RPTU

DISSERTATION

Optimization based Algorithms for Task Planning and Predictive Motion Control in Robotics

Ausgeführt zum Zwecke der Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

unter der Leitung von
Prof. Dr.- Ing. Naim Bajcinca
Lehrstuhl für Mechatronik in Maschinenbau und Fahrzeugtechnik

genehmigt vom
Fachbereich Maschinenbau und Verfahrenstechnik
der
Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau

Vorgelegt von
Herrn
Dipl.-Ing. Argtim Tika
aus Kërçovë, Nordmazedonien

Kaiserslautern, Februar 2024

Rheinland-Pfälzische Technische Universität Kaiserslautern-Landau
67663 Kaiserslautern, Gottlieb-Daimler-Straße 47

Tag der Disputation:	22.12.2023

Promotionskommission

Vorsitzender:	Prof. Dr.-Ing. Sergiy Antonyuk
Erster Berichterstatter:	Prof. Dr.-Ing. Naim Bajcinca
Zweiter Berichterstatter:	Prof. Dr. rer. nat. Karsten Berns

D 386

Prindërve të mi

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Mechatronik in Maschinenbau und Fahrzeugtechnik (MEC) im Fachbereich Maschinenbau und Verfahrenstechnik der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau (RPTU).

Mein besonderer Dank gilt meinem Doktorvater Prof. Dr.-Ing. Naim Bajcinca für die Möglichkeit unter seiner Betreuung meine wissenschaftliche Karriere zu starten. Besonders bin ich dankbar für sein hohes Interesse an meiner Arbeit, die zahlreichen fachlichen Anregungen und Diskussionen, und vor allem für das entgegengebrachte Vertrauen und die Freiheit mich fachlich und persönlich in vielen Bereichen weiterzuentwickeln. Des Weiteren möchte ich Herrn Prof. Dr.-Ing. Sergiy Antonyuk für die Übernahme des Vorsitzes der Promotionskommission und Prof. Dr. rer. nat. Karsten Berns für das Interesse an meiner Arbeit und die Erstellung des Zweigutachtens danken.

Ich bin zahlreichen Kolleginnen und Kollegen am Lehrstuhl für die Hilfsbereitschaft und die vielen anregenden fachlichen und persönlichen Diskussionen zu großem Dank verpflichtet. Vor allem möchte ich an dieser Stelle den hervorragenden kollegialen Zusammenhalt hervorheben und mich für die fruchtbare Kooperation während der vergangenen Jahre, die zahlreichen Ratschläge sowie Unterstützung beim Aufbau von Prüfständen und Demonstratoren im Labor herzlich bedanken.

Darüber hinaus möchte ich mich bei den Studenten bedanken, die unter meiner Betreuung im Rahmen von Abschlussarbeiten zum Gelingen dieser Arbeit beigetragen haben. Dankbar bin ich auch den industriellen Kooperationspartnern die im Zuge der Forschungsprojekte CooPick, KIMKO, und KORINS, gefördert von der Arbeitsgemeinschaft industrieller Forschungsvereinigungen (AiF) unter den Förderkennzeichen ZF4335706DB7, ZF4335711PO9, und ZF4335715DB9, das Zustandekommen dieser Arbeit unterstützt haben.

Der größte Dank gebührt meiner Familie, meinen Freunden und besonders meinen Eltern für die uneingeschränkte Unterstützung während meines Lebens und meiner wissenschaftlichen Laufbahn.

Kaiserslautern, Februar 2024

Abstract

Since their introduction, robots have primarily influenced the industrial world, providing new opportunities and challenges for humans and machinery. With the introduction of lightweight robots and mobile robot platforms, the field of robot applications has been expanded, diversified, and brought closer to society. The increased degree of digitalization and the personalization of goods and products require an enhanced and flexible robot deployment by operating several multi-robot systems along production processes, industrial applications, assembly and packaging lines, transport systems, etc.

Efficient and safe robot operation relies on successful task planning followed by the computation and execution of task-performing motion trajectories. This thesis addresses these issues by developing, implementing, and validating optimization-based methods for task and trajectory planning in robotics, considering certain optimality and performance criteria. The focus is mainly on the time optimality of the presented approaches with respect to both execution and computation time without compromising safe robot use.

Driven by a systematic approach, the basis for the algorithm development is established first by modeling the kinematics and dynamics of the considered robots and identifying required dynamic parameters. In a further step, time-optimal task and trajectory planning algorithms for a single robotic arm are developed. Initially, a hierarchical approach is introduced consisting of two decoupled optimization-based control policies, a binary problem for task planning, and a continuous model predictive trajectory planning problem. The two layers of the hierarchical structure are then merged into a monolithic layer, resulting in a hybrid structure in the form of a mixed-integer optimization problem for inherent task and trajectory planning.

Motivated by a multi-robot deployment, the hierarchical control structure for time-optimal task and trajectory planning is extended for the case of a two-arm robotic system with highly overlapping operational spaces, leading to challenging robot motions with high inter-robot collision potential. To this end, a novel predictive approach for collision avoidance is proposed based on a continuous approximation of the robot geometry, resulting in a nonlinear optimization problem capable of online applications with real-time requirements. Towards a mobile and flexible robot platform, a model predictive path-following controller for an omnidirectional mobile robot is introduced. Here, a time-minimal approach is also applied, which consists of the robot following a given parameterized path as accurately as possible and at maximum speed.

The performance of the proposed algorithms and methods is experimentally analyzed and validated under real conditions on robot demonstrators. Implementation details, including the resulting hardware and software architecture, are presented, followed by a detailed description of the results. Concrete and industry-oriented demonstrators for integrating robotic arms in existing manual processes and the indoor navigation of a mobile robot complete the work.

Kurzfassung

Roboter haben seit deren Einführung die Industriewelt weitgehend geprägt und für Mensch und Maschine neue Möglichkeiten und Herausforderungen geschaffen. Mit der Einführung von Leichtbaurobotern und mobilen Roboterplattformen wurde das Robotereinsatzgebiet erweitert, diversifiziert und näher an die Gesellschaft herangeführt. Der zunehmende Digitalisierungsgrad und die Personalisierung von Waren und Gütern erfordern einen steigernden und flexiblen Robotereinsatz, auch durch den Betrieb von Multirobotersystemen entlang von Produktionsprozessen, industriellen Anwendungen, Montage- und Verpackungslinien, Transportsystemen usw.

Ein effizienter und sicherer Roboterbetrieb stützt sich auf eine erfolgreiche Aufgabenplanung gefolgt von der Berechnung und Ausführung von aufgabengerechten Trajektorien. In dieser Hinsicht befasst sich die vorliegende Arbeit mit der Entwicklung, Implementierung und Validierung von optimierungsbasierten Methoden zur Aufgaben- und Trajektorienplanung unter Berücksichtigung von gewissen Optimalitäts- und Performancekriterien. Der Fokus liegt dabei vor allem auf der Zeitoptimalität sowohl hinsichtlich der Ausführungs- als auch der Berechnungszeit, ohne den sicheren Robotereinsatz zu beeinträchtigen.

Getrieben durch einen systematischen Ansatz wird durch die Modellierung der Kinematik und Dynamik der Roboter und der Identifikation von dynamischen Parametern die Grundlage für die Algorithmenentwicklung geschaffen. In einem ersten Schritt wird für einen Roboterarm eine hierarchische Regelungsstruktur vorgeschlagen, bestehend aus zwei entkoppelten zeitoptimalen Optimierungsproblemen, einem binären zur Aufgabenplanung und einem kontinuierlichen modellprädiktiven Trajektorienplanungsproblem. Anschließend werden beide Ebenen des hierarchischen Ansatzes zu einer monolithischen Einheit zusammengeführt was in einer hybriden Struktur in Form eines gemischt ganzzahligen Optimierungsproblems für inhärente Aufgaben- und Trajektorienplanung resultiert.

Motiviert durch einen Multirobotereinsatz wird für ein zweiarmiges Robotersystem mit stark überlappenden Arbeitsbereichen ein zeitoptimaler Algorithmus nach dem Schema der hierarchischen Struktur entwickelt. Für die Kollisionsvermeidung zwischen den Roboterarmen wird ein neuartiger prädiktiver Ansatz vorgeschlagen, der auf eine kontinuierliche Approximation der Robotergeometrie aufbaut und in einem nichtlinearen Optimierungsproblem fähig für den Einsatz in Onlineanwendungen mit Echtzeitanforderungen resultiert. Auf dem Weg zu einer mobilen und flexiblen Roboterplattform wird ein modellprädiktiver Regler vorgestellt, mit dem ein omnidirektionaler mobiler Roboter einem vorgegebenen parametrisierten Pfad mit maximaler Geschwindigkeit folgen kann.

Die Performance der vorgeschlagenen Algorithmen wird auf Demonstratoren experimentell analysiert und validiert. Dabei werden stets Implementierungsdetails, einschließlich der Hard- und Softwarearchitektur präsentiert, gefolgt von einer ausführlichen Ergebnisbeschreibung. Konkrete und industrienahe Anwendungsfälle zur Integration von Roboterarmen in bestehende manuelle Prozesse und die Indoor-Navigation eines mobilen Roboters runden und schließen die Arbeit ab.

Contents

1 Introduction **1**
1.1 Task planning . . . 2
1.2 Motion planning . . . 3
1.2.1 Sampling-base approaches . . . 3
1.2.2 Optimization-based approaches . . . 4
1.3 Path-following control . . . 6
1.4 Goal and overview of this work . . . 7

2 Mathematical Modeling **9**
2.1 Kinematics of robot manipulators . . . 9
2.1.1 Forward kinematics . . . 10
2.1.1.1 Denavit-Hartenberg convection . . . 12
2.1.1.2 Denavit-Hartenberg parameters of the UR5 robot . . . 14
2.1.2 Inverse kinematics . . . 17
2.2 Manipulator Jacobian . . . 23
2.3 Dynamics of robot manipulators . . . 25
2.3.1 Lagrange method . . . 25
2.3.1.1 Kinetic and potential energy . . . 26
2.3.1.2 Generalized torque . . . 27
2.3.1.3 Equation of motion . . . 29
2.3.2 Newton-Euler methods . . . 29
2.4 Kinematics of an omnidirectional mobile robot . . . 32
2.5 Dynamics of an omnidirectional mobile robot . . . 35

3 Dynamic Parameter Identification **37**
3.1 Identification model . . . 38
3.2 Excitation trajectory . . . 39
3.3 Parameter estimation . . . 41

4 Task and Trajectory Planning **49**
4.1 Problem formulation . . . 50
4.2 Hierarchical control structure . . . 51
4.2.1 Task scheduling . . . 51
4.2.1.1 Minimum-distance scheduling . . . 52
4.2.1.2 Minimum-time scheduling . . . 53
4.2.2 Trajectory planning . . . 55
4.2.2.1 Prediction model . . . 56
4.2.2.2 MPC-based trajectory planning . . . 57

4.3 Hybrid control structure . . . 58
  4.3.1 Single-step hybrid controller . . . 59
  4.3.2 Multiple-step hybrid controller . . . 62
4.4 Implementation . . . 64
4.5 Experimental results . . . 67

5 Cooperative Task and Trajectory Planning **75**
5.1 Problem formulation . . . 76
5.2 Hierarchical control structure . . . 77
  5.2.1 Task scheduling . . . 78
  5.2.2 Centralized trajectory planning . . . 80
    5.2.2.1 Prediction model . . . 81
    5.2.2.2 Collision avoidance . . . 81
    5.2.2.3 Time-optimal MPC . . . 88
  5.2.3 Distributed trajectory planning . . . 90
5.3 Experimental results . . . 93

6 Predictive Path-Following Control for Mobile Robots **103**
6.1 Problem formulation . . . 103
6.2 Predictive-based path-following control . . . 105
6.3 Experimental results . . . 107

7 Demonstrators **111**
7.1 Bottle sorting system . . . 111
7.2 Fruit packing station . . . 114
  7.2.1 Task scheduling and trajectory planning . . . 115
  7.2.2 Simulation results . . . 119
  7.2.3 Experimental results . . . 124
7.3 Mobile robot platform . . . 134

8 Summary and Outlook **139**

List of Figures

2.1 UR5 robot with classical Denavit-Hartenberg parameters. 15
2.2 UR5 robot with modified Denavit-Hartenberg parameters. 16
2.3 Schematic illustration for computing the joint angle θ_1. 17
2.4 Schematic illustration for computing the joint angle θ_5. 19
2.5 Schematic illustration for computing the joint angle θ_6. 20
2.6 Schematic illustration for computing the joint angle θ_3. 21
2.7 Schematic illustration for computing the joint angles θ_2 and θ_4. 22
2.8 Omnidirectional mobile robot. 32
2.9 External torque on a mecanum wheel. 36

3.1 Solutions for the optimized persistent excitation trajectory. 42
3.2 Persistent excitation trajectory in the joint space. 43
3.3 End-effector path resulting from the persistent excitation trajectory. . . . 43
3.4 Comparison between predicted $\hat{\tau}_j$ and measured τ_j joint torques. 46

4.1 Experimental setup with robot manipulator performing pick-and-place tasks. 50
4.2 Hierarchical control structure. 51
4.3 Hybrid controller for online task and trajectory planning. 58
4.4 System architecture for a single robot manipulator using ROS. 64
4.5 ROS and RTDE reading time delays. 65
4.6 Joint position and velocity tracking error. 66
4.7 Experimental results for task scheduling and trajectory planning involving multiple task points of the same class. 68
4.8 Solver time over MPC iterations. 71
4.9 Experimental results for task scheduling and trajectory planning involving multiple task points of different classes. 73

5.1 Two robots performing pick-and-place tasks in a shared operational space. 76
5.2 Hierarchical control structure for cooperative robot control. 77
5.3 Robot geometry approximation with smooth Bézier curves. 82
5.4 2D projection method for collision avoidance. 86
5.5 3D optimal perpendicular vector calculation for collision avoidance. 87
5.6 Switch functions for robot motion coordination in distributed robot control. 91
5.7 Synchronized vs. unsynchronized trajectory planning for both robots. . . 92
5.8 Synchronized vs. unsynchronized planning for each robot individually. . . 93
5.9 System architecture for cooperative robot control using ROS. 94
5.10 Time-optimal task sequences for Robot 1 and Robot 2. 96
5.11 Picture sequences demonstrating cooperative robot task execution. 97

5.12 Minimum distance between the robots computed during the experiment. . 99
5.13 Robot joint position, velocity, and acceleration trajectories. 99
5.14 Solver time over iterations for cooperative robot control. 100
5.15 Minimum-distance task sequences for Robot 1 and Robot 2. 101
5.16 Minimum-time task sequence for single robot manipulator. 101

6.1 Path-following control of a mobile robot. 104
6.2 System architecture for mobile robot path-following control. 107
6.3 Path-following control experiments using a circular reference path. 109
6.4 Path-following control experiments using a lemniscate reference path. . . 109
6.5 Path-following control experiments using a pinched circular reference path. 110
6.6 Mobile robot platform with omnidirectional wheels. 110

7.1 Experimental setup of a robot-based bottle sorting system. 112
7.2 Experimental demonstration of the robot-based bottle sorting system. . . 113
7.3 Experimental setup of a dual-arm robot integration into existing packing processes. 114
7.4 Schematic illustration of the fruit packing station. 117
7.5 Dynamic simulation model of the experimental setup. 119
7.6 Simulation results: optimal task scheduling results for both robots. 120
7.7 Simulation results: paths of the robots' end-effectors. 120
7.8 Simulation results: collision avoidance using 2D projection method. 122
7.9 Simulation results: collision avoidance using 3D optimal vector calculation. 123
7.10 Simulation results: minimum distance between approximating Bézier curves. 123
7.11 System architecture for cooperative pick-and-place with dynamic task points. 124
7.12 Experimental setup in Test Case 1 for collision avoidance validation. . . . 125
7.13 Task scheduling sequences for both robots in Test Case 1. 126
7.14 Experimental validation of the collision avoidance. 127
7.15 Minimum distance between approximating Bézier curves. 128
7.16 Computed minimum final time for Test Case 1. 128
7.17 Robot joint position, velocity, and acceleration trajectories. 129
7.18 Solver time over iterations in Test Case 1. 130
7.19 Cooperative pick-and-place robot task execution on a conveyor belt. . . . 132
7.20 Computed minimum final time for Test Case 2. 133
7.21 Solver time over iterations in Test Case 2. 133
7.22 Mobile robot demonstrator. 134
7.23 Occupancy map of the environment including PRM nodes. 135
7.24 Control points of validation paths generated with PRM and A*. 135
7.25 Experimental results of path-following control using a closed reference path. 136
7.26 Experimental results of path-following control using an open reference path. 136
7.27 Computation time of the path-following controller over MPC iterations. . 137

List of Tables

2.1 Denavit-Hartenberg parameters for the UR5. 15
2.2 Modified (Khalil-Kleinfinger) Denavit-Hartenberg parameters for the UR5. 16
2.3 Torque vectors expressed in the local body-attached link frames. 28
2.4 Geometric parameters of the mobile robot. 34

3.1 Normalized estimation errors. 45
3.2 Regrouped parameters and standard deviations. 47

4.1 Tuned PID parameters. 67
4.2 Maximum angular distances for Test Case 1 when applying 1S-HMPC. . . 70
4.3 Maximum angular distances for Test Case 2 when applying 1S-HMPC. . . 72
4.4 Maximum angular distances for Test Case 3 when applying 1S-HMPC. . . 74

5.1 Parameters used for experiments. 95
5.2 Experimental results for $N_T = 10$ and $N_C = 5$. 100

7.1 Experimental results for Test Case 1 using different horizon lengths. . . . 130
7.2 Experimental results for Test Case 1a and $N_T = 15$. 131
7.3 Experimental results for Test Case 1a, with $N_T = 15$ and $N_C = 8$. 131
7.4 Experimental results for Test Case 2. 133

CHAPTER 1

Introduction

Robotic systems have significantly revolutionized industrial workplaces and continuously increased the level of automation in various industrial sectors. The use of robots is wide-ranging and steadily expanding with increasing digitalization and Industry 4.0. It involves tasks in multiple domains such as packaging, palletizing, sorting, assembly, and the execution of specific functions like machine tending, drilling, welding, soldering, painting, etc. [1]. Most of the robotic applications fall under the category of pick-and-place tasks and are well-suited as benchmarking frameworks in robotics due to their widespread use in various automated processes and operations. Prior to executing the pick-and-place tasks, task planning is usually performed to assign a task or a sequence of tasks to the robots. Task planning is then followed by point-to-point trajectory generation to perform the assigned tasks.

With the increased diversity and the personalization of goods and products, production in short batches becomes more attractive, leading to multiple robot tasks of different classes. Despite increasing automation, manual methods are currently still considered the only cost-effective solution for the production of small batches, characteristic of high-mix, low-volume manufacturers [2]. The increased flexibility appears to be a key element towards higher levels of operational efficiency and productivity. This can be achieved by empowering workers to work directly with automated systems or by facilitating robot integration, enabling adaptations of robotic solutions to new workspaces [3].

Motivated by the increase of efficiency and productivity, in some applications, multi-robot systems are deployed with robotic manipulators operating in close proximity and in cooperation with each other, sharing a common working environment. In such cases, the coordinated execution of optimal discrete task sequences is typically of interest, resulting in a challenging problem due to the excessively large number of involved combinatorial motion scenarios. Hence, increasing the performance of robotic applications involving multiple goals requires efficient task scheduling and the subsequent execution of fast and collision-free motion trajectories. Robot task allocation and scheduling is usually performed separately from the trajectory planning. However, approaches also exist to tightly couple the two problems or consider them jointly within the framework of an algorithm. Nevertheless, most methods follow the traditional way used for repetitive tasks of planning and programming robot applications offline [4].

This work considers robot applications involving multiple pick-and-place tasks performed by a single or two robot manipulators. The problem formulations are motivated by specific industrial applications about the integration of robot manipulators into existing manual processes. Therefore, optimization-based algorithms are introduced, addressing the problem of time-optimal robot task execution. This involves task scheduling and trajectory planning by considering both problems independently within the scope of a hierarchic framework or combined in the form of a hybrid control structure. In addition, towards a flexible mobile robot platform, a motion control algorithm for fast and accurate indoor path following is presented.

1.1 Task planning

The research on robot task scheduling has been primarily focused on minimizing the cycle time by determining an optimal sequence of a set of unordered task points in the three dimensional (3D) operational space $\mathcal{W}$ (robot working space). Task scheduling problems are often modeled as an extension of the traveling salesman problem (TSP) [5]. An overview of related TSP-like combinatorial problems for application in robot task sequencing is provided in [4]. The scope of the survey paper is limited to offline sequencing algorithms for a single robot manipulator and also gives a brief insight into related planning domains, such as multi-robot task planning, production scheduling, robot path planning, etc. In [6], a modified TSP is introduced to obtain the optimal traveling schedule for a 3 DoF (degree of freedom) robot performing drilling / spot welding tasks.

