this typical example with collision avoidance. Furthermore, the rate of convergence for different conjugate gradients and line search methods is discussed. A quantitative evaluation of the proposed methods is given in section 5.2.4.

General Remarks on the Optimal Solution Fig. 5.1 shows several snapshots of the instantaneous and the optimized solution: While the optimal solution (blue robot) avoids the obstacle by adjusting its motion plan from the outset, the instantaneous approach (gray robot) reacts only when the minimum distance falls below the activation distance d_a. The drawback of this late reaction becomes apparent when the costs for joint velocities and collision avoidance are regarded in fig. 5.2: At $t \approx 2.75\,\mathrm{s}$: the robot approaches the obstacle with greater proximity and has to apply high joint velocities in order to avoid it. By comparison, the optimized solution accepts slightly higher velocities at the beginning in order to achieve a more favorable configuration near the obstacle.

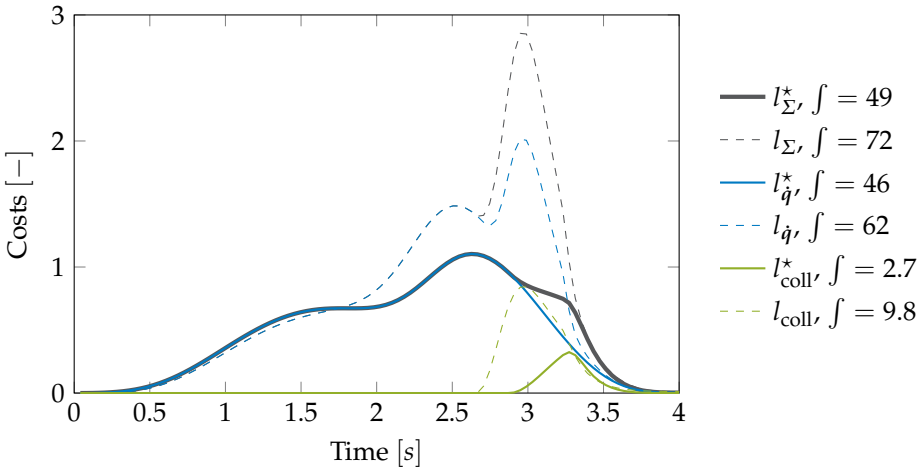


Figure 5.2: Cost as a function of time of the instantaneous *(dashed)* and the optimized solution *(solid)* for the scenario shown in fig. 5.1 with the parameters given in table 5.1. The individual costs for joint velocities $l_{\dot{q}}(t)$ *(blue)* and collision avoidance $l_{\mathrm{coll}}(t)$ *(green)* are cumulated to the overall costs $l_\Sigma(t)$ *(gray)*.

Conjugate Gradient First, different approaches modifying the search direction using conjugate gradients (section 5.2.2) are compared, while the step size α remains fixed as $\alpha = \alpha_0$. All approaches yield similar optimization results. However, their rate of convergence differs: As depicted in section 5.2.3, the algorithm of Fletcher and Reeves (1964) shows the fastest convergence compared to the other approaches for the collision avoidance scenario. Furthermore, it should be noted that the algorithm of Hestènes and Stiefel (1952) converges faster than the steepest descent while no improvement can be observed with the formulation of Polak and Ribiere (1969). Additionally, the proportional gain β of the conjugate gradients is shown over the iterations. While for the method proposed by Fletcher and Reeves (1964) $\beta \geq 0$ holds, β may become negative for the Hestènes and Stiefel (1952) and Polak and Ribiere (1969) algorithms.

Line Search Line search algorithms aim to find the optimal step size for descent as far as possible along the given search direction. The line search approaches

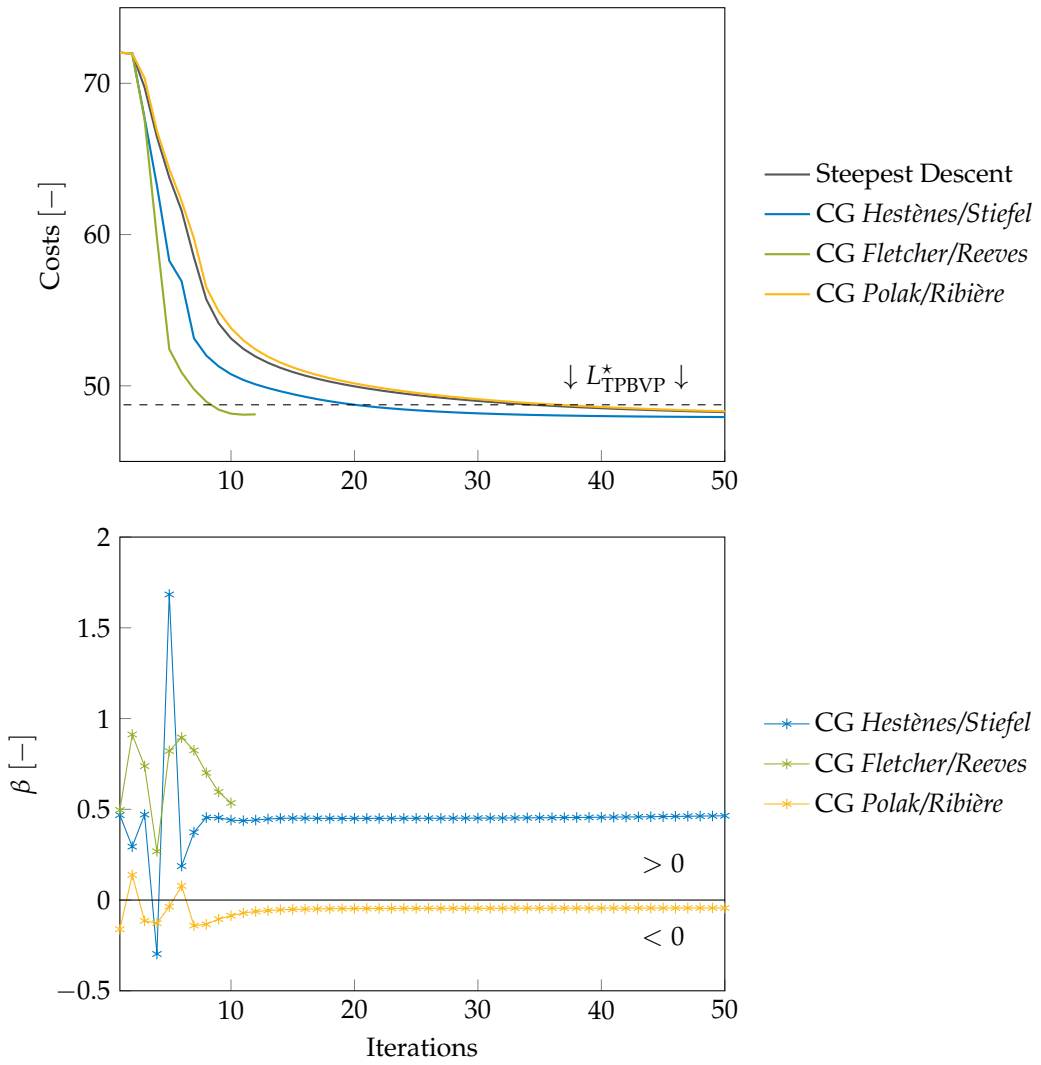


Figure 5.3: Convergence for different conjugate gradient algorithms for the collision avoidance example: The value of the cost function L and the β factor are plotted as a function of the the number of iterations. For comparison, the costs from the TPBVP solution $L^{\star}_{\mathrm{TPBVP}}$ are drawn as a dashed line in the upper figure.

presented in section 5.2.2 are applied to the collision avoidance example with the simple steepest descent search direction. The results are shown in fig. 5.4: The backtracking and explicit methods use information from the previous iteration. Thus, they descend with a fixed step size in the first iteration. By contrast, the adaptive scheme approximates the optimal $\alpha^{\star}$ by three evaluations of the cost function $L(\alpha)$ and already descends faster at iteration 1. The resulting step sizes are also depicted in fig. 5.4. One may note that all algorithms with variable step sizes show a faster convergence compared to the conservatively chosen fixed step size.

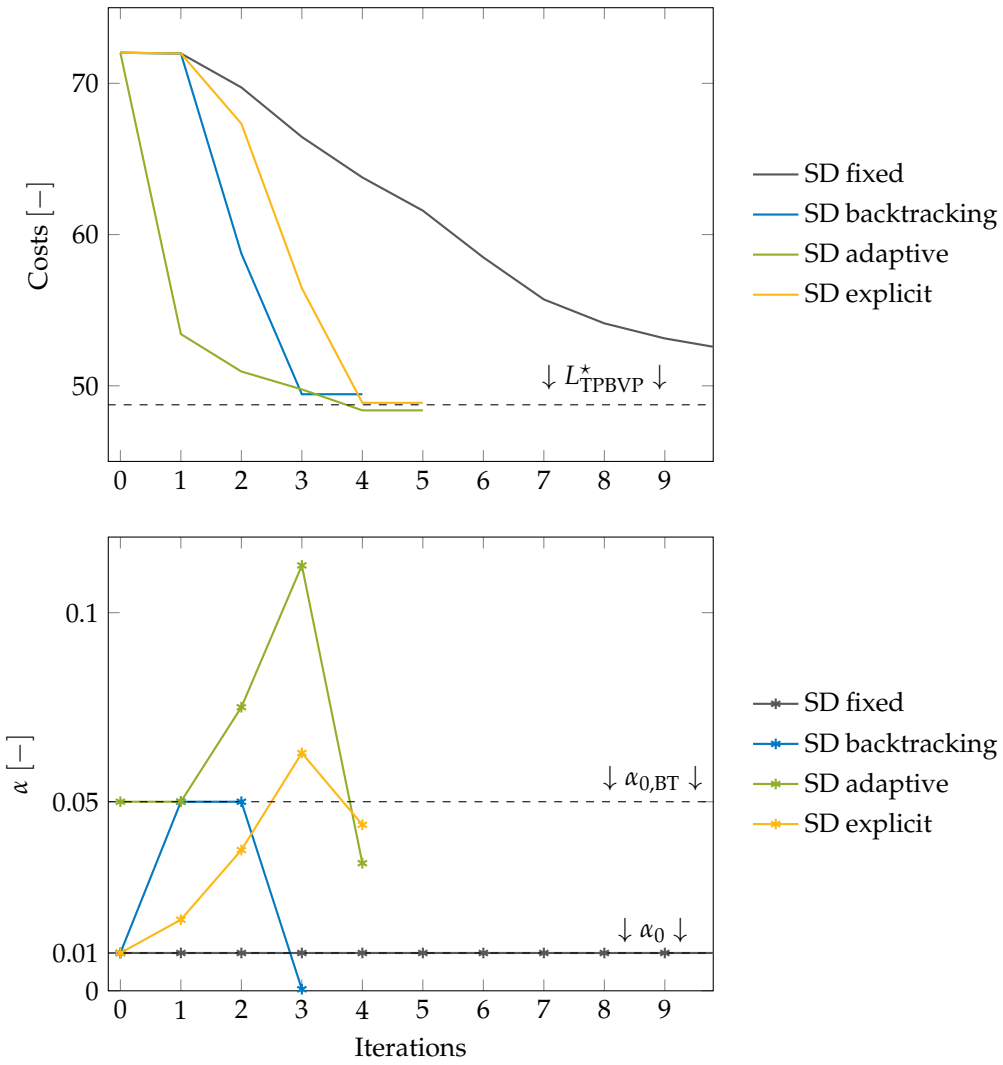


Figure 5.4: Convergence for different line search algorithms for the collision avoidance example: The value of the cost function L and the factor β calculated by the conjugate gradient algorithms section 5.2.2 are shown as a function of number of iterations.

5.2.4 Evaluation of the Numerical Methods

Problem Formulation

The presented approaches and variations are evaluated for the example given in section 5.2.3 using random initial configurations $\boldsymbol{q}_0$ and task space goal positions $\boldsymbol{w}_{\text{end}}$. For this evaluation, the cost function is modified as follows:

$$l(\boldsymbol{q}, \dot{\boldsymbol{q}}, \hat{\boldsymbol{u}}) = \zeta_{\text{cmf}} l_{\text{cmf}}(\boldsymbol{q}) + \zeta_{\text{jla}} l_{\text{jla}}(\boldsymbol{q}) + \zeta_{\dot{q}} \underbrace{l_{\dot{q}}(\boldsymbol{q}, \dot{\boldsymbol{q}})}_{\frac{1}{2}\dot{\boldsymbol{q}}^T \dot{\boldsymbol{q}}} + \zeta_{\hat{u}} \underbrace{l_{\hat{u}}(\hat{\boldsymbol{u}})}_{\frac{1}{2}\hat{\boldsymbol{u}}^T \hat{\boldsymbol{u}}} \tag{5.42}$$

Parameter	Symbol	Value
Evaluation		
Number of experiments	n_{exp}	100
Varied Parameters		
Initial Configuration	$\boldsymbol{q}^0$	$q_i^0 \in \mathbb{R} \cap [-2/3\pi, 2/3\pi]$
Task Space Goal	$\boldsymbol{w}_{\text{end}}$	$w_{\text{end},x} \in \mathbb{R} \cap [-1.5, 1.5]$
		$w_{\text{end},y} \in \mathbb{R} \cap [0, 3]$
Cost Function		
Scaling factor for collision avoidance	ζ_{coll}	0
Joint Limits	$[q_{\text{min,i}}, q_{\text{max,i}}]$	$[-\pi, \pi]$
Comfort Pose		
Scaling factor	ζ_{cmf}	0.5
Comfort Pose	$\boldsymbol{q}_{\text{cmf}}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}^T$
Joint Limit Avoidance		
Scaling factor	ζ_{jla}	1
Lower Soft Limit	$q_{\text{min,soft,i}}$	$q_{\text{min,i}} + \pi/2$
Upper Soft Limit	$q_{\text{max,soft,i}}$	$q_{\text{max,i}} - \pi/2$
Scaling factor for velocity penalty	$\zeta_{\dot{\boldsymbol{q}}}$	2
Scaling factor for input damping	$\zeta_{\hat{\boldsymbol{u}}}$	2

Table 5.2: Adaption of parameters from table 5.1 for the statistical evaluation presented in section 5.2.4 of the 4-DOF robot following a straight-line task space path while optimizing joint velocities and its kinematic configuration.

$$
\begin{aligned}
l_{\text{cmf}}(\boldsymbol{q}) &= \frac{1}{2} \sum_{i=1}^{n} \frac{(q_i - q_{\text{cmf}})^2}{(q_{i,\text{max}} - q_{i,\text{min}})^2} \\
l_{\text{jla}}(\boldsymbol{q}) &= \sum_{i=1}^{n} H_{\text{jla},i} \\
H_{\text{jla},i} &= \begin{cases} \frac{(q_i - q_{\text{min,soft},i})^{\sigma}}{(q_{\text{min},i} - q_{\text{min,soft},i})^{\sigma}} & \text{if } (q_i < q_{\text{min,soft},i}) \\ \frac{(q_i - q_{\text{max,soft},i})^{\sigma}}{(q_{\text{max},i} - q_{\text{max,soft},i})^{\sigma}} & \text{else if } (q_i > q_{\text{max,soft},i}) \\ 0 & \text{else} \end{cases}
\end{aligned}
\tag{5.43}
$$

For the quantitative comparison of the numerical approaches in many random settings, the term for collision avoidance is omitted. The parameters shown in table 5.1 are retained, except for modifications listed in table 5.2.

The following results are obtained solving the dynamic optimization problem for $n_{\text{exp}} = 100$ random settings. The initial configuration $\boldsymbol{q}_0$ and goal position

$\boldsymbol{w}_{\text{end}}$ have a uniform distribution within the intervals

$$\begin{aligned} q_{0,i} &\in \mathbb{R} \cap [-2/3\pi, 2/3\pi] \\ w_{\text{end},x} &\in \mathbb{R} \cap [-1.5, 1.5] \\ w_{\text{end},y} &\in \mathbb{R} \cap [0, 3]. \end{aligned} \tag{5.44}$$

Evaluation Criteria

For the evaluation, the following aspects are considered:

- *Improvement:* Integral of optimal costs $l^\star(\cdot, t)$ over time compared to the instantaneous costs $l_0(\cdot, t)$:

$$\text{Improvement} := \frac{\int_{t_0}^{t_{\text{end}}} l^\star(\cdot)\,\mathrm{d}t - \int_{t_0}^{t_{\text{end}}} l_0(\cdot)\,\mathrm{d}t}{\int_{t_0}^{t_{\text{end}}} l_0(\cdot)\,\mathrm{d}t} \tag{5.45}$$

  A negative value means an improvement w.r.t. the costs while a positive value is a deterioration (which is considered as a failure).
- *Relative Improvement*: Difference between the improvement obtained by the method yielding the minimal costs and the considered method. A value of 0 means that a method yields the minimal costs among the considered solutions.
- *Success Rate:* A solution is considered as successful when its cost improvement differs by 5% relative to the method yielding the minimal costs. A value of 1 means 100% success while a value of 0 means no success.
- *Computation Time:* Computation time of gradient based methods $T_{\text{calc,GM}}$ is compared relatively to the computation time needed by the TPBVP solution ($T_{\text{calc,BVP}}$):

$$\text{Rel. Computation Time} := \frac{T_{\text{calc,GM}} - T_{\text{calc,BVP}}}{T_{\text{calc,BVP}}} \tag{5.46}$$

Absolute Results

In order to evaluate the expectable improvement and computation time of the optimization, a statistical evaluation of the reference TPBVP solution is performed. The cumulative costs for joint velocities $L_{\dot{\boldsymbol{q}}}$, comfort pose L_{cmf} and joint limit avoidance L_{jla} resulting from the optimization are compared to those of the initial guess. The costs for input damping $L_{\hat{\boldsymbol{u}}}$ are neglected since they are not taken into account by the instantaneous solution. As depicted in fig. 5.5, the costs are reduced in average by 16% while the distribution is skew symmetric. It should be pointed out that this quantification of the improvement relies on the absolute

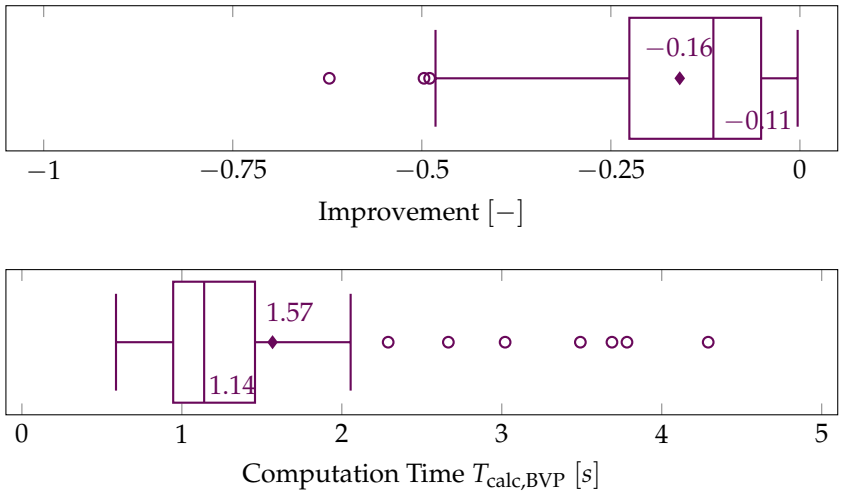


Figure 5.5: *Top:* Improvement of the cumulative costs for joint velocities $L_{\dot{q}}$, comfort pose L_{cmf} and joint limit avoidance L_{jla} of the TPBVP solution relative to the initial guess (instantaneous solution). *Bottom:* Absolute computation time needed for solving the TPBVP. Time measured using MATLAB R2016A, UBUNTU 15.10 on an *Intel® i5-4310U@ 2.00GHz*. The average value is denoted by ♦ and the median as the vertical line in the box. Tukey Boxplot, Whiskers ±1.5 Interquartile Range.