Since the robot task execution is performed in the operational space and the robot is controlled in the configuration space $\mathcal{C}$, one obvious extension of the TSP algorithm is the application of inverse kinematics (IK), as shown in [7] by introducing simplified kinematic equations, in determining optimal task sequences. A further modification of the TSP to account for multiple solutions of the inverse kinematics is presented in [8]. This idea is extended in [9] combining task sequencing with path planning for an articulated robot with obstacle avoidance in the robot's 2D working environment. An adaptation considering collision avoidance with static obstacles in the 3D real-word environment is introduced in [10]. A further TSP modification is presented in [11] for the case of a two-robot work cell. The robots are each modeled as first-degree B-spline curves to plan collision-free motions. Kinematic and dynamic constraints, as well as collision avoidance with the environment, are not considered. Another idea involving two robots is presented in [12]. Therein, timed Petri nets and the uniform cell decomposition approach are used to model the robot tasks and ensure collision-free operation. In all cases, the optimization is performed offline using genetic algorithms (GA). The GA approach is extended in [13] to account for the relative replacement of the robot base. A TSP approach for task sequencing considering both robot kinematics and dynamics is proposed in [14] for a fruit picking scenario. In [15], an asymmetric TSP algorithm for minimum-time motion planning for point-to-point tasks is introduced, using predetermined travel times and considering the cost dependency of the motion direction due to gravity and kinematics without accounting for multiple IK solutions.

Task scheduling and trajectory or path planning are often closely coupled and are considered in some applications as a combined planning framework. In [16], a batch scheduling approach for pick-and-place task scheduling and motion coordination in combination with velocity tuning for a dual-arm robot is presented. Alternatively, an incremental scheduling method is considered for online coordination based on coordination diagrams. A simultaneous task allocation and motion scheduling for a dual-arm robot is also considered in [17] by introducing an offline constraint optimization problem. An offline approach for multi-robotic task sequencing and path planning is presented in [18]. The problem is modeled as a multiple generalized TSP and solved based on a modified

GA algorithm. The applications found in the literature are mainly concerned with the offline optimization for repetitive tasks in which the cycle time is to be minimized. The reported computation times are still not suitable for an online application with dynamically moving target points.

1.2 Motion planning

Collision-free motion planning of robotic manipulators in dynamically changing environments requires algorithms capable of planning and updating robot trajectories in real time. This research area has been widely studied in the last decades, presenting different solutions and algorithms, mainly categorized as global and local planning methods. Path and trajectory planning can also be considered separately by defining a geometric path and then assigning a time law to it. The following gives a brief overview of related work, focusing more on optimization-based approaches as more related to this work.

1.2.1 Sampling-base approaches

Sampling-based approaches such as rapidly-exploring random tree (RRT) [19] and probabilistic roadmap (PRM) [20] are, especially in global motion planning, widely used for path planning of robotic manipulators. They involve a time-consuming preprocessing stage of sampling the collision-free configuration space and generating a graph-based representation like a roadmap or a tree data structure. Sampling-based approaches are well-suited for high-dimensional configuration spaces and can be successfully used for path planning in multi-robot systems [21–23]. In [21], a PRM-based method consisting of the composition of elementary roadmaps is presented and validated on simulations using multi-robot arm systems in constrained environments. An adaptation of the RRT algorithm for the discrete case of a graph (dRRT) is presented in [22], enabling rapid exploration of high-dimensional configuration space for multi-robot motion planning. To provide path-quality guarantees by identifying conditions for convergence to optimal paths, an asymptotically-optimal extension of the dRRT approach is presented in [23], denoted as dRRT*. Since these algorithms are computationally demanding, the paths are computed offline followed by an open-loop execution, making them more suitable for application in static working environments. To react on immediate environment changes path deformation approaches like the elastic band method in [24] can be used. Prioritized planning methods, like coordination along fixed independent paths and coordination along independent roadmaps, as discussed in [25], can also be applied to avoid collisions in a multi-robot system. A fixed-path coordination for two dual-arm robots in the presence of assembly precedence constraints is discussed in [26]. Further coordination methods are presented in [27] for the case when the robots' paths are given and in [28] for the case of offline computed robot trajectories, i. e., given paths along with velocity profiles.

Sampling-based approaches have been subject to various adaptations resulting in algorithms capable of dealing with dynamically changing environments where continuous replanning is needed. An asymptotically optimal and single-query algorithm for quick replanning based on RRTs is presented in [29] ($\mathrm{RRT}^{\mathrm{X}}$). For motion planning of robotic arms in dynamic environments, an algorithm based on RRTs in configuration-time-space (CT-RRT) is discussed in [30], where the path of the other arm is considered as a dynamic obstacle. Continuous sampling-based replanning in [31] is achieved by parallelly executing multiple RRTs as the robot executes the first action of the best motion plan from the previous planning period. In the context of probabilistic roadmap approaches, dynamic roadmaps (DRM) are introduced in [32] for real-time path planning in changing environments. Motion planning based on the composition of two separate DRMs for a dual-arm robot is presented in [33], incorporating prioritized and coordinated planning along fixed paths or graphs for collision avoidance. To follow the generated path, a model predictive control (MPC) algorithm is proposed in [34], where an additional constraint is optionally considered to avoid collisions locally in case the robot deviates from the path, or the planner fails to find a valid path while replanning. MPC-based path-following approaches are also presented in [35] and [36].

1.2.2 Optimization-based approaches

Local planning algorithms solve an optimization-based motion planning problem in each sampling step and continuously update the robot motion plan without requiring prior information on the working environment. One of the first and widely used local planning approaches for collision-free trajectory generation are artificial potential field methods proposed in [37]. These methods are mainly used for mobile robots [38] or single manipulation arms [39] and consist of generating a potential field with repulsive terms in the vicinity of obstacles and attracting terms at the target point. The generated artificial forces, resulting from the gradient of the potential field, drive the robot away from the obstacles toward the goal. In general, it is challenging to create a potential field with a single minimum at the target point so that the planner can get stuck in local minima and the robot fails to reach its target.

As an alternative to the artificial potential field approach, a local method for obstacle avoidance based on the introduction of virtual velocity dampers and the existence of separating hyperplanes is presented in [40] and described in more detail in [41] and [42]. The algorithm can also be applied to multi-robot systems by representing each manipulator link by a hierarchical description with convex volumes (primitives) to efficiently update the environment model as the robots move. This results in an optimization problem, where collision avoidance is translated into geometric constraints in the robot configuration space. Before the constrained optimization problem is solved, a list of primitive pairs lying at a distance less than a threshold is computed, which need to be separated. However, the computation times achieved when applying this method are unsuitable for online applications. The velocity damper approach is extended in [43] to replace the local planner with a boundary following method and avoid local minima. The algorithm is evaluated using simulations with robot systems consisting of two and three robots, having five DoF each. Although it is stated to be real-time capable, no computation times are given, and no implementation in an experimental setup is realized. A further extension is proposed in [44] for the use of non-strictly convex objects as geometric models.

Another optimization-based algorithm for online planning of collision-free robot trajectories involving two robot manipulators is presented in [45]. The parts of the robots are modeled by spherical shells, generating a geometric representation of the manipulators and their surroundings by a list of geometric primitives. Given the geometric model, a distance check is performed and a set of linear inequality constraints on the joint velocities is computed. Only the joint velocities are used as optimization variables, resulting in a convex optimization problem with a quadratic cost function minimizing the error between the actual and the desired end-effector velocity. This algorithm falls in the category of prioritized planning [46] since the first robot follows a preprogrammed trajectory, and the second one has to reach a target while accounting for collision avoidance with the other robot. A multi-robot trajectory optimization approach using the alternating direction method of multipliers (ADMM) is proposed in [47]. This approach requires the existence of a strictly feasible initial trajectory, and the presented computation times are, for a setup with two robotic arms, not applicable for online applications. A local optimization approach for multi-arm payload manipulation is presented in [48]. However, the paper focuses on the teleoperation-based simultaneous guidance of multiple collaborative arms without considering any collision avoidance constraints.

CHOMP [49] and TrajOpt [50] represent two optimization-based algorithms for motion planning which can cover a wide range of robotic applications, mainly in static environments. The performance of the planners highly depends on the provided trajectory initialization. In some cases, an optimization problem needs to be solved to obtain an initialization trajectory, or multiple trajectory initialization is required to find a feasible solution. Both algorithms formulate trajectory planning as an unconstrained optimization problem penalizing the smoothness of the path and the proximity to obstacles. Implementing collision avoidance as penalties in the cost function has the drawback that the planner can converge to local minima of the cost function, which do not correspond to collision-free robot motions. These algorithms are mainly used in simulations since they rely on a geometry representation of the working environment. This poses a challenge when transferring the motion planning from simulation to reality and involves a preprocessing stage. Depending on the application, additional sensors like cameras and laser scanners may be required to generate the geometry representation using voxel grids (CHOMP) or meshes (TrajOpt), or additional software tools like OpenRAVE have to be considered. In addition, CHOMP and TrajOpt are not complete planners, and postprocessing of the generated paths is required when using ROS (Robot Operating System) to send control commands to a real robot. For the considered robot arms, this involves the motion planning framework MoveIt adding additional processing time in the control loop. This, combined with the lack of robustness in ensuring collision-free motion planning, makes them less suitable for real-world applications with dynamic target points where continuous replanning is required.

MPC algorithms are increasingly used in the field of robot manipulators not only for following a given path, but also for point-to-point trajectory generation. A predictive control algorithm as a fixed-time optimization problem is proposed in [51], in which the deviation of states and control inputs are penalized. Some approaches regarding time-optimal MPC concerning stabilizing properties, real-time application and achieving minimum number of control interventions can be found in [52] and [53]. In [54], an MPC algorithm for reference tracking combined with a sliding mode controller for uncertainties compensation is introduced. An MPC-based framework for semiautonomous teleoperation of a robot manipulator, including collision avoidance with the environment, is presented in [55]. Collision avoidance with dynamic obstacles is considered in [56] in the context of a human-robot collaboration, where the robot parts and the dynamic obstacles are modeled using convex objects. Further MPC-based approaches considering single robot manipulators are presented in [57], [58] and [59]. Considering two robotic manipulators, a hierarchical approach for MPC-based time-optimal planning is described in [60]. For collision avoidance, a standard approach is considered by modeling the robot's shape and obstacles using a composition of spheres and swept sphere lines. Collision avoidance is ensured by imposing minimum distance constraints. The algorithm is validated on a simple experimental setup using two planar robot arms with two DoF each. Recent publications on MPC and dual-arm manipulation focus on cooperative object transportation without considering collision avoidance between the robot arms, see, e. g., [61], [62].

The listed planning algorithms encounter limitations when implementing motion planning on real applications with robots sharing a common workspace with dynamically moving target points. ADMM, for instance, requires a strictly feasible (collision-free) initial trajectory which is difficult to provide. The PRM-based approach [33] requires offline preprocessing and the outcome is a prioritized path planning which requires an additional path-following algorithm to finally generate the control inputs for the robot arms. Experimental results presented in [45] using an interior-point optimizer also result in prioritized planning with one robot following a preprogrammed trajectory. CHOMP and TrajOpt require pre- and postprocessing for an adequate environment representation and compatibility with the ROS interface and do not guarantee collision-free motion.

1.3 Path-following control

Path-following control is often encountered in various fields involving autonomous systems, such as mobile robots and robotic arms, unmanned aircraft systems (UAS), unmanned ground vehicles (UGV), and, more recently, autonomous driving cars. In path-following applications the controlled system is guided to a predefined reference path, and the system follows it as accurately as possible while accounting for relevant kinematic and dynamic limitations. The geometric reference path is often, especially in mobile robotics for use in indoor environments with static obstacles, generated by applying sampling-based approaches such as the already mentioned RRT [19] and PRM [20] methods.

Optimal control methods based on MPC are widely applied in different applications for the design of path-following controllers, see, e. g., [63–65]. In [63] and [65], the path-following problem is formulated in the Frenet-Serret frame considering an omnidirectional mobile robot, respectively, an autonomous driving car. An MPC-based controller for trajectory tracking and path following of underactuated two- and three-dimensional moving vehicles is addressed in [64]. Focusing on path tracking for automated road vehicles, [66] provides a comprehensive overview and classification of different MPC formulations published in recent years. In [67], a path-following predictive controller for nonlinear constrained systems is presented, and sufficient stability conditions are provided. The approach is evaluated on simulations using a simplified kinematic vehicle model. Further predictive-based path-following approaches are introduced in [68] and [69]. While other presented approaches mainly use nominal prediction models, in [69], a disturbance observer-based MPC scheme for nonholonomic vehicles with coupled input constraints and disturbances is considered. Recently, an MPC approach implemented using deep neural network (DNN) is introduced in [70] for camera-based lane-following control of an autonomous vehicle. In addition to MPC, further path-following control methods like backstepping [71] and feedback linearization [72] can also be used.

Towards minimum-time path-following strategies, a model predictive contouring control (MPCC) approach is presented in [73] as an extension to [67], aiming to minimize the distance to a given reference path while maximizing the progress along it. This method is reformulated in [74] in the form of a local MPCC algorithm, suitable for real-time collision-free navigation of a mobile robot in indoor environments. However, the focus lies on following a given reference path and velocity while accounting for collision avoidance with static and dynamic objects. A predictive contouring control method for time-optimal quadrotor flight is introduced in [75], aiming to find a trade-off between path progress maximization and minimizing path error.

1.4 Goal and overview of this work

The main purpose of this work is to develop optimization-based approaches for task scheduling and collision-free trajectory planning of robot manipulators relevant to specific use cases motivated by existing industrial applications. With the goal of increasing the efficiency of robotic systems involving single or multiple robot manipulators, time criteria are considered as performance indicators for algorithm development. By deploying multiple robots in the presence of various tasks from different classes, the idea is to compute an optimal distribution of the tasks among the robots by satisfying both task and robot-relevant constraints. For fast and coordinated task execution, predictive control methods are introduced, generating minimum-time robot trajectories online. A similar motion planning strategy is introduced for a maximum speed path-following control of an omnidirectional mobile robot, which can be equipped with two robotic arms forming a flexible robotic workplace for indoor applications.

Time-optimal planning and control strategies in the sense of minimum-time robot task and trajectory planning are known to lead to aggressive system behavior, enhancing the need for fast and reliable control algorithms capable for online applications in dynamic environments. Therefore, during the development, implementation and validation of the algorithms, attention is also paid to the computation time to ensure that, especially the algorithms for motion planning, are sufficiently fast to allow continuous replanning at runtime. In addition, the idea is to develop novel algorithms that address the two essential problems in robotic automation, task sequencing and trajectory planning, and are easy to implement and use in existing industrial applications.

Compared to existing sampling-based path planning approaches, the proposed predictive methods do not require a time-consuming prepossessing stage and generate robot trajectories directly in the configuration space, which can be forwarded to the local robot controller using ROS without involving additional path-following or motion generation controllers. Moreover, as opposed to the mentioned optimization-based approaches, the algorithms introduced in this work include relevant collision avoidance restrictions as state-dependent constraints in the optimization problem, ensuring that computed feasible solutions correspond to collision-free trajectories.

In order to lay the foundation for the algorithm development, Chapter 2 summarizes the basics for the mathematical modeling of the considered robots. Using a systematic approach, Section 2.1 presents the kinematics of the serial robot manipulator, including forward and inverse kinematics in Section 2.1.1, respectively Section 2.1.2, followed by the differential kinematics in Section 2.2. Section 2.3 gives an insight into the dynamics of the robot manipulators and the analytic expression of the motion equations in closed form, applying Lagrange formalism in Section 2.3.1 and Newton-Euler methods in Section 2.3.2. The kinematics and dynamics of the considered omnidirectional mobile robot are described in Section 2.4 and Section 2.5, respectively.

For high-fidelity robot simulation models, the dynamic parameters of the robot manipulators are identified in Chapter 3. Therefore, the system model is expressed linearly in a set of base inertial parameters estimated utilizing optimized trajectories. For the optimization-based generation of persistent excitation trajectory suitable for parameter identification, a memetic algorithm is introduced, presented in [76], which represents a metaheuristic combination of genetic and gradient-based algorithms.

Considering a single robot manipulator, Chapter 4 presents algorithms for task scheduling and predictive-based trajectory planning by introducing two control structures, see, [77], [78]. The hierarchic controller in Section 4.2 consists of two independent optimization problems, an integer bilinear programming (IBLP) problem for computing an optimal task execution sequence and a continuous nonlinear programming (NLP) problem for online trajectory planning. For the task planning, two methods are presented based on modifications of the TSP. Minimum-distance scheduling is performed in the robot's operational space by minimizing the Euclidean distance of the robot end-effector. Minimum-time scheduling is introduced to minimize robot cycle time by transforming the scheduling problem in the robot's configuration space. A time-optimal algorithm is presented for the MPC-based trajectory planning, making the robots reach the assigned goals in the minimum possible time. The hybrid controller described in Section 4.3 combines the two optimization problems into a single algorithm for task and trajectory planning by introducing a mixed-integer predictive control method.

Chapter 5 is dedicated to the cooperative synchronous execution of pick-and-place tasks by two robot manipulators sharing a confined working environment. The robot arrangement, in conjunction with the distributed tasks, leads to challenging robot motion with high collision potential between the robot manipulators. Therefore, the hierarchic control structure is extended to the case of a multi-robot system, introducing safety-related constraints in both task scheduling and trajectory planning layers. By making use of the high structural flexibility of the hierarchic controller, two architectures are presented for the MPC-based trajectory planning layer: a centralized one common to both robots [79], and a distributed MPC scheme where each robot has its local controller [80]. Collision avoidance between robot manipulators is subject to nonlinear constraints applied along the prediction horizon, increasing the dimension and computation time of the trajectory planning problem. Therefore, this work presents a novel approach based on a continuous approximation of the robot's geometry and the introduction of tangent separating planes, resulting in a nonlinear optimization algorithm capable of planning collision-free robot trajectories online, see, [81], [82].

A path-following feedback controller for an omnidirectional mobile robot based on nonlinear model predictive control is introduced in Chapter 6 as presented in [83]. The problem is formulated in the Frenet-Serret frame with the predictive controller aiming to drive the mobile robot approach and follow a parametrized geometric path while maximizing the robot speed, i. e., the covered robot distance. With the goal of finding a tradeoff between path-tracking error and maximum possible speed, an automatic procedure is used to select suitable weighting parameters for the controller.

Chapter 7 presents suitable modifications of the introduced optimization-based algorithms and their implementation on specific industry-related demonstrators. The focus is more on integrating robot manipulators into existing manual processes to increase automation and efficiency. A simplified pipeline of indoor navigation is presented for the mobile robot, using existing methods for localization and global path planning and the proposed predictive controller for maximum speed path-following control.

Finally, some concluding remarks in Chapter 8 finalize the work and give a brief outlook on possible future research topics.

CHAPTER 2

Mathematical Modeling

Mathematical models build the basis for the design and development of advanced techniques to understand, analyze, and control the behavior of physical systems. Considering that robots equipped with end-effectors execute their tasks in the workspace $\mathcal{W}$ (operational space) and the motion control is realized in the robots configuration space $\mathcal{C}$ (joint space), providing mathematical models that map between the robot configuration and operational space is crucial for many robot applications. Depending on whether these models describe the relation between the end-effector position and velocity to the position and velocity of the robot joints or how forces and torques map to accelerations, they represent kinematic or dynamic robot equations. There exist different approaches on how to model and represent the kinematics and dynamics of robotic systems [84–88]. This chapter provides basic information regarding the mathematical modeling of kinematics and dynamics of robot manipulators and mobile robots. In particular, the modeling of serial-linked robot manipulators with rigid links is considered, and the kinematic and dynamic equations of the UR5 Robot from Universal Robots[1] are derived. In the domain of mobile robots, there are various drive systems that result in different kinematic and dynamic representations and constraints, thus impacting the planning and control of robot motions. This chapter derives the kinematic and dynamic equations of a wheeled mobile robot with an omnidirectional drive system.

2.1 Kinematics of robot manipulators

Robotic systems are composed of interconnected rigid bodies. In the case of robot manipulators, the rigid robot parts -links- are connected by joints forming a kinematic chain. In order to perform the kinematic analysis, each link is associated with at least one body-attached coordinate system. The motions of the rigid bodies and the entire kinematic chain can be expressed as a combination of rotation and translation motions in conjunction with transformations between the different body-related coordinate systems. Robot kinematic models determine the relation between the configuration and operational space for kinematic quantities like position, velocity, and acceleration vectors. In this section, based on rigid body motions, the forward and inverse kinematics of a serial robot manipulator are derived. The forward and inverse position kinematics is followed by the differential kinematics, describing the relation between the velocities in the operational space and the joint space by computing the manipulator Jacobian.