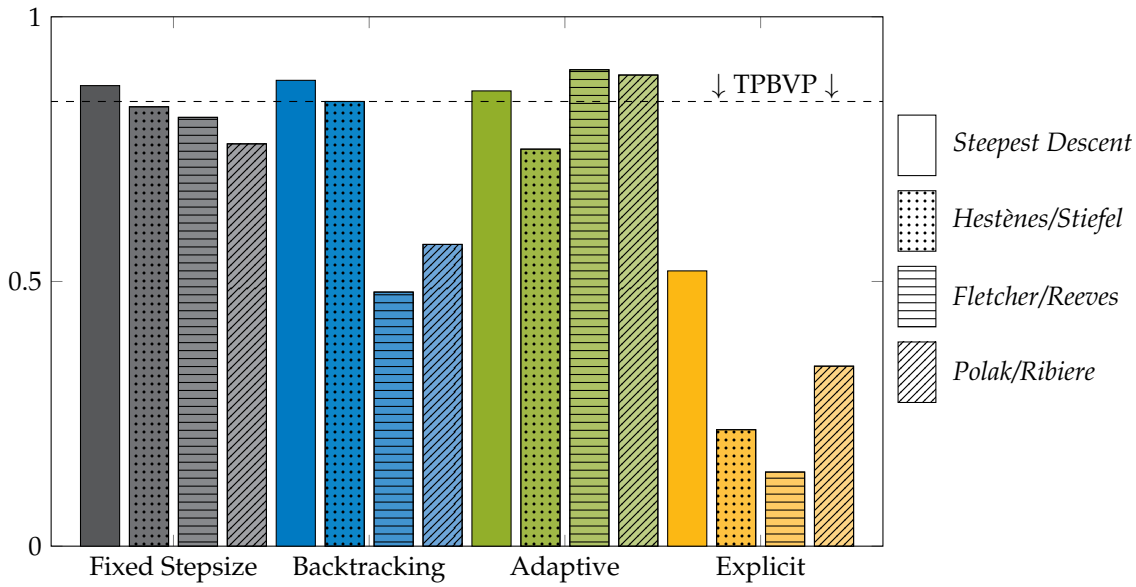


Figure 5.6: Success Rate of the various combinations of conjugate gradient and line search algorithms. For comparison, the success rate of the TPBVP solution is drawn as a dashed line.

weighting of the cost function in this example. In order to quantify the computational effort, the computation time $T_{\text{calc,BVP}}$ is measured as well: The average computation time is 1.57 s while half of the experiments converge after 1.14 s using the MATLAB R2016A routine BVP4C on an *Intel® i5-4310U@ 2.00GHz*.

Relative Results

Results obtained by the different methods are shown relative to the TPBVP (→ computation time) or minimal cost (→ relative improvement) solution. In the following, the approaches presented in section 5.2.2 for calculating the conjugate gradient and in section 5.2.2 for determining the optimal step size are combined with each other.

Success Rate In fig. 5.7 the success rate of the conjugate gradient/line search combinations is shown. As defined in section 5.2.4, an optimization is considered as *successful* when the resulting improvement compared to the instantaneous solution by less than 5% relative to the solution yielding the minimal costs. While the TPBVP solution is successful in 84% of the experiments, the highest success rate is achieved by the adaptive line search algorithm and the Fletcher/Reeves conjugate gradient. Stable results are also obtained using a fixed step size. The explicit line search algorithm performed the worst for all combinations. However, the convergence depends heavily on the parametrization of the cost function, the initial step sizes and the problem itself.

A more detailed illustration of the convergence is given in fig. 5.7. This diagram shows the difference of the improvement achieved by the respective method to the method yielding the minimal costs ("most optimal" solution). In particular, a bad convergence is obtained by the backtracking line search with the Fletcher/Reeves and Polak/Ribière conjugate gradient and the explicit line search (all CG versions). By contrast, the versions with fixed step size and with adaptive line search yield the best results.

Computation Time Apart from the quality of the results as investigated in the previous paragraph, the computational effort is of particular importance for real world applications. In fig. 5.8 the relative computation times of the different methods are given w.r.t. the computation time of the TPBVP solution. In doing so, only successful experiments are considered. While the quality of the results is high for the fixed step size algorithms, computation times are longer. However, using the conjugate gradient methods, in particular the Fletcher/Reeves algorithm, the convergence can be accelerated significantly. A similar behavior can be observed using the backtracking line search, which shows even faster convergence but worse reliability. The best results are achieved at a high quality level by the adaptive line search strategy in combination with the Fletcher/Reeves conjugate gradient. The lowest computational effort is observed for the explicit line search methods. However, its optimization process is often stopped prematurely, before the solution is optimal.

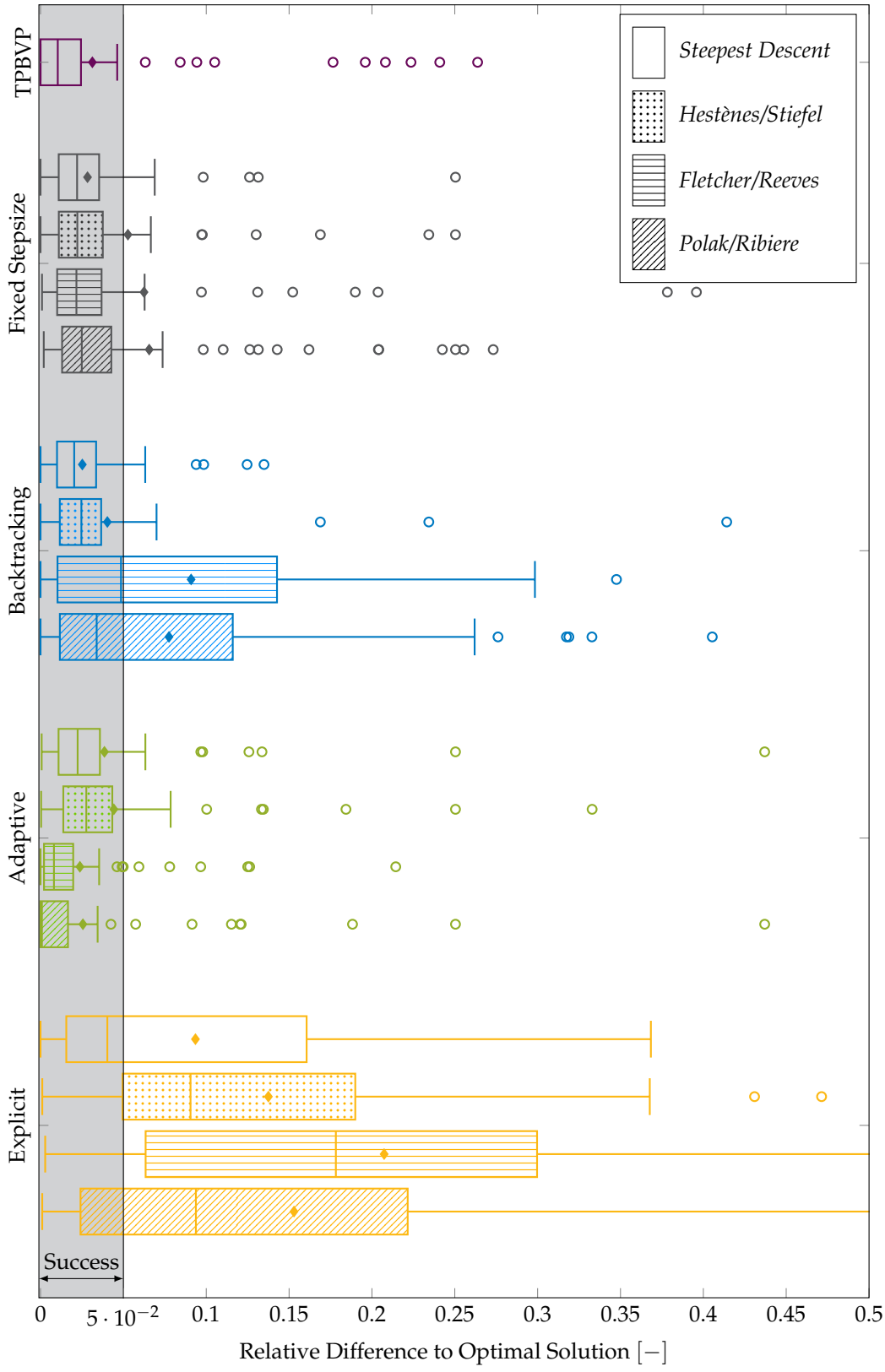


Figure 5.7: Relative Improvement: Difference of improvement relative to the best solution. A value of 0 means the solution with minimum costs. All samples within the range of 5% are considered in section 5.2.4 as a success (*gray*). The average value is denoted by ♦ and the median as the vertical line in the box. Tukey Boxplot, Whiskers ±1.5 Interquartile Range.

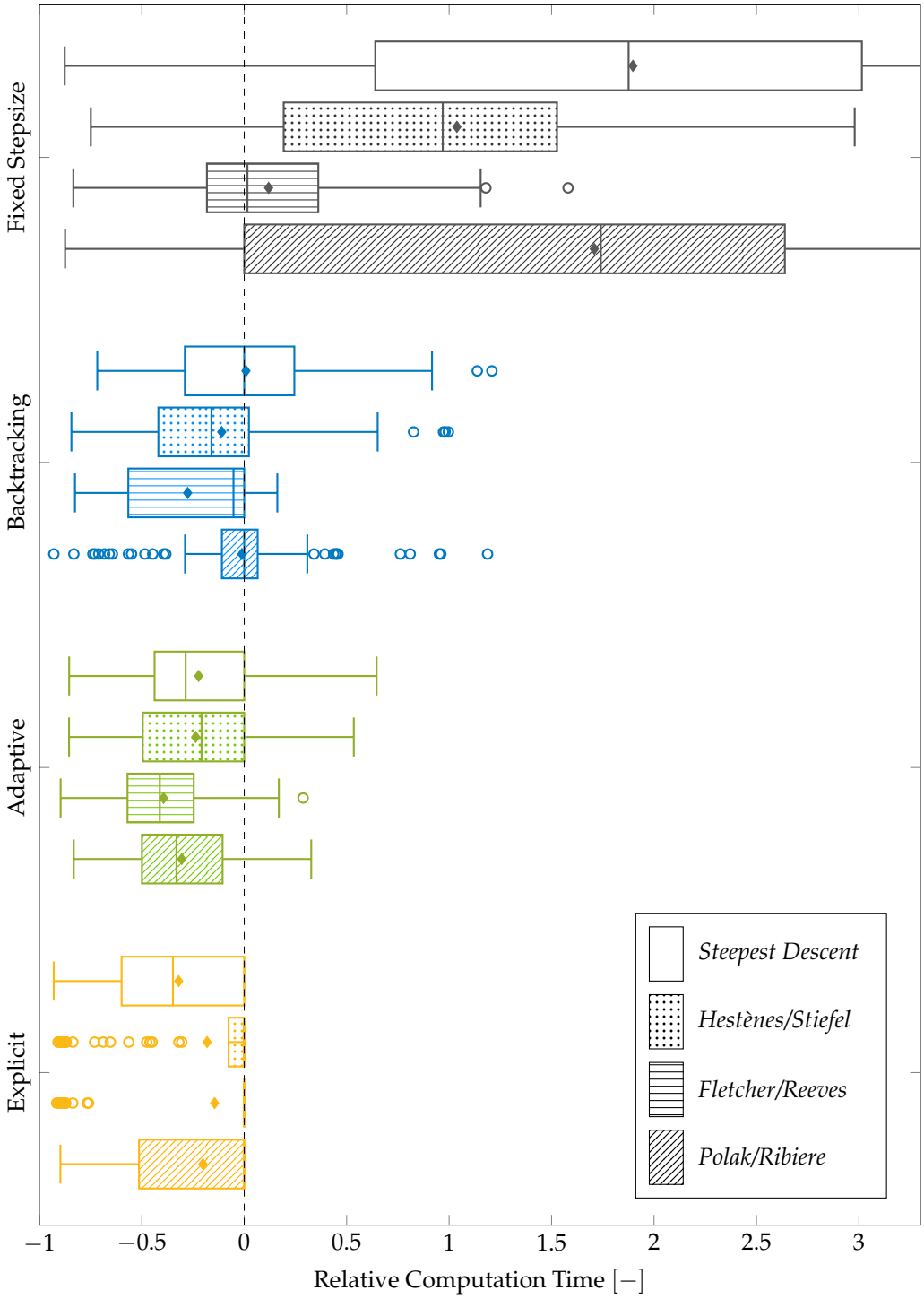


Figure 5.8: Comparison of the computation time T_{calc} needed for the solution of the dynamic optimization problem relative to the solution as a TPBVP. The average value is denoted by ♦ and the median as the vertical line in the box. Tukey Boxplot, Whiskers ± 1.5 Interquartile Range.

5.3 Real Time Application

The numerical methods presented in the previous section are capable of solving dynamic optimization problems formulated as TPBVPs. Thus, one can determine the optimal joint space trajectory for a given task space trajectory optimizing the nullspace motion of a redundant robot offline, i.e. before initiating the motion. For several applications, a real time optimization is desirable: For example, the task space trajectory may be modified while moving or new obstacles can appear. Furthermore, when the trajectory is optimized online, the motion can be started immediately. Even under the assumption that the optimization problem is sufficiently well known in advance, the computational load for longer trajectories is usually too high to be computed in one time interval of a typical real time scheme.

A well-known approach from control theory is the model predictive control (MPC) or Moving Horizon scheme (Diehl et al. 2009). This section indicates how the MPC scheme can be applied to the optimization problem for resolving redundancy using the numerical gradient methods presented in the previous sections.

5.3.1 MPC Algorithm

The main idea of MPC can be summarized as follows: A large optimal control problem is divided into several smaller sub-problems for a finite prediction horizon T_{horizon} starting from the current time increment t^k. Once the sub-problem is solved, the initial part (usually smaller than the prediction horizon) of the calculated input trajectory $\boldsymbol{u}^\star(\tau_k')$ with $\tau_k' := [t_k, t_k + \Delta t_{\text{mpc}}]$ and $\Delta t_{\text{mpc}} < T_{\text{horizon}}$ is applied to the system while the sub-problem of the subsequent time window τ_{k+1} is being solved. The basic scheme is depicted in alg. 6.

Algorithm 6 *Model Predictive Control* (MPC) Algorithm

$t_{\text{start}} = t^0$, $T_{\text{end}} = t^N$, $\Delta t_{\text{mpc}} = t^{k+1} - t^k$
for $k = 0$ to N **do**
  $\tau_k \leftarrow [t_k, t^k + T_{\text{horizon}}]$
  $\tau_k' \leftarrow [t_k, t^k + \Delta t_{\text{mpc}}]$
  determine $\boldsymbol{u}^\star(\tau_k)$ using $\boldsymbol{u}^0(\tau_k) = \boldsymbol{u}^\star_{\text{previous}}(\tau_k)$
  apply $\boldsymbol{u}^\star(\tau_k')$
end for

5.3.2 Implementation

Real Time Capability

In order to assure the real time capability of the implementation, each calculation routine should provide a feasible solution within a fixed time span. The numerical solution of the dynamic optimization problem contradicts this hard real time constraint since the number of iterations to convergence varies. Assuming that each iteration requires a similar computation time, this issue can obviously be resolved by limiting the number of iterations. This is only applicable in case the "not yet converged" or *sub-optimal* solution still provides a feasible trajectory. This requirement is inherently fulfilled by the considered system formulation shown in section 5.2.1. Using a "warm start", i.e. the previous solution $\boldsymbol{u}^{\star}_{\text{previous}}$ initializes the subsequent optimization problem, convergence to the optimal solution over several iterations is shown by Graichen and Kugi (2010).

In Schütz, Buschmann, et al. (2014) the author reports the real time implementation for the 9-DOF CROPS manipulator allowing one iteration per sub-problem. Thereby, the prediction horizon is 0.15 s with a sampling time of 10 ms.

Continuity Considerations

While solving the optimization problem for a moving horizon and subsequently applying the optimal control inputs $\boldsymbol{u}^{\star}(\tau')$, one has to assure that the overall control trajectory $\boldsymbol{u}^{\star}(t)$ is continuous. Solving the OCP for the system formulation on the velocity level eq. (5.5) by a MPC scheme illustrates this problem: As depicted in fig. 5.9-*left* for a single joint velocity of a 4-DOF collision avoidance example, the resulting trajectory (*blue*) is non-smooth, although the single trajectories solving the sub-problem are individually smooth. This can be explained by the fact that the solution has non-zero initial nullspace velocities since nullspace accelerations are not taken into account by the cost function. This issue is encountered in the extended system formulation presented in section 5.1.3. Using a simple state augmentation, the new system input $\hat{\boldsymbol{u}}$ for accelerations instead of velocities in nullspace is defined. This extension results in continuity on the velocity level as shown in fig. 5.9-*right*.

Prediction Horizon

The prediction horizon T_{horizon} has a significant influence on the optimization result. This insight is evident: The larger the prediction horizon, the earlier the optimization takes future events into account. I.e., the manipulator can adapt its motion earlier to choose a more favorable configuration. Regarding the squared joint velocities $\dot{\boldsymbol{q}}^T\dot{\boldsymbol{q}}$ of the self-collision scenario (depicted in fig. 5.10), this effect

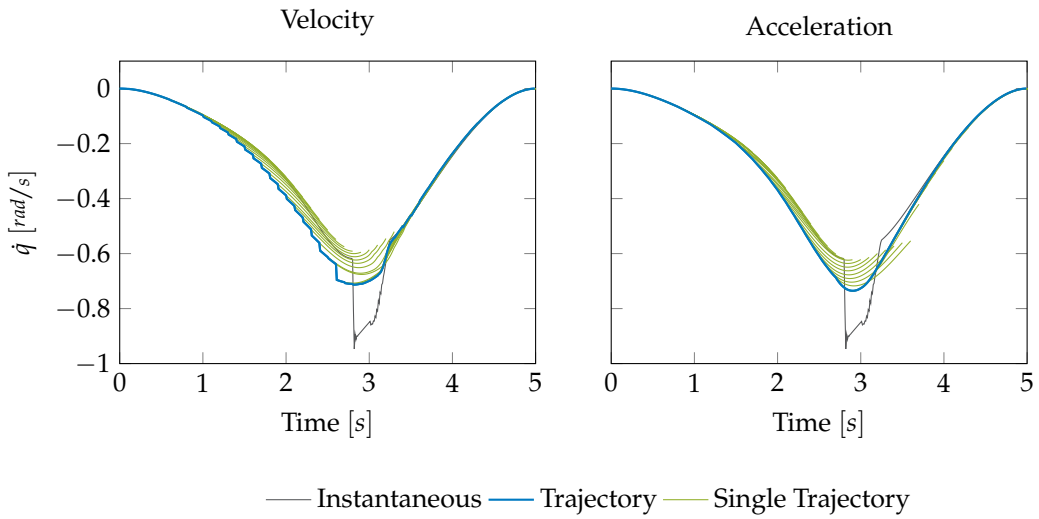


Figure 5.9: Joint velocity for the 4-DOF robot at the collision avoidance scenario using the MPC solution. Thereby, the system formulation with an input on the velocity level (*left*) shows discontinuities while the one on the acceleration level as presented in section 5.1.3 remains smooth (*right*).

can be quantified: The manipulator moves on a straight-line task space path from bottom to top. While moving, the manipulator prevents a self-collision of the arm with the prismatic joint by an evasive nullspace movement. Following the instantaneous solution (*gray*) as shown in fig. 5.11, high joint velocities are needed to avoid the collision. Using a large prediction horizon (e.g. $T_{\text{horizon}} = 2\,\text{s}$, *green*), the collision is anticipated early and high velocities are prevented. By comparison, the smaller horizon of $T_{\text{horizon}} = 0.15\,\text{s}$ (*blue*) yields a much smaller improvement compared to the larger horizon.

5.4 Remarks

In this chapter, a novel method for the optimization of inverse kinematics for redundant robots has been introduced. By applying the method in a moving horizon calculation scheme, a predictive optimization of the respective nullspace motion can be applied in real time. Thus, future constraints and events can already be taken into account during the calculation of the current time increment. This method is based on a suitable system formulation for an indirect optimization scheme. Two contributions of this thesis, the decoupling and state augmentation enable its efficient solution. Besides the numerical solution as a TPBVP, the (conjugate) gradient method is shown in greater detail. Using the example of a redundant 4-DOF pendulum, the efficiency and robustness of several conjugate gradient versions and adaptive step size algorithms are examined. By evaluating numerous random scenarios of this example, the combination of the Fletcher/Reeves conjugate gradient and the adaptive line search algorithm

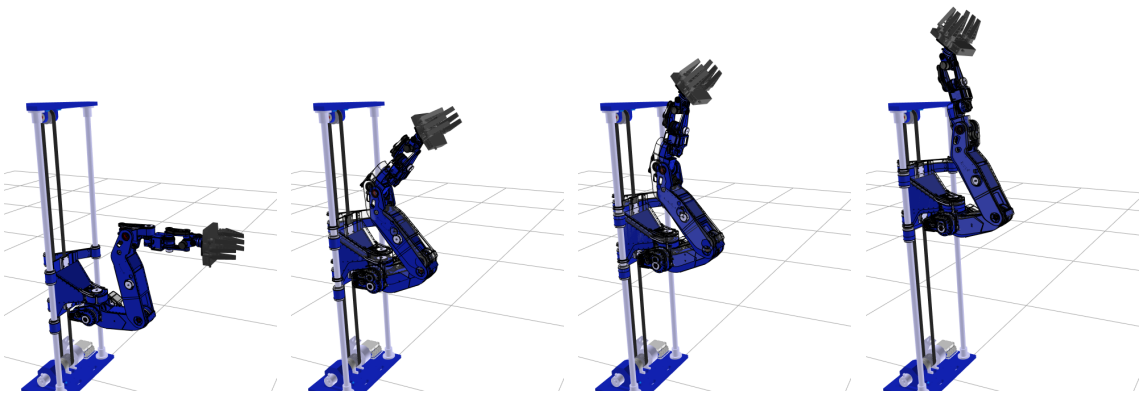


Figure 5.10: Collision Avoidance scenario for 9-DOF harvesting manipulator (prototype 1). The end effector moves on a straight-line path from bottom to top. A self-collision of the arm with the manipulator's prismatic joint is avoided by suitable nullspace movements.

showed the best results regarding success rate and computation time. Furthermore, the real time application of the proposed method in a moving horizon scheme was introduced.

The solution of optimal control problems by an indirect method has several advantages: First, it offers highly accurate solutions to the dynamic problem. Second, the numerical solution e.g. by the conjugate gradient method is computationally efficient. Iterations can be interrupted before the problem is fully converged, since the current estimate of the optimal solution is always applicable. This fact is of particular importance regarding real time applications. However, several drawbacks have to be considered when applying the proposed method. The trajectory is optimized *locally* around the current estimate of the optimal solution. This makes it sensitive to the initial guess. Since optimization takes place

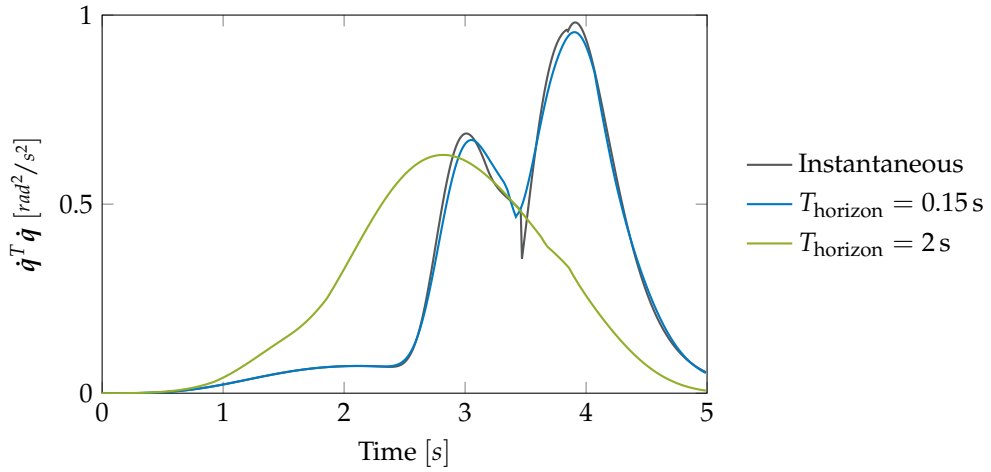


Figure 5.11: Squared joint velocities of revolute joints in self-collision scenario for the harvesting robot calculated by local inverse kinematics and MPC approach with a moving horizon of 0.15s and 2s.

only in nullspace for the proposed system formulation, an initialization of zeros or using the gradient $\nabla_q H$ of the current timestep leads to good results. Although the optimization may end in a local minimum, the solution can be expected to be more favorable compared with the instantaneous calculation scheme (cf. section 4.1.2). Another drawback is the sensitivity of the method to the relative weights within the cost functional. The coefficients $\zeta_{(\cdot)}$ have to be adjusted carefully according to the respective system and scenario. Furthermore, constraints are only taken into account as soft constraints by penalty terms. In the experiments, the latter fact has shown to be no limitation for practical usage.

While the proposed approach shows great potential for the efficient and optimized redundancy resolution, two main aspects may be addressed in further research: the improvement of the collision avoidance formulation and the extension of the prediction horizon. Using strictly convex geometries and/or a hierarchical scheme from large to more detailed geometry models may be beneficial for convergence of the collision avoidance cost function. A more efficient implementation of the proposed approach and a faster CPU allows a larger prediction time window and/or more iterations at each planning step.

Chapter 6

Trajectory Optimization by Direct Methods

The previous chapter introduced a novel trajectory optimization approach for redundant robots and the nullspace motion was optimized using an indirect optimization method. By contrast, this chapter presents the trajectory optimization using a direct method. The dynamic optimization problem is transformed to a static one by discretization. The problem can be solved with off-the-shelf nonlinear programming algorithms, which are able to solve the high-dimensional problem reliably. Using only the minimum number of constraints, the search space for optimal, admissible trajectories is very large. Thus, considerable savings in terms of maximum velocities and energy can be achieved. An extensive evaluation of the proposed approach shows its applicability for real world scenarios such as the harvesting of sweet pepper fruit in a greenhouse. The method as well as a summary of related results is presented in earlier work by the author (Schütz et al. 2015), while first implementations were done in collaboration with Reuß (2013a,b).

6.1 Objectives and Application

The method developed in this chapter has to meet several design criteria regarding the optimization goal itself as well as the practical implementation. Major optimization objectives can be summarized as follows:

- *Minimal set of task space constraints:* Using only the necessary constraints regarding the task space allows maximum flexibility in order to achieve secondary objectives. E.g. for selective harvesting tasks, constraints on the end effector imposed along the path can be freed while only the goal pose is important.

- *Energy optimal trajectory:* Joint velocities and accelerations are minimized while the end time is fixed.
- *Respect joint limits:* Kinematic joint limits for position, velocity and acceleration must be respected.
- *Collision avoidance:* Obstacles as well as self-collisions must be avoided.

Regarding the application of the proposed method for a real world scenario, the following requirements have to be considered regarding the implementation:

- *Fault detection:* Failures of the optimized trajectory must be detected reliably. Possible errors are an undetected collision or the violation of hard constraints such as kinematic limits.
- *Failure strategy:* In case of failures, an alternative solution to the trajectory optimization has to be found.
- *Computational time:* Offline numerical computation of the optimal trajectory has to be feasible within reasonable time. An evaluation of a large number of samples gives an indication regarding expected computational times and their distribution.

While the developed method remains general, the application considered in this thesis is the harvest of sweet peppers by the CROPS manipulator (prototype 1) in its 9-DOF configuration. The typical scenario in the greenhouse and the abstract simulation environment are shown in fig. 6.1: The manipulator has to move from its initial position at the storage container to the fruit, while avoiding collision with itself and the stems. Typical parameters are given in section 6.4.1. The cycle time of harvesting one fruit is in the range of a few seconds. Thus, the next trajectory can be pre-planned and optimized within the timespan of actually harvesting one fruit.

6.2 The Constrained Problem

Firstly, the optimal control problem (OCP) considered in this chapter is formulated as a constrained nonlinear optimization problem continuous in time. System states $\boldsymbol{x}$ are chosen as the robot's joint angles $\boldsymbol{q}$ while its inputs $\boldsymbol{u}$ are represented by joint velocities $\dot{\boldsymbol{q}}$. The mapping between the task space $\mathcal{W} \in \mathbb{R}^m$ and joint space $\mathcal{C} \in \mathbb{R}^n$ is given by forward kinematics mapping $\boldsymbol{\Phi}(\cdot)$.

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{q} \\ \dot{\boldsymbol{x}} &= \boldsymbol{u} = \dot{\boldsymbol{q}} \\ \boldsymbol{w} &= \boldsymbol{\Phi}\left(\boldsymbol{q}\right). \end{aligned} \tag{6.1}$$

(a) CROPS robot system (Gen. 1) in greenhouse. **(b)** Simplified environment.

Figure 6.1: *Left:* Harvesting sweet peppers with the CROPS manipulator system (Gen. 1) in a greenhouse. *Right:* Plants are modeled by vertical stems, simulating the environment for trajectory optimization.

A cost functional L defined by an integral over a time interval $[t_0, t_{\text{end}}]$ of some cost function $l(\cdot)$, is minimized by the system states $\boldsymbol{x}(\cdot)$ and the inputs $\boldsymbol{u}(\cdot)$. Equality conditions $\boldsymbol{g}(\cdot)$ as well as the space of admissible states $\mathcal{X}_t$ and controls $\mathcal{U}_t$ have to be respected. Thus, the constrained optimization problem can be written in its time-continuous form as follows:

$$\begin{aligned} \min_{\boldsymbol{x}(\cdot),\boldsymbol{u}(\cdot)} \; & L\left(\boldsymbol{x},\boldsymbol{u},t\right) = \int_{t_0}^{t_{\text{end}}} l\left(\boldsymbol{x}(t),\boldsymbol{u}(t),t\right) \mathrm{d}t \\ \text{w.r.t.} \quad & \dot{\boldsymbol{x}} = f\left(\boldsymbol{u}(t),t\right) \\ & \boldsymbol{g}\left(\boldsymbol{x}(t),\boldsymbol{u}(t)\right) = \boldsymbol{0} \\ & \boldsymbol{x}(t) \in \mathcal{X}_t, \quad \boldsymbol{u} \in \mathcal{U}_t \\ & t \in [t_0, t_{\text{end}}] \end{aligned} \tag{6.2}$$

The resulting set of optimal controls $\boldsymbol{u}^\star(t)$ and states $\boldsymbol{x}^\star(t)$ is considered to be the optimal trajectory for the given task.

6.2.1 Discretization

Instead of formulating optimality conditions according to Pontryagin's minimum principle, the OCP is discretized w.r.t. time. Thus, the dynamic problem is transformed to a static optimization problem. Thanks to this transformation it can be solved using nonlinear programming methods. The time interval $[t_0, t_{\text{end}}]$ is discretized in N steps with the increment Δt. Variables and functions are expressed

with the index k:

$$\begin{aligned} t_0 = t^0 < t^1 < ... < t^N \leq t_{\text{end}} \\ \Delta t = t^k - t^{k-1} \\ \boldsymbol{x}^k, \boldsymbol{u}^k := \boldsymbol{x}(t^k), \boldsymbol{u}(t^k) \\ l^k := l(\boldsymbol{x}^k, \boldsymbol{u}^k) \\ \boldsymbol{f}^k = \boldsymbol{f}(\boldsymbol{x}^k, \boldsymbol{u}^k) \\ k \in \mathbb{N}, \quad 0 \leq k \leq N \end{aligned} \tag{6.3}$$

Using the discrete counterparts of variables and functions $(\cdot)^k$, the problem eq. (6.2) can be written in its discrete form while the integral is solved using the trapezoidal rule.

$$\begin{aligned} \min_{\boldsymbol{x}(\cdot), \boldsymbol{u}(\cdot)} \; & L(\boldsymbol{x}, \boldsymbol{u}) = \sum_{k=0}^{N-1} \frac{t^{k+1} - t^k}{2} \left[l^k + l^{k+1} \right] \\ \text{w.r.t.} \quad & \dot{\boldsymbol{x}}^k = \boldsymbol{f}^k \\ & \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{0} \\ & \boldsymbol{x}_k \in \mathcal{X}_k, \quad \boldsymbol{u}_k \in \mathcal{U}_k \\ & k \in \mathbb{N}, \quad 0 \leq k \leq N. \end{aligned} \tag{6.4}$$

In the following, functions and variables are always used in their time-discrete form.

6.2.2 Cost Function

The cost function l^k is the sum of arbitrary sub-functions l_i^k, weighted by scalar quantities ζ_i. For the "harvesting sweet peppers" scenario, dynamic (l_{dyn}^k) as well as collision costs (l_{coll}^k) are taken into account. By contrast to the cost functions presented in section 4.2, there is no term for joint limit avoidance since compliance with joint limits is assured by box constraints (cf. section 6.2.3).

$$l^k = \zeta_{\text{dyn}}^k l_{\text{dyn}}^k + \zeta_{\text{coll}} l_{\text{coll}}^k \tag{6.5}$$

Dynamic Costs Dynamic costs are defined by the squared joint velocities (pseudo-energy). The weighting factor ζ_{dyn}^k increases quadratically from the initial value $\zeta_{\text{dyn},0}^k$ to its maximum value $\zeta_{\text{dyn},N}^k$ (cf. fig. 6.2). This adaptation is motivated by the reasoning that the robot system should slow down close to the goal:

$$\begin{aligned} l_{\text{dyn}}^k &= \left(\dot{\boldsymbol{q}}^k \right)^T \dot{\boldsymbol{q}}^k \\ \zeta_{\text{dyn}}^k &= \zeta_{\text{dyn},0} + \frac{\zeta_{\text{dyn,N}} - \zeta_{\text{dyn},0}}{N^2} k^2 \end{aligned} \tag{6.6}$$

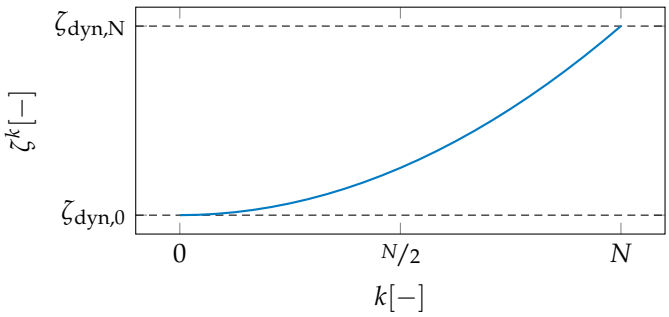


Figure 6.2: Weight ζ^k of dynamic costs l^k_{dyn} as a function of k.

Collision Avoidance Costs Collisions are avoided using the cost function l^k_{coll} which corresponds to H_{coll} at the time instance t^k as presented in section 4.2.2. The objective function l^k_{coll} as the time-dependent variant of eq. (4.31) can be written as:

$$\begin{aligned} l^k_{\text{coll}} &= \begin{cases} \frac{s_0}{3}\left(d_a - d^k\right)^3 & \text{if } d^k < d_a \\ 0 & \text{else} \end{cases} \\ s_0 &= 3\frac{l_{\text{coll,max}}}{d_a^3} \end{aligned} \tag{6.7}$$

6.2.3 Constraints

Apart from the cost functional, several equality and inequality constraints have to be defined. Firstly, the system differential equation is discretized by the trapezoidal rule and results in a set of $(N-1)$ equality constraints:

$$\begin{aligned} \boldsymbol{f}^k(\boldsymbol{x}^k, \boldsymbol{u}^k) &= \dot{\boldsymbol{x}}^k = \dot{\boldsymbol{q}}^k = \boldsymbol{u}^k \\ \dot{\boldsymbol{x}}^k &\approx \frac{\boldsymbol{x}^{k+1} - \boldsymbol{x}^k}{t^{k+1} - t^k} = \frac{1}{2}\left[\boldsymbol{f}^k + \boldsymbol{f}^{k+1}\right] = \frac{1}{2}\left[\boldsymbol{u}^k + \boldsymbol{u}^{k+1}\right]\,;\ \forall k \in [0; N-1] \end{aligned} \tag{6.8}$$

Other constraints are obtained from the given task: At the start ($t = t^0$), the initial configuration $\boldsymbol{q}^0$ and velocity $\dot{\boldsymbol{q}}^0$ are known. Furthermore, a desired goal in task space $\boldsymbol{w}_g$[1] and joint velocity $\dot{\boldsymbol{q}}_g$ at the end time ($t^N = t_{\text{end}}$) can be defined. Thus, start and end conditions can be summarized as follows using the function of forward kinematics $\boldsymbol{\Phi}(\cdot)$:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{u} \end{bmatrix}\Big|_{t^0} &= \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \end{bmatrix}\Big|_{t^0} = \begin{bmatrix} \boldsymbol{q}^0 \\ \dot{\boldsymbol{q}}^0 \end{bmatrix} \\ \boldsymbol{\Phi}(\boldsymbol{q}^N) &= \boldsymbol{w}_{\text{g}} \in \mathcal{W} \in \mathbb{R}^m \\ \dot{\boldsymbol{q}}^N &= \dot{\boldsymbol{q}}_g \end{aligned} \tag{6.9}$$

[1] In case of the scenario given in section 6.1, the goal is given by the desired pose of the gripper.

The system states $\boldsymbol{x}^k$ and inputs $\boldsymbol{u}^k$ can be bounded by the kinematic limits $\mathcal{X}$ and $\mathcal{U}$ of the robot system using box constraints $[\cdot]$ with the minimum $(\cdot)_{\text{min}}$ and maximum $(\cdot)_{\text{max}}$ values.

$$\begin{aligned} \boldsymbol{x}^k &\in \mathcal{X} := [\boldsymbol{x}_{\text{min}}, \boldsymbol{x}_{\text{max}}] \in \mathbb{R}^n \\ \boldsymbol{u}^k &\in \mathcal{U} := [\boldsymbol{u}_{\text{min}}, \boldsymbol{u}_{\text{max}}] \in \mathbb{R}^n \end{aligned} \tag{6.10}$$

6.3 Numerical Solution

The optimization problem is defined in the previous section 6.2. Its properties are summarized as follows:

- The problem is discretized by time increment Δt and a total number of steps N.
- Assuming a robot with n-DOF, the problem has nN states $\boldsymbol{x}^k$ and nN inputs $\boldsymbol{u}^k$.
- The discrete system equation is expressed by nN equality constraints.
- Start and end conditions result in $3n + m$ equality constraints.
- The states and inputs are bounded by $4nN$ inequality constraints.
- The end time t_{end} is fixed.

For an efficient numerical solution, a feasible initial guess is beneficial. Additionally, a feasible initial guess may be applied to the system in case the optimization fails. This section recapitulates briefly the initialization and optimization routine. Furthermore, the specific structure for interfacing the problem with the solver is shown.

6.3.1 Initialization

A feasible initial guess is given by a heuristically planned end effector trajectory (Baur 2015; Schütz et al. 2015) which is mapped to configuration space using the inverse kinematics algorithm described in section 4.1.2. While the workspace path is calculated within some milliseconds in advance, the inverse kinematics algorithm is computed online. Thus, the scheme can be summarized as follows:

1. Initialization
   (a) Planning of feasible task space trajectory by a heuristic planner.
   (b) (Real time) solution of inverse kinematics by the automatic supervisory control approach.
2. Optimization (optional)

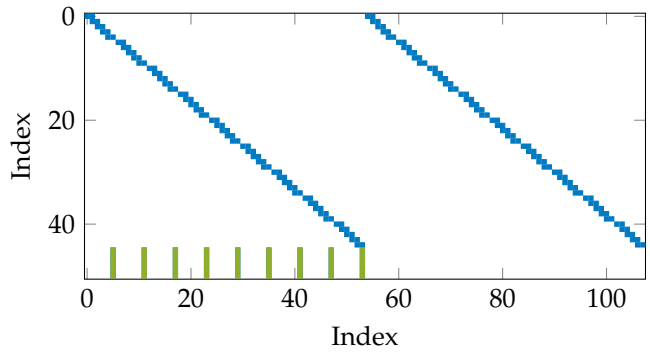


Figure 6.3: Sparse Matrix structure of constraints' Jacobian $\frac{\partial g(x)}{\partial x}$ with $N = 6$, $n = 9$, $m = 6$. In *blue*, the entries of the trapezoidal rule are shown while workspace constraints are marked in *green*.

(a) Discretization of the OCP to a static problem.
(b) Numerical solution of the problem.

3. Application of the computed trajectory.

6.3.2 Solver

In this work the algorithm IPOPT is used to solve the static optimization problem. IPOPT is an implementation of an interior-point algorithm with a filter line search method for large nonlinear programming problems (Wächter and Biegler 2006). The linear solver is implemented by the routine MA27. First-order derivatives of the cost function and of the equality constraints are calculated analytically. The limited memory quasi-newton BFGS[2] routine approximates the Hessian of the problem. The Jacobian matrix of the equality constraints has a sparse matrix structure, which is depicted in fig. 6.3. States $\boldsymbol{x}$ and inputs $\boldsymbol{u}$ are concatenated in the optimization vector $\boldsymbol{x}_{\text{opt}}$.

$$\begin{aligned} \boldsymbol{x}_{\text{opt}} &= \begin{bmatrix} q_0^0 \, q_0^1 \, \dots \, q_0^N & q_1^0 \, \dots \, q_n^N & \dot{q}_0^0 \, \dots \, \dot{q}_0^N & \dot{q}_1^0 \, \dots \, \dot{q}_n^N \end{bmatrix} \\ \boldsymbol{x}_{\text{opt}} &\in \mathbb{R}^{2n(N+1)}. \end{aligned} \tag{6.11}$$

6.4 Evaluation

The proposed optimization scheme is evaluated in a realistic test scenario regarding success rate, benefit and computational time. Two different initialization methods (*heuristic* and *straight-line* task space path) are investigated.