---

[1] Universal Robots

2.1.1 Forward kinematics

Robot forward kinematics describes the position and orientation (pose) of robot parts in the operational space as a function of the joint coordinates $\mathbf{q} = [\theta_1, \ldots, \theta_n]^{\mathrm{T}} \in \mathcal{C}$, with n denoting the number of robot joints and $\theta_j, j \in \{1, \ldots, n\}$, the respective joint angles. Mostly, the pose of the robot end-effector attached to the last robot link is of interest when deriving the forward kinematic equations. By introducing a homogeneous coordinate representation for the position vector, rotation, and translation operationals from the coordinate system of robot link j to the coordinate system of link $j-1$ can be combined into a single homogeneous transformation of the form

$$\mathbf{T}_{j-1}^{j} = \begin{bmatrix} \mathbf{R}_{j-1}^{j} & {}_{j-1}\mathbf{p}_j \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.1}$$

Here, $\mathbf{R}_{j-1}^{j} \in SO(3)^2$ represents an orthogonal rotation matrix to transform a vector expressed in the coordinate frame attached to link j to the coordinate system of link $j-1$. ${}_{j-1}\mathbf{p}_j$ denotes the vector from the origin of the frame of link $j-1$ to the origin of the frame attached to link j expressed in the frame of link $j-1$. The vector $\mathbf{0}$ is a row zero vector $\mathbf{0}^{1\times 3}$, and the value 1 a scale factor denoting that the transformed homogeneous coordinates of the position vector correspond to the physical coordinates. Considering an additional robot link, i. e., $j-2$, the transformation from the coordinate system of Link j to the coordinate system of link $j-2$ is given by

$$\mathbf{T}_{j-2}^{j} = \mathbf{T}_{j-2}^{j-1}\,\mathbf{T}_{j-1}^{j}\,. \tag{2.2}$$

The homogeneous transformation $\mathbf{T}_{j-1}^{j}$ can be decomposed to pure translation and rotation transformations in the form

$$\mathbf{T}_{j-1}^{j} = \mathbf{T}_{T\,j-1}^{j}\,\mathbf{T}_{R\,j-1}^{j} = \begin{bmatrix} \mathbf{I} & {}_{j-1}\mathbf{p}_j \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{j-1}^{j} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.3}$$

with the 3×3 identity matrix $\mathbf{I}$. Let $i \in \{x, y, z\}$ denote the local axis of a coordinate system, θ the rotation angle, and d the displacement length, a set of basic homogeneous transformations is then defined by

$$\mathbf{T}_T(i, d) = \begin{bmatrix} \mathbf{I} & d\mathbf{e}_i \\ \mathbf{0} & 1 \end{bmatrix}, \quad \mathbf{T}_R(i, \theta) = \begin{bmatrix} \mathbf{R}_{i,\theta} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.4}$$

Here, $\mathbf{e}_i$ represent unit vectors of the respective coordinate axis and $\mathbf{R}_{i,\theta}$ rotation matrices by the angle θ about the axis $i \in \{x, y, z\}$, i. e.,

$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}, \mathbf{R}_{y,\theta} = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \mathbf{R}_{z,\theta} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.5}$$

The 4×4 homogeneous transformation matrices of the form (2.1) represent a special case of homogeneous coordinates, widely used in the fields of robotics and computer

---

[2] Special Orthogonal group of order 3.

graphics. A set containing all these matrices is generated by the basic homogeneous transformations (2.4) and is denoted by $E(3)$[3] [87].

For a robot manipulator with n rigid links, the homogeneous transformation from the coordinate system $(o_n x_n y_n z_n)$ of the last link n to the inertial frame of reference $(o_0 x_0 y_0 z_0)$ is given by

$$\mathbf{T}_0^n = \prod_{j=1}^{n} \mathbf{T}_{j-1}^{j} = \begin{bmatrix} \mathbf{R}_0^n & {}_0\mathbf{p}_n \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.6}$$

with the rotation matrix $\mathbf{R}_0^n$ and the vector ${}_0\mathbf{p}_n$ from the center of the inertial frame to the center of the frame attached to the last robot link. For a given rotation $\mathbf{R}_n^{\mathrm{e}}$ and displacement ${}_n\mathbf{p}_{\mathrm{e}}$ of the end-effector relative to the coordinate frame $(o_n x_n y_n z_n)$ of the last robot link, the position and orientation of the robot end-effector in the inertial frame is given by

$$\mathbf{T}_0^{\mathrm{e}} = \begin{bmatrix} \mathbf{R}_0^n & {}_0\mathbf{p}_n \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R}_n^{\mathrm{e}} & {}_n\mathbf{p}_{\mathrm{e}} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_0^{\mathrm{e}} & {}_0\mathbf{p}_{\mathrm{e}} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.7}$$

Given that the columns of the rotation matrix $\mathbf{R}_0^{\mathrm{e}} \in \mathbb{R}^{3\times 3}$ are unit vectors and mutually orthogonal to each other, they define six independent constraints by nine matrix elements, implying thus the existence of only three independent variables. Therefore, the orientation of a rigid body, i. e., an arbitrary $SO(3)$ rotation matrix, can be represented using only three independent quantities [87]. In general, a minimal representation of the special orthonormal group $SO(\mathfrak{m})$ requires $\mathfrak{m}(\mathfrak{m}-1)/2$ parameters [89]. Let $\mathbf{R}_{\mathrm{e}}$ denote a minimal representation of the rotation matrix $\mathbf{R}_0^{\mathrm{e}}$ with only three angular parameters $\boldsymbol{\phi}_{\mathrm{e}}(\mathbf{R}_{\mathrm{e}}) \in \mathbb{R}^3$ determining the orientation of the robot end-effector. The forward robot kinematic equation describing the pose of the end-effector relative to the robot base can then be written in the form

$$\mathbf{f}_{\mathrm{kin}}(\mathbf{q}) = \begin{bmatrix} {}_0\mathbf{p}_{\mathrm{e}} \\ \boldsymbol{\phi}_{\mathrm{e}}(\mathbf{R}_{\mathrm{e}}) \end{bmatrix} \tag{2.8}$$

as a function of the joint angles $\mathbf{q} = [\theta_1, \ldots, \theta_n]^{\mathrm{T}}$.

In robotics, the Euler angles $\boldsymbol{\phi} = [\varphi, \vartheta, \psi]^{\mathrm{T}}$ are often used for the minimal representation of a rigid body orientation in space. There exist two minimal parametrizations of rotations using the Euler angles obtained by composing a ZYZ or ZYX (Roll-Pitch-Yaw) sequence of three basic rotations (2.5), i. e.,

$$\mathbf{R}_{ZYZ}(\boldsymbol{\phi}) = \mathbf{R}_{z,\varphi}\mathbf{R}_{y,\vartheta}\mathbf{R}_{z,\psi}\,, \tag{2.9}$$

respectively

$$\mathbf{R}_{ZYX}(\boldsymbol{\phi}) = \mathbf{R}_{z,\varphi}\mathbf{R}_{y,\vartheta}\mathbf{R}_{x,\psi}\,. \tag{2.10}$$

Both minimal representations are subject to representational singularities that have to be taken into account. For a given end-effector rotation matrix $\mathbf{R}_0^{\mathrm{e}}$ and its minimal representation $\mathbf{R}_{\mathrm{e}}$ using Euler angles, e. g., $\mathbf{R}_{\mathrm{e}}(\boldsymbol{\phi}_{\mathrm{e}}) = \mathbf{R}_{ZYZ}(\boldsymbol{\phi}_{\mathrm{e}})$, the set of Euler angles $\boldsymbol{\phi}_{\mathrm{e}}(\mathbf{q})$ corresponding to $\mathbf{R}_0^{\mathrm{e}}(\mathbf{q})$ is determined by solving the inverse problem considering that $\mathbf{R}_0^{\mathrm{e}}(\mathbf{q}) \overset{!}{=} \mathbf{R}_{\mathrm{e}}(\boldsymbol{\phi}_{\mathrm{e}})$. For more information regarding the parametrization of rotations and solving the inverse problem of computing the minimal angles see, e. g., [87] and [89].

To compute and analyze the kinematics of the robot, each link j is associated with a rigidly attached coordinate frame $(o_j x_j y_j z_j)$. Regarding the distribution of the coordinate

---

[3] Euclidean group of order 3.

frames, no further assumptions have been made so far. They can even be assigned to lie outside the physical link itself, provided they are rigidly attached to them. This means that, when the robot joint j is actuated, the link j and the frame attached to it undergo a movement that results in no change in the pose of the coordinate frame $(o_j x_j y_j z_j)$ relative to link j. However, for the choice of coordinate systems, a standard, the Denavit-Hartenberg convection, has been established, which entails further simplifications in the systematic description of the geometry of industrial robots.

2.1.1.1 Denavit-Hartenberg convection

When deriving the homogeneous transformation matrices of the form (2.1), it is to be expected that six parameters are needed, three for the position and three for the orientation to describe the pose of a robot link in the kinematic chain relative to the pose of the preceding link. The Denavit-Hartenberg (DH) convection [90] uses only four parameters to describe the spatial relationship between the coordinate frames of two successive links, by introducing two constraints to the placement of the frames [87], namely:

1. The axis x_j is perpendicular to the axis z_{j-1}.
2. The axis x_j intersects the axis z_{j-1}.

Following the DH convection each homogeneous transformation matrix can be expressed as a function of four parameters, two link (link length a_j and twist angle α_j) and two joint (offset length d_j and joint angle θ_j) parameters, and can be represented as a product of four basic transformations (2.4). However, there exist two, respectively, three popular representations for the kinematics of serial manipulators based on the DH convection:

- Classical DH convention
- Modified DH convention
  - Corke notation
  - Khalil-Kleinfinger notation

Regardless of the notation used, given the correct choice of the DH parameters, all convections lead to the same result, i. e., the same end-effector pose relative to the inertial frame. Nevertheless, each representation has its technical benefits and drawbacks. The differences and commonalities between different DH convection are discussed in [91] and [92]. In the following, we will briefly describe the classical and modified DH formalism for serial-linked robot manipulators with rotary joints.

Classical DH convection

The classical DH representation is the standard convection originally introduced by Danavit and Hartenberg [90], which consists of attaching the link coordinate frames to the far (distal) end of each link. Following this approach the DH parameters are defined as follows:

- α_j is the angle from the z_{j-1} axis to the z_j axis about the x_j axis;
- a_j is the distance from the origin o_j of the (j)-th coordinate frame $(o_j x_j y_j z_j)$ to the intersection of the x_j and z_{j-1} axes along the x_j axis, i. e., the distance from the z_{j-1} axis to the z_j axis along the axis x_j;
- θ_j is the joint angle from the x_{j-1} axis to the x_j axis about the z_{j-1} axis;
- d_j is the offset distance from the origin o_{j-1} of the $(j-1)$-th frame to the intersection of the z_{j-1} axis with the x_j axis along the z_{j-1} axis, i. e., the distance from the x_{j-1} axis to the x_j axis along the axis z_{j-1};

Using the classical DH parameters the homogeneous transformation between link j and $j-1$ is given as a product of four basic transformations, two rotations and two translations, i. e.,

$$\begin{aligned}\mathbf{T}_{j-1}^{j} &= \mathbf{T}_R(z,\theta_j)\mathbf{T}_T(z,d_j)\mathbf{T}_T(x,a_j)\mathbf{T}_R(x,\alpha_j)\\ &= \begin{bmatrix} \cos(\theta_j) & -\sin(\theta_j)\cos(\alpha_j) & \sin(\theta_j)\sin(\alpha_j) & a_j\cos(\theta_j)\\ \sin(\theta_j) & \cos(\theta_j)\cos(\alpha_j) & -\cos(\theta_j)\sin(\alpha_j) & a_j\sin(\theta_j)\\ 0 & \sin(\alpha_j) & \cos(\alpha_j) & d_j\\ 0 & 0 & 0 & 1 \end{bmatrix}.\end{aligned} \tag{2.11}$$

Modified DH convection: Corke notation

The modified DH parameters were first introduced by Craig in [84], where as opposed to the classical convection, the link coordinate frames are attached to the near (proximal) rather than to the far (distal) end of each link. This modified notation is, in some ways, clearer and tidier, and it is widely used and described in many robotics books [92]. By using this approach, the DH parameters are defined as follows:

- α_{j-1} is the angle from the z_{j-1} axis to the z_j axis about the x_{j-1} axis;
- a_{j-1} is the distance from the z_{j-1} axis to the z_j axis along the x_{j-1} axis;
- θ_j is the angle from the x_{j-1} axis to the x_j axis about the z_j axis;
- d_j is the distance from the x_{j-1} axis to the x_j axis along the z_j axis.

The homogeneous transformation matrix between two successive links is, in this case, given by

$$\begin{aligned}\mathbf{T}_{j-1}^{j} &= \mathbf{T}_R(x,\alpha_{j-1})\mathbf{T}_T(x,a_{j-1})\mathbf{T}_T(z,d_j)\mathbf{T}_R(z,\theta_j)\\ &= \begin{bmatrix} \cos(\theta_j) & -\sin(\theta_j) & 0 & a_{j-1}\\ \sin(\theta_j)\cos(\alpha_{j-1}) & \cos(\theta_j)\cos(\alpha_{j-1}) & -\sin(\alpha_{j-1}) & -d_j\sin(\alpha_{j-1})\\ \sin(\theta_j)\sin(\alpha_{j-1}) & \cos(\theta_j)\sin(\alpha_{j-1}) & \cos(\alpha_{j-1}) & d_j\cos(\alpha_{j-1})\\ 0 & 0 & 0 & 1 \end{bmatrix}.\end{aligned} \tag{2.12}$$

Modified DH convection: Khalil-Kleinfinger notation

Khalil and Kleinfinger presented in [93] another modification of the DH convection, which, analogous to the Corke notation, attaches the coordinate frames to the near (proximal) end of each link. The difference to the Corke notation consists in the definition of the two link parameter (a_j, α_j). The following parameters are required to define the (j)-th frame relative to the frame $(j-1)$:

- α_j is the angle from the z_{j-1} axis to the z_j axis about the x_{j-1} axis;
- a_j is the distance from the z_{j-1} axis to the z_j axis along the x_{j-1} axis;
- θ_j is the angle from the x_{j-1} axis to the x_j axis about the z_j axis;
- d_j is the distance from the x_{j-1} axis to the x_j axis along the z_j axis.

The homogeneous transformation matrix between two successive links resulting by applying this modified DH convection is given by

$$\begin{aligned}\mathbf{T}_{j-1}^{j} &= \mathbf{T}_R(x,\alpha_j)\mathbf{T}_T(x,a_j)\mathbf{T}_R(z,\theta_j)\mathbf{T}_T(z,d_j)\\ &= \begin{bmatrix} \cos(\theta_j) & -\sin(\theta_j) & 0 & a_j \\ \sin(\theta_j)\cos(\alpha_j) & \cos(\theta_j)\cos(\alpha_j) & -\sin(\alpha_j) & -d_j\sin(\alpha_j) \\ \sin(\theta_j)\sin(\alpha_j) & \cos(\theta_j)\sin(\alpha_j) & \cos(\alpha_j) & d_j\cos(\alpha_j) \\ 0 & 0 & 0 & 1 \end{bmatrix}.\end{aligned} \tag{2.13}$$

The motivation behind this modification is to develop a method which can be used easily and without ambiguity in tree structured and robots with closed kinematic chain (closed-loop robots). Furthermore, attaching the coordinate frame $(o_j x_j y_j z_j)$ to link j such that axis z_j is along the axis of joint j also simplifies the dynamic model of the robot [88] and is therefore often commonly used in parameter estimation literature and textbooks, see, e. g., [85, 86, 94].

2.1.1.2 Denavit-Hartenberg parameters of the UR5 robot

In this work, the UR5 robot from Universal Robots is used for experiments, which consists of seven links and six rotational joints, the base joint, shoulder joint, elbow joint and three wrist joints. Hence, $n = 6$ holds for the considered robot arm. Link zero is the robot mounting flange and is referred to as the robot base, and link six, i. e., wrist three, is the tool flange.

To derive and analyze the forward and inverse robot kinematics the classical DH convection is used. The resulting distribution of the coordinate frames for each link and the corresponding DH parameters are shown in Figure 2.1 and Table 2.1, respectively. The coordinate frame $(o_0 x_0 y_0 z_0)$, which represent also the inertial frame, is attached to the robot base, and the last frame $(o_6 x_6 y_6 z_6)$ to the last robot link. The last robot link is referred to also as the end-effector, since an end-effector is rigidly attached to the tool flange. The tool offset and orientation can be systematically considered in the homogeneous transformation, see (2.6) and (2.7). Hence, the homogeneous transformation describing the position ${}_0\mathbf{p}_6(\mathbf{q})$ and orientation $\mathbf{R}_0^6(\mathbf{q})$ of the robot's end-effector relative to the base frame reads

$$\mathbf{T}_\mathrm{b}^\mathrm{e}(\mathbf{q}) = \mathbf{T}_0^6(\mathbf{q}) = \prod_{j=1}^{6} \mathbf{T}_{j-1}^{j}(\theta_j, \alpha_j, a_j, d_j) = \begin{bmatrix} \mathbf{R}_0^6(\mathbf{q}) & {}_0\mathbf{p}_6(\mathbf{q}) \\ \mathbf{0} & 1 \end{bmatrix}, \tag{2.14}$$

with the matrix $\mathbf{T}_{j-1}^{j}$ from (2.11), the DH parameters from Table 2.1 and the vector of the joint angles $\mathbf{q}^\mathrm{T} = [\theta_1, \ldots, \theta_6]$.

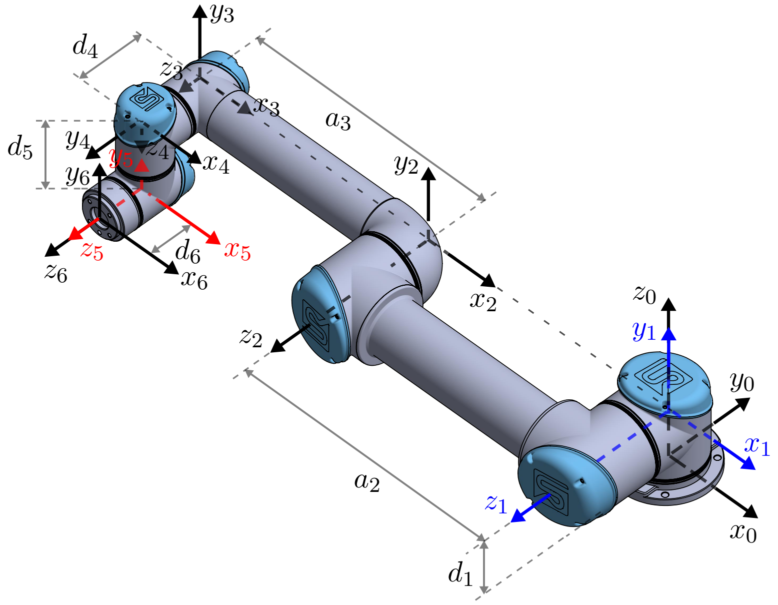


Figure 2.1: UR5 from Universal Robots with coordinate frames and Denavit-Hartenberg parameters according to the classical convection.

Table 2.1: Denavit-Hartenberg parameters for the UR5.

Link j	α_j(rad)	a_j(m)	θ_j	d_j(m)	Link j	α_j(rad)	a_j(m)	θ_j	d_j(m)
1	$\frac{\pi}{2}$	0	θ_1	0.089	4	$\frac{\pi}{2}$	0	θ_4	0.109
2	0	-0.425	θ_2	0	5	$-\frac{\pi}{2}$	0	θ_5	0.094
3	0	-0.392	θ_3	0	6	0	0	θ_6	0.082

As already mentioned, the choice of the coordinate systems and the corresponding DH parameters according to the Khalil-Kleinfinger notation leads to simplifications in the dynamic robot model, which is particularly advantageous for parameter identification. Therefore, when deriving the robot model for parameter identification the Khalil-Kleinfinger representation of the robot geometry is used, leading to the coordinate frame assignment as depicted in Figure 2.2 with the modified DH parameters shown in Table 2.2. Due to the choice of the coordinate frames where the first frame $(o_0x_0y_0z_0)$ is not attached to robot base and the last one not directly to the robots last link, two additional translations are needed represent the pose of the end-effector relative to the robot base, i. e.,

$$\mathbf{T}_{\mathrm{b}}^{\mathrm{e}}(\mathbf{q}) = \mathbf{T}_T(z, d_1)\mathbf{T}_0^6(\mathbf{q})\mathbf{T}_T(z, d_6)\,. \tag{2.15}$$

Here, d_1 and d_6 correspond to the offset distances from Table 2.1, as they do not occur as parameters in the modified DH convection. The matrix

$$\mathbf{T}_0^6(\mathbf{q}) = \prod_{j=1}^{6} \mathbf{T}_{j-1}^{j}(\theta_j, \alpha_j, a_j, d_j)\,, \tag{2.16}$$

with the homogeneous transformation matrix $\mathbf{T}_{j-1}^{j}$(2.13) and the DH parameters from Table 2.2 represent the pose of the last coordinate frame relative to the first one.

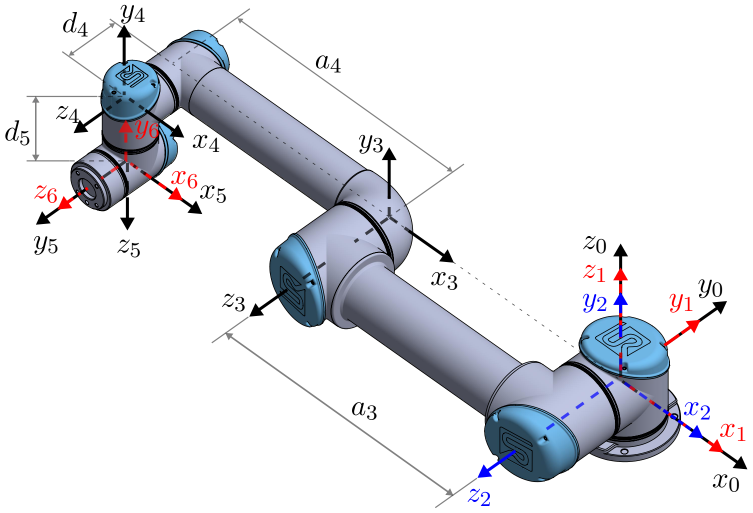


Figure 2.2: UR5 from Universal Robots with coordinate frames and Denavit-Hartenberg parameters based on the modified DH representation following the Khalil-Kleinfinger notation.

Table 2.2: Modified (Khalil-Kleinfinger) Denavit-Hartenberg parameters for the UR5.

Link j	α_j(rad)	a_j(m)	θ_j	d_j(m)	Link j	α_j(rad)	a_j(m)	θ_j	d_j(m)
1	0	0	θ_1	0	4	0	-0.392	θ_4	0.109
2	$\frac{\pi}{2}$	0	θ_2	0	5	$\frac{\pi}{2}$	0	θ_5	0.094
3	0	-0.425	θ_3	0	6	$-\frac{\pi}{2}$	0	θ_6	0

2.1.2 Inverse kinematics

In comparison to forward robot kinematics, which deals with determining the pose of the end-effector for given joint coordinates $\mathbf{q}$, inverse kinematics (IK) is concerned with the inverse problem of finding the joint variables as a function of the end-effector position and orientation. The inverse kinematics problem is generally more complex than forward kinematics because it involves solving nonlinear equations where multiple solutions might exist, and sometimes even no solution exists for some desired poses. The existence of multiple IK solutions implies that different joint angle configurations lead to the same end-effector pose. Two approaches can be used to determine the inverse kinematics: a closed-loop analytic solution based on geometric and algebraic equations or an iterative numerical solution. The method preference usually depends on the robot and the application since there is no general approach to derive the inverse kinematics of serial-linked robot manipulators. The analytical solution of the considered UR5 robot is discussed in many technical reports and paper, see, e. g., [95–98]. Based on the approach presented in [95], the following describes how the inverse kinematics is calculated analytically in a more systematic way.

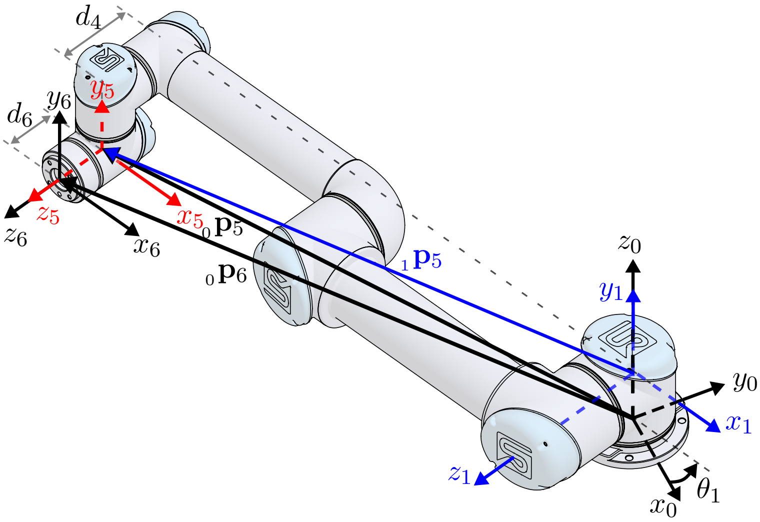


Figure 2.3: Schematic illustration for computing the joint angle θ_1.

For a given position ${}_0\mathbf{p}_6 = [{}_0p_{6x}, {}_0p_{6y}, {}_0p_{6z}]^\mathrm{T}$ and orientation of the robot end-effector relative to the base frame, the homogeneous transformation matrix can be written as

$$\mathbf{T}_0^6 = \begin{bmatrix} \mathbf{R}_0^6 & {}_0\mathbf{p}_6 \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} {}_0\hat{x}_{6x} & {}_0\hat{y}_{6x} & {}_0\hat{z}_{6x} & {}_0p_{6x} \\ {}_0\hat{x}_{6y} & {}_0\hat{y}_{6y} & {}_0\hat{z}_{6y} & {}_0p_{6y} \\ {}_0\hat{x}_{6z} & {}_0\hat{y}_{6z} & {}_0\hat{z}_{6z} & {}_0p_{6z} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.17}$$

where the columns $[{}_0i_{6x}, {}_0i_{6y}, {}_0i_{6z}]^\mathrm{T}, i \in \{\hat{x}, \hat{y}, \hat{z}\}$ represent unit vectors defining the axis of end-effector frame relative to the robot base expressed in the coordinate frame of the latter one. To find the joint angle θ_1, firstly the homogeneous transformation matrix from the coordinate frame $(o_5x_5y_5z_5)$ to the frame $(o_1x_1y_1z_1)$ as function of the end-effector pose is computed in the form

$$\mathbf{T}_1^5 = \mathbf{T}_1^0(\theta_1)\mathbf{T}_0^6\mathbf{T}_6^5(\theta_6)\,, \tag{2.18}$$

with $\mathbf{T}_1^0 = \left(\mathbf{T}_0^1\right)^{-1}$ and $\mathbf{T}_6^5 = \left(\mathbf{T}_5^6\right)^{-1}$. For the position vector ${}_1\mathbf{p}_5$ from the origin o_1 to the origin o_5 expressed in the coordinate frame $(o_1 x_1 y_1 z_1)$, $\mathbf{e}_z^{\mathrm{T}}\,{}_1\mathbf{p}_5 = d_4$ holds, see Figure 2.3, yielding

$$\mathbf{e}_z^{\mathrm{T}}\,{}_1\mathbf{p}_5 = \mathbf{T}_1^5[3,4] = \left({}_0\hat{z}_{6y} d_6 - {}_0p_{6y}\right)\cos(\theta_1) - \left({}_0\hat{z}_{6x} d_6 - {}_0p_{6x}\right)\sin(\theta_1) \overset{!}{=} d_4\,, \tag{2.19}$$

where $\mathbf{T}[3,4]$ denotes the matrix element in the 3-rd row and 4-th column. Equation (2.19) can be written in a simplified form by computing the position vector of the origin o_5 relative to the robot base frame, i. e.,

$$ {}_0\mathbf{p}_5 = {}_0\mathbf{p}_6 + \mathbf{R}_0^6 \begin{bmatrix} 0 \\ 0 \\ -d_6 \end{bmatrix} = \begin{bmatrix} {}_0p_{6x} - {}_0\hat{z}_{6x} d_6 \\ {}_0p_{6y} - {}_0\hat{z}_{6y} d_6 \\ {}_0p_{6z} - {}_0\hat{z}_{6z} d_6 \end{bmatrix} = \begin{bmatrix} {}_0p_{5x} \\ {}_0p_{5y} \\ {}_0p_{5z} \end{bmatrix} \tag{2.20}$$

and substituting the first two components of (2.20) into (2.19). The resulting equation

$$-{}_0p_{5y}\cos(\theta_1) + {}_0p_{5x}\sin(\theta_1) = d_4 \tag{2.21}$$

can then be written as

$$-\sqrt{{}_0p_{5x}^2 + {}_0p_{5y}^2}\left(\sin(\gamma_1)\cos(\theta_1) - \cos(\gamma_1)\sin(\theta_1)\right) = d_4\,, \tag{2.22}$$

with $\gamma_1 = \arctan\left({}_0p_{5y}/{}_0p_{5x}\right)$. Finally, the two solutions for θ_1 corresponding to the robot shoulder configurations "*left*" and "*right*" are given by

$$\theta_1 = \arctan\left(\frac{{}_0p_{5y}}{{}_0p_{5x}}\right) \pm \arccos\left(\frac{d_4}{\sqrt{{}_0p_{5x}^2 + {}_0p_{5y}^2}}\right) + \frac{\pi}{2}\,. \tag{2.23}$$

The angle θ_1 is always defined if an inverse solution exists since there is no possible robot configuration which results in $\sqrt{{}_0p_{5x}^2 + {}_0p_{5y}^2} \le |d_4|$, see Figure 2.3.

Next, the joint angle θ_5 is computed using the projection of the position vector ${}_1\mathbf{p}_6$ on the z_1-axis of the coordinate frame $(o_1 x_1 y_1 z_1)$, i. e.,

$$\mathbf{e}_z^{\mathrm{T}}\,{}_1\mathbf{p}_6 = \mathbf{T}_1^6[3,4] = {}_0p_{6x}\sin(\theta_1) - {}_0p_{6y}\cos(\theta_1) \overset{!}{=} d_4 + d_6\cos(\theta_5)\,, \tag{2.24}$$

which depends on θ_5 and the previously computed angle θ_1, see Figure 2.4. Here, $\mathbf{T}_1^6 = \mathbf{T}_1^0\mathbf{T}_0^6$ represents the transformation of the given end-effector pose $\mathbf{T}_0^6$ (2.17) to the coordinate frame of link one by applying the inverse of the homogeneous transformation $\mathbf{T}_0^1$ from (2.11). Note that equation (2.24) can also be derived in a systematic way by comparing the given pose of the end-effector relative to the coordinate frame $(o_1 x_1 y_1 z_1)$ with the computed end-effector pose by successively applying the homogeneous transformation (2.11) from link six up to link one, i. e.,

$$\mathbf{T}_1^0(\theta_1)\mathbf{T}_0^6 \overset{!}{=} \prod_{j=2}^{6} \mathbf{T}_{j-1}^j(\theta_j, \alpha_j, a_j, d_j)\,. \tag{2.25}$$

Using (2.24), the two solutions for the angle θ_5 corresponding to the robot wrist being "*up*" or "*down*", respectively, are given by

$$\theta_5 = \pm\arccos\left(\frac{{}_0p_{6x}\sin(\theta_1) - {}_0p_{6y}\cos(\theta_1) - d_4}{d_6}\right). \tag{2.26}$$

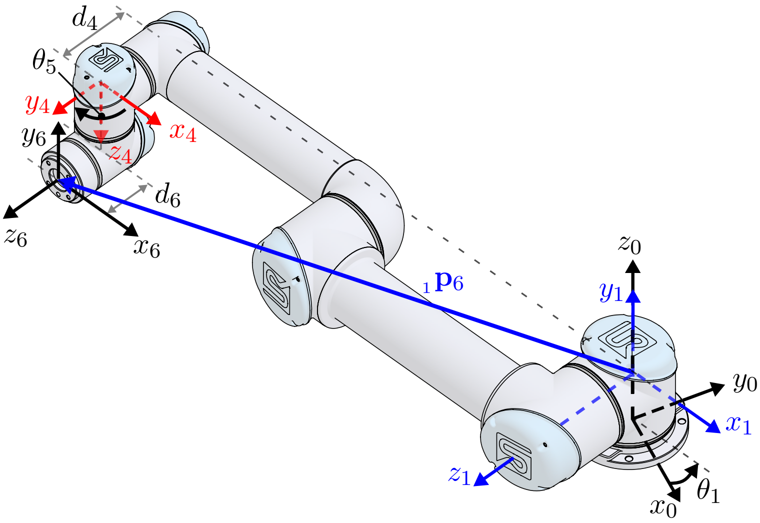


Figure 2.4: Schematic illustration for computing the joint angle θ_5.

This solution is defined as long as the argument of arccos $(\cdot)$ has a magnitude not greater than one, or $|{}_0p_{6x}\sin(\theta_1) - {}_0p_{6y}\cos(\theta_1) - d_4| \leq |d_6|$.

The joint angle θ_6 is determined by analyzing the z_1 axis of the coordinate frame $(o_1x_1y_1z_1)$, which, as can be seen in Figure 2.5, is always parallel to the z_2 and z_3 axes of the respective coordinate frames. Therefore, seen from the coordinate frame of the end-effector the orientation $[{}_6\hat{z}_{1x}, {}_6\hat{z}_{1y}, {}_6\hat{z}_{1z}]^{\mathrm{T}}$ of the z_1 axis depends only on the angles θ_5 and θ_6

$$\mathbf{R}_6^1\,\mathbf{e}_z = \begin{bmatrix} {}_6\hat{z}_{1x} \\ {}_6\hat{z}_{1y} \\ {}_6\hat{z}_{1z} \end{bmatrix} = \begin{bmatrix} \cos(\theta_6)\sin(\theta_5) \\ -\sin(\theta_6)\sin(\theta_5) \\ \cos(\theta_5) \end{bmatrix} . \tag{2.27}$$

Here, $\mathbf{R}_6^1$ denotes the rotation matrix from the coordinate frame of link one to the frame of link six and $\mathbf{e}_z = [0, 0, 1]^{\mathrm{T}}$ is the unit vector. Next, we compute the orientation of the z_1 axis as function of the end-effector pose by computing first the orientation relative to the base frame, i. e.,

$$\mathbf{R}_0^1\,\mathbf{e}_z = \begin{bmatrix} {}_0\hat{z}_{1x} \\ {}_0\hat{z}_{1y} \\ {}_0\hat{z}_{1z} \end{bmatrix} = \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} . \tag{2.28}$$

and transforming it back to the end-effector frame by multiplying with the inverse $\mathbf{R}_6^0 = (\mathbf{R}_0^6)^{-1}$ of the rotation matrix from (2.17), yielding

$$\begin{bmatrix} {}_6\hat{z}_{1x} \\ {}_6\hat{z}_{1y} \\ {}_6\hat{z}_{1z} \end{bmatrix} = \mathbf{R}_6^0 \begin{bmatrix} {}_0\hat{z}_{1x} \\ {}_0\hat{z}_{1y} \\ {}_0\hat{z}_{1z} \end{bmatrix} = \begin{bmatrix} {}_6\hat{x}_{0x} & {}_6\hat{y}_{0x} & {}_6\hat{z}_{0x} \\ {}_6\hat{x}_{0y} & {}_6\hat{y}_{0y} & {}_6\hat{z}_{0y} \\ {}_6\hat{x}_{0z} & {}_6\hat{y}_{0z} & {}_6\hat{z}_{0z} \end{bmatrix} \begin{bmatrix} \sin(\theta_1) \\ -\cos(\theta_1) \\ 0 \end{bmatrix} = \begin{bmatrix} {}_6\hat{x}_{0x}\sin(\theta_1) - {}_6\hat{y}_{0x}\cos(\theta_1) \\ {}_6\hat{x}_{0y}\sin(\theta_1) - {}_6\hat{y}_{0y}\cos(\theta_1) \\ {}_6\hat{x}_{0z}\sin(\theta_1) - {}_6\hat{y}_{0z}\cos(\theta_1) \end{bmatrix} . \tag{2.29}$$

Equating the first two components of (2.27) and (2.29) leads

$$\begin{aligned} \cos(\theta_6) &= \frac{{}_6\hat{x}_{0x}\sin(\theta_1) - {}_6\hat{y}_{0x}\cos(\theta_1)}{\sin(\theta_5)} , \\ \sin(\theta_6) &= \frac{-{}_6\hat{x}_{0y}\sin(\theta_1) + {}_6\hat{y}_{0y}\cos(\theta_1)}{\sin(\theta_5)} , \end{aligned} \tag{2.30}$$

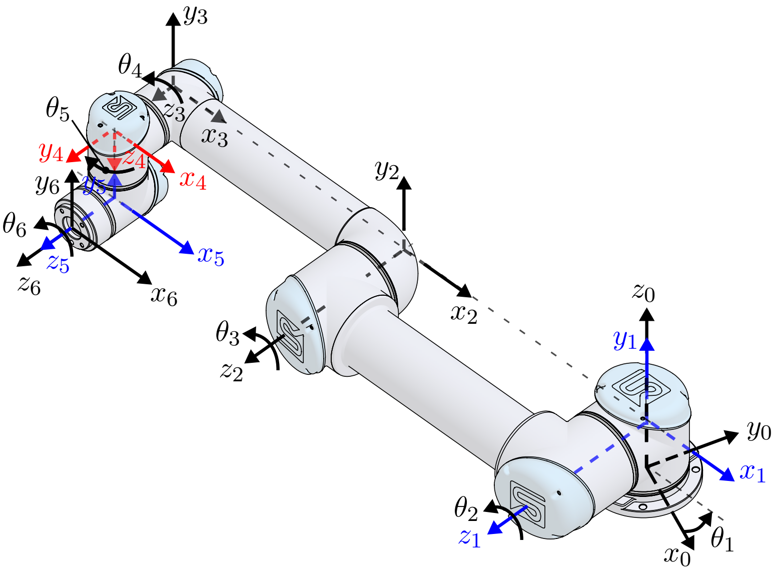


Figure 2.5: Schematic illustration for computing the joint angle θ_6.

and finally, to the solution of the joint angle θ_6

$$\theta_6 = \arctan\left(\frac{-{}_6\hat{x}_{0y}\sin(\theta_1) + {}_6\hat{y}_{0y}\cos(\theta_1)}{{}_6\hat{x}_{0x}\sin(\theta_1) - {}_6\hat{y}_{0x}\cos(\theta_1)}\right). \tag{2.31}$$

Note that for $\sin(\theta_5) = 0$, which also results in both numerators of (2.30) being zero, the solution is undetermined. This corresponds to the robot configuration where the axis z_6 and z_5 are aligned collinearly with axes z_1, z_2, and z_3, as shown in Figure 2.5. In this case θ_6 can be set to an arbitrary value between $[-2\pi, 2\pi]$.

With the determined joint angles θ_1, θ_5 and θ_6 the transformation matrices $\mathbf{T}_0^1$, $\mathbf{T}_5^6$ and $\mathbf{T}_4^5$ are known, allowing us to define the pose of the coordinate frame $(o_4x_4y_4z_4)$ relative to the frame $(o_1x_1y_1z_1)$ of link one as a function of the given end-effector pose $\mathbf{T}_0^6$ in the form

$$\mathbf{T}_1^4 = \mathbf{T}_1^0(\theta_1)\mathbf{T}_0^6\mathbf{T}_6^4(\theta_5, \theta_6)\,, \tag{2.32}$$

where $\mathbf{T}_1^0 = \left(\mathbf{T}_0^1\right)^{-1}$ and $\mathbf{T}_6^4 = \left(\mathbf{T}_4^5\mathbf{T}_5^6\right)^{-1}$. The coordinates of the position vector ${}_1\mathbf{p}_4$ from the origin o_1 to the origin o_4 expressed in the frame of the first link are then given by ${}_1p_{4x} = \mathbf{T}_1^4[1,4], {}_1p_{4y} = \mathbf{T}_1^4[2,4]$ and ${}_1p_{4z} = \mathbf{T}_1^4[2,4]$, where $\mathbf{T}_1^4[i,j]$ denotes the matrix element in i-th row and j-th column. As can be seen in Figure 2.6, the components ${}_1p_{4x}$ and ${}_1p_{4y}$ of the vector ${}_1\mathbf{p}_4$ form with the DH parameters a_2 and a_3 a triangle on the x_1y_1-plane. By applying the law of cosines, the determination of θ_3 is then reduced to a planar problem given by

$${}_1p_{4x}^2 + {}_1p_{4y}^2 = a_2^2 + a_3^2 - 2a_2a_3\cos(\pi - \theta_3)\,. \tag{2.33}$$

The right-hand side of equation (2.33) can also be computed by successively applying the homogeneous transformation (2.11) from link four up to link one since

$$\mathbf{T}_1^4 = \mathbf{T}_1^0(\theta_1)\mathbf{T}_0^6\mathbf{T}_6^4(\theta_5, \theta_6) \overset{!}{=} \prod_{j=2}^{4} \mathbf{T}_{j-1}^{j}(\theta_j, \alpha_j, a_j, d_j) \tag{2.34}$$

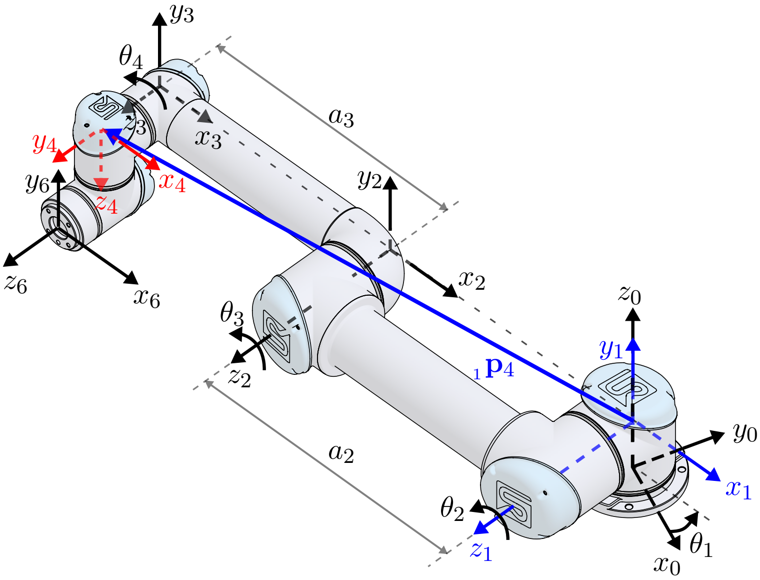


Figure 2.6: Schematic illustration for computing the joint angle θ_3.

holds true. From (2.33) the two solutions of joint angle θ_3 corresponding to the robot configurations "*elbow up*" and "*elbow down*" are given by

$$\theta_3 = \pm \arccos\left(\frac{{}_1p_{4x}^2 + {}_1p_{4y}^2 - a_2^2 - a_3^2}{2a_2a_3}\right) . \tag{2.35}$$

The solutions for the joint angle θ_3 are defined for $\left|{}_1p_{4x}^2 + {}_1p_{4y}^2 - a_2^2 - a_3^2\right| \leq |2a_2a_3|$.