[2] *Broyden–Fletcher–Goldfarb–Shanno* algorithm for an approximation of Newton's method.

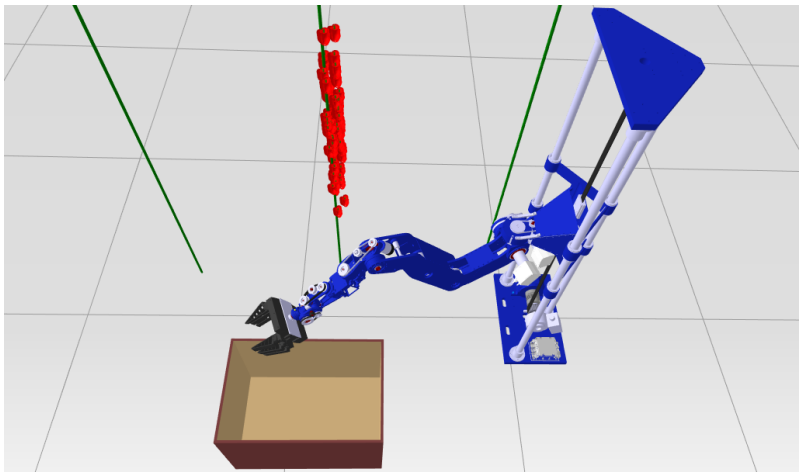


Figure 6.4: Evaluation setup with 79 sweet peppers (fruit size is downscaled in this figure), position data from real plant data measured by Wageningen Greenhouse & Horticulture.

6.4.1 Test Setup

The proposed trajectory optimization scheme is tested using fruit positions of 79 sweet peppers. The fruit positions are measured in a greenhouse by Wageningen Greenhouse and Horticulture relative to their main stem position. Details about growing system and fruit distribution around the stem are presented by Bac (2015) and Baur (2015). Stems can be approximated by vertical cylinders which are surrounded radially by fruit. The robot system in a greenhouse and its abstracted simulation environment are pictured in fig. 6.1. In the test setup, all fruit positions are projected to one representative stem position (cf. fig. 6.4). The set of parameters is summarized in table 6.1.

6.4.2 Typical Result

The performance, the general idea and the benefit of the proposed method can be illustrated by examining a typical result of the proposed optimization. The chosen harvesting scenario is depicted in fig. 6.5: The stem obstructs the direct straight-line path between start position (fig. 6.5-1) of the end effector and the fruit (fig. 6.5-6). Thus, a movement along this line path would cause a collision of the manipulator with the stem. The heuristic task space planning approach circumvents the stem and provides a collision free initial guess. The resulting end effector path is drawn in *red* and the manipulator is transparent (fig. 6.5). By comparison to the optimized solution (*green* path, blue/solid manipulator), start and goal of both end effector paths are the same due to the constraints on the initial configuration $\boldsymbol{q}_0$ and the goal pose $\boldsymbol{w}_g$. The path itself, however, is different. The optimized solution provides a more beneficial movement of the

Parameter	Symbol	Value
Maximum number of iterations	-	1000
Trajectory end time	t_{end}	10 s
Discretization step size	Δt	100 ms
Number of discretization steps	N	100
Degrees-of-freedom	n	9
Dimension of task space	m	6
Weight dynamic cost, initial value	$\zeta_{\text{dyn},0}$	0.5
Weight dynamic cost, maximum value	$\zeta_{\text{dyn},N}$	1.0
Weight collision cost	ζ_{coll}	10.0
Maximum collision cost	$l_{\text{coll,max}}$	5
Activation distance	d_a	Self-collisions: 0.05 m
		Collisions with the environment: 0.07 m
Stem position	x_{stem}	0.241 m
	y_{stem}	0.81 m
Stem radius	r_{stem}	0.015 m
Number of fruit	-	79

Table 6.1: Set of parameters used for evaluation of the proposed approach.

joints regarding dynamic costs while avoiding collisions[3]. The initial guess solution is constrained by the task space trajectory, which is far from being optimal regarding joint velocities. In fig. 6.7 dynamic costs are plotted over time: Velocities can be significantly reduced compared to the initial trajectory. Although the end effector path length is increased (cf. fig. 6.5), velocity peaks are evened out by choosing more favorable robot configurations. In this scenario, optimization took 76 iterations (time: 3.84 s[4]) and reduced the dynamic costs by 22.0% relative to the initial guess. By comparison to the initialization by the feasible initial guess, optimization based on the straight-line task space path converged as well, but took 177 iterations (time: 7.57 s). A statistical evaluation of all tested fruit positions is given in section 6.4.3.

The optimized trajectory is applied to the CROPS manipulator: The computed trajectory with the discretization increment of $\Delta t = 100$ ms is sent to the real time control unit and up-sampled to $\Delta t_{\text{manipulator}} = 5$ ms using spline interpolation. In fig. 6.6 the manipulator is depicted executing the optimized trajectory.

6.4.3 Statistics

A statistical evaluation of the proposed optimization scheme with a sufficient number of trials allows quantification of the expectable results. In the following, the median value of the results is preferred for evaluation instead of the average.

[3] By contrast to the dynamic costs, weights for collision avoidance are chosen to be higher (cf. table 6.1).

[4] Desktop computer, Intel i3-2100@3.1 GHz, single-threaded.

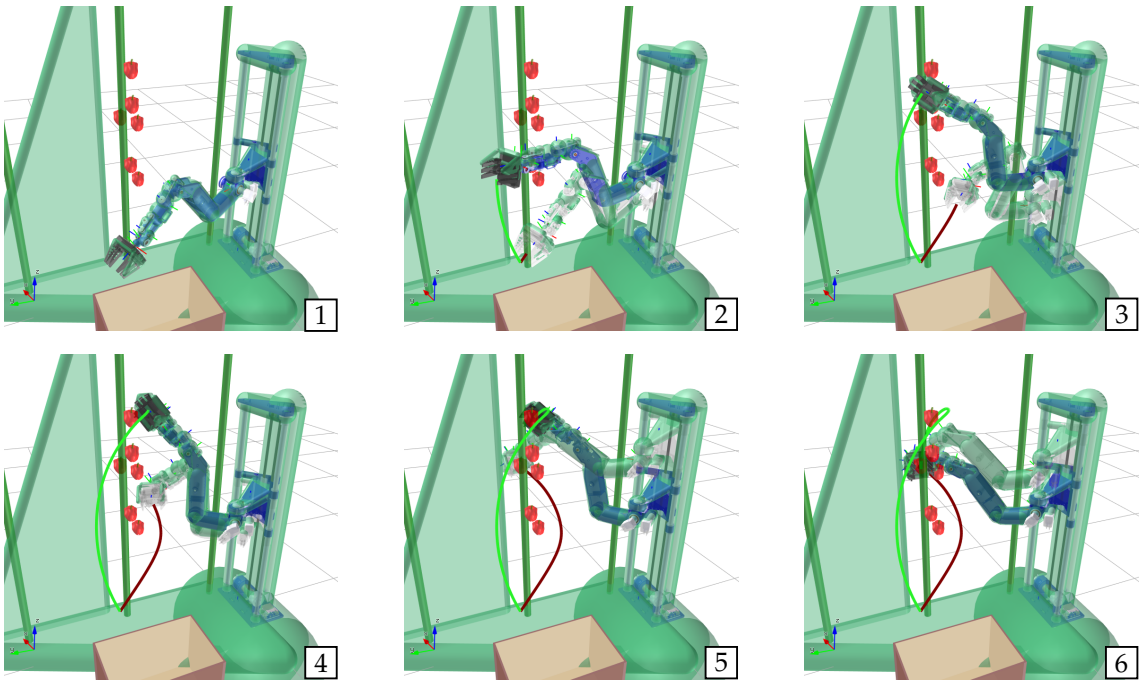


Figure 6.5: Typical result of optimized trajectory. Previously published in Schütz et al. (2015).

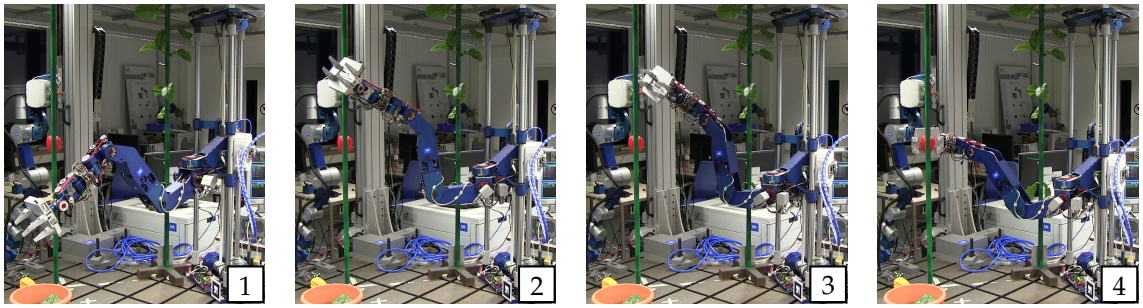


Figure 6.6: CROPS manipulator (prototype 1) executing an optimized trajectory.

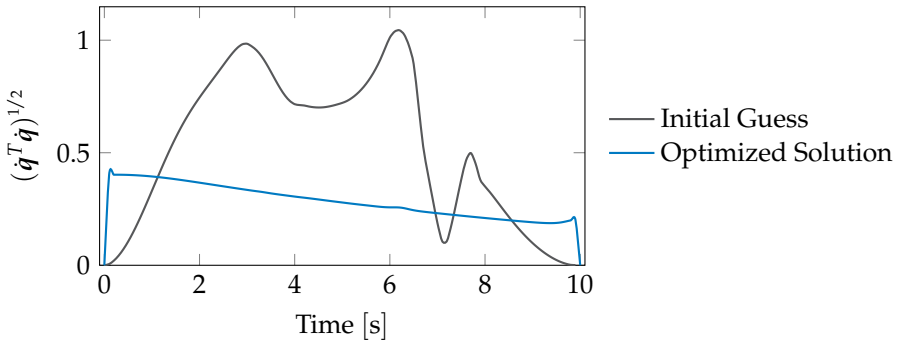


Figure 6.7: Dynamic costs over time of the initial guess (gray) and the optimized trajectory (blue) for the scenario shown in fig. 6.5.

Parameter	Value
Optimization convergence rate	94%
Remaining dynamic costs after optimization median (average)	36% (29%)
Iterations median (average)	Heuristic path init.: 60 (71.3) Straight-line path init.: 81 (92.2)
Computational time median (average)	Heuristic path init.: 3.4 s (5.0 s) Straight-line path init.: 4.8 s (6.6 s)

Table 6.2: Detailed results of the evaluation

Its allocation base is the number of trials so it gives a better estimate for the expectable result. However, the average is also given in the boxplot figures. All quantitative results are summarized in table 6.2.

The optimization converges in 94% of all runs. A failure occurs in case a collision occurs for the optimized solution or the optimization does not converge within the maximum number of iterations. Experiments showed that the feasibility of the initial guess has only a negligible influence on the overall success rate. Both, the straight line as well as the heuristic task space path, yield comparable success rates. Assuming that the resulting trajectory has to be collision free, dynamic costs are taken into account for evaluating the actual benefit of the trajectory optimization. Considering the median of the resulting costs, a reduction down to 36% of the initial costs (heuristic path initialization), corresponding to 29% on average is achieved (cf. fig. 6.8). For an evaluation of the computational time and iterations[5], the maximum number of iterations is not limited. Using the straight-line task space path initialization, the problem converges after 81 iterations (median). Using a collision-free feasible heuristic task space path, only 60 iterations (median) are needed which corresponds to a reduction by 26%. In accordance with the iterations, the median of time to convergence using the straight-line approach is 4.78 s and for the heuristic path initialization 3.44 s (reduction by 28%). Average values and boxplots are given in fig. 6.9.

6.5 Discussion

In this chapter an offline trajectory optimization approach using a direct method is presented. The optimal constrained problem is discretized w.r.t. time and solved by a nonlinear programming solver. Desired behavior of the computed trajectory can be expressed in terms of cost function as well as equality and inequality constraints. This formulation allows the use of a minimum number of

[5] Computational times measured on the same computer as in section 6.4.2, Intel i3-2100@3.1 GHz, single-threaded.

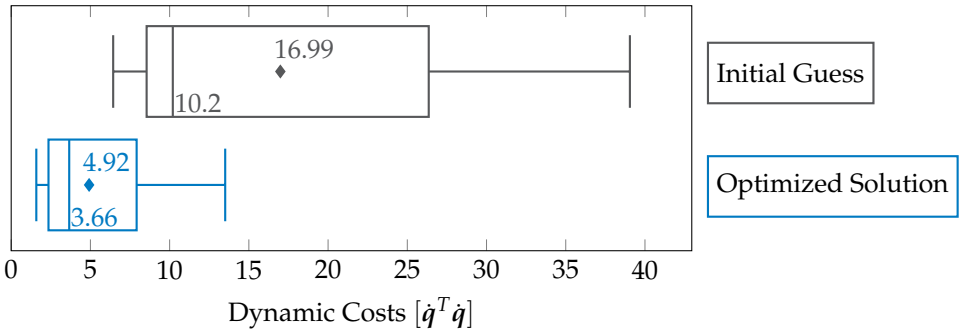
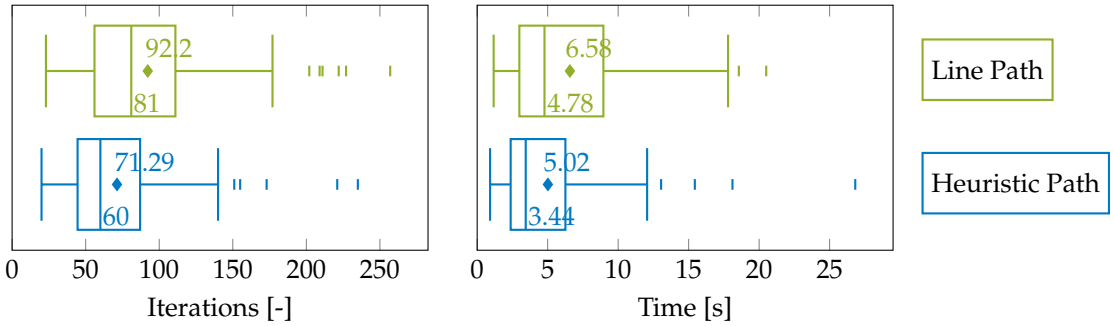


Figure 6.8: Comparision of dynamic costs l_{dyn} of initial guess (*gray*) and optimized solution (*blue*) with heuristic path as initial guess. The average value is denoted by ♦ and the median as the vertical line in the box. Tukey Boxplot, Whiskers ±1.5 Interquartile Range.



(a) Iterations **(b)** Computation time.

Figure 6.9: Computational effort for solving the optimization problem based on a straight-line task space path (green) initialization and a collision free heuristic path (blue). The average value is denoted by ♦ and the median as the vertical line in the box. Tukey Boxplot, Whiskers ±1.5 Interquartile Range.

constraints to achieve the desired goal while providing maximum flexibility for the trajectory optimization. In the following, results are discussed w.r.t. the influence of the initial guess, the design objectives presented in section 6.1, current limitations of the approach and its relation to the indirect method presented in chapter 5.

Influence of Initial Guess The initial guess is computed by a predefined task space trajectory which is mapped to the robot joints by an inverse kinematics algorithm. While the time parametrization is fixed by a 5^{th} order polynomial (Baur 2015), the geometric task space path is planned from start to goal pose either by using a simple linear interpolation ("straight line") or by a heuristic path circumventing the stem. While the heuristic approach provides a collision free end effector path, the line path in most cases does not. For the presented test case, the feasibility in terms of collisions has a negligible impact on the success rate of the numerical optimization. However, the solver converges with a smaller num-

ber of iterations, i.e. using less computational time, where a feasible trajectory is provided.

Failure Detection and Strategy In 6% of the harvesting cycles, a collision is not avoided in the final solution and the optimized trajectory is discarded. Collision avoidance is implemented as a soft constraint using a penalty term in the cost function and is therefore not rigidly enforced. Another problem is caused by the fixed time discretization and time dependency of the collision cost function. In some configurations, fast or very close passing or even penetration of the obstacle is observed. This behavior is enabled by the small dimensions of the thin stem of the plant in this scenario. An interesting strategy is presented by Zucker et al. (2013) for coping with this problem: The cost function is integrated along the arc length of the path, not w.r.t. time. This makes the collision penalty invariant with time and prevents this behavior.

Application and Calculation Time While the method proposed in chapter 5 is also suitable for a real time optimization, the method presented in this section is evaluated in an offline scheme. By contrast to a real time application, it can be applied as follows to the harvesting sweet pepper scenario: Assuming the scene is registered by some vision system and fruit/stem positions are known, the manipulator can start harvesting the first fruit instantaneously by applying the heuristic task space trajectory and the real time inverse kinematics algorithm. Each harvesting cycle is expected to take 5-10 s. While executing the first trajectory, the manipulator can already start the optimization of the picking of the next fruit. In case the optimization fails and the initial guess provides a feasible task space trajectory, the latter can be used as a fall-back solution.

Current Limitations The presented experiments are conducted with a fixed end time. For real world application, this approach has to be extended to another problem formulation with a free end time or a suitable post-processing strategy. Since velocities are considerably lowered by the proposed optimization and a smooth trajectory is calculated, execution times can be lowered as well. This can be done by scaling the time interval iteratively until joint velocity and acceleration limits are met.

Comparison to Indirect Methods Both the approaches developed in chapter 5 (indirect method) and in this chapter (direct method) aim at the optimization of a robot's trajectory. Both minimize joint velocities, penalty terms, etc. while respecting task space constraints. However, they are based on different problem formulations: The predictive IK algorithm optimizes the nullspace motion of a redundant manipulator, i.e. its task space trajectory remains fixed. By contrast, the problem formulation presented in this chapter only considers the task space

goal pose, while the joint space and the task space trajectory can be freely chosen. Furthermore, the indirect approach considers e.g. kinematic limits by penalty functions, i.e. as soft constraints. The approach of this chapter is able to include joint limits as hard constraints via box constraints. Unlike the method of chapter 5, the direct method in this chapter takes advantage of off-the-shelf optimization algorithms. This facilitates enormously the implementation by the user, e.g. new constraints can be integrated more easily.

Summary The trajectory optimization approach presented in this chapter allows a significant reduction of joint velocities by alleviating the restrictive task space trajectory constraints while taking into account only a minimum set of conditions. Evaluation of a large number of realistic harvesting scenarios proved that the method takes a reasonable amount of time and can be performed during the short idle states of robot systems.

Chapter 7

Conclusion

By contrast to traditional industrial production, modern robots have to cope with changing tasks in dynamic environments. Trends towards flexible production in industry and service oriented robotics make new demands on robots' design and planning capabilities. Providing a maximum of flexibility, modern robots are equipped with more degrees of freedom than necessary for most of their usual tasks. Resolving this redundancy in an optimal way while preserving the online planning capabilities is one of the key challenges in modern motion planning. In the following, contributions and main achievements of the thesis are summarized and discussed. Furthermore, an outlook and recommendations are given for possible future research.

7.1 Summary

This thesis presented novel motion planning approaches for redundant manipulators in the context of an agricultural robot system. Driven by the requirements of the respective application, e.g. the autonomous harvesting of sweet peppers, planning and optimization algorithms for a flexible, redundant robot were presented and evaluated.

To this day, many robot system were developed in order to automate agricultural applications. However, selective harvesting tasks of single, high quality crops are still performed by human workers. The harvesting period is short and the utilization rates of specialized robot systems are expected to be low. Thus, the objective of the European research project CROPS was the development of a multi-purpose agricultural robot system which is able to harvest sweet peppers, apples, grapes or to spray precisely diseased regions of fruit. In order to provide sufficient flexibility for the manifold scenarios, a modular manipulator with up to nine degrees of freedom was developed at the Institute of Applied Mechanics. Based on the doctoral theses of Baur (2015) and Pfaff (2015), new abilities and

methods were added to this manipulator: For situations in which an autonomous operation fails, a concept for teleoperation by a haptic joystick was developed. This enables the user to control the manipulator's end effector in real time while receiving haptic feedback from the arm. Furthermore, another concept for tactile perception, apart from the tactile sensor of Pfaff (2015), was developed: Without additional sensors, external forces can be estimated using only information from motor currents, the torsion in the joints and the dynamic model of the arm. By comparison to the high quality output of the tactile sensor module, which can be used closed loop, the torque estimator yields a poorer signal quality, but can still be used for collision detection with the environment.