To determine the joint angle θ_2 the pose $\mathbf{T}_2^4$ of the coordinate frame $(o_4x_4y_4z_4)$ is computed relative to the frame $(o_2x_2y_2z_2)$ of the second link in the form

$$\mathbf{T}_2^1(\theta_2)\mathbf{T}_1^4 \overset{!}{=} \prod_{j=3}^{4} \mathbf{T}_{j-1}^{j}(\theta_j, \alpha_j, a_j, d_j) , \tag{2.36}$$

with $\mathbf{T}_1^4$ from (2.32) and the homogeneous transformations (2.11). Comparing the ${}_2p_{4y}$ components of the vector ${}_2\mathbf{p}_4$, see Figure 2.7, yields

$$-{}_1p_{4x} \sin(\theta_2) + {}_1p_{4y} \cos(\theta_2) = -a_3 \sin(\theta_3) , \tag{2.37}$$

which can then be written as

$$-\sqrt{{}_1p_{4x}^2 + {}_1p_{4y}^2}\left(\sin(\theta_2)\cos(\gamma_2) - \cos(\theta_2)\sin(\gamma_2)\right) = -a_3 \sin(\theta_3) , \tag{2.38}$$

with $\gamma_2 = \arctan\left({}_1p_{4y}/{}_1p_{4x}\right)$. Finally, the solution for θ_2 is given by

$$\theta_2 = \arctan\left(\frac{{}_1p_{4y}}{{}_1p_{4x}}\right) - \arcsin\left(\frac{-a_3 \sin(\theta_3)}{\sqrt{{}_1p_{4x}^2 + {}_1p_{4y}^2}}\right) . \tag{2.39}$$

The solution for θ_2 is defined if the argument of $\arcsin(\cdot)$ is within $[-1, 1]$, which is always the case if an inverse kinematics solution exists.

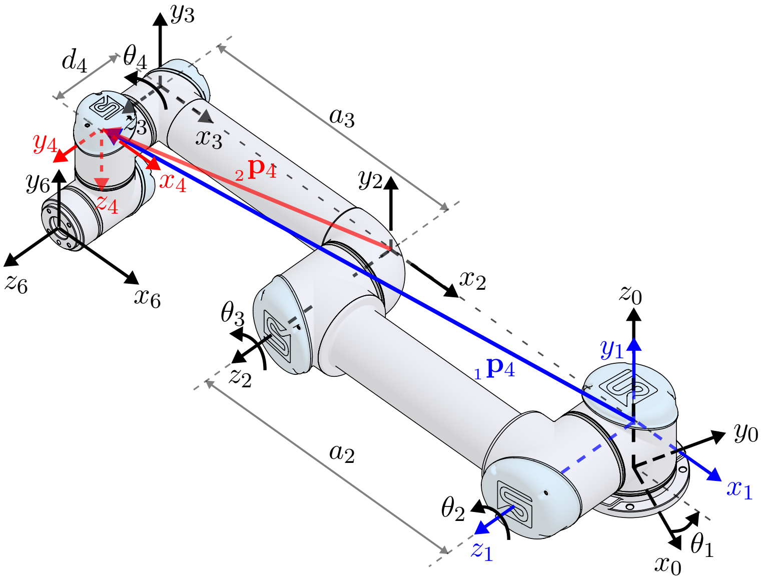


Figure 2.7: Schematic illustration for computing the joint angles θ_2 and θ_4.

The last remaining joint angle θ_4 can be computed by transforming the matrix $\mathbf{T}_1^4$ from (2.32) back to the coordinate frame $(o_3 x_3 y_3 z_3)$, i. e.,

$$\mathbf{T}_3^4 = \mathbf{T}_3^1(\theta_2, \theta_3)\mathbf{T}_1^4 = \begin{bmatrix} {}_3\hat{x}_{4x} & {}_3\hat{y}_{4x} & {}_3\hat{z}_{4x} & {}_3p_{4x} \\ {}_3\hat{x}_{4y} & {}_3\hat{y}_{4y} & {}_3\hat{z}_{4y} & {}_3p_{4y} \\ {}_3\hat{x}_{4z} & {}_3\hat{y}_{4z} & {}_3\hat{z}_{4z} & {}_3p_{4z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \overset{!}{=} \begin{bmatrix} \cos(\theta_4) & 0 & \sin(\theta_4) & 0 \\ \sin(\theta_4) & 0 & -\cos(\theta_4) & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \tag{2.40}$$

Thus, θ_4 can be determined from the direction of the x_4 axis relative to the frame of link three in the form

$$\theta_4 = \arctan\left(\frac{{}_3\hat{x}_{4y}}{{}_3\hat{x}_{4x}}\right) . \tag{2.41}$$

With the solutions for θ_2 (2.39), θ_4 (2.41), θ_6 (2.31), and the respective double solutions for θ_1 (2.23), θ_3 (2.35) and θ_5 (2.26), there exist in total eight geometric robot configurations ($c = 8$) which correspond to the same end-effector pose. Considering that the revolute joints are limited to $\theta_j \in [-2\pi, 2\pi]$, $j \in \{1, \dots, n\}$, also the $\pm 2\pi$ solutions, i. e., $\theta_j - \operatorname{sgn}(\theta_j)2\pi$, should be considered as possible solutions, resulting in 2^n solutions per robot configuration and $2^n c$ possible joint configurations for a given end-effector pose. Let $p \in \mathcal{P}$ denote a task point in the robot's operational space $\mathcal{W}$. This results for the considered robotic arm with six rotational joints ($n = 6$) in a set $\mathcal{C}_p$ of 512 possible joint configurations for a given task-related robot end-effector position and orientation. This set is usually reduced to a set of feasible robot joint configurations since robots typically operate in working environments surrounded by static obstacles, which limit their operational space. Besides the presented analytical approach, several numerical approaches exist to compute the robot joint configurations for a given end-effector pose, which generally converge to a single solution.

2.2 Manipulator Jacobian

The forward robot kinematics $\mathbf{f}_{\text{kin}}(\mathbf{q}) = \left[{}_0\mathbf{p}_{\text{e}}^{\text{T}}, \boldsymbol{\phi}_{\text{e}}^{\text{T}}\right]^{\text{T}}$ from (2.8) represents the relation between the robot joints $\mathbf{q} \in \mathbb{R}^n$ and the pose of the end-effector in operational space relative to the inertial frame. The time derivative of the forward kinematics equation results in a set of equations

$$\begin{bmatrix} {}_0\dot{\mathbf{p}}_{\text{e}} \\ \dot{\boldsymbol{\phi}}_{\text{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_p \\ \mathbf{J}_\phi \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}_a \dot{\mathbf{q}} \tag{2.42}$$

describing the relationship between the velocities in the operational space $\mathcal{W}$ and joint velocities $\dot{\mathbf{q}} \in \mathbb{R}^n$ in configuration space $\mathcal{C}$. In (2.42), $\mathbf{J}_a \in \mathbb{R}^{6\times n}$ denotes the analytical manipulator Jacobian with its translational $\mathbf{J}_p$ and rotational $\mathbf{J}_\phi$ parts. The name analytical refers to the analytical approach used to compute the Jacobian via differentiation of the direct kinematics equation with respect to the joint variables, i. e.,

$${}_0\dot{\mathbf{p}}_{\text{e}} = \frac{\partial {}_0\mathbf{p}_{\text{e}}(\mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}_p \dot{\mathbf{q}} \,, \quad \dot{\boldsymbol{\phi}}_{\text{e}} = \frac{\partial \boldsymbol{\phi}_{\text{e}}(\mathbf{q})}{\partial \mathbf{q}} \dot{\mathbf{q}} = \mathbf{J}_\phi \dot{\mathbf{q}} \,. \tag{2.43}$$

Computing the rotational analytical Jacobian $\mathbf{J}_\phi$ is not straightforward since it includes determining the minimal representation angles $\boldsymbol{\phi}_{\text{e}}(\mathbf{q})$ as function of the robot joints $\mathbf{q}$ by solving an inverse problem, see Section 2.1.1. To avoid this, a geometric approach can also be used to find the relationship between the velocities in joint and configuration space as

$$\begin{bmatrix} {}_0\dot{\mathbf{p}}_{\text{e}} \\ {}_0\boldsymbol{\omega}_{\text{e}} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_p \\ \mathbf{J}_\omega \end{bmatrix} \dot{\mathbf{q}} = \mathbf{J}_g \dot{\mathbf{q}} \,, \tag{2.44}$$

with the geometric Jacobian $\mathbf{J}_g \in \mathbb{R}^{6\times n}$, and its translational $\mathbf{J}_p$ and rotational $\mathbf{J}_\omega$ parts. The geometric approach consists of determining the contributions of each joint velocity to the end-effector linear ${}_0\dot{\mathbf{p}}_{\text{e}}$ and angular ${}_0\boldsymbol{\omega}_{\text{e}}$ velocity components in the form, see, e. g., [85]

$${}_0\dot{\mathbf{p}}_{\text{e}} = \sum_{j=1}^{n} \frac{\partial {}_0\mathbf{p}_{\text{e}}(\mathbf{q})}{\partial q_j} \dot{q}_j = \sum_{j=1}^{n} \mathbf{j}_{p_j} \dot{q}_j \,, \quad {}_0\boldsymbol{\omega}_{\text{e}} = \sum_{j=1}^{n} \mathbf{j}_{\omega_j} \dot{q}_j \,. \tag{2.45}$$

Considering revolute robot joints and the classical DH convection, which consist of rigidly assigning a coordinate frame $(o_{j-1}x_{j-1}y_{j-1}z_{j-1})$ to robot joint j, the angular joint velocity expressed in the local frame is given by

$${}_{j-1}\boldsymbol{\omega}_j = {}_{j-1}\mathbf{e}_{z,j-1} \dot{\theta}_j \,, \tag{2.46}$$

with ${}_{j-1}\mathbf{e}_{z,j-1} = \mathbf{e}_z = [0, 0, 1]^{\text{T}}$. ${}_{j-1}\boldsymbol{\omega}_j$ can also be referred to as the angular velocity of link j relative to link $j-1$ expressed in the coordinate frame of the later one. Transforming equation (2.46) to the inertial frame defines the contribution of joint velocity $\dot{\theta}_j$ to the angular end-effector velocity

$$\mathbf{j}_{\omega_j} \dot{q}_j = \mathbf{R}_0^{j-1} {}_{j-1}\boldsymbol{\omega}_j = {}_0\mathbf{e}_{z,j-1} \dot{\theta}_j \,, \tag{2.47}$$

where ${}_0\mathbf{e}_{z,j-1} = \mathbf{R}_0^{j-1} {}_{j-1}\mathbf{e}_{z,j-1}$. Finally, with $\dot{q}_j = \dot{\theta}_j$, the geometric rotational Jacobian can be written as

$$\mathbf{J}_\omega = \begin{bmatrix} \mathbf{j}_{\omega_1} & \mathbf{j}_{\omega_2} & \cdots & \mathbf{j}_{\omega_n} \end{bmatrix} = \begin{bmatrix} {}_0\mathbf{e}_{z,0} & {}_0\mathbf{e}_{z,1} & \cdots & {}_0\mathbf{e}_{z,n-1} \end{bmatrix} \,. \tag{2.48}$$

To calculate the translational geometric Jacobian the end-effector pose relative to the inertial robot frame is expressed as

$$ {}_{0}\mathbf{p}_{\mathrm{e}} = {}_{0}\mathbf{p}_{j-1} + \mathbf{R}_0^{j-1}\,{}_{j-1}\mathbf{p}_j + \mathbf{R}_0^{j}\,{}_{j}\mathbf{p}_{\mathrm{e}}\,. \tag{2.49} $$

The contribution of robot joint j to the end-effector linear velocity is determined by allowing only joint angle θ_j to move and keeping all other joints still. The derivative of (2.49) with respect to θ_j defines the j-th column of $\mathbf{J}_p$, denoted as $\mathbf{j}_{p_j}$, and is given by, see, e. g., [87]

$$ \begin{aligned} \mathbf{j}_{p_j} = \frac{\partial_0 \mathbf{p}_{\mathrm{e}}}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}\left(\mathbf{R}_0^{j-1}\,{}_{j-1}\mathbf{p}_j + \mathbf{R}_0^{j}\,{}_{j}\mathbf{p}_{\mathrm{e}}\right) \\ &= \mathbf{R}_0^{j-1}\frac{\partial_{j-1}\mathbf{p}_j}{\partial \theta_j} + \frac{\partial \mathbf{R}_0^{j}}{\partial \theta_j}\,{}_{j}\mathbf{p}_{\mathrm{e}} \\ &= {}_{0}\mathbf{e}_{z,j-1} \times \left({}_{0}\mathbf{p}_e - {}_{0}\mathbf{p}_{j-1}\right)\,. \end{aligned} \tag{2.50} $$

The translational geometric Jacobian is then defined as

$$ \mathbf{J}_p = \begin{bmatrix} \mathbf{j}_{p_1} & \mathbf{j}_{p_2} & \cdots & \mathbf{j}_{p_n} \end{bmatrix}\,. \tag{2.51} $$

For the translational Jacobian, the analytical and geometric approaches lead to the same matrix. This is not the case for the rotational Jacobian since the analytic rotational Jacobian depends on the used minimal rotation representation. If the Euler angles $\boldsymbol{\phi}_{\mathrm{e}} = [\varphi_{\mathrm{e}}, \vartheta_{\mathrm{e}}, \psi_{\mathrm{e}}]^{\mathrm{T}}$ are used to parameterize the end-effector rotation, a relation between the angular velocity ${}_{0}\boldsymbol{\omega}_{\mathrm{e}}$ and the rotational velocity $\dot{\boldsymbol{\phi}}_{\mathrm{e}}$ can be found. The time derivative of the rotation matrix resulting from the ZYZ Euler parametrization (2.9) is given by

$$ \dot{\mathbf{R}}_{ZYZ}(\boldsymbol{\phi}_{\mathrm{e}}) = \mathbf{S}({}_{0}\boldsymbol{\omega}_{\mathrm{e}})\mathbf{R}_{ZYZ}(\boldsymbol{\phi}_{\mathrm{e}})\,, \tag{2.52} $$

with the skew-symmetric matrix $\mathbf{S}$ and the angular velocity vector

$$ {}_{0}\boldsymbol{\omega}_{\mathrm{e}} = \mathbf{T}_{ZYZ}(\boldsymbol{\phi}_e)\dot{\boldsymbol{\phi}}_{\mathrm{e}}\,. \tag{2.53} $$

Here, the matrix

$$ \mathbf{T}_{ZYZ}(\boldsymbol{\phi}_e) = \begin{bmatrix} 0 & -\sin(\varphi_{\mathrm{e}}) & \cos(\varphi_{\mathrm{e}})\sin(\vartheta_{\mathrm{e}}) \\ 0 & \cos(\varphi_{\mathrm{e}}) & \sin(\varphi_{\mathrm{e}})\sin(\vartheta_{\mathrm{e}}) \\ 1 & 0 & \cos(\vartheta_{\mathrm{e}}) \end{bmatrix} \tag{2.54} $$

represents a transformation matrix defining the relation between the time derivative of the Euler angles and the angular velocities, see, e. g., [84, 85]. Combining equation (2.53) with (2.42) and (2.44) yields the relation between the analytical and geometric Jacobian

$$ \mathbf{J}_a = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{ZYZ}^{-1}(\boldsymbol{\phi}_e) \end{bmatrix} \begin{bmatrix} \mathbf{J}_p \\ \mathbf{J}_\omega \end{bmatrix} \tag{2.55} $$

provided $\det(\mathbf{T}_{ZYZ}(\boldsymbol{\phi}_e)) \neq 0$.

2.3 Dynamics of robot manipulators

Mathematical dynamic models in robotics describe the relation between applied generalized forces and the resulting robot joint motion and play a crucial role in the simulation and control of robotic systems. In the literature exist different approaches mainly based on Lagrangian and Newtonian mechanics, which are used to derive a closed mathematical model of a dynamic system. The most commonly used modeling approaches in robotics are the Lagrangian formalism, the Newton-Euler method, and the Projected Newton-Euler method, which all result in an equivalent description of the dynamics in the form of a multi-body dynamic system formulated as

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})\,. \tag{2.56}$$

Here, $\boldsymbol{\tau} \in \mathbb{R}^n$ denotes the vector of generalized torques, $\mathbf{q} \in \mathbb{R}^n$ denotes the vector of generalized coordinates, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the symmetric positive definite generalized inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n \times n}$ represents the coefficients of the centrifugal (proportional to $\dot{q}_i^2$) and Coriolis (proportional to $\dot{q}_i \dot{q}_j, i \neq j$) forces and $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ is the vector of potential forces. The dynamic model can be further extended by considering additional phenomena like friction, joint elasticity etc. Equation (2.56) represents the inverse dynamic model that, for a given set of robot parameters, provides joint torques as function of joint positions, velocities and accelerations. Inverse dynamic models play an important role in the design and operational of robotic systems, in computing torques and selecting suitable actuators, or in identifying dynamic parameters. Given that the generalized inertia matrix is positive definite, the robot joint accelerations can be described in terms of the joint positions, velocities and torques, leading the direct dynamic model

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q}) \left(\boldsymbol{\tau} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q})\right)\,, \tag{2.57}$$

which is mainly used for simulation and testing purposes to analyze the performance of applied control strategies, cf., e. g., [84].

2.3.1 Lagrange method

The Lagrange method provides an analytical approach for deriving the equation of motion of physical systems and is closely related to the d'Almbert and Hamilton principles [99]. The main idea is to consider the motion of the robotic system in the configuration $\mathcal{C}$ space by introducing a set of generalized coordinates that automatically satisfy the kinematic constrains of the system. With the set of generalized coordinates $\mathbf{q}$ and generalized velocities $\dot{\mathbf{q}}$, a scalar energy-based function called the Lagrangian function

$$L = T - U \tag{2.58}$$

is defined, which for mechanical rigid body systems corresponds to the difference between the total kinetic energy T and the total potential energy U. The equation of motion can then be calculated using the Euler-Lagrange equations, known also as Euler-Lagrange equations of the second kind, consisting of a system of second order differential equations for the generalized coordinates $\mathbf{q}$, i. e.,

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_j} - \frac{\partial L}{\partial q_j} = \tau_j\,, \quad j = 1, \ldots, n\,. \tag{2.59}$$

The Lagrangian formalism is often used to derive the dynamic equations in closed form, which are then used for controller design or to obtain a better insight into the structure of the equations. It also exhibits high structural flexibility and allows systematic consideration of various effects such as friction, joint elasticity, actuator transmission system etc.

2.3.1.1 Kinetic and potential energy

The total kinetic energy T stored in the moving links and actuators of robot manipulators is a function of the robot joint coordinates $\mathbf{q}$ and velocities $\dot{\mathbf{q}}$, and is given by

$$T = \sum_{j=1}^{n} \left(T_{Lj} + T_{Aj}\right) \,. \tag{2.60}$$

Here, $T_{Lj} = T_{L_tj} + T_{L_rj}$ and $T_{Aj} = T_{A_tj} + T_{A_rj}$ denote the kinetic energy of link j and actuator j, respectively, consisting each of a translational (T_{L_tj}, T_{A_tj}) and rotational (T_{L_rj}, T_{A_rj}) component. In this work, lightweight robot manipulators are used, and it is assumed that the mass of the stator of the actuator is considered in the mass distribution of the link. Thus, the contributions of the translational kinetic energy of the stators are included in the respective parts of the kinetic energy of the links given by

$$T_{L_tj} = \frac{1}{2} m_{Lj}\, {}_0\dot{\mathbf{p}}_{Lcj}^{\mathrm{T}}\, {}_0\dot{\mathbf{p}}_{Lcj} \,, \tag{2.61}$$

with the link mass m_{Lj}, and the velocity ${}_0\dot{\mathbf{p}}_{Lcj}$ of the vector to the center of the mass ${}_0\mathbf{p}_{Lcj}$ expressed in the inertial frame. The rotational component of the kinetic energy of link j is given by

$$T_{L_rj} = \frac{1}{2} {}_0\boldsymbol{\omega}_j^{\mathrm{T}} \mathbf{R}_0^j \boldsymbol{\Theta}_{Lcj} \left(\mathbf{R}_0^j\right)^{\mathrm{T}} {}_0\boldsymbol{\omega}_j^{\mathrm{T}} \,, \tag{2.62}$$

where ${}_0\boldsymbol{\omega}_j$ represents the angular link velocity vector expressed in the inertial frame and $\boldsymbol{\Theta}_{Lcj}$ is the inertia tensor of link j with respect to its center of mass expressed in the local body-attached frame. By applying the transformation $\mathbf{R}_0^j \boldsymbol{\Theta}_{Lcj} \left(\mathbf{R}_0^j\right)^{\mathrm{T}}$ with the rotation matrix $\mathbf{R}_0^j$, the inertia tensor is transformed to the inertial frame of reference.

The total kinetic energy of the rotor j, including the transmission system of the actuator, is calculated by

$$T_{Aj} = T_{A_tj} + T_{A_rj} = \frac{1}{2} m_{Mj}\, {}_0\dot{\mathbf{p}}_{Mcj}^{\mathrm{T}}\, {}_0\dot{\mathbf{p}}_{Mcj} + \frac{1}{2} \Theta_{Aj} \dot{q}_j^2 \,, \tag{2.63}$$

with the rotor mass m_{Mj}, the velocity $\dot{\mathbf{p}}_{Mcj}$ of the vector to the center of the mass relative to the inertial frame, and the moment of inertia $\Theta_{Aj} = \iota_j^2 \Theta_{Mj}$ of the rotor including actuator transmission. Assuming a rigid transmission, the transmission ratio ι_j of actuator j is given by the constant velocity relation $\iota_j = \dot{q}_{Mj}/\dot{q}_j$, with the joint velocity $\dot{q}_j$ and the angular rotor velocity $\dot{q}_{Mj}$. Note that, (2.63) neglects the gyroscopic effects of the rotor, which take place due to the movement of the link where the actuator is rigidly mounted. However, this approximation is justified by the fact that the actuators of the considered lightweight robot have a relatively small rotor mass and high transmission ratio [86].

The total kinetic energy of robot link j can be expressed as

$$T_{Lj} = \frac{1}{2} m_{Lj}\, \dot{\mathbf{q}}^{\mathrm{T}} \mathbf{J}_{pLj}^{\mathrm{T}} \mathbf{J}_{pLj} \dot{\mathbf{q}} + \frac{1}{2} \dot{\mathbf{q}}^{\mathrm{T}} \mathbf{J}_{\omega j}^{\mathrm{T}} \mathbf{R}_0^j \boldsymbol{\Theta}_{Lcj} \left(\mathbf{R}_0^j\right)^{\mathrm{T}} \mathbf{J}_{\omega j} \dot{\mathbf{q}} \,, \tag{2.64}$$

with the analytical translational Jacobian $\mathbf{J}_{pcj}$ and the geometric rotational Jacobian $\mathbf{J}_{\omega j}$ given by

$$\mathbf{J}_{pLj} = \frac{\partial_0 \mathbf{p}_{Lcj}}{\partial \mathbf{q}}\,, \quad \mathbf{J}_{\omega j} = \begin{bmatrix} {}_0\mathbf{e}_{z,0} & {}_0\mathbf{e}_{z,1} & \cdots & {}_0\mathbf{e}_{z,j-1} & \mathbf{0}^{3\times n-j} \end{bmatrix}\,. \tag{2.65}$$

Analogous, the total kinetic energy of actuator j is given by

$$T_{Aj} = \frac{1}{2} m_{Mj}\, \dot{\mathbf{q}}^{\mathrm{T}} \mathbf{J}_{pMj}^{\mathrm{T}} \mathbf{J}_{pMj} \dot{\mathbf{q}} + \frac{1}{2} \dot{\mathbf{q}}^{\mathrm{T}} \mathbf{\Theta}_{Aj} \dot{\mathbf{q}}\,, \tag{2.66}$$

with

$$\mathbf{J}_{pMj} = \frac{\partial_0 \mathbf{p}_{Mcj}}{\partial \mathbf{q}} \quad \text{and} \quad \mathbf{\Theta}_{Aj} = \mathrm{diag}([0, \cdots, \Theta_{Aj}, \cdots, 0])\,. \tag{2.67}$$

For simplicity, let ${}_0\mathbf{p}_{cj}$ represent the vector to the center of mass of the combined body of mass m_j composed of link j and rotor of the actuators rigidly attached to it. The total translational kinetic energy of link j is then written as

$$T_{tj} = \frac{1}{2} m_j\, {}_0\dot{\mathbf{p}}_{cj}^{\mathrm{T}}\, {}_0\dot{\mathbf{p}}_{cj} = \frac{1}{2} m_j\, \dot{\mathbf{q}}^{\mathrm{T}} \mathbf{J}_{pcj}^{\mathrm{T}} \mathbf{J}_{pcj} \dot{\mathbf{q}}\,, \quad \text{with} \quad \mathbf{J}_{pcj} = \frac{\partial_0 \mathbf{p}_{cj}}{\partial \mathbf{q}}\,. \tag{2.68}$$

Finally, the total kinetic energy of the robot manipulator can be expressed as

$$T(\mathbf{q}, \dot{\mathbf{q}}) = \frac{1}{2} \dot{\mathbf{q}}^{\mathrm{T}} \mathbf{M}(\mathbf{q}) \dot{\mathbf{q}}\,, \tag{2.69}$$

with the generalized inertia matrix

$$\mathbf{M}(\mathbf{q}) = \sum_{j=1}^{n} \left(m_j \mathbf{J}_{pcj}^{\mathrm{T}} \mathbf{J}_{pcj} + \mathbf{J}_{\omega j}^{\mathrm{T}} \mathbf{R}_0^j \mathbf{\Theta}_{Lcj} \left(\mathbf{R}_0^j\right)^{\mathrm{T}} \mathbf{J}_{\omega j} + \mathbf{\Theta}_{Aj} \right)\,. \tag{2.70}$$

In this work, rigid robot links are assumed, and the elasticity between the motors and the links is not considered. Thus, the total potential energy stored in the manipulator corresponds to the sum of the potential energy of the individual links resulting from the gravitational forces and is given by

$$U(\mathbf{q}) = -\sum_{j=1}^{n} m_j\, {}_0\boldsymbol{g}^{\mathrm{T}}\, {}_0\mathbf{p}_{cj} \tag{2.71}$$

Here, ${}_0\boldsymbol{g}^{\mathrm{T}} = [0, 0, -g]$ represents the vector of the gravitational acceleration g, that, with respect to the inertial frame, always points towards the negative z_0-axis. m_j and ${}_0\mathbf{p}_{cj}$ denote the mass and the vector to the center of the mass of link j, including the actuator, expressed in the inertial frame.

2.3.1.2 Generalized torque

For robot manipulators with serial links usually it is quite straightforward to obtain the vector $\boldsymbol{\tau}$ of generalized torques since the actuators act in the direction of the generalized coordinates. If external torque and force vectors acting on the robot links are given with respect to the local body-attached Cartesian frame, the generalized torques and forces can be computed by projecting each Cartesian vector onto the space of generalized coordinates using Jacobian matrices. Let ${}_j\mathfrak{T}_j$ represent the Cartesian torque acting on robot link j expressed in the body-attached link frame. The generalized torques can then be computed by

$$\boldsymbol{\tau} = \sum_{j=1}^{n} \mathbf{J}_{\omega j}^{\mathrm{T}} \left(\mathbf{R}_0^j \, {}_j\boldsymbol{\mathfrak{T}}_j \right) , \tag{2.72}$$

with the rotational Jacobian $\mathbf{J}_{\omega j} = \left[{}_0\mathbf{e}_{z,0},\ {}_0\mathbf{e}_{z,1}, \cdots, {}_0\mathbf{e}_{z,j-1}, \mathbf{0}^{3\times n-j}\right]$ of link j, see (2.48), and the transformation matrix $\mathbf{R}_0^j$ expressing the torque vectors in the inertial frame. Similarly, the contributions of external forces ${}_j\boldsymbol{\mathfrak{F}}_j$ can be computed by projecting the Cartesian vectors using the translational Jacobian of the points where forces are acting.

For the UR5 robot manipulator with the body-attached coordinate frames shown in Figure 2.1, the torque vectors in the local frames are given in Table 2.3, considering the action-reaction principle of actuators acting between body links. Applying transformation (2.72) with the Jacobian matrices (2.48) yields the generalized torque vector

$$\boldsymbol{\tau} = \begin{bmatrix} \tau_1 & \tau_2 & \tau_3 & \tau_4 & \tau_5 & \tau_6 \end{bmatrix}^{\mathrm{T}} . \tag{2.73}$$

Table 2.3: Torque vectors expressed in the local body-attached link frames.

Link j	1	2	3	4	5	6
${}_j\boldsymbol{\mathfrak{T}}_j$	$\begin{bmatrix} 0 \\ \tau_1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ \tau_2 - \tau_3 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ \tau_3 - \tau_4 \end{bmatrix}$	$\begin{bmatrix} 0 \\ \tau_4 \\ -\tau_5 \end{bmatrix}$	$\begin{bmatrix} 0 \\ -\tau_5 \\ -\tau_6 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ \tau_6 \end{bmatrix}$

Robot joints and transmission systems exhibit friction phenomena, which are significant for the performance of the robot manipulator. Assuming that there is no interaction of the robot with the environment, the generalized torques can be divided into driving torques $\boldsymbol{\tau}_d$ and friction torques $\boldsymbol{\tau}_f$, i. e., $\boldsymbol{\tau} = \boldsymbol{\tau}_d - \boldsymbol{\tau}_f$. There exist different approaches to model friction effects in control applications, mainly consisting of static or dynamic models [100]. In robotics, usually static friction models are considered, which generally consist of three parts: Coulomb friction, static friction, and viscous friction [84]. In this work, a static friction model describing only the Coulomb and viscous components is used, given by

$$\tau_{fj} = F_{cj} \operatorname{sign}(\dot{q}_j) + F_{vj}\dot{q}_j . \tag{2.74}$$

The friction model is linear in the Coulomb F_{cj} and viscous F_{vj} friction parameters and neglects nonlinearities due to Stribeck phenomena at low rotational speed. Considering all joints $j \in \{1, \ldots, n\}$ of the robot manipulator, the generalized friction torque can be written as

$$\boldsymbol{\tau}_f = \operatorname{diag}\left([\operatorname{sign}(\dot{q}_1), \ldots, \operatorname{sign}(\dot{q}_n)]\right)\mathbf{F}_c + \operatorname{diag}\left([\dot{q}_1, \ldots, \dot{q}_n]\right)\mathbf{F}_v , \tag{2.75}$$

where $\mathbf{F}_c = [F_{c1}, \ldots, F_{cn}]^{\mathrm{T}}$ and $\mathbf{F}_v = [F_{v1}, \ldots, F_{vn}]^{\mathrm{T}}$.

The friction parameters are identified in Chapter 3 along with other dynamic robot parameters. However, friction parameters highly depend on the operation conditions, like temperature, lubrication type, humidity etc. Therefore, for some applications, online methods are applied to estimate and compensate friction forces during runtime [84].

2.3.1.3 Equation of motion

The equations of motion of a multibody robotic system obtained by applying the Euler-Lagrange equations (2.59) are generally expressed in the form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau}_d + \boldsymbol{\tau}_{\text{ext}}\,, \tag{2.76}$$

with the inertia matrix $\mathbf{M}(\mathbf{q})$ from (2.70) and the vector

$$\boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) - \boldsymbol{\tau}_f \tag{2.77}$$

containing centrifugal and Coriolis forces $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$, potential forces $\mathbf{g}(\mathbf{q})$, and generalized friction forces $\boldsymbol{\tau}_f$, see (2.75). $\boldsymbol{\tau}_d$ represents actuator driving torques and $\boldsymbol{\tau}_{\text{ext}}$ generalized forces and torques due to the robot interaction with the environment.

The Coriolis matrix $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is computed directly from the inertia matrix by using the Christoffel symbols of the first king defined as

$$c_{jlk} = \frac{1}{2}\left(\frac{\partial m_{kl}}{\partial q_j} + \frac{\partial m_{kj}}{\partial q_l} - \frac{\partial m_{jl}}{\partial q_k}\right)\,, \tag{2.78}$$

with m_{jl} denoting the $(j,l)^{\text{th}}$ element of the generalized inertia matrix $\mathbf{M}(\mathbf{q})$. The $(k,l)^{\text{th}}$ element of the Coriolis matrix is then given by, cf., e. g., [87],

$$\mathbf{C}[k,l] = \sum_{j=1}^{n} c_{jlk}\dot{q}_j\,. \tag{2.79}$$

The vector of the generalized potential forces resulting from the potential energy $U(\mathbf{q})$ is given by

$$\mathbf{g}^{\mathrm{T}}(\mathbf{q}) = \frac{\partial U(\mathbf{q})}{\partial \mathbf{q}}\,. \tag{2.80}$$

Finally, the equation of motions of the robot are given in the form (2.56), i. e.,

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau}\,, \tag{2.81}$$

where $\boldsymbol{\tau} = \boldsymbol{\tau}_d - \boldsymbol{\tau}_f + \boldsymbol{\tau}_{\text{ext}}$.

2.3.2 Newton-Euler methods

Newton-Euler formulation consists of a force balance approach and describes the motion of the physical system in the Cartesian space. Iterative Newton-Euler methods are often used in robotics for real-time computation of the inverse dynamic model by applying a forward recursion for propagating link velocities and accelerations, followed by a backward recursion for propagating forces and torques, cf., e. g., [84, 85].

In the Lagrange formulation, the equations of motion of a multibody system are derived starting from the total Lagrangian of the system without the need to account for internal reaction forces between the links. However, in the Newton-Euler formulation, each link is separated at the joints and considered as a single unit. In this case, constraint forces, which are imposed by the joints and limit the relative motion between links, must be introduced as external forces. Assuming rigid robot links, the Newton-Euler equations for link j with respect to its center of mass are given by

$$m_j \, {}_0\ddot{\mathbf{p}}_{cj} - {}_0\mathfrak{F}_{cj} \tag{2.82}$$

$$\mathbf{R}_0^j \boldsymbol{\Theta}_{cj} \left(\mathbf{R}_0^j\right)^{\mathrm{T}} {}_0\dot{\boldsymbol{\omega}}_j + {}_0\boldsymbol{\omega}_j \times \left(\mathbf{R}_0^j \boldsymbol{\Theta}_{cj} \left(\mathbf{R}_0^j\right)^{\mathrm{T}} {}_0\boldsymbol{\omega}_j\right) - {}_0\mathfrak{T}_j \,. \tag{2.83}$$

Here, ${}_0\ddot{\mathbf{p}}_{cj}$ denotes the absolute acceleration of the link center of mass relative to the inertial frame, and ${}_0\mathfrak{F}_{cj}$ are the resultant external forces that act through the center of the mass, expressed also in the inertial frame. Note that forces that do not act through the center of mass have do be transformed to an equivalent force through the center of mass and an additional torque term. The Euler equations (2.83) are also expressed with respect to the inertial frame, where

$${}_0\boldsymbol{\omega}_j = \sum_{i=1}^{j} \mathbf{j}_{\omega i} \dot{q}_i \,, \quad \text{with} \quad \mathbf{j}_{\omega i} = {}_0\mathbf{e}_{z,i-1} \,, \tag{2.84}$$

represents the angular velocity of link j relative to the inertial frame, see (2.45) and (2.47). ${}_0\dot{\boldsymbol{\omega}}_j$ is the absolute angular acceleration of link j, and ${}_0\mathfrak{T}_j$ are the resultant external torques. The constant inertia tensor $\boldsymbol{\Theta}_{cj} = \boldsymbol{\Theta}_{L_{cj}}$ with respect to the center of the mass of link j is transformed to the inertial frame using the rotation matrix $\mathbf{R}_0^j$.

An advantage of the Lagrangian formulation is the ability to provide closed-form equations of motion directly in generalized coordinates in compliance with kinematic constraints imposed by constraint conditions between interconnected bodies. However, Newton-Euler equations can be also used to analytically derive equation of motions in Configuration space by projecting Cartesian forces and torques onto the space of generalized coordinates using Jacobian matrices. With the translational $\mathbf{J}_{p_{cj}}$ and rotational $\mathbf{J}_{\omega j}$ Jacobian the Newton-Euler equations can be written as

$$\begin{bmatrix} m_j \mathbf{J}_{p_{cj}} \\ \mathbf{R}_0^j \boldsymbol{\Theta}_{cj} \left(\mathbf{R}_0^j\right)^{\mathrm{T}} \mathbf{J}_{\omega j} \end{bmatrix} \ddot{\mathbf{q}} + \begin{bmatrix} m_j \dot{\mathbf{J}}_{p_{cj}} \dot{\mathbf{q}} \\ \mathbf{R}_0^j \boldsymbol{\Theta}_{cj} \left(\mathbf{R}_0^j\right)^{\mathrm{T}} \dot{\mathbf{J}}_{\omega j} \dot{\mathbf{q}} + \mathbf{J}_{\omega j} \dot{\mathbf{q}} \times \left(\mathbf{R}_0^j \boldsymbol{\Theta}_{cj} \left(\mathbf{R}_0^j\right)^{\mathrm{T}} \mathbf{J}_{\omega j} \dot{\mathbf{q}}\right) \end{bmatrix} - \begin{bmatrix} {}_0\mathfrak{F}_{cj} \\ {}_0\mathfrak{T}_j \end{bmatrix} . \tag{2.85}$$

The first variational principle encountered in the science of mechanics is the principle of virtual work, which controls the equilibrium of a mechanical systems and is fundamental for the development of analytical mechanics [101]. If, for a mechanical multibody system with constraint conditions, the principle of virtual works applies, then the dynamics of the system can be analyzed without having to evaluate the constraint forces, cf., e. g., [85, 87, 99]. For mechanical manipulators with time-invariant, holonomic constraints, the principle applies, denoting the fact that configuration constraints define forces which do not perform work in the direction of the virtual displacements, i. e., infinitesimal displacements consistent with the constraints.

Let $\delta\mathbf{p}_{cj}$ denote the virtual displacement and $\delta\boldsymbol{\phi}_{cj}$ the variation of an infinitesimal rotation of a local frame attached to the center of the mass of link j. For the virtual displacements to be consistent with the joints, they must satisfy the equation

$$\begin{bmatrix} \delta\mathbf{p}_{cj} \\ \delta\boldsymbol{\phi}_{cj} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{p_{cj}} \\ \mathbf{J}_{\omega j} \end{bmatrix} \delta\mathbf{q} \,. \tag{2.86}$$

Applying the expressions for virtual displacements of generalized coordinates $\delta\mathbf{q}$ and evaluating the principle of virtual work for multibody systems yields the Projected Newton-Euler formulation, see, e. g., [102],

$$\begin{aligned}\mathbf{0} =& \sum_{j=1}^{n}\Bigg(\begin{bmatrix}\mathbf{J}_{pcj}\\ \mathbf{J}_{\omega j}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}m_j\mathbf{J}_{pcj}\\ \mathbf{R}_0^j\boldsymbol{\Theta}_{cj}\left(\mathbf{R}_0^j\right)^{\mathrm{T}}\mathbf{J}_{\omega j}\end{bmatrix}\ddot{\mathbf{q}}\\ &+\begin{bmatrix}\mathbf{J}_{pcj}\\ \mathbf{J}_{\omega j}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}m_j\dot{\mathbf{J}}_{pcj}\dot{\mathbf{q}}\\ \mathbf{R}_0^j\boldsymbol{\Theta}_{cj}\left(\mathbf{R}_0^j\right)^{\mathrm{T}}\dot{\mathbf{J}}_{\omega j}\dot{\mathbf{q}}+\mathbf{J}_{\omega j}\dot{\mathbf{q}}\times\left(\mathbf{R}_0^j\boldsymbol{\Theta}_{cj}\left(\mathbf{R}_0^j\right)^{\mathrm{T}}\mathbf{J}_{\omega j}\dot{\mathbf{q}}\right)\end{bmatrix}-\begin{bmatrix}\mathbf{J}_{pcj}\\ \mathbf{J}_{\omega j}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}{}_0\mathfrak{F}_{cj}\\ {}_0\mathfrak{T}_j\end{bmatrix}\Bigg).\end{aligned} \tag{2.87}$$

Following the assumptions made in Section 2.3.1 for the modeling of the robot arm by considering a simplified rotor dynamics and neglecting a potential interaction of the robot with the environment, the external forces and torques can be expressed in the form

$$\sum_{j=1}^{n}\mathbf{J}_{pcj}^{\mathrm{T}}\,{}_0\mathfrak{F}_{cj}=\sum_{j=1}^{n}\mathbf{J}_{pcj}^{\mathrm{T}}\left(m_j\,{}_0\boldsymbol{g}\right)=-\left(\frac{\partial U(\mathbf{q})}{\partial\mathbf{q}}\right)^{\mathrm{T}} \tag{2.88}$$

and

$$\sum_{j=1}^{n}\mathbf{J}_{\omega j}^{\mathrm{T}}\,{}_0\mathfrak{T}_j=\boldsymbol{\tau}-\sum_{j=1}^{n}\boldsymbol{\Theta}_{Aj}\ddot{\mathbf{q}}\,. \tag{2.89}$$

Regrouping terms in (2.87) results in the closed-form representation (2.81) of the robot dynamic model, with the generalized inertia matrix

$$\mathbf{M}(\mathbf{q})=\sum_{j=1}^{n}\left(m_j\mathbf{J}_{pcj}^{\mathrm{T}}\mathbf{J}_{pcj}+\mathbf{J}_{\omega j}^{\mathrm{T}}\mathbf{R}_0^j\boldsymbol{\Theta}_{Lcj}\left(\mathbf{R}_0^j\right)^{\mathrm{T}}\mathbf{J}_{\omega j}+\boldsymbol{\Theta}_{Aj}\right)\,, \tag{2.90}$$

the Coriolis and centrifugal forces

$$\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}=\sum_{j=1}^{n}\left(m_j\mathbf{J}_{pcj}^{\mathrm{T}}\dot{\mathbf{J}}_{pcj}\dot{\mathbf{q}}+\mathbf{J}_{\omega j}^{\mathrm{T}}\left(\mathbf{R}_0^j\boldsymbol{\Theta}_{cj}\left(\mathbf{R}_0^j\right)^{\mathrm{T}}\dot{\mathbf{J}}_{\omega j}\dot{\mathbf{q}}+\mathbf{J}_{\omega j}\dot{\mathbf{q}}\times\left(\mathbf{R}_0^j\boldsymbol{\Theta}_{cj}\left(\mathbf{R}_0^j\right)^{\mathrm{T}}\mathbf{J}_{\omega j}\dot{\mathbf{q}}\right)\right)\right)\,, \tag{2.91}$$

and the vector of gravitational forces

$$\mathbf{g}(\mathbf{q})=-\sum_{j=1}^{n}\mathbf{J}_{pcj}^{\mathrm{T}}\left(m_j\,{}_0\boldsymbol{g}\right)\,. \tag{2.92}$$

It is to be noted that by the analytic derivation of the equations of motion (2.87), the linear momentum conservation law and the angular momentum conservation law are expressed with respect to the same coordinate frame. However, since the projections lead to a scalar product of a gradient with a vector, the Newton and Euler equations can be expressed in different coordinate frames as long as the gradient and the corresponding vector are defined with respect to the same coordinate frame. Given that the inertia tensors $\boldsymbol{\Theta}_{cj}$ are computed with respect to the local body-attached frames and thus constant, it is convenient to define the projected Euler equations in the respective local frames. In this case, the Jacobian $\mathbf{J}_{\omega j}$ has to be expressed also with respect to the the same body-attached coordinate frames. On the other hand, the projected Newton equations are generally expressed with respect to an inertial frame of reference.