By contrast to classical industrial settings, positions of fruit and obstacles differ in agricultural scenarios from scene to scene. Thus, paths and trajectories have to be re-planned for every target. The basic strategy, as described by Baur (2015), is to determine first a suitable task space trajectory, i.e. a trajectory for the manipulator's end effector. In a second step, the inverse kinematics are calculated in real time in an instantaneous scheme while solving redundancy by a gradient descent of the next timestep. This thesis recapitulates this method and presents an additional important secondary objective, the comfort pose. Furthermore, it encounters two important drawbacks of these approaches: First, the former motion planning approach relies on obstacle data from external vision sensors. In cluttered environments such as greenhouses, obstacles may be occluded by leaves which may cause unforeseen collisions. In this thesis tactile feedback was integrated into the planning approach. Using the sensor information of a tactile cover, contact forces between manipulator arm and environment are reduced by evasive movements of the arm. Different control approaches are presented that enable a controlled reaction while differentiating between nullspace and task space reactions. Second, the instantaneous method only uses information of one subsequent timestep for the optimization-based redundancy resolution. However, information about future events such as kinematic limits or self-collisions are available. Thus, this thesis develops a novel approach for optimizing the nullspace movement over a certain time window, named as predictive inverse kinematics. This approach relies on a numerically efficient method which even enables a real time implementation of the optimization scheme. The problem itself is solved by the conjugate gradent method. Due to the developed system formulation, even suboptimal trajectories that result from a preliminary abortion of the iterations may be applied. Besides, several aspects of the numerical optimization were investigated. This includes approaches for calculating the conjugate gradent in combination with different line search methods. An evaluation of a four-DOF pendulum showed that a combination of the Fletcher/Reeves conjugate gradent and the adaptive line search method yielded the best results. Additionally, an offline optimization approach for the joint trajectories was evaluated using realistic harvesting scenarios based on real data from greenhouses. The problem definition goes without the separation between task and joint spaces and relies only on a minimal set of constraints. The average success rates of 94% , an expectable re-

duction of dynamic costs of 36% and computational times in the range of a few seconds promote the use of the proposed method for real world scenarios.

7.2 Discussion and Future Directions

Motivated by the application of a nine-DOF agricultural manipulator, this thesis developed several motion planning approaches for dealing with cluttered environments and an optimal redundancy resolution. In this section, I will discuss the main contributions as introduced in section 1.1 and also give an outlook on future research directions. Furthermore, I will emphasize the importance of the developed approaches in the context of a future scenario regarding the application of autonomous manipulator in agriculture.

7.2.1 Submodules

Modeling Precise dynamic models enable a high quality estimation of quantities of a manipulator that are not directly measurable. While the concept of proprioceptive torque estimation yielded promising results, its performance was directly correlated to the accuracy of the dynamics model. Influences such as varying temperatures, fabrication and actuator tolerances made a reliable and accurate parameter identification difficult. Hence, I assume that the quality of the dynamics model could only be slightly improved by a more extensive parameter identification. I therefore doubt that the model accuracy and performance can be sufficient for the implementation of tactile feedback approaches based on a proprioceptive torque estimation at the CROPS manipulator. However, the estimated torques are still applicable for emergency functions, such as collision detection.

Teleoperation Interface The implementation of a teleoperation concept is a valuable extension for the CROPS manipulator system. Several public demonstrations showed that the teleoperation interface enables even non-expert users to pick fruit in a laboratory scenario. Another benefit is the usage of the system for investigation related to future space robotics research topics. E.g. the influence of time delays on user experience and robust control of the manipulator can be evaluated. Further investigations can include the development of user displays for controlling the manipulator remotely or the development of semi-autonomous planning and control approaches for dealing with long delays. A further valuable extension of the system is the integration of a 6D instead of the current 3D joystick. This enables the user to command end effector orientations as well. Furthermore, an adaption of the inverse kinematics algorithm that allows an arbitrary definition of the manipulator task space would enhance the user experience and prospects.

Planning based on Tactile Feedback The performance of the planning approaches based on tactile feedback in experiments heavily depends on the sensor signal quality of the tactile cover. In more dynamic scenarios or facing very stiff obstacles, the robot occasionally shows unintuitive or even unstable reactions. One obvious solution is to improve the signal quality of the tactile sensor, i.e. to minimize delays, isolate dynamic forces of the cover and improve the accuracy of the resulting force vector direction. However, I expect further improvements to be limited while keeping the efforts on a reasonable level. Therefore, additional security functions and saturations could be implemented in order to assure a fail-safe human-robot interaction. While the presented planning framework does not limit significantly the capabilities of the manipulator, further work could investigate the transition from nullspace to task space reaction in more detail. Additionally, the robustness of the feedback linearization control approaches could be improved in order to enable a smooth human-robot interaction.

Predictive Inverse Kinematics Aspects of the predictive inverse kinematics algorithm such as system formulation and numerical optimization method were extensively examined in this thesis. Although a prototype implementation was done for the CROPS manipulator, one major future challenge is the robust implementation for real systems. A more efficient implementation could yield higher optimization horizons or more iterations per planning step. Especially, a larger horizon would be beneficial. Furthermore, the current collision avoidance approach seems to be not well suited for the optimization scheme. Due to the non-convexity of the problem, the optimization routine occasionally converges to local minima. More convex problem formulations could help to resolve this issue. According to my opinion, a promising and robust approach would be the use of the predictive optimization only for kinematic limits and joint velocities. Collision avoidance could be integrated in a subsequent step in a standard instantaneous scheme.

Offline Trajectory Optimization The offline trajectory optimization allows the user to integrate arbitrary constraints and objectives. One drawback of the presented problem formulation regarding collision avoidance is the time parametrization of the trajectory. Since collision avoidance is formulated as a soft constraint and evaluated at fixed time increments, an optimal trajectory could "miss" an obstacle or pass it very fast. Three approaches may encounter this problem. First, obstacle geometries could be adapted. Introducing a hierarchical scheme, an outer, more "inflated" volume could help to avoid local minima. A second, inner volume as a more exact geometry representation may help derive a more exact solution. Second, collision avoidance may be inserted as a hard inequality constraint in the formulation of the optimization problem. A third option could be a transition from a time-based to an arc-length based parametrization of the trajectory as proposed by Zucker et al. (2013). Another limitation of the system formulation is the fixed end-time and time parametrization. Thus, the end-time

of current trajectories has to be specified by the user in advance. One approach to add more flexibility is to either scale the trajectory afterwards, according to velocity and acceleration limits of the manipulator, or to insert the time directly into $\boldsymbol{q}$ in the problem formulation.

7.2.2 Application of Autonomous Manipulators for Agriculture

Lessons Learned in the CROPS Project In 2014, the CROPS agricultural robot was the first autonomous system to harvest sweet peppers in a greenhouse. However, two years later there is still no commercial system available on the market, neither for selective harvesting of sweet peppers nor for apples. Major drawbacks of the CROPS harvesting system were the long cycle times and low overall success rates. According to my opinion, both drawbacks could be improved significantly by a better integration of the single system components. Due to a lack of time and resources, this was not feasible within the CROPS project. Under the assumption that prices for flexible manipulators do not drop significantly in the next years, the multipurpose approach of the CROPS project may also be a knockout criterion for commercialization. A wide variety of applications requires a flexible design of the manipulator with many joints. Customized kinematics, optimally chosen for one specific application, may reduce the system complexity and costs. Especially regarding the conservative farmers' community, this could increase the acceptance of autonomous systems. Additionally, growing conditions of greenhouses or orchards should be developed side-by-side with the robot system. Concepts known as vertical farming are promising approaches that may also be applied to high value crops such as peppers. Furthermore, a robust major motion planning strategy including a degradation concept is needed for a successful commercial implementation that copes with the elevated requirements of agricultural scenarios.

Context of the Thesis in a Future Scenario The approaches and algorithms developed in this thesis can be regarded as necessary and useful tools for an enhanced robust motion planning strategy. In the following, I will outline one promising scenario for the autonomous operation of robot manipulators in agriculture.

Both in greenhouses and orchards basic geometries of the environment are known. This may include the position and height of plants, the width of corridors, the area where fruit can be expected or fixed surroundings such as wires or other constructions. This static information has to be extended by real time visual information of cameras or other vision sensors. Based on this information, an optimized trajectory for the manipulator is determined by the application of direct methods as proposed in chapter 6. From this trajectory, a rough task space trajectory that leads the end effector to the fruit can be derived. However, this trajectory may still involve collisions, since the path is obstructed by soft obstacles

such as leaves. The task space trajectory is given to the real time inverse kinematics planner including the extension for tactile planning (chapter 4) and the predictive optimization (chapter 5). Both modules are based on the ASC system formulation and can therefore be sequentially combined. The predictive IK optimization keeps the manipulator away from kinematic limits and joint velocity peaks. At the same time, the tactile planning module reduces contact forces with the environment in an adaptive prioritized scheme. Thus, the intended task space trajectory can be followed as precisely as possible. During operation, an observer monitors the current "health" status of the arm. Based on an accurate dynamics model (chapter 3), this observer recognizes previously undetected collisions with the environment or hardware defects. In case of hazards or dead ends regarding the motion planning module, the manipulator can be remotely recovered by the teleoperation interface.

In summary, the contributions developed in this thesis have a great potential to advance the autonomous operation of robots in difficult environments such as agriculture. In combination with appropriately adapted cultivation methods and a successful integration of all system components, advanced agricultural tasks such as the selective harvesting of high quality fruit could be automated in the near future.

Appendix A

Manipulator Parameters

A.1 Kinematics

Frame i	${}_i\boldsymbol{r}_{pi}$ [m]	$\begin{bmatrix}\alpha & \beta & \gamma\end{bmatrix}^T$ [rad]
1	$\begin{bmatrix}0.115 & -0.017 & 0.796\end{bmatrix}^T$	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
2	$\begin{bmatrix}-0.115 & 0.48 & 0\end{bmatrix}^T$	$\begin{bmatrix}0 & 0 & \pi/2\end{bmatrix}^T$
3	$\begin{bmatrix}0 & 0 & -0.2\end{bmatrix}^T$	$\begin{bmatrix}\pi/2 & 0 & 0\end{bmatrix}^T$
4	$\begin{bmatrix}0.35 & 0 & 0\end{bmatrix}^T$	$\begin{bmatrix}\pi & 0 & 0\end{bmatrix}^T$
5	$\begin{bmatrix}0.35 & 0 & 0\end{bmatrix}^T$	$\begin{bmatrix}\pi & 0 & 0\end{bmatrix}^T$
6	$\begin{bmatrix}0.18 & 0 & 0\end{bmatrix}^T$	$\begin{bmatrix}-\pi/2 & 0 & 0\end{bmatrix}^T$
7	$\begin{bmatrix}0.15 & 0 & 0\end{bmatrix}^T$	$\begin{bmatrix}0 & \pi/2 & 0\end{bmatrix}^T$
8	$\begin{bmatrix}0 & 0 & 0.1\end{bmatrix}^T$	$\begin{bmatrix}0 & -\pi/2 & 0\end{bmatrix}^T$
9	$\begin{bmatrix}0.065 & 0 & 0\end{bmatrix}^T$	$\begin{bmatrix}0 & \pi/2 & 0\end{bmatrix}^T$

Table A.1: Kinematic parameters of the CROPS manipulator prototype 2.

A.2 Dynamics

Body	m_i [kg]	$\boldsymbol{r}_{OCi}$ [m]	$\boldsymbol{I}_i^{\langle C_i \rangle}$ [gm^2]
1	15.203	$[-0.113 \quad 0.188 \quad 0.104]^T$	$\begin{bmatrix} 619.584 & 43.334 & -9.220 \\ 43.334 & 176.102 & 141.988 \\ -9.220 & 141.988 & 549.512 \end{bmatrix}$
2	6.443	$[-0.043 \quad -0.007 \quad -0.151]^T$	$\begin{bmatrix} 43.344 & -3.024 & 8.064 \\ -3.024 & 68.544 & -0.298 \\ 8.064 & -0.298 & 43.344 \end{bmatrix}$
3	5.083	$[0.255 \quad 0 \quad -0.018]^T$	$\begin{bmatrix} 15.195 & -0.012 & -15.195 \\ -0.012 & 120.547 & 0.006 \\ -15.195 & 0.006 & 116.495 \end{bmatrix}$
4	4.347	$[0.177 \quad 0 \quad -0.036]^T$	$\begin{bmatrix} 11.268 & 0.376 & -10.016 \\ 0.376 & 102.664 & 0.136 \\ -10.016 & 0.136 & 102.664 \end{bmatrix}$
5	2.288	$[0.131 \quad 0.010 \quad -0.008]^T$	$\begin{bmatrix} 3.894 & -0.582 & -1.298 \\ -0.582 & 14.278 & -0.220 \\ -1.298 & -0.220 & 14.278 \end{bmatrix}$
6	1.379	$[0.086 \quad 0 \quad -0.017]^T$	$\begin{bmatrix} 1.244 & 0.011 & -0.717 \\ 0.011 & 6.220 & 0.017 \\ -0.717 & 0.017 & 6.220 \end{bmatrix}$
7	0.460	$[-0.039 \quad 0 \quad 0.047]^T$	$\begin{bmatrix} 1.069 & 0 & 0.355 \\ 0 & 1.362 & -0.001 \\ 0.355 & -0.001 & 0.659 \end{bmatrix}$
8	1.293	$[0.016 \quad 0 \quad 0.004]^T$	$\begin{bmatrix} 2.174 & -0.004 & 0.085 \\ -0.004 & 3.261 & -0.006 \\ 0.085 & -0.006 & 2.174 \end{bmatrix}$
9	0.217	$[0 \quad 0 \quad 0.010]^T$	$\begin{bmatrix} 0.132 & 0 & 0.001 \\ 0 & 0.130 & 0 \\ 0.001 & 0 & 0.172 \end{bmatrix}$

Table A.2: Dynamic parameters of the moving bodies of the CROPS manipulator prototype 2.

Appendix B

Modeling Data

B.1 Joint Stiffness

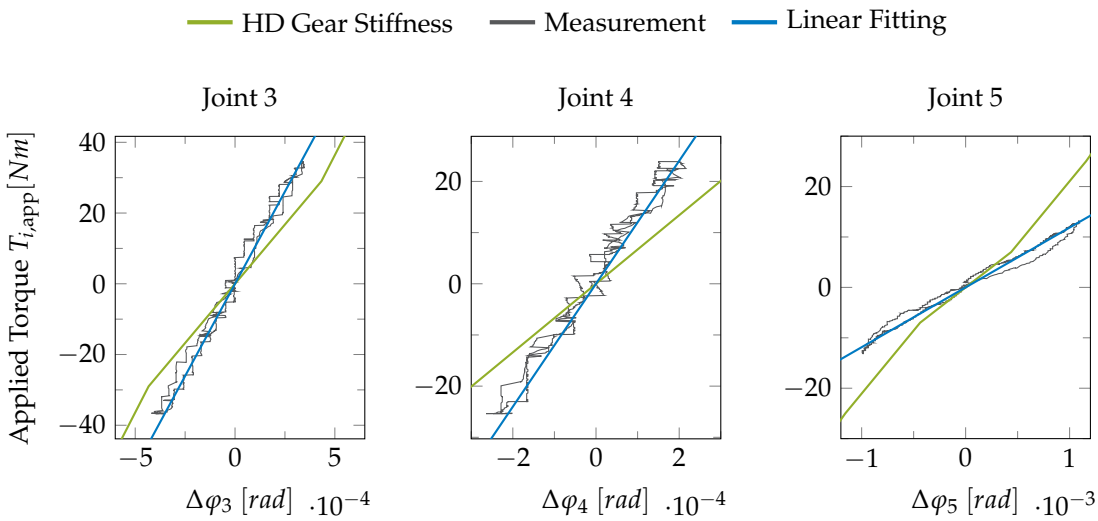


Figure B.1: Comparison of measured stiffness (gray) from quasi-static experiment and HarmonicDrive catalog data (green). Refer to section 3.2.3 for joint 2 and joint 6.

B.2 Inverse Dynamics Model

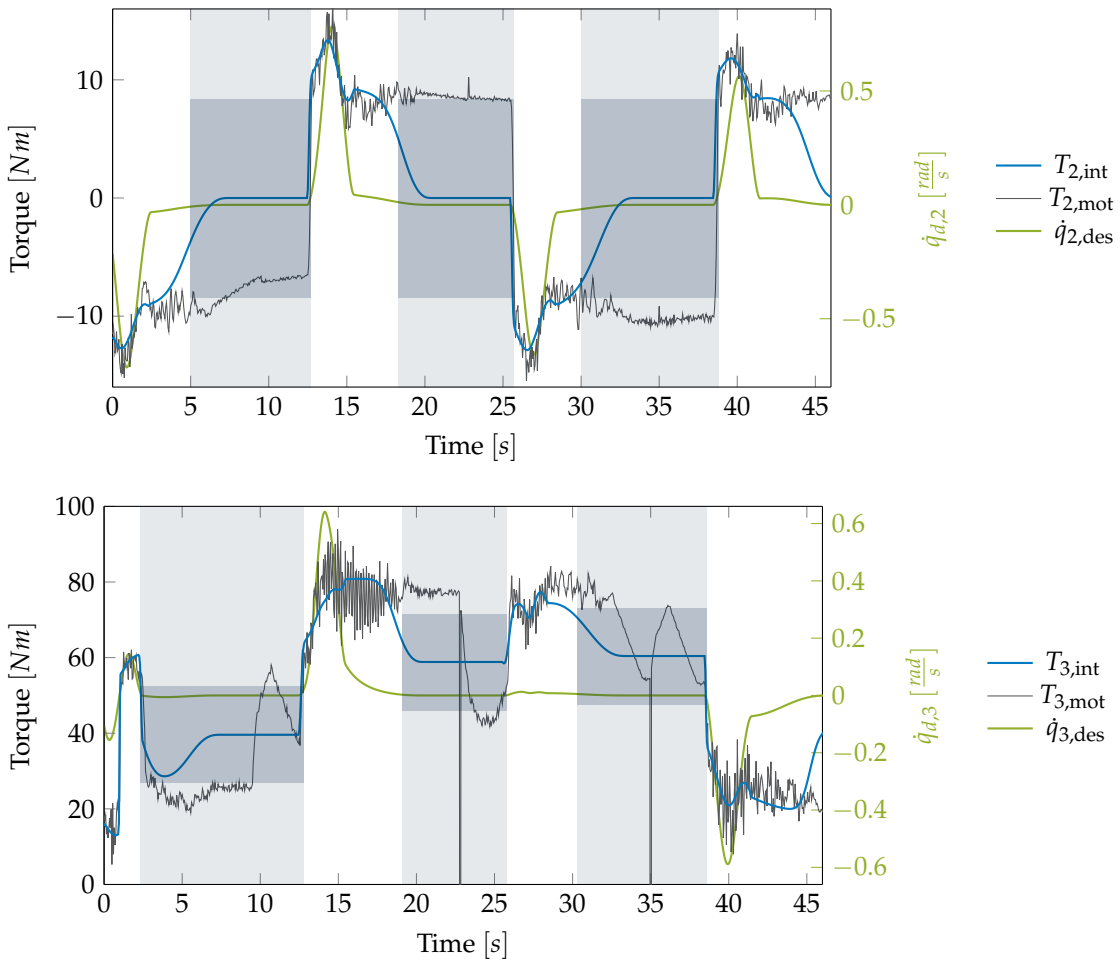


Figure B.2: Validation of the inverse dynamics model joint 2 and 3. *Blue:* Computed torque inverse dynamics model T_{int}. *Black:* Motor torque T_{mot}. *Green:* Desired joint velocity $\dot{q}$. *Gray area:* Velocities close to zero. *Dark-gray:* Sticking friction band. Cf. section 3.2.4.

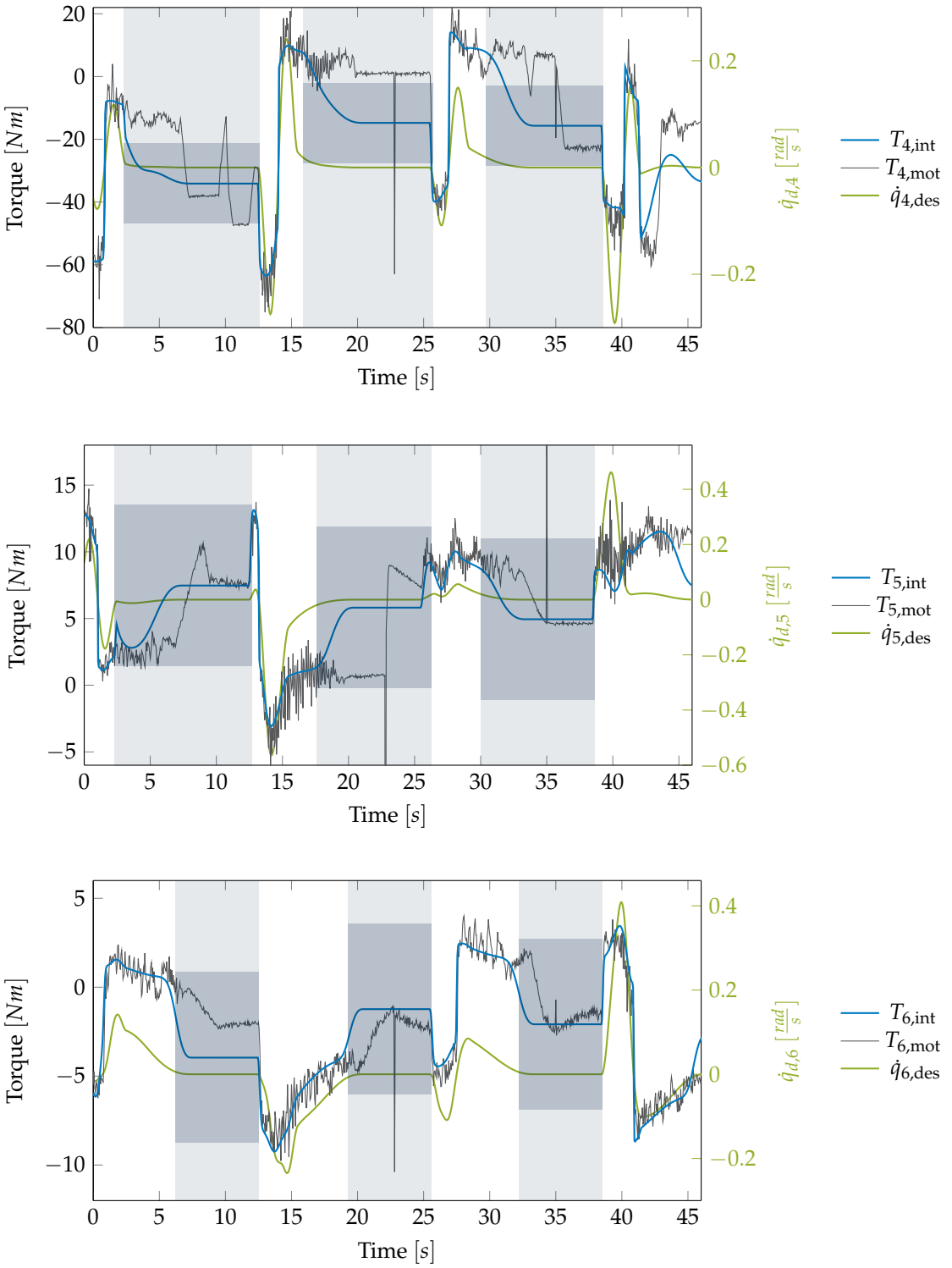


Figure B.3: Validation of the inverse dynamics model joint 4-6. *Blue:* Computed torque inverse dynamics model T_{int}. *Black:* Motor torque T_{mot}. *Green:* Desired joint velocity $\dot{q}$. *Gray area:* Velocities close to zero. *Dark-gray:* Sticking friction band. Cf. section 3.2.4.

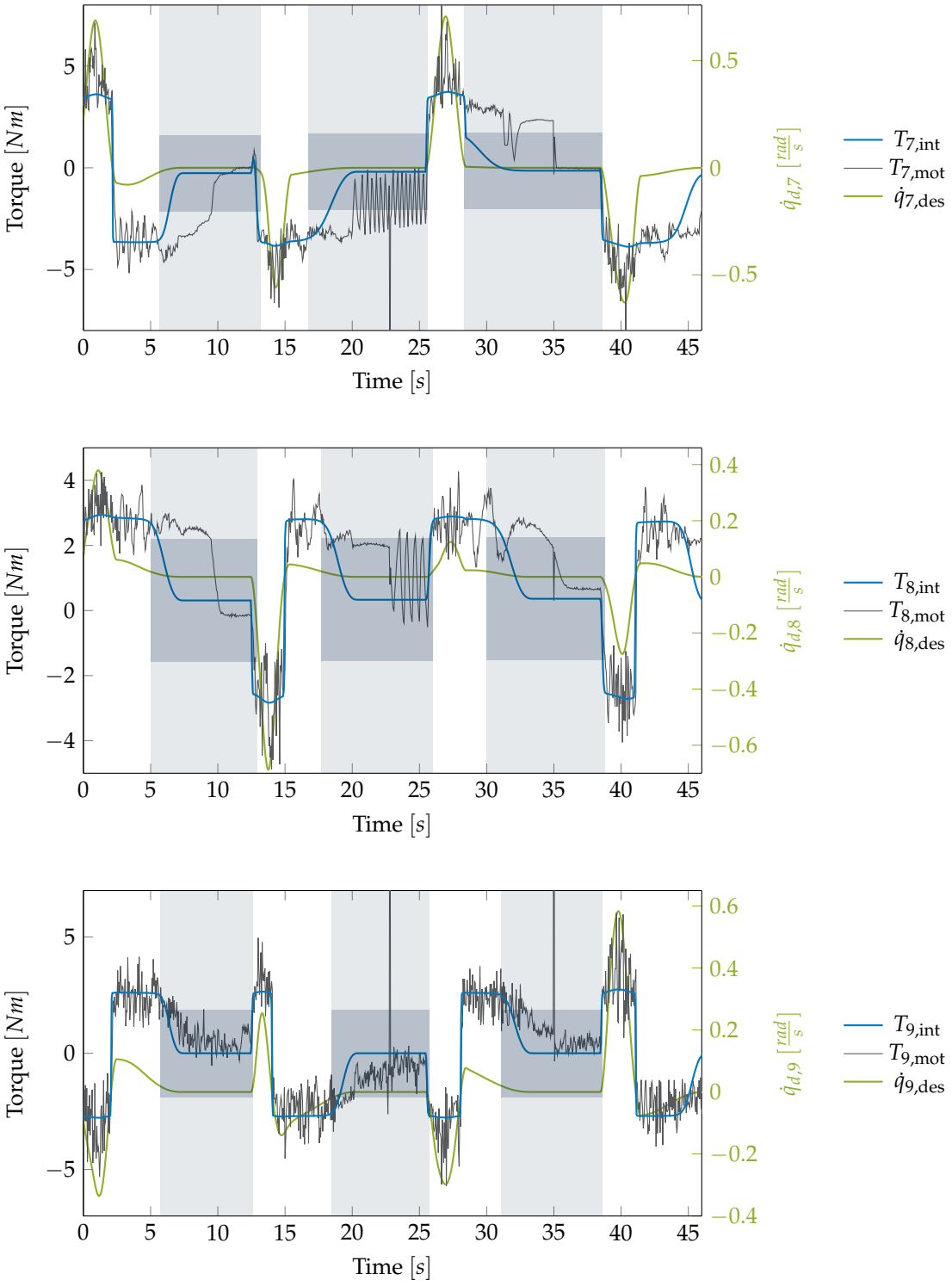
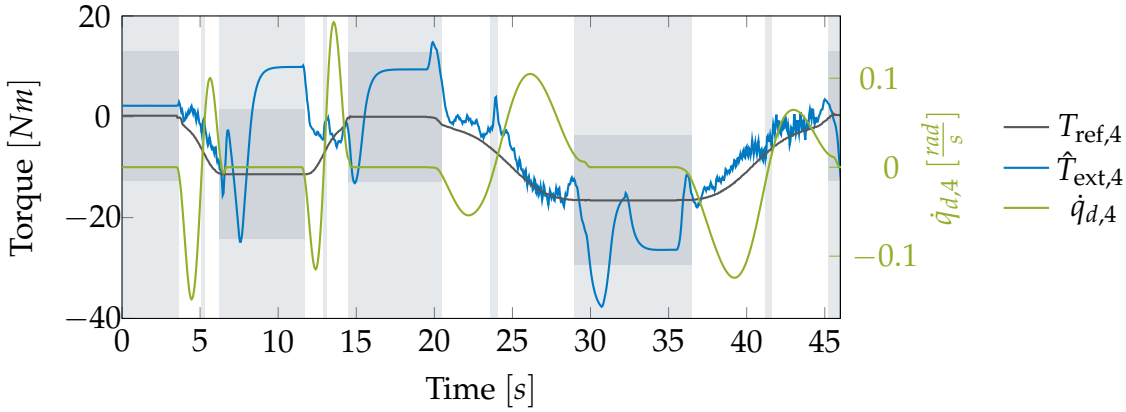
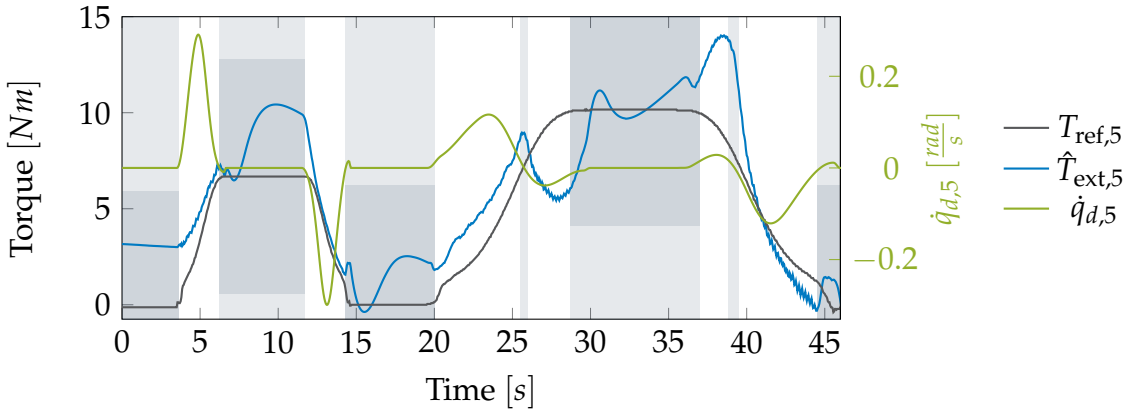


Figure B.4: Validation of the inverse dynamics model joint 7-9. *Blue:* Computed torque inverse dynamics model T_{int}. *Black:* Motor torque T_{mot}. *Green:* Desired joint velocity $\dot{q}$. *Gray area:* Velocities close to zero. *Dark-gray:* Sticking friction band. Cf. section 3.2.4.

B.3 Torque Estimation

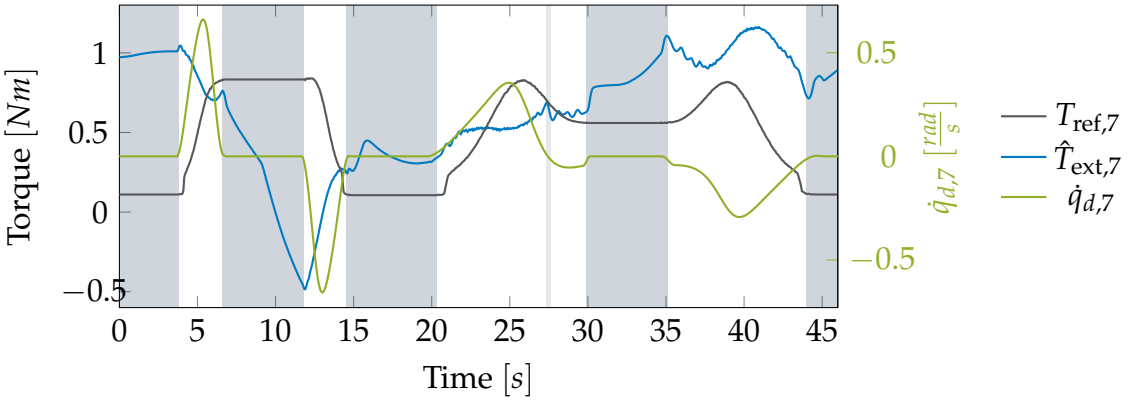


(a) Joint 4.

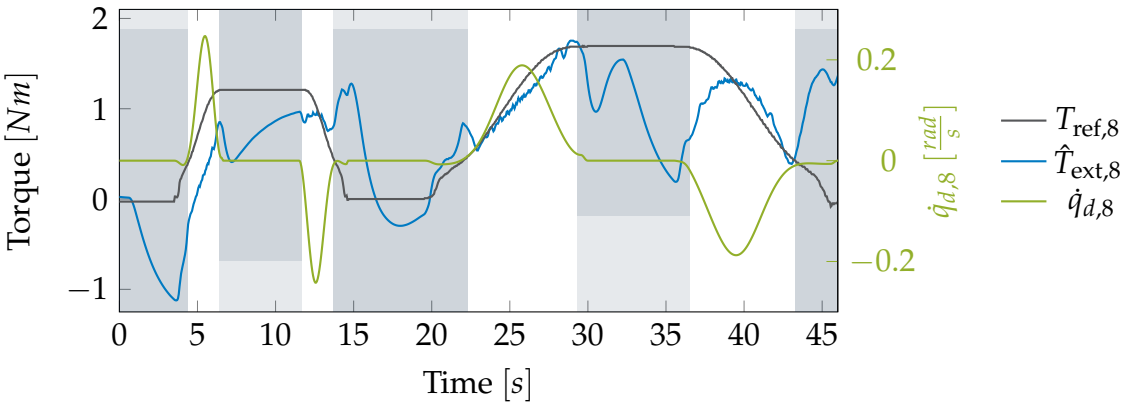


(b) Joint 5.

Figure B.5: Results for joint 4 and 5 in addition to fig. 3.18. Reference torque $T_{\text{ref},i}$ *(black)*, estimated torque $\hat{T}_{\text{est},i}$ (*blue*), desired joint velocity $\dot{q}_{d,i}$ (*green*). *Gray area:* joint velocities close to zero. *Dark-gray area*: corresponding sticking friction band.



(a) Joint 7.



(b) Joint 8.

Figure B.6: Results for joint 7 and 8 in addition to fig. 3.18. Reference torque $T_{\text{ref},i}$ *(black)*, estimated torque $\hat{T}_{\text{est},i}$ (*blue*), desired joint velocity $\dot{q}_{d,i}$ (*green*). *Gray area:* joint velocities close to zero. *Dark-gray area*: corresponding sticking friction band.

List of Figures

References

ASUSTeK Computer Inc. (2016). *Xtion PRO LIVE*. https://www.asus.com/3D-Sensor/Xtion_PRO_LIVE/, *accessed on* Oct. 16th, 2016.

Bac, C. W., J. Hemming, R. Barth, E. Wais, and E. J. V. Henten (2015). „Performance evaluation of a harvesting robot for sweet-pepper“. In: *Autonomous Robots* (submitted), pp. 1–22.

Bac, C. W., E. J. Van Henten, J. Hemming, and Y. Edan (2014). „Harvesting Robots for High-value Crops: State-of-the-art Review and Challenges Ahead“. In: *Journal of Field Robotics* 31.6, pp. 888–911. ISSN: 1556-4967. DOI: 10.1002/rob.21525.

Bac, C. (2015). „Improving obstacle awareness for robotic harvesting of sweet-pepper“. PhD Thesis. Wageningen University. ISBN: 9789462571808.

Baeten, J., K. Donné, S. Boedrij, W. Beckers, and E. Claesen (2008). „Autonomous Fruit Picking Machine: A Robotic Apple Harvester“. In: *Springer Tracts in Advanced Robotics*. Ed. by C. Laugier and R. Siegwart. Vol. 42. Springer Tracts in Advanced Robotics. Springer Berlin / Heidelberg, pp. 531–539. ISBN: 9783540754039. DOI: 10.1007/978-3-540-75404-6_51.

Barzilai, J. and J. M. Borwein (1988). „Two-point step size gradient methods“. In: *IMA Journal of Numerical Analysis* 8.1, pp. 141–148. ISSN: 02724979. DOI: 10.1093/imanum/8.1.141.

Baur, J. (2015). „Agricultural Manipulators“. Dissertation. Technische Universität München.

Baur, J., J. Pfaff, H. Ulbrich, and T. Villgrattner (2012). „Design and development of a redundant modular multipurpose agricultural manipulator“. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 823–830. DOI: 10.1109/AIM.2012.6265928.

Bellman, R. (1954). „Some Problems in the Theory of Dynamic Programming“. In: *Econometrica* 22.1, pp. 37–48. ISSN: 00129682.

Berenson, D., S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner (2009). „Manipulation planning with Workspace Goal Regions“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 618–624. ISBN: 978-1-4244-2788-8. DOI: 10.1109/ROBOT.2009.5152401.

Bertram, D., J. Kuffner, R. Dillmann, and T. Asfour (2006). „An integrated approach to inverse kinematics and path planning for redundant manipulators“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1874–1879. ISBN: 0-7803-9505-0. DOI: 10.1109/ROBOT.2006.1641979.

Bhattacharjee, T., A. Jain, S. Vaish, M. D. Killpack, and C. C. Kemp (2013). „Tactile sensing over articulated joints with stretchable sensors“. In: *World Haptics Conference (WHC)*, pp. 103–108. ISBN: 9781479900886. DOI: 10.1109/WHC.2013.6548392.

Bicchi, a., J. Salisbury, and P. Dario (1989). „Augmentation of grasp robustness using intrinsic tactile sensing“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 302–307. ISBN: 0-8186-1938-4. DOI: 10.1109/ROBOT.1989.100005.

Bobrow, J. E., S. Dubowsky, and J. S. Gibson (1985). „Time-Optimal Control of Robotic Manipulators Along Specified Paths“. In: *The International Journal of Robotics Research* 4.3, pp. 3–17. ISSN: 0278-3649. DOI: 10.1177/027836498500400301.

Bontsema, J., J. Hemming, E. P. W. Saeys, Y. Edan, A. Shapiro, M. Hočevar, T. Hellström, R. Oberti, M. Armada, H. Ulbrich, J. Baur, B. Debilde, S. Best, S. Evain, A. Münzenmaier, and O. Ringdahl (2014). „CROPS: high tech agricultural robots“. In: *International Conference of Agricultural Engineering (AgEng)*. Zurich, pp. 6–10. ISBN: 978-0-9930236-0-6.

Brock, O. and O. Khatib (2002). „Elastic Strips: A Framework for Motion Generation in Human Environments“. In: *The International Journal of Robotics Research* 21.12, pp. 1031–1052. ISSN: 0278-3649. DOI: 10.1177/027836490202101200 2.

Buschmann, T. (2010). „Simulation and Control of Biped Walking Robots“. Dissertation. Technische Universität München. ISBN: 978-3-86853-804-5.

— (2014). „Dynamics and Control of Redundant Robots“. Habilitation. Technische Universität München.

Chen, P. C. and Y. K. Hwang (1998). „SANDROS: A dynamic graph search algorithm for motion planning“. In: *IEEE Transactions on Robotics and Automation* 14.3, pp. 390–403. ISSN: 1042296X. DOI: 10.1109/70.678449.

Cheng, F. T., T. H. Chen, and Y. Y. Sun (1994). „Resolving Manipulator Redundancy Under Inequality Constraints“. In: *IEEE Transactions on Robotics and Automation* 10.1, pp. 65–71. ISSN: 1042296X. DOI: 10.1109/70.285587.

Cho, R. (2011). *Vertical Farms: From Vision to Reality*. blogs.ei.columbia.edu/2011/10/13/vertical-farms-from-vision-to-reality/, *accessed on* June 12th, 2016.

Choset, H., W. Burgard, S. Hutchinson, G. Kantor, L. E. Kavraki, K. Lynch, and S. Thrun (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementation*. Cambridge, MA.: MIT Press. ISBN: 9780262033275.

Craig, J. J. (2005). *Introduction to Robotics, Mechanics & Control*. Pearson/Prentice Hall. ISBN: 9780201543629.

Cutkosky, M. R., R. D. Howe, and W. R. Provancher (2008). „Force and Tactile Sensors“. In: *Springer Handbook of Robotics*. Chap. 19, pp. 455–476. ISBN: 978-3-540-23957-4. DOI: 10.1007/978-3-540-30301-5_20.

Dahiya, R. S., P. Mittendorfer, M. Valle, G. Cheng, and V. J. Lumelsky (2013). „Directions toward effective utilization of tactile skin: A review“. In: *IEEE Sensors Journal* 13.11, pp. 4121–4138. ISSN: 1530437X. DOI: 10.1109/JSEN.2013.2279056.