2.4 Kinematics of an omnidirectional mobile robot

Different drive mechanisms and wheel typologies are used in mobile robotics to design robot platforms with high motion flexibility and maneuverability capable of performing tasks in narrow spaces and environments with static and dynamic obstacles. Depending on the type of wheels and motion capabilities, wheeled mobile robots can be classified into two major categories, omnidirectional and nonholonomic [103]. Nonholonomic mobile robots are subject to nonholonomic constraints imposing a specific relation between the robot's generalized coordinates $\mathbf{q}_{\mathrm{m}}$ and possible generalized velocities $\dot{\mathbf{q}}_{\mathrm{m}}$ at any time, preventing the robot from being able to move in any direction in a two-dimensional (2D) plane. In contrast, holonomic or omnidirectional robots can perform complex motion combinations in any direction. They usually employ non-steered omniwheels or mecanum wheels, which are driven forward or backward and allow sideways sliding, imposing no velocity constraints on the robot's chassis [103]. This work considers an omnidirectional mobile robot equipped with four mecanum wheels, as shown in Figure 2.8.

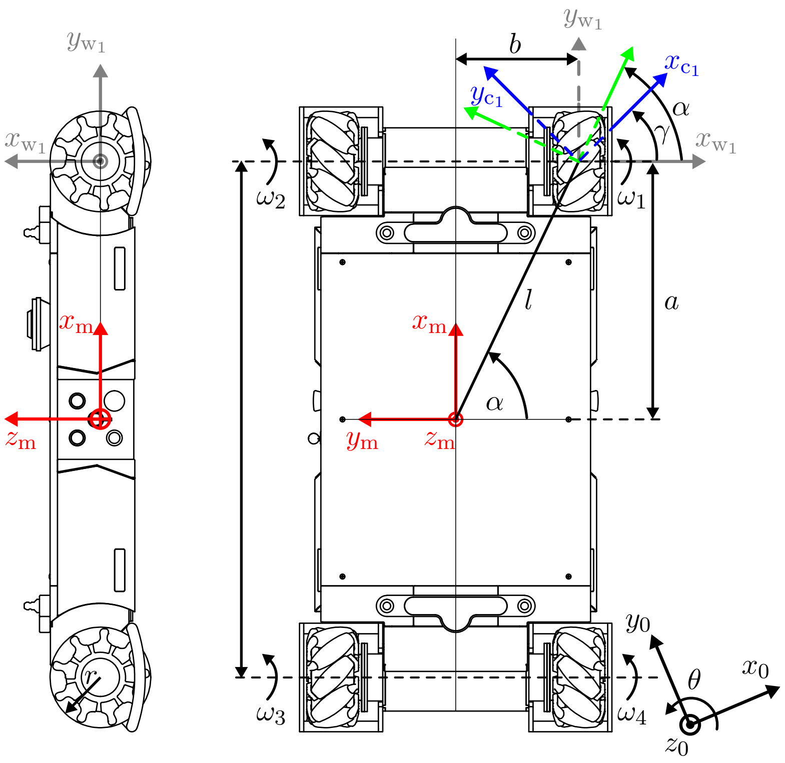


Figure 2.8: Mobile robot with mecanum wheels[4].

The kinematic model of a mobile robot describes the relation between the wheel speeds $\boldsymbol{\omega}_{\mathrm{m}} = [\omega_1, \omega_2, \omega_3, \omega_4]^{\mathrm{T}}$ and the robot's chassis velocity represented by the generalized velocities $\dot{\mathbf{q}}_{\mathrm{m}} = [v_x, v_y, \omega]^{\mathrm{T}}$. In order to derive the kinematics of the mecanum wheeled mobile robot, it is assumed that the robot is rolling without slipping on hard, horizontal, flat ground. Each mecanum wheel consists of nine passive rollers mounted around the wheel circumference. These rollers rotate freely about their rotation axis, which forms an angle γ with the wheel's rotation axis, see Figure 2.8. The coordinate frame $(o_{\mathrm{m}}x_{\mathrm{m}}y_{\mathrm{m}}z_{\mathrm{m}})$ is attached to the robot chassis, and to each wheel is assigned a coordinate frame

---

[4] DONKEYmotion

$(o_{\mathrm{w}_i} x_{\mathrm{w}_i} y_{\mathrm{w}_i} z_{\mathrm{w}_i}), i \in \{1, \ldots, 4\}$, as schematically illustrated for wheel one. In addition, the roller which is in contact with the ground is associated with a coordinate frame $(o_{\mathrm{c}_i} x_{\mathrm{c}_i} y_{\mathrm{c}_i} z_{\mathrm{c}_i})$ such that the frame origin is at the contact point and the x-axis shows in the direction of the rollers rotation axis. The constant parameters $\alpha = \arctan(b/a)$ and $l = \sqrt{a^2 + b^2}$ describe the position of the wheel frame relative to the robot frame, and the angle θ describes the orientation of the robot frame relative to the inertial frame.

Let ${}_0\mathbf{p}_\mathrm{m}$ represent the vector from the origin of the inertial frame $(o_0 x_0 y_0 z_0)$ to the origin of the local robot frame $(o_\mathrm{m} x_\mathrm{m} y_\mathrm{m} z_\mathrm{m})$ expressed in inertial coordinates. The vector ${}_0\mathbf{p}_\mathrm{w}$ from the inertial frame to the center of the omnidirectional wheel, relative to the inertial frame, is given by

$$ {}_0\mathbf{p}_\mathrm{w} = {}_0\mathbf{p}_\mathrm{m} + \mathbf{R}_0^\mathrm{m}\,{}_\mathrm{m}\mathbf{p}_\mathrm{mw}\,. \tag{2.93} $$

Here, ${}_\mathrm{m}\mathbf{p}_\mathrm{mw} = \mathbf{R}_\mathrm{m}^\mathrm{w}\,[l,\ 0,\ 0]^\mathrm{T}$ represents the position of the wheel center relative to the local robot frame, with the rotation matrices $\mathbf{R}_\mathrm{m}^\mathrm{w} = \mathbf{R}_{z,(\alpha - \frac{\pi}{2})}, \mathbf{R}_0^\mathrm{m} = \mathbf{R}_{z,\theta}$ from the wheel to the robot frame and from the robot to the inertial frame, respectively. Given that $\alpha = \mathrm{const.}$, the velocity ${}_0\mathbf{v}_\mathrm{w} = {}_0\dot{\mathbf{p}}_\mathrm{w}$ of the wheel center is given by

$$ {}_0\mathbf{v}_\mathrm{w} = {}_0\mathbf{v}_\mathrm{m} + \dot{\mathbf{R}}_0^\mathrm{m}\,{}_\mathrm{m}\mathbf{p}_\mathrm{mw} = {}_0\mathbf{v}_\mathrm{m} + \mathbf{S}\mathbf{R}_0^\mathrm{m}\,{}_\mathrm{m}\mathbf{p}_\mathrm{mw}\,, \tag{2.94} $$

with the skew-symmetric matrix $\mathbf{S}(\omega) = \dot{\mathbf{R}}_0^\mathrm{m}\,(\mathbf{R}_0^\mathrm{m})^\mathrm{T}$, where $\omega = \dot{\theta}$. Transforming the velocity vector to the local robot frame yields

$$ \begin{aligned} (\mathbf{R}_0^\mathrm{m})^\mathrm{T}\,{}_0\mathbf{v}_\mathrm{w} &= (\mathbf{R}_0^\mathrm{m})^\mathrm{T}\,{}_0\mathbf{v}_\mathrm{m} + (\mathbf{R}_0^\mathrm{m})^\mathrm{T}\,\mathbf{S}\mathbf{R}_0^\mathrm{m}\,{}_\mathrm{m}\mathbf{p}_\mathrm{mw} \\ {}_\mathrm{m}\mathbf{v}_\mathrm{w} &= (\mathbf{R}_0^\mathrm{m})^\mathrm{T}\,{}_0\mathbf{v}_\mathrm{m} + \mathbf{S}\,{}_\mathrm{m}\mathbf{p}_\mathrm{mw}. \end{aligned} \tag{2.95} $$

Using ${}_\mathrm{m}\mathbf{v}_\mathrm{w}$, the velocity at the contact point of the wheel roller relative to the local robot frame is given by

$$ {}_\mathrm{m}\mathbf{v}_\mathrm{c} = {}_\mathrm{m}\mathbf{v}_\mathrm{w} + {}_\mathrm{m}\boldsymbol{\omega}_\mathrm{w} \times {}_\mathrm{m}\mathbf{r}_\mathrm{w} = {}_\mathrm{m}\mathbf{v}_\mathrm{w} + \begin{bmatrix} 0 \\ \dot{\phi} \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ -r \end{bmatrix} = {}_\mathrm{m}\mathbf{v}_\mathrm{w} - \begin{bmatrix} r\dot{\phi} \\ 0 \\ 0 \end{bmatrix}, \tag{2.96} $$

with the angular wheel velocity vector ${}_\mathrm{m}\boldsymbol{\omega}_\mathrm{w} = [0,\ \dot{\phi},\ 0]^\mathrm{T}$ and the wheel radius r. Expressing the velocity at the roller contact point with respect to the roller-attached coordinate frame $(o_\mathrm{c} x_\mathrm{c} y_\mathrm{c} z_\mathrm{c})$ by applying the transformation matrix $\mathbf{R}_\mathrm{m}^\mathrm{c} = \mathbf{R}_{z,(\gamma - \frac{\pi}{2})}$ yields

$$ (\mathbf{R}_\mathrm{m}^\mathrm{c})^\mathrm{T}\,{}_\mathrm{m}\mathbf{v}_\mathrm{c} = (\mathbf{R}_\mathrm{m}^\mathrm{c})^\mathrm{T}\,{}_\mathrm{m}\mathbf{v}_\mathrm{w} - (\mathbf{R}_\mathrm{m}^\mathrm{c})^\mathrm{T} \begin{bmatrix} r\dot{\phi} \\ 0 \\ 0 \end{bmatrix}. \tag{2.97} $$

Substituting (2.95) into (2.97) gives

$$ \begin{aligned} {}_\mathrm{c}\mathbf{v}_\mathrm{c} &= (\mathbf{R}_\mathrm{m}^\mathrm{c})^\mathrm{T}\,(\mathbf{R}_0^\mathrm{m})^\mathrm{T}\,{}_0\mathbf{v}_\mathrm{m} + (\mathbf{R}_\mathrm{m}^\mathrm{c})^\mathrm{T}\mathbf{S}\,{}_\mathrm{m}\mathbf{p}_\mathrm{mw} - (\mathbf{R}_\mathrm{m}^\mathrm{c})^\mathrm{T} \begin{bmatrix} r\dot{\phi} \\ 0 \\ 0 \end{bmatrix} \\ {}_\mathrm{c}\mathbf{v}_\mathrm{c} &= \left[(\mathbf{R}_\mathrm{m}^\mathrm{c})^\mathrm{T} + \left[0\ 0\ \frac{1}{\omega}\,(\mathbf{R}_\mathrm{m}^\mathrm{c})^\mathrm{T}\mathbf{S}\,{}_\mathrm{m}\mathbf{p}_\mathrm{mw}\right]\right](\mathbf{R}_0^\mathrm{m})^\mathrm{T}\,{}_0\mathbf{v}_\mathrm{m} - (\mathbf{R}_\mathrm{m}^\mathrm{c})^\mathrm{T} \begin{bmatrix} r\dot{\phi} \\ 0 \\ 0 \end{bmatrix}, \end{aligned} \tag{2.98} $$

with

$$ {}_\mathrm{m}\mathbf{v}_\mathrm{m} = (\mathbf{R}_0^\mathrm{m})^\mathrm{T}\,{}_0\mathbf{v}_\mathrm{m} = \begin{bmatrix} v_x & v_y & \omega \end{bmatrix}^\mathrm{T} \tag{2.99} $$

representing the robot velocity relative to the local robot frame. The wheels rolling without slipping condition at the contact point implies the constraint

$$\mathbf{e}_x^{\mathrm{T}}\,{}_{\mathrm{c}}\mathbf{v}_{\mathrm{c}} = \mathbf{e}_x^{\mathrm{T}} \left(\mathbf{R}_{\mathrm{m}}^{\mathrm{c}}\right)^{\mathrm{T}} {}_{\mathrm{m}}\mathbf{v}_{\mathrm{c}} = 0\,, \tag{2.100}$$

yielding the kinematic constraint equation for wheel $i \in \{1, \dots, 4\}$

$$\begin{bmatrix} \sin(\gamma_i) & -\cos(\gamma_i) & -l\sin(\alpha_i - \gamma_i) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = r_i \dot{\phi}_i \sin(\gamma_i)\,. \tag{2.101}$$

The kinematic constraints for the four wheels are then given by

$$\begin{bmatrix} \sin(\gamma_1) & -\cos(\gamma_1) & -l\sin(\alpha_1 - \gamma_1) \\ \sin(\gamma_2) & -\cos(\gamma_2) & -l\sin(\alpha_2 - \gamma_2) \\ \sin(\gamma_3) & -\cos(\gamma_3) & -l\sin(\alpha_3 - \gamma_3) \\ \sin(\gamma_4) & -\cos(\gamma_4) & -l\sin(\alpha_4 - \gamma_4) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} = \begin{bmatrix} r_1\dot{\phi}_1 \sin(\gamma_1) \\ r_2\dot{\phi}_2 \sin(\gamma_2) \\ r_3\dot{\phi}_3 \sin(\gamma_3) \\ r_4\dot{\phi}_4 \sin(\gamma_4) \end{bmatrix} \tag{2.102}$$

which, with the geometric parameters from Table 2.4 and $\dot{\mathbf{q}}_{\mathrm{m}} = {}_{\mathrm{m}}\mathbf{v}_{\mathrm{m}}$, lead the following inverse robot kinematics equation

$$\boldsymbol{\omega}_{\mathrm{m}} = \mathbf{J}_{\mathrm{m}}\dot{\mathbf{q}}_{\mathrm{m}}\,, \quad \text{with} \quad \mathbf{J}_{\mathrm{m}} = \frac{1}{r}\begin{bmatrix} 1 & 1 & l\sqrt{2}\sin(\alpha + \frac{\pi}{4}) \\ 1 & -1 & -l\sqrt{2}\sin(\alpha + \frac{\pi}{4}) \\ 1 & 1 & -l\sqrt{2}\sin(\alpha + \frac{\pi}{4}) \\ 1 & -1 & l\sqrt{2}\sin(\alpha + \frac{\pi}{4}) \end{bmatrix}\,, \tag{2.103}$$

where $\mathbf{J}_{\mathrm{m}} \in \mathbb{R}^{4\times 3}$ is the Jacobian matrix with constant parameters, $\boldsymbol{\omega}_{\mathrm{m}} = [\dot{\phi}_1, \dots, \dot{\phi}_4]^{\mathrm{T}}$, and $r = r_i, \forall i \in \{1, \dots, 4\}$. An omnidirectional mobile robot requires at least three wheels to achieve an arbitrary motion on a 2D plane. For a mobile robot equipped with more than three wheels the forward kinematics equations are overdetermined, i. e., $\operatorname{rank}(\mathbf{J}_{\mathrm{m}}) = 3$. The forward kinematics can be obtained by applying the Least Squares approach

$$\min_{\dot{\mathbf{q}}_{\mathrm{m}}} ||\boldsymbol{\omega}_{\mathrm{m}} - \mathbf{J}_{\mathrm{m}}\dot{\mathbf{q}}_{\mathrm{m}}||_2^2\,, \tag{2.104}$$

leading the optimal solution

$$\dot{\mathbf{q}}_{\mathrm{m}} = \left(\mathbf{J}_{\mathrm{m}}^{\mathrm{T}}\mathbf{J}_{\mathrm{m}}\right)^{-1}\mathbf{J}_{\mathrm{m}}^{\mathrm{T}}\boldsymbol{\omega}_{\mathrm{m}} = \mathbf{J}_{\mathrm{m}}^{\dagger}\boldsymbol{\omega}_{\mathrm{m}}\,, \tag{2.105}$$

with the Moore-Penrose pseudoinverse matrix

$$\mathbf{J}_{\mathrm{m}}^{\dagger} = \frac{r}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ \frac{1}{l\sqrt{2}\sin(\alpha + \frac{\pi}{4})} & \frac{-1}{l\sqrt{2}\sin(\alpha + \frac{\pi}{4})} & \frac{-1}{l\sqrt{2}\sin(\alpha + \frac{\pi}{4})} & \frac{1}{l\sqrt{2}\sin(\alpha + \frac{\pi}{4})} \end{bmatrix}\,. \tag{2.106}$$

Table 2.4: Geometric parameters of the mobile robot.

Wheel i	1	2	3	4
γ_i	$-\frac{\pi}{4}$	$\frac{\pi}{4}$	$-\frac{\pi}{4}$	$\frac{\pi}{4}$
α_i	α	$\pi - \alpha$	$\pi + \alpha$	$2\pi - \alpha$

2.5 Dynamics of an omnidirectional mobile robot

The dynamics of a mobile robot can be derived analogous to the dynamics of robot manipulators or other mechanical systems by applying the Lagrangian formalism or the Newton-Euler equations. This section applies the Lagrange method to derive the equation of motion for the introduced omnidirectional mobile robot with mecanum wheels. Assuming that the robot is moving on a plane ground, the potential energy is considered to be zero, resulting in the Lagrange function being equal to the total kinetic energy of the system, i. e., $L = T_{\mathrm{m}}$.