De Luca, A., A. Albu-Schäffer, S. Haddadin, and G. Hirzinger (2006). „Collision detection and safe reaction with the DLR-III lightweight manipulator arm“.

In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1623–1630. ISBN: 142440259X. DOI: 10.1109/IROS.2006.282053.

De Luca, A. and R. Mattone (2005). „Sensorless robot collision detection and hybrid force/motion control“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 999–1004. ISBN: 078038914X. DOI: 10.1109/ROBOT.2005.1570247.

Diehl, M., H. Ferreau, and N. Haverbeke (2009). „Efficient numerical methods for nonlinear MPC and moving horizon estimation“. In: *Nonlinear Model Predictive Control*. Ed. by L. Magni, D. Raimondo, and F. Allgöwer. Vol. 384. Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg, pp. 391–417. ISBN: 978-0-85729-500-2. DOI: 10.1007/978-3-642-01094-1_32.

Dogar, M. and S. Srinivasa (2011). „A Framework for Push-Grasping in Clutter“. In: *Robotics: Science and Systems VII*. DOI: 10.15607/RSS.2011.VII.009.

Duchaine, V., N. Lauzier, M. Baril, M.-A. Lacasse, and C. C. Gosselin (2009). „A flexible robot skin for safe physical human robot interaction“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3676–3681. ISBN: 978-1-4244-2788-8. DOI: 10.1109/ROBOT.2009.5152595.

Eberman, B. S. and J. K. Salisbury (1990). „Determinantion of Manipulator Contact Information from Joint Torque Measurements“. English. In: *Experimental Robotics I*. Ed. by V. Hayward and O. Khatib. Vol. 139. Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg, pp. 463–473. ISBN: 978-3-540-52182-2. DOI: 10.1007/bfb0042535.

El Khoury, A., F. Lamiraux, and M. Taix (2013). „Optimal motion planning for humanoid robots“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3136–3141. ISBN: 978-1-4673-5643-5. DOI: 10.1109/ICRA.2013.6631013.

ERO Gerätebau GmbH (2016). *ERO Grape Harvesters*. www.ero-geraetebau.de/, *accessed on* June 6th, 2016.

Escande, A., N. Mansard, and P. B. Wieber (2010). „Fast resolution of hierarchized inverse kinematics with inequality constraints“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3733–3738. ISBN: 9781424450381. DOI: 10.1109/ROBOT.2010.5509953.

— (2014). „Hierarchical quadratic programming: Fast online humanoid-robot motion generation“. In: *The International Journal of Robotics Research* 33.7, pp. 1006–1028. ISSN: 0278-3649. DOI: 10.1177/0278364914521306.

Featherstone, R. (1987). *Robot Dynamics Algorithms*. English. Vol. 22. Boston, Dordrecht, Lancaster: Springer US. ISBN: 978-1-4757-6437-6. DOI: 10.1007/978-0-387-74315-8.

Fishel, J. A., G. E. Loeb, R. A. Peck, C. H. Lin, B. Matulevich, and V. Pandit (2014). „Compliant tactile sensor with fluid-filled, sponge-like material“. Patent US20140069212A1.

Fishel, J. A., V. J. Santos, and G. E. Loeb (2008). „A robust micro-vibration sensor for biomimetic fingertips“. In: *IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pp. 659–663. ISBN: 9781424428830. DOI: 10.1109/BIOROB.2008.4762917.

Flacco, F., A. De Luca, and O. Khatib (2015). „Control of Redundant Robots Under Hard Joint Constraints: Saturation in the Null Space“. In: *IEEE Transactions on Robotics* 31.3, pp. 637–654. ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2418582.

Fletcher, R. and C. M. M. Reeves (1964). „Function minimization by conjugate gradients“. In: *The Computer Journal* 7.2, pp. 149–154. ISSN: 0010-4620. DOI: 10.1093/comjnl/7.2.149.

Frank, B., C. Stachniss, N. Abdo, and W. Burgard (2011). „Efficient motion planning for manipulation robots in environments with deformable objects“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2180–2185. ISBN: 9781612844541. DOI: 10.1109/IROS.2011.6048633.

Fruit Growers News (2012). *Robotic strawberry harvesters demoed in California*. fruitgrowersnews.com/article/robotic-strawberry-harvesters-demoed-in-california1/, *accessed on* June 12th, 2016.

Gauchel, W. and S. Saller (2012). „Adaptive gripper jaws for high-value crops harvesting“. In: *8th International Fluid Power Conference*. Dresden.

Graichen, K. and B. Käpernick (2011). „A Real-Time Gradient Method for Nonlinear Model Predictive Control“. In: *Frontiers of Model Predictive Control* 2010, pp. 9–28. DOI: 10.5772/37638.

Graichen, K. and A. Kugi (2010). „Stability and incremental improvement of suboptimal MPC without terminal constraints“. In: *IEEE Transactions on Automatic Control* 55.11, pp. 2576–2580. ISSN: 00189286. DOI: 10.1109/TAC.2010.2057912.

Grand d'Esnon, A., G. Rabatel, R. Pellenc, A. Journeau, and M. Aldon (1987). „Magali: a Self-Propelled Robot To Pick Apples“. In: *ASAE Annual International Meeting*. Baltimore, MD.

Haas, V. (1975). „An alternate form of the conjugate gradient method for dynamic optimization“. In: *IEEE Transactions on Automatic Control* 20.4, pp. 581–582. ISSN: 0018-9286. DOI: 10.1109/TAC.1975.1101029.

Haddadin, S., A. Albu-Schäffer, A. De Luca, and G. Hirzinger (2008). „Collision detection and reaction: A contribution to safe physical human-robot interaction“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3356–3363. ISBN: 9781424420582. DOI: 10.1109/IROS.2008.4650764.

Harrell, R. C., P. D. Adsit, T. A. Pool, and R. Hoffman (1990). „The Florida Robotic Grove-Lab“. In: *Transactions of the ASAE* 33.2, pp. 391–399. ISSN: 00012351. DOI: 10.13031/2013.24066.

Harrell, R. C., D. C. Slaughter, and P. D. Adsit (1989). „A fruit-tracking system for robotic harvesting“. In: *Machine Vision and Applications* 2.2, pp. 69–80. ISSN: 09328092. DOI: 10.1007/BF01212369.

Hemming, J., W. Bac, and Y. Edan (2014). *CROPS Deliverable D5.7: Report of prototype test under laboratory and field conditions, including suggestions for improvement*. Project Report.

Hemming, J., W. Bac, B. van Tuijl, R. Barth, J. Bontsema, E. Pekkeriet, and E. van Henten (2014). „A robot for harvesting sweet-pepper in greenhouses“. In: *International Conference of Agricultural Engineering (AgEng)*. Zurich, pp. 6–10.

Hemming, J., J. Bontsema, W. Bac, Y. Edan, B. van Tuijl, R. Barth, and E. Pekkeriet (2014). *Final Report Sweet-Pepper Harvesting Robot*. Technical Report December. Wageningen UR.

Henten, E. van (2004). „Greenhouse mechanization: State of the art and future perspective“. In: *International Symposium on Greenhouses, Environmental Controls and In-house Mechanization for Crop Production*, pp. 55–70.

Hestènes, M. and E. Stiefel (1952). „Methods of conjugate gradients for solving linear systems“. In: *Journal cf Research of the National Bureau of Standards* 49.6, pp. 81–85. ISSN: 0091-0635. DOI: 10.6028/jres.049.044.

Hollerbach, J. and K. S. K. Suh (1985). „Redundancy resolution of manipulators through torque optimization“. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 4, pp. 308–316. ISBN: 0882-4967. DOI: 10.1109/ROBOT.1985.1087285.

Hornung, A., K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard (2013). „OctoMap: An efficient probabilistic 3D mapping framework based on octrees“. In: *Autonomous Robots* 34.3, pp. 189–206. ISSN: 09295593. DOI: 10.1007/s10514-012-9321-0.

Howe, R. D. (1993). „Tactile sensing and control of robotic manipulation“. In: *Advanced Robotics* 8.3, pp. 245–261. ISSN: 0169-1864. DOI: 10.1163/156855394X00356.

IFR (2015a). *World Robotics 2015 Industrial Robots*. Technical Report. VDMA Robotics & Automation.

— (2015b). *World Robotics - Industrial Robots*. Technical Report. VDMA Robotics & Automation.

Irie, N., N. Taguchi, T. Horie, and T. Ishimatsu (2009). „Asparagus harvesting robot coordinated with 3-D vision sensor“. In: *IEEE International Conference on Industrial Technology*. ISBN: 1424435064. DOI: 10.1109/ICIT.2009.4939556.

Iwata, H., H. Hoshino, T. Morita, and S. Sugano (2001). „Force detectable surface covers for humanoid robots“. In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1205–1210. ISBN: 0-7803-6736-7. DOI: 10.1109/AIM.2001.936882.

Jacobson, D. H. and D. Q. Mayne (1970). *Differential Dynamic Programming*. New York, New York, USA: American Elsevier Publishing Company, Inc. ISBN: 0444000704.

Jain, A. and M. Killpack (2013). „Reaching in clutter with whole-arm tactile sensing“. In: *The International Journal of Robotics Research* 32.4, pp. 458–482. ISSN: 0278-3649. DOI: 10.1177/0278364912471865.

Jamisola, R. S., P. Kormushev, A. Bicchi, and D. G. Caldwell (2014). „Haptic exploration of unknown surfaces with discontinuities“. In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 1255–1260. ISBN: 9781479969340. DOI: 10.1109/IROS.2014.6942718.

Kálmán, R. E. (1960). „A new approach to linear filtering and prediction problems“. In: *Journal of Basic Engineering* 82, pp. 35–45.

Kanoun, O., F. Lamiraux, and P. B. Wieber (2011). „Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task“. In: *IEEE Transactions on Robotics* 27.4, pp. 785–792. ISSN: 15523098. DOI: 10.1109/TRO.2011.2142450.

Käpernick, B. and K. Graichen (2014). „The gradient based nonlinear model predictive control software GRAMPC“. In: *European Control Conference*, pp. 1170–1175. ISBN: 978-3-9524269-1-3. DOI: 10.1109/ECC.2014.6862353.

Karaman, S. and E. Frazzoli (2011). „Sampling-based algorithms for optimal motion planning“. In: *International Journal of Robotics Research (IJRR)* 30.7, pp. 846–894. ISSN: 0278-3649. DOI: 10.1177/0278364911406761.

Kavraki, L. E., P. Švestka, J. C. Latombe, and M. H. Overmars (1996). „Probabilistic roadmaps for path planning in high-dimensional configuration spaces“. In: 12.4, pp. 566–580. ISSN: 1042296X. DOI: 10.1109/70.508439.

Kawamura, N., K. Namikawa, T. Fujimura, and M. URA (1985). „Study on Agricultural Robot (Part 2)“. In: *Journal of the Society of Agricultural Machinery Japan* 47.2, pp. 177–182. DOI: 10.11357/jsam1937.46.3_353.

Khalil, W. and E. Dombre (2004). „Chapter 6 - Inverse kinematic model of serial robots“. In: *Modeling, Identification and Control of Robots*. Vol. 56, pp. 117–144. ISBN: 978-1-903996-66-9. DOI: 10.1016/B978-190399666-9/50005-1.

Khatib, O. (1983). „Dynamic Control of Manipulators in Operational Space“. In: *Proc. of the Sixth CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*. New Delhi, India, pp. 1128–1131.

Khatib, O. (1986). „Real-time Obstacle Avoidance for Manipulators and Mobile Robots“. In: *IEEE International Conference on Robotics and Automation (ICRA)* 2.1, pp. 90–98. ISSN: 0278-3649. DOI: 10.1109/ROBOT.1985.1087247.

Killpack, M. D. and C. C. Kemp (2013). „Fast reaching in clutter while regulating forces using model predictive control“. In: *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, pp. 146–153. DOI: 10.1109/HUMANOIDS.2013.7029969.

Kim, W., F. Tendick, S. Ellis, and L. Stark (1987). „A comparison of position and rate control for telemanipulations with consideration of manipulator system dynamics“. In: *IEEE Journal on Robotics and Automation* 3.5, pp. 426–436. DOI: 10.1109/JRA.1987.1087117.

Klein, C. A. and C. H. Huang (1983). „Review of Pseudoinverse Control for Use with Kinematically Redundant Manipulators“. In: *IEEE Transactions on Systems, Man and Cybernetics* 13.2, pp. 245–250. ISSN: 21682909. DOI: 10.1109/TSMC.1983.6313123.

Klessig, R. and E. Polak (1972). „Efficient Implementations Of Polak-Ribiere Conjugate Gradient Algorithm“. In: *Siam Journal On Control* 10.3, pp. 524–549. ISSN: 0036-1402. DOI: 10.1137/0310040.

Knight, W. (2014). *How Human-Robot Teamwork Will Upend Manufacturing*. https://www.technologyreview.com/s/530696/how-human-robot-teamwork-will-upend-manufacturing/, *accessed on* Nov. 15th, 2016.

Kolter, J. Z. and A. Y. Ng (2009). „Task-space trajectories via cubic spline optimization“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1675–1682. ISBN: 9781424427895. DOI: 10.1109/ROBOT.2009.5152554.

Kondo, N. (1995). „Harvesting Robot Based on Physical-Properties of Grapevine“. In: *Japan Agricultural Research Quarterly* 29.3, pp. 171–177. ISSN: 0021-3551.

Kondo, N., Y. Nitshitsuji, P. P. Ling, and K. C. Ting (1996). „Visual feedback guided robotic cherry tomato harvesting“. In: *Transactions of the ASAE* 39.6, pp. 2331–2338. ISSN: 0001-2351.

Kondo, N., M. Monta, T. Fujiura, Y. Shibano, and K. Mohri (1994). „Control Method for 7 DOF Robot to Harvest Tomato“. In: *Proceedings of the Asian Control Conference*. Tokyo, pp. 1–4.

Kondo, N. and K. C. Ting (1998). *Robotics for bioproduction systems*. St Joseph, MI: American Society of Agricultural Engineers. ISBN: 0929355946.

Kopp, R., R. Mcgill, H. Moyer, and G. Pinkham (1964). „Several Trajectory Optimization Techniques“. In: *Computing methods in optimization*, pp. 65–89.

Kröger, T. (2010). „On-Line Trajectory Generation in Robotic Systems: Basic Concepts for Instantaneous Reactions to Unforeseen (Sensor) Events“. In: *Springer Tracts in Advanced Robotics* 58.1, pp. 1–230. ISSN: 16107438. DOI: 10.1007/978-3-642-05175-3_1.

Lasdon, L., S. Mitter, and a. Waren (1967). „The conjugate gradient method for optimal control problems“. In: *IEEE Transactions on Automatic Control* 12.2, pp. 132–138. ISSN: 0018-9286. DOI: 10.1109/TAC.1967.1098538.

Latombe, J.-C. (1991). *Robot motion planning*. Vol. 54. The Kluwer international series in engineering and computer science. Boston, MA., p. 430.

LaValle, S. M. (1998). *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. Iowa State University, pp. 98–11.

LaValle, S. M. (2006). *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press. ISBN: 9780511546877. DOI: 10.1017/CBO9780511546877.

Lee, M. H. (2000). „Tactile Sensing: New Directions, New Challenges“. In: *The International Journal of Robotics Research* 19.7, pp. 636–643. ISSN: 0278-3649. DOI: 10.1177/027836490001900702.

Lee, M. and H. Nicholls (1999). „Review Article Tactile sensing for mechatronics—a state of the art survey“. In: *Mechatronics* 9.1, pp. 1–31. ISSN: 09574158. DOI: 10.1016/S0957-4158(98)00045-2.

Liégeois, A. (1977). „Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms“. In: *IEEE Transactions on Systems, Man, and Cybernetics* 7.12, pp. 868–871. ISSN: 0018-9472. DOI: 10.1109/TSMC.1977.4309644.

Löffler, K., M. Gienger, F. Pfeiffer, and H. Ulbrich (2004). „Sensors and control concept of a biped robot“. In: *IEEE Transactions on Industrial Electronics* 51.5, pp. 972–980. ISSN: 02780046. DOI: 10.1109/TIE.2004.834948.

Lohmeier, S. (2010). „Design and Realization of a Humanoid Robot for Fast and Autonomous Bipedal Locomotion“. Dissertation. Technische Universität München. ISBN: 9783868537345.

Lozano-Pérez, T. (1983). „Spatial Planning: A Configuration Space Approach“. In: *IEEE Transactions on Computers*, pp. 108–120. ISSN: 00189340. DOI: 10.1109/TC.1983.1676196.

Luh, J. Y. S., R. P. C. Paul, and M. W. Walker (1980). „Resolved-Acceleration Control of Mechanical Manipulators“. In: *IEEE Transactions on Automatic Control* 25.3, pp. 468–474. ISSN: 15582523. DOI: 10.1109/TAC.1980.1102367.

Machulis, K. (2010). *libnifalcon - Open source driver for the Novint Falcon*. qdot . github.io/libnifalcon/index.html, *accessed on* Mar. 14th, 2016.

Martin, D. P., J. Baillieul, and J. M. Hollerbach (1989). „Resolution of Kinematic Redundancy Using Optimization Techniques“. In: *IEEE Transactions on Robotics and Automation* 5.4, pp. 529–533. ISSN: 1042296X. DOI: 10.1109/70.88067.

Martin, S. and N. Hillier (2009). „Characterisation of the Novint Falcon Haptic Device for Application as a Robot Manipulator“. In: *Australasian Conference on Robotics and Automation (ACRA)*.

Mittendorfer, P. and G. Cheng (2011). „Humanoid multimodal tactile-sensing modules“. In: *IEEE Transactions on Robotics* 27.3, pp. 401–410. ISSN: 15523098. DOI: 10.1109/TRO.2011.2106330.

Monta, M., N. Kondo, and Y. Shibano (1995). „Agricultural robot in grape production system“. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 3, pp. 2504–2509. ISBN: 0-7803-1965-6. DOI: 10.1109/ROBOT.1995.525635.

Munckhof (2016). *Pluk-O-Trak*. www.munckhof.org/, *accessed on* Oct. 14th, 2016.

Muscato, G., M. Prestifilippo, N. Abbate, and I. Rizzuto (2005). „A prototype of an orange picking robot: past history, the new robot and experimental results“. In: *Industrial Robot: An International Journal* 32.2, pp. 128–138. ISSN: 0143-991X. DOI: 10.1108/01439910510582255.

Nakamura, Y. (1991). *Advanced Robotics: Redundancy and Optimization*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. ISBN: 0201151987.

Nakamura, Y. and H. Hanafusa (1987a). „Optimal Redundancy Control of Robot Manipulators“. In: *The International Journal of Robotics Research* 6.1, pp. 32–42. ISSN: 0278-3649. DOI: 10.1177/027836498700600103.

— (1987b). „Optimal Redundancy Control of Robot Manipulators“. In: *The International Journal of Robotics Research* 6.1, pp. 32–42. ISSN: 0278-3649. DOI: 10.1177/027836498700600103.

Nevins, J. L. and D. E. Whitney (1979). „Schrifttum / Literature“. English. In: *at - Automatisierungstechnik*. Vol. 27. International Centre for Mechanical Sciences 1-12. Springer Berlin Heidelberg, pp. 273–288. ISBN: 9783662403938. DOI: 10.1524/auto.1979.27.112.11.

New Holland Agriculture (2016). *New Holland Grape Harvester*. agriculture . newholland.com, *accessed on* Dec. 21st, 2015.

Nguyen, T. T., E. Kayacan, J. De Baedemaeker, and W. Saeys (2013). „Task and motion planning for apple harvesting robot“. In: *IFAC Proceedings Volumes*. Aalto University, Espoo, Finland, pp. 247–252. ISBN: 9783902823441. DOI: 10.3182/20130828-2-SF-3019.00063.

Nguyen, T. and J. Keresztes (2014). „Apple detection algorithm for robotic harvesting using a RGB-D camera“. In: *International Conference of Agricultural Engineering (AgEng)*. Zurich, pp. 6–10.

Niemeyer, G., C. Preusche, and G. Hirzinger (2008). „Telerobotics“. In: *Springer Handbook of Robotics*. Ed. by B. Siciliano and O. Khatib. Berlin, Heidelberg: Springer Berlin / Heidelberg. Chap. 31, pp. 741–757.

Nocedal, J. and S. J. Wright (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Berlin. ISBN: 978-0387-30303-1.

Novint Technologies Inc. (2012). *Falcon*. www.novint.com/, *accessed on* May 5th, 2016.

Oberti, R. and M. Hocevar (2014). *CROPS Deliverable D7.7: Report on evaluation of both prototype sprayers by laboratory experiments, follow up*. Project Report.

Oxbo International Corporation (2014). *Oxbo Harvesting Machines*. www . oxbocorp.com/, *accessed on* Dec. 21st, 2015.

Papageorgiou, M., M. Leibold, and M. Buss (2012). *Optimierung*. Springer Berlin Heidelberg, p. 541. ISBN: 978-3-540-34012-6. DOI: 10.1007/978-3-540-34013-3.

Park, J. and O. Khatib (2008). „Robot multiple contact control“. In: *Robotica* 26.05, pp. 667–677. ISSN: 0263-5747. DOI: 10.1017/S0263574708004281.

Patil, S., J. van den Berg, and R. Alterovitz (2011). „Motion planning under uncertainty in highly deformable environments“. In: *Robotics: Science and Systems*. Los Angeles, CA, USA. ISBN: 9780262517799. DOI: 10.15607/RSS.2011.VII.033.

Pellenc SA (2015). *Pellenc Grape Harvester*. www.pellenc.com/, *accessed on* Dec. 21st, 2015.

Perez, A., R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez (2012). „LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2537–2542. ISBN: 9781467314039. DOI: 10.1109/ICRA.2012.6225177.

Peterson, D. L., B. S. Bennedsen, W. C. Anger, and S. D. Wolford (1999). „A systems approach to robotic bulk harvesting of apples“. In: *Transactions of the ASAE* 42.4, pp. 871–876. ISSN: 0001-2351.

Pfaff, J. (2015). „Entwicklung eines modularen Manipulators für den Garten- und Weinbau“. Dissertation. Technische Univerität München.

Pfeiffer, F. and R. Johanni (1986). „A concept for manipulator trajectory planning“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1399–1405. ISBN: 0818606959. DOI: 10.1109/ROBOT.1986.1087500.

Polak, E. and G. Ribiere (1969). „Note sur la convergence de méthodes de directions conjuguées“. fra. In: *Rev. Franc. Inform. Rech. Oper.* 3.16, pp. 35–43. ISSN: 0764-583X.

Pontryagin, L. S., V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko (1962). *The mathematical theory of optimal processes*. New York [u.a.]: Interscience Publ. ISBN: 978-2-88124-077-5.

Quigley, M., K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng (2009). „ROS: an open-source Robot Operating System“. In: *ICRA Workshop on Open Source Software*.

Quinlan, S. and O. Khatib (1993). „Elastic bands: connecting path planning and control“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 802–807. ISBN: 0-8186-3450-2. DOI: 10.1109/ROBOT.1993.291936.

Ratliff, N., M. Zucker, J. A. Bagnell, and S. Srinivasa (2009). „CHOMP: Gradient optimization techniques for efficient motion planning“. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 489–494. ISBN: 978-1-4244-2788-8. DOI: 10.1109/ROBOT.2009.5152817.

Rimon, E. and D. E. Koditschek (1988). „Exact robot navigation using cost functions: The case of distinct spherical boundaries in En“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1791–1796. ISBN: 0-8186-0852-8. DOI: 10.1109/ROBOT.1988.12325.

ROS.org (2016). *ROS - Robot Operating System*. www.ros.org/about-ros/, *accessed on* Aug. 5th, 2016.

Salisbury J., J. (1984). „Interpretation of contact geometries from force measurements“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 240–247. ISBN: 0262022079. DOI: 10.1109/ROBOT.1984.1087180.

Sarig, Y. (1993). „Robotics of Fruit Harvesting: A State-of-the-art Review“. In: *Journal of Agricultural Engineering Research* 54.4, pp. 265–280. ISSN: 00218634. DOI: dx.doi.org/10.1006/jaer.1993.1020.

Scarfe, A. J., R. C. Flemmer, H. H. Bakker, and C. L. Flemmer (2009). „Development of an autonomous kiwifruit picking robot“. In: *IEEE International Conference on Autonomous Robots and Agents (ICARA)*, pp. 380–384. ISBN: 9781424427130. DOI: 10.1109/ICARA.2000.4804023.

Schertz, C. E. and G. K. Brown (1968). „Basic considerations in mechanizing citrus harvest.“ In: *Transactions of the ASAE* 11.3, pp. 343–348. ISSN: 0001-2351. DOI: 10.13031/2013.39405.

Schlemmer, M. (1996). „On-line trajectory optimization for kinematically redundant robot-manipulators and avoidance of moving obstacles“. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 474–479. ISBN: 0-7803-2988-0. DOI: 10.1109/ROBOT.1996.503821.

Schmitz, A., P. Maiolino, M. Maggiali, L. Natale, G. Cannata, and G. Metta (2011). „Methods and technologies for the implementation of large-scale robot tactile sensors“. In: *IEEE Transactions on Robotics* 27.3, pp. 389–400. ISSN: 15523098. DOI: 10.1109/TRO.2011.2132930.

Schulman, J., J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel (2013). „Finding locally optimal, collision-free trajectories with sequential convex optimization“. In: *Robotics: Science and Systems*. Vol. 9. 1, pp. 1–10. ISBN: 978-981-07-3937-9. DOI: 10.1.1.387.4642.

Schwienbacher, M. (2013). „Efficient Algorithms for Biped Robots – Simulation, Collision Avoidance and Angular Momentum Tracking“. Dissertation. Technische Universität München.

Schwienbacher, M., T. Buschmann, S. Lohmeier, V. Favot, and H. Ulbrich (2011). „Self-collision avoidance and angular momentum compensation for a biped humanoid robot“. In: pp. 581–586. ISBN: 9781612843865. DOI: 10.1109/ICRA.2011.5980350.

Shiller, Z. and S. Dubowsky (1991). „On Computing the Global Time Optimal Motions of Robotic Manipulators in the Presence of Obstacles“. In: *IEEE Transactions on Robotics and Automation* 7.6, pp. 785–797. ISSN: 1042296X. DOI: 10.1109/70.105387.

Shin, K. and N. McKay (1986). „A dynamic programming approach to trajectory planning of robotic manipulators“. In: *IEEE Transactions on Automatic Control* 31.6, pp. 491–500. ISSN: 0018-9286. DOI: 10.1109/TAC.1986.1104317.

Siciliano, B., L. Sciavicco, L. Villani, and G. Oriolo (2009). *Robotics: Modelling, Planning and Control*. Ed. by J. Grimble and A. M. Johnson. Springer Verlag London Ltd. DOI: 10.1007/978-1-84628-642-1.

Siciliano, B. (1990). „Kinematic control of redundant robot manipulators: A tutorial". In: *Journal of Intelligent and Robotic Systems* 3.3, pp. 201–212. ISSN: 09210296. DOI: 10.1007/BF00126069.

Siemens PLM Software Inc. (2015). *PLM Components: Kineo*. www . plm . automation . siemens . com / en{_} us / products / open / kineo / index . shtml, *accessed on* Jan. 1st, 2016.

Singh, S. K. and M. C. Leu (1991). „Manipulator Motion Planning in the Presence of Obstacles and Dynamic Constraints". In: *The International Journal of Robotics Research* 10.2, pp. 171–187. ISSN: 0278-3649. DOI: 10.1177/027836499101000208.

Son, J. S., M. R. Cutkosky, and R. D. Howe (1996). „Comparison of contact sensor localization abilities during manipulation". In: *Robotics and Autonomous Systems* 17.4, pp. 217–233. ISSN: 09218890. DOI: 10.1016/0921-8890(95)00068-2.

Stilman, M., J. U. Schamburek, J. Kuffner, and T. Asfour (2007). „Manipulation planning among movable obstacles". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3327–3332. ISBN: 1424406021. DOI: 10 . 1109/ROBOT.2007.363986.

Strohmayr, M. W. and D. Schneider (2013). „The DLR artificial skin step II: Scalability as a prerequisite for whole-body covers". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4721–4728. ISBN: 9781467363587. DOI: 10.1109/IROS.2013.6697036.

Strohmayr, M. W., H. Worn, and G. Hirzinger (2013). „The DLR artificial skin step I: Uniting sensitivity and collision tolerance". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1012–1018. ISBN: 9781467356411. DOI: 10.1109/ICRA.2013.6630697.

Stryk, O. V. (1994). „Optimal control of the industrial robot Manutec r3". In: *Computational Optimal Control*. Ed. by R. Bulirsch and D. Kraft. Vol. 115. ISNM International Series of Numerical Mathematics. Birkhäuser Basel, pp. 367–382. ISBN: 978-3-7643-5015-4. DOI: 10.1007/978-3-0348-8497-6_30.

Sucan, I., M. Moll, and L. Kavraki (2014). *Open Motion Planning Library: A Primer*. ompl.kavrakilab.org/OMPL_Primer.pdf, *accessed on* Oct. 13th, 2014.

Tassa, Y., T. Erez, and W. D. Smart (2008). „Receding Horizon Differential Dynamic Programming". In: *Advances in Neural Information Processing Systems 20*. Ed. by J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis. Curran Associates, Inc., pp. 1465–1472.

Tassa, Y., T. Erez, and E. Todorov (2012). „Synthesis and stabilization of complex behaviors through online trajectory optimization". In: *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pp. 4906–4913. ISBN: 9781467317375. DOI: 10.1109/IROS.2012.6386025.

Tassa, Y., N. Mansard, and E. Todorov (2014). „Control-limited differential dynamic programming". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1168–1175. ISBN: 978-1-4799-3685-4. DOI: 10.1109/ICRA.2014.6907001.

Tenzer, Y., L. P. Jentoft, and R. D. Howe (2014). „The feel of MEMS barometers: Inexpensive and easily customized tactile array sensors“. In: *IEEE Robotics and Automation Magazine* 21.3, pp. 89–95. ISSN: 10709932. DOI: 10.1109/MRA.2014.2310152.

Tillett, N. (1993). „Robotic Manipulators in Horticulture: A Review“. In: *Journal of Agricultural Engineering Research* 55.2, pp. 89–105. ISSN: 00218634. DOI: 10.1006/jaer.1993.1035.

Todorov, E. and W. Li (2003). „Optimal control methods suitable for biomechanical systems“. In: *IEEE International Conference on Engineering in Medicine and Biology Society (IEMBS)*, pp. 1758–1761. ISBN: 0-7803-7789-3. DOI: 10.1109/IEMBS.2003.1279748.

Trinidad, J. B. (2011). „Machine for Automatically Harvesting Fruit Cultivated in Rows“. Patent EP 2371204A1.

Tsuji, T., Y. Kaneko, and S. Abe (2009). „Whole-body force sensation by force sensor with shell-shaped end-effector“. In: *IEEE Transactions on Industrial Electronics* 56.5, pp. 1375–1382. ISSN: 02780046. DOI: 10.1109/TIE.2009.2014748.

Ulbrich, H. (1996). *Maschinendynamik*. Stuttgart: Teubner Studienbücher. ISBN: 3-519-03233-3.

Van den Munckhof, M. and P. Van den Munckhof (1994). „Vehicle and a trailer for same“. Patent EP 0580205A1.

Van Henten, E. J., J. Hemming, B. A. J. Van Tuijl, J. G. Kornet, J. Meuleman, J. Bontsema, and E. A. Van Os (2002). „An autonomous robot for harvesting cucumbers in greenhouses“. In: *Autonomous Robots* 13.3, pp. 241–258. ISSN: 09295593. DOI: 10.1023/A:1020568125418.

Vande Weghe, M., D. Ferguson, and S. S. Srinivasa (2008). „Randomized path planning for redundant manipulators without inverse kinematics“. In: *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, pp. 477–482. ISBN: 9781424418626. DOI: 10.1109/ICHR.2007.4813913.

Villani, L. and J. De Schutter (2008). „Force Control“. In: *Handbook of Robotics*. Ed. by B. Siciliano and O. Khatib. Springer Berlin / Heidelberg. Chap. 7, pp. 161–185.

Vukobratovic, M. and N. Kircanski (1986). *Kinematics and Trajectory Synthesis of Manipulation Robots*. Springer-Verlag Berlin Heidelberg. ISBN: 978-3-642-82197-4.

Wächter, A. and L. T. Biegler (2006). „On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming“. In: *Mathematical Programming* 106.1, pp. 25–57. ISSN: 0025-5610. DOI: 10.1007/s10107-004-0559-y.

Wang, D. and Y. Hamam (1992). „Optimal Trajectory Planning of Manipulators With Collision Detection and Avoidance“. In: *The International Journal of Robotics Research* 11.5, pp. 460–468. ISSN: 0278-3649. DOI: 10.1177/027836499201100503.

Ward, G. (2003). *New mechanical and robotic harvesting technologies to increase fruit production efficiency Glynn Ward Department of Agriculture Western Australia*. English. Technical Report. Sydney.

Whitney, D. (1969). „Resolved Motion Rate Control of Manipulators and Human Prostheses“. In: *IEEE Transactions on Man Machine Systems* 10.2, pp. 47–53. ISSN: 0536-1540. DOI: 10.1109/TMMS.1969.299896.

Wouters, N., K. Vandevoorde, B. Lenaerts, T. T. Nguyen, S. Evain, L. L. Guttierez, K. Rutten, B. De Ketelaere, and W. Saeys (2014). *CROPS Deliverable D6.8: Test report of prototype including suggestions for improvement*. Project Report.

Zhang, H., S. Ahmad, and G. Liu (2015). „Torque Estimation for Robotic Joint With Harmonic Drive Transmission Based on Position Measurements“. In: *IEEE Transactions on Robotics* 31.2, pp. 322–330. ISSN: 15523098. DOI: 10.1109/TRO.2015.2402511.

Zillich, M. and W. Feiten (2012). „A versatile tactile sensor system for covering large and curved surface areas“. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 20–24. ISBN: 9781467317375. DOI: 10.1109/IROS.2012.6385806.

Zucker, M., N. Ratliff, a. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. a. Bagnell, and S. S. Srinivasa (2013). „CHOMP: Covariant Hamiltonian optimization for motion planning“. In: *The International Journal of Robotics Research* 32.9-10, pp. 1164–1193. ISSN: 0278-3649. DOI: 10.1177/0278364913488805.

Author's Publications

Baur, J., S. Dendorfer, J. Pfaff, C. Schütz, T. Buschmann, and H. Ulbrich (2014). „Experimental Friction Identification in Robot Drives". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China, pp. 6006–6011. DOI: 10.1109/ICRA.2014.6907744.

Baur, J., C. Schütz, J. Pfaff, T. Buschmann, and H. Ulbrich (2014). „Path Planning for a Fruit Picking Manipulator". In: *International Conference of Agricultural Engineering (AgEng)*. Zurich, pp. 6–10.

Baur, J., C. Schütz, J. Pfaff, and H. Ulbrich (2014). „Modeling and Control of Pneumatic Artificial Muscles in an Antagonistic Set-up". In: *International Conference on Multibody System Dynamics (IMSD/ACMD)*. Busan, Korea.

Hildebrandt, A.-C., C. Schütz, D. Wahrmann, R. Wittmann, and D. Rixen (2016). „A Flexible Robotic Framework for Autonomous Manufacturing Processes: Report from the European Robotics Challenge Stage 1". In: *IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*.

Oberti, R., M. Marchi, P. Tirelli, A. Calcante, M. Iriti, M. Hočevar, J. Baur, J. Pfaff, C. Schütz, and H. Ulbrich (2014). „CROPS Agricultural Robot: Application to Selective Spraying of Grapevine's Diseases". In: *Proceedings of the Second RHEA International Conference on Robotics and associated High-technologies and Equipment for Agriculture*.

Oberti, R., M. Marchi, P. Tirelli, A. Calcante, M. Iriti, E. Tona, M. Hočevar, J. Baur, J. Pfaff, C. Schütz, and H. Ulbrich (2016). „Selective spraying of grapevines for disease control using a modular agricultural robot". In: *Biosystems Engineering*. ISSN: 15375110. DOI: 10.1016/j.biosystemseng.2015.12.004.

Pfaff, J., J. Baur, and C. Schütz (2014). „Design of Drive Units for Agricultural Robots". In: *International Conference of Agricultural Engineering (AgEng)*. Zurich, pp. 6–10.

Schütz, C., J. Baur, J. Pfaff, T. Buschmann, and H. Ulbrich (2015). „Evaluation of a direct optimization method for trajectory planning of a 9-DOF redundant fruit-picking manipulator". In: *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2660–2666. DOI: 10.1109/ICRA.2015.7139558.

Schütz, C., J. Pfaff, J. Baur, T. Buschmann, and H. Ulbrich (2014). „A modular robot system for agricultural applications". In: *International Conference of Agricultural Engineering (AgEng)*. Zurich, pp. 6–10.

Schütz, C. (2014). *CROPS - 3rd periodical scientific report of WP1*. Project Report. München, Germany.

Schütz, C., J. Baur, J. Pfaff, T. Buschmann, and H. Ulbrich (2014). „Multipurpose Redundant Manipulators for Agricultural Tasks". In: *Austrian Robotics Workshop (ARW)*.

Schütz, C., T. Buschmann, J. Baur, J. Pfaff, and H. Ulbrich (2014). „Predictive online inverse kinematics for redundant manipulators". In: *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, pp. 5056–5061. DOI: 10.1109/ICRA.2014.6907600.

Schütz, C., J. Pfaff, F. Sygulla, D. Rixen, and H. Ulbrich (2015). „Motion Planning for Redundant Manipulators in Uncertain Environments based on Tactile Feedback". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany, pp. 6387–6394. ISBN: 978-1-4799-9994-1. DOI: 10.1109/IROS.2015.7354290.

Schütz, C., T. Ponn, F. Sygulla, D. Rixen, and H. Ulbrich (2016). „Proprioceptive Estimation of External Joint Torques at a 9-DOF Manipulator". In: *International Conference on Multibody System Dynamics (IMSD)*. Montreal, Canada.

Sygulla, F., C. Schütz, and D. Rixen (2016). „Adaptive Motion Control in Uncertain Environments using Tactile Feedback". In: *IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1277–1284. ISBN: 978-1-5090-2065-2. DOI: 10.1109/AIM.2016.7576946.

Ulbrich, H., J. Baur, J. Pfaff, and C. Schütz (2015). „Design and Realization of a Redundant Modular Multipurpose Agricultural Robot". In: *Proceedings of the XVII International Symposium on Dynamic Problems of Mechanics (DINAME)*. Natal, Brazil.

Student Theses

Berger, J. (2015). „Predictive Trajectory Optimization using Differential Dynamic Programming“. Master's Thesis. Technische Universität München.

Fagerer, P. (2014). „Heuristische Bahnplanung für einen Ernteroboter“. Bachelor's Thesis. Technische Universität München.

Fiordelisio Junior, E. (2015). „Design of a Security Brake for a Lightweight Robot Arm“. Semester Thesis. Technische Universität München.

Hazirbas, C. (2013). „Vision System for a Harvesting Robot“. Interdisciplinary Project. Technische Universität München.

Hirth, M. (2015). „Bahnplanung für ein Fahrzeug mit Vierradlenkung“. Master's Thesis. Technische Universität München.

Kiener, M. (2014). „Sampling-basierte Planungsalgorithmen für einen Ernteroboter“. Bachelor's Thesis. Technische Universität München.

Lochner, F. (2015). „Zeitliche Parametrierung vorgeplanter Bahnen“. Semester Thesis. Technische Universität München.

Miller, C. (2015). „Dynamische Modellierung eines Ernteroboters“. Bachelor's Thesis. Technische Universität München.

Ponn, T. (2016). „Schätzung externer Momente für einen modularen Manipulator“. Semester Thesis. Technische Universität München.

Reuß, F. (2013a). „Optimale Lösungsansätze der inversen Kinematik für redundante Manipulatoren“. Semester Thesis. Technische Universität München.

— (2013b). „Optimale Trajektorienplanung für redundante Manipulatoren“. Diploma Thesis. Technische Universität München.

Roder, S. (2015). „Auswertung eines taktilen Sensorelementes am Ernteroboter“. Bachelor's Thesis. Technische Universität München.

Schaller, R. (2016). „Haptische Benutzerschnittstelle für einen redundanten Manipulator“. Interdisciplinary Project. Technische Universität München.

Scheurer, M. (2013). „Bahnplanung für einen Ernteroboter“. Semester Thesis. Technische Universität München.

Smith, T. (2014). „Differentielle Dynamische Programmierung zur Trajektorienoptimierung“. Bachelor's Thesis. Technische Universität München.

Sygulla, F. (2015). „Motion Planning with Tactile Feedback“. Master's Thesis. Technische Universität München.

Yousefpour, S. (2015). „3D Image Processing for a Harvesting Manipulator“. Interdisciplinary Project. Technische Universität München.