The translational kinetic energy of the mobile robot is given by

$$T_{\mathrm{m}_t} = \frac{1}{2} m_{\mathrm{m}} \, {}_0\dot{\mathbf{p}}_{c_{\mathrm{m}}}^{\mathrm{T}} \, {}_0\dot{\mathbf{p}}_{c_{\mathrm{m}}} \,, \tag{2.107}$$

where ${}_0\dot{\mathbf{p}}_{c_{\mathrm{m}}}$ represents the absolute velocity vector of the center of the robot mass m_{m}. The position of the center of the mass relative to the inertial frame can be written as

$${}_0\mathbf{p}_{c_{\mathrm{m}}} = {}_0\mathbf{p}_{\mathrm{m}} + \mathbf{R}_0^{\mathrm{m}} \, {}_{\mathrm{m}}\mathbf{p}_{c_{\mathrm{m}}} \,, \tag{2.108}$$

with the vector ${}_0\mathbf{p}_{\mathrm{m}}$ from the inertial frame to the origin of the body-attached robot frame and ${}_{\mathrm{m}}\mathbf{p}_{c_{\mathrm{m}}}$ representing the vector to the center of the mass relative to the local robot frame. The velocity vector can be computed as

$${}_0\dot{\mathbf{p}}_{c_{\mathrm{m}}} = {}_0\mathbf{v}_{\mathrm{m}} + \dot{\mathbf{R}}_0^{\mathrm{m}} \, {}_{\mathrm{m}}\mathbf{p}_{c_{\mathrm{m}}} = \mathbf{R}_0^{\mathrm{m}} \, {}_{\mathrm{m}}\mathbf{v}_{\mathrm{m}} + \dot{\mathbf{R}}_0^{\mathrm{m}} \, {}_{\mathrm{m}}\mathbf{p}_{c_{\mathrm{m}}} \,, \tag{2.109}$$

with ${}_{\mathrm{m}}\mathbf{v}_{\mathrm{m}} = \dot{\mathbf{q}}_{\mathrm{m}}$, see (2.99), and finally expressed in the form

$${}_0\dot{\mathbf{p}}_{c_{\mathrm{m}}} = \mathbf{J}_{p_{c\mathrm{m}}} \dot{\mathbf{q}}_{\mathrm{m}} \,, \tag{2.110}$$

where

$$\mathbf{J}_{p_{c\mathrm{m}}}(\theta, {}_{\mathrm{m}}\mathbf{p}_{c_{\mathrm{m}}}) = \frac{\partial_0 \dot{\mathbf{p}}_{c_{\mathrm{m}}}}{\partial \dot{\mathbf{q}}_{\mathrm{m}}} \tag{2.111}$$

denotes the Jacobian matrix. Substituting (2.110) into (2.107) yields the translational kinetic energy

$$T_{\mathrm{m}_t} = \frac{1}{2} m_{\mathrm{m}} \dot{\mathbf{q}}_{\mathrm{m}}^{\mathrm{T}} \mathbf{J}_{p_{c\mathrm{m}}}^{\mathrm{T}} \mathbf{J}_{p_{c\mathrm{m}}} \dot{\mathbf{q}}_{\mathrm{m}} \,. \tag{2.112}$$

The rotational kinetic energy is given by

$$T_{\mathrm{m}_r} = \frac{1}{2} \Theta_{c\mathrm{m}} \dot{\theta}^2 + \frac{1}{2} \sum_{i=1}^{4} \left(m_{\mathrm{w}_i} (r\omega_i)^2 + \Theta_{\mathrm{w}_i} \, \omega_i^2 \right) \,, \tag{2.113}$$

with the moment of inertia $\Theta_{c\mathrm{m}}$ of the robot around the z-axis (rotation axis) passing through the center of the mass, the wheel masses m_{w_i}, and the moment of inertia Θ_{w_i} of the wheels around the rotation axis. Assuming identical mecanum wheels, equation (2.113) can be written in the form

$$T_{\mathrm{m}_r} = \frac{1}{2} \dot{\mathbf{q}}_{\mathrm{m}}^{\mathrm{T}} \mathbf{\Theta}_{c\mathrm{m}} \dot{\mathbf{q}}_{\mathrm{m}} + \frac{1}{2} \left(m_{\mathrm{w}_i} r^2 + \Theta_{\mathrm{w}_i} \right) \boldsymbol{\omega}_{\mathrm{m}}^{\mathrm{T}} \boldsymbol{\omega}_{\mathrm{m}} \,, \tag{2.114}$$

with $\mathbf{\Theta}_{c\mathrm{m}} = \operatorname{diag}([0, 0, \Theta_{c\mathrm{m}}])$ and $\boldsymbol{\omega}_{\mathrm{m}} = [\omega_1, \ldots, \omega_4]^{\mathrm{T}}$. By substituting the inverse kinematics equation $\boldsymbol{\omega}_{\mathrm{m}} = \mathbf{J}_{\mathrm{m}} \dot{\mathbf{q}}_{\mathrm{m}}$, see (2.103), into (2.114), the total kinetic energy of the robot can be expressed as

$$T_{\mathrm{m}} = T_{\mathrm{m}_t} + T_{\mathrm{m}_r} = \frac{1}{2} \dot{\mathbf{q}}_{\mathrm{m}}^{\mathrm{T}} \mathbf{M}_{\mathrm{m}}(\mathbf{q}_{\mathrm{m}}) \dot{\mathbf{q}}_{\mathrm{m}} \,, \tag{2.115}$$

where

$$\mathbf{M}_{\mathrm{m}}(\mathbf{q}_{\mathrm{m}}) = m_{\mathrm{m}} \mathbf{J}_{pcm}^{\mathrm{T}} \mathbf{J}_{pcm} + \boldsymbol{\Theta}_{cm} + \left(m_{\mathrm{w}_i} r^2 + \Theta_{\mathrm{w}_i} \right) \mathbf{J}_{\mathrm{m}}^{\mathrm{T}} \mathbf{J}_{\mathrm{m}} \tag{2.116}$$

is the generalized mass matrix. Not that, the mass matrix depends only on the coordinate θ, i. e., $\mathbf{M}_{\mathrm{m}}(\mathbf{q}_{\mathrm{m}}) = \mathbf{M}_{\mathrm{m}}(\theta)$, and the Coriolis matrix $\mathbf{C}_{\mathrm{m}}(\theta, \dot{\theta})$ can be easily computed by using the Christoffel symbols of the first king (2.78).

The external torque acting on the robot wheel i, neglecting friction phenomena of the actuator, is given by $\tau_{\mathrm{w}_i} - F_{\mathrm{c}_{ri}} r \operatorname{sign}(\omega_i)$, see Figure 2.9, leading to the vector of the external moments

$$\boldsymbol{\mathfrak{T}}_{\mathrm{m}} = \boldsymbol{\tau}_{\mathrm{w}} - \operatorname{diag}\left([r \operatorname{sign}(\omega_1), \ldots, r \operatorname{sign}(\omega_4)]\right) \mathbf{F}_{\mathrm{c}_r} \, . \tag{2.117}$$

Here, $\boldsymbol{\tau}_{\mathrm{w}} = [\tau_{\mathrm{w}_1}, \ldots, \tau_{\mathrm{w}_4}]^{\mathrm{T}}$ represent the actuator torque vector, and $\mathbf{F}_{\mathrm{c}_r} = [F_{\mathrm{c}_{r1}}, \ldots, F_{\mathrm{c}_{r4}}]^{\mathrm{T}}$ contains coefficients of friction forces acting at the contact point of the rollers. Let, ${}_{\mathrm{m}}\boldsymbol{\mathfrak{F}}_{\mathrm{m}}$ represent the external force vector acting on the robot body, expressed with respect to the local robot frame. For the power delivered to the mechanical system by the external forces, respectively, torques, the following holds

$$\boldsymbol{\mathfrak{T}}_{\mathrm{m}}^{\mathrm{T}} \boldsymbol{\omega}_{\mathrm{m}} = {}_{\mathrm{m}}\boldsymbol{\mathfrak{F}}_{\mathrm{m}}^{\mathrm{T}} \dot{\mathbf{q}}_{\mathrm{m}} \, , \tag{2.118}$$

yielding, using (2.103), the vector of the external forces

$${}_{\mathrm{m}}\boldsymbol{\mathfrak{F}}_{\mathrm{m}} = \mathbf{J}_{\mathrm{m}}^{\mathrm{T}} \boldsymbol{\mathfrak{T}}_{\mathrm{m}} \, . \tag{2.119}$$

Substituting (2.117) into (2.119) leads to

$${}_{\mathrm{m}}\boldsymbol{\mathfrak{F}}_{\mathrm{m}} = \mathbf{J}_{\mathrm{m}}^{\mathrm{T}} \boldsymbol{\tau}_{\mathrm{w}} - \mathbf{J}_{\mathrm{m}}^{\mathrm{T}} \mathbf{H}(\boldsymbol{\omega}_{\mathrm{m}}) \mathbf{F}_{\mathrm{c}_r} \, , \tag{2.120}$$

with $\mathbf{H}(\boldsymbol{\omega}_{\mathrm{m}}) = \operatorname{diag}\left([r \operatorname{sign}(\omega_1), \ldots, r \operatorname{sign}(\omega_4)]\right)$.

Finally, the equations of motion of the omnidirectional robot derived by applying the Lagrange equations, i. e.,

$$\frac{\mathrm{d}}{\mathrm{d}t} \left(\frac{\partial T_{\mathrm{m}}}{\partial \dot{\mathbf{q}}_{\mathrm{m}}} \right)^{\mathrm{T}} - \left(\frac{\partial T_{\mathrm{m}}}{\partial \mathbf{q}_{\mathrm{m}}} \right)^{\mathrm{T}} = {}_{\mathrm{m}}\boldsymbol{\mathfrak{F}}_{\mathrm{m}} \, , \tag{2.121}$$

can be expressed in the form

$$\begin{aligned} \mathbf{M}_{\mathrm{m}}(\mathbf{q}_{\mathrm{m}}) \ddot{\mathbf{q}}_{\mathrm{m}} + \mathbf{C}_{\mathrm{m}}(\mathbf{q}_{\mathrm{m}}, \dot{\mathbf{q}}_{\mathrm{m}}) \dot{\mathbf{q}}_{\mathrm{m}} + \mathbf{J}_{\mathrm{m}}^{\mathrm{T}} \mathbf{H}(\boldsymbol{\omega}_{\mathrm{m}}) \mathbf{F}_{\mathrm{c}_r} &= \mathbf{J}_{\mathrm{m}}^{\mathrm{T}} \boldsymbol{\tau}_{\mathrm{w}} \\ \boldsymbol{\omega}_{\mathrm{m}} &= \mathbf{J}_{\mathrm{m}} \dot{\mathbf{q}}_{\mathrm{m}} \, . \end{aligned} \tag{2.122}$$

Note that the equations of motion are derived with respect to the local body-attached robot frame but can easily transformed to the inertial frame or any other frame of reference.

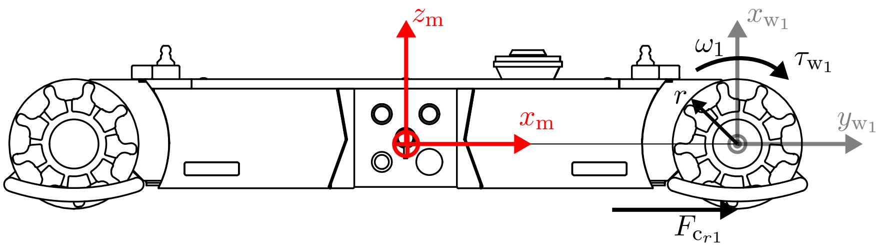


Figure 2.9: External torque on a mecanum wheel.

CHAPTER 3

Dynamic Parameter Identification

In advanced industrial robot applications, one is typically enforced to apply model-based control techniques, such as Computed Torque control that rely on accurate mathematical modeling of the robot manipulator at hand. These approaches are often afflicted with insufficiently known values of dynamical parameters involving inertia elements, friction parameters, etc. Robot manufacturers usually provide dynamic parameters, but they are often incomplete or refer to specific application scenarios and thus do not cover the versatility of different situations, like advanced applications with robot operation in dynamic working environments. Consequently, estimating dynamic parameters is often a mandatory step in developing appropriate control strategies.

This chapter presents a general procedure for parameter estimation of robot manipulators, which makes use of a formulation of the dynamic robot model in terms of symbolic parameter aggregates, i. e., a set of symbolic expressions involving multiple parameters. Roughly speaking, in this approach, one strives to construct proper manipulation trajectories that enable collections of data sets suitable for accurately estimating all involved dynamic parameters. It is important to emphasize that the results of the identification procedure are highly dependent on the synthesis of suitable identifying manipulation trajectories, which typically amount to proper formulations of corresponding trajectory planning optimization problems. The involved optimization problems generally feature, due to the periodic and non-convex nature of the cost function and constraints, a large number of local minima. The L^2 norm condition number of the regressor matrix is commonly chosen for the cost function. The optimization is further subject to constraints to respect the physical robot limitations and avoid collisions between the robot links and the working environment while computing persistent excitation trajectories. To solve the resulting optimization problem gradient-based numerical local solvers [104], sometimes in conjunction with multiple starting points [105], [106] can be applied. Genetic algorithms (GA) have been also used for finding the global minima in certain problem classes, as they are able to cover the whole search space [107–109].

In order to combine the advantages of the latter approaches, a customized metaheuristic algorithm, named memetic algorithm, introduced in [76] and [110], is used. Memetic algorithms (MAs), also known as "Hybrid Evolutionary Algorithms" (hybrid EAs), combine different optimization ideas, such as population-based methods used in genetic algorithms and local search methods, while trying to exploit any given knowledge about the problem [111]. This chapter will study the benefits and efficiency of the proposed computation technique in obtaining excitation robot trajectories suitable for highly accurate parameter estimation of the UR5 robot.

3.1 Identification model

The inverse dynamic model (2.81) of robot manipulators with rigid transmissions is linear in the dynamic parameter vector $\boldsymbol{\varrho} \in \mathbb{R}^{n_\varrho}$ and can be expressed in the form

$$\boldsymbol{\tau}_d = \mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\varrho}\,, \tag{3.1}$$

with the regression matrix $\mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \in \mathbb{R}^{n \times n_\varrho}$, representing a key feature for identifying dynamic parameters, see, e. g., [86], [89]. However, when applying the Lagrangian or Newton-Euler formalism expressed with respect to the center of the mass of the links, see Section 2.3.1 and Section 2.3.2, the resulting dynamic model is not linear in the inertial parameters. For instance, the forces $m_j\,{}_0\ddot{\mathbf{p}}_{cj}$ in the Newton equation (2.82) make use of the mass m_j and the constant vector ${}_j\mathbf{p}_{ocj}$ of the center of mass relative to the link coordinate frame, both subject to identification. Following the notation presented in [86], the recursive Newton-Euler equations are reformulated and expressed with respect to the origin o_j of the local link frames $(o_j x_j y_j z_j)$ in the form

$$\begin{aligned} \mathfrak{F}_j &= m_j\,\ddot{\mathbf{p}}_{oj} + \dot{\boldsymbol{\omega}}_j \times m_j\,\mathbf{p}_{ocj} + \boldsymbol{\omega}_j \times (\boldsymbol{\omega}_j \times m_j\,\mathbf{p}_{ocj}) \\ \mathfrak{T}_j &= \boldsymbol{\Theta}_{oj}\,\dot{\boldsymbol{\omega}}_j + \boldsymbol{\omega}_j \times (\boldsymbol{\Theta}_{oj}\,\boldsymbol{\omega}_j) + m_j\,\mathbf{p}_{ocj} \times \dot{\mathbf{p}}_{oj}\,. \end{aligned} \tag{3.2}$$

Applying recursively the regrouped Newton-Euler equations, results in a dynamic model of the form (3.1), which is linear in the link parameters $\boldsymbol{\varrho}_j, \forall j \in \{1, \ldots, n\}$, cf., e. g., [112], [113]. The link parameters are composed of inertial and friction parameters and can be written in the form

$$\boldsymbol{\varrho}_j = \begin{bmatrix} \boldsymbol{\varrho}_{\Theta j}^{\mathrm{T}} & \boldsymbol{\varrho}_{Pj}^{\mathrm{T}} & \boldsymbol{\varrho}_{mj}^{\mathrm{T}} & \boldsymbol{\varrho}_{fj}^{\mathrm{T}} & \boldsymbol{\varrho}_{Aj}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}. \tag{3.3}$$

Here, the parameter vector $\boldsymbol{\varrho}_{\Theta j}^{\mathrm{T}} = [\Theta_{xxj}, \Theta_{xyj}, \Theta_{xzj}, \Theta_{yyj}, \Theta_{yzj}, \Theta_{zzj}]$ consists of components of the constant inertial tensor $\boldsymbol{\Theta}_{oj}$ expressed with respect to the origin o_j of the local link frame. $\boldsymbol{\varrho}_{Pj} = m_j \mathbf{p}_{ocj} = [P_{xj}, P_{yj}, P_{zj}]^{\mathrm{T}}$ represents the first moments of the links, and $\boldsymbol{\varrho}_{mj} = m_j$ is the mass of link j including all masses of the rotors attached to it. The parameter vector $\boldsymbol{\varrho}_{fj}^{\mathrm{T}} = [F_{cj}, F_{vj}]$ contains the Coulomb and viscous Friction parameters, and $\boldsymbol{\varrho}_{Aj} = \Theta_{Aj}$ is the rotor inertia.

The regression matrix $\mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ depends on the robot's geometry and is considered to be known. Therefore, the modified DH convection is used, following the Khalil-Kleinfinger notation with the body-attached coordinate frames shown in Figure 2.2 and the DH parameters given in Table 2.2. Since some columns of $\mathbf{W}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ are linearly dependent, not all parameters can be identified. Some cannot be identified at all, because they do not have any or have only a neglectable effect on the dynamic model. Others can only be identified in linear combination with other parameters. Therefore, some columns of the matrix and the according parameters are regrouped leading to a minimum set of linear independent parameter, called base parameter set, that determines the dynamic model completely.

The base parameter set is not unique and consists of the actual physical robot parameters or a linear combination of those. There are numerous ways to reduce the model to this set of parameters, e. g., numerically, using the singular value decomposition (SVD) or QR decomposition [114] or symbolically by finding a Groebner basis, see, e. g., [115], [116], of the equations. Deduced from the terms of the energy model of serial robots there exists a closed form symbolic solution [117]. Following this approach, this work applies symbolic regrouping with the workflow presented in [110]. The system dynamics (3.1) expressed in the base parameters $\boldsymbol{\varrho}_b \in \mathbb{R}^{\underline{n}_\varrho}$ is then given by

$$\boldsymbol{\tau}_d = \mathbf{W}_b(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\boldsymbol{\varrho}_b\,, \tag{3.4}$$

with the regression matrix $\mathbf{W}_b(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \in \mathbb{R}^{n\times\underline{n}_\varrho}$. For the system identification, it is required for the robot to execute motion trajectories that can sufficiently excite the identified dynamic parameters. During the experiment, measurements of $\boldsymbol{\tau}_d(t_i), \mathbf{q}(t_i), \dot{\mathbf{q}}(t_i), \ddot{\mathbf{q}}(t_i)$ are taken at multiple time instances $t_{i+1} \geq t_i, t_i \in \{t_0, \dots t_T\}$, and the system identification is reduced to an overdetermined linear parameter estimation problem of the form

$$\mathbf{X}\boldsymbol{\varrho}_b = \boldsymbol{\tau}_b\,, \tag{3.5}$$

with the observation matrix $\mathbf{X}$ and the measured torque vector $\boldsymbol{\tau}_b$ given by

$$\mathbf{X} = \begin{bmatrix} \mathbf{W}_b(\mathbf{q}(t_0), \dot{\mathbf{q}}(t_0), \ddot{\mathbf{q}}(t_0)) \\ \vdots \\ \mathbf{W}_b(\mathbf{q}(t_T), \dot{\mathbf{q}}(t_T), \ddot{\mathbf{q}}(t_T)) \end{bmatrix} \quad \text{and} \quad \boldsymbol{\tau}_b = \begin{bmatrix} \boldsymbol{\tau}_d(t_0) \\ \vdots \\ \boldsymbol{\tau}_d(t_T) \end{bmatrix}. \tag{3.6}$$

The optimization-based computation of a persistent excitation trajectory that satisfies robot constraints generally leads to an infinite-dimensional problem. In the following section, the trajectory-finding problem is reduced to a finite-dimensional optimization problem by introducing parameterized robot trajectories.

3.2 Excitation trajectory

The quality of the robot base parameters, identified by solving the overdetermined system (3.5) applying the Least Squares approach

$$\hat{\boldsymbol{\varrho}}_b = \mathbf{X}^\dagger \boldsymbol{\tau}_b\,, \tag{3.7}$$

with the pseudoinverse matrix $\mathbf{X}^\dagger = \left(\mathbf{X}^\mathrm{T}\mathbf{X}\right)^{-1}\mathbf{X}^\mathrm{T}$, depends highly on the reference robot trajectory. The entries of $\mathbf{W}_b(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ consist of functions with a variety of periodic terms of the robot joints $\mathbf{q}$, multiplied by velocity $\dot{\mathbf{q}}$, acceleration $\ddot{\mathbf{q}}$, and gravity terms. This implies that for certain possible data points, some terms might vanish. For some joint configurations, the periodic terms or groups of periodic terms are zero, and if velocities and accelerations are zero, the corresponding terms are zero, too. On the other hand, to excite the gravity-dependent terms, a few positions without any robot movement are sufficient. A good selection of data points that satisfies a sufficient excitation of all relevant terms inside $\mathbf{W}_b(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ is provided by a persistently exciting reference trajectory. Finding such a trajectory corresponds to solving a nonlinear optimization problem of the form

$$\begin{aligned} \min_{\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}} \quad & f_q(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \\ \text{s.t.} \quad & \mathbf{q}_\mathrm{min} \leq \mathbf{q} \leq \mathbf{q}_\mathrm{max} \\ & |\dot{\mathbf{q}}| \leq \dot{\mathbf{q}}_\mathrm{max} \\ & |\ddot{\mathbf{q}}| \leq \ddot{\mathbf{q}}_\mathrm{max} \\ & \mathbf{g}_q(\mathbf{q}) \leq \mathbf{0}\,. \end{aligned} \tag{3.8}$$

There exist many ideas how to choose the objective function $f_q(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ [85, 86, 118], e. g., the D-optimality criterion [105], or the Hadamard inequality [119]. The most common is the condition number $\kappa(\mathbf{X}) = \mathrm{cond}(\mathbf{X}(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}))$, which describes how much the error in the observation matrix will influence the solution $\hat{\boldsymbol{\varrho}}_b$. The optimization must respect constraints due to maximal joint range, velocities and acceleration, including nonlinear constraints to prevent collisions with the robot's workspace and between robot links.

Discretizing and solving the optimization problem (3.8) returns unconnected points of configurations, which have to be connected to one trajectory. One possible way is to apply interpolation (curve fitting) to the points [94], but the resulting trajectory is not guaranteed to have the same objective function value. The standard way is to integrate trajectory models and their parameters into the optimization model. Trajectory models might be spline-variants [106] or variants of periodic excitation [120–123]. A popular model is the trajectory parametrization by Fourier series, which, for robot joint j, is given as follows

$$q_j(t) = \sum_{k=1}^{K} \left(\frac{a_{jk}}{k\omega_f} \sin(k\omega_f t) - \frac{b_{jk}}{k\omega_f} \cos(k\omega_f t) \right) + q_{j0} \, . \tag{3.9}$$

The fundamental angular frequency ω_f is chosen to be equal for all joints to ensure periodicity. a_{jk} and b_{jk} are the amplitudes for the $1 \dots K$ harmonic terms. q_{j0} is the offset and the trajectory evolves along time t. The terms for the velocities and accelerations are obtained by differentiating (3.9), i. e.,

$$\begin{aligned} \dot{q}_j(t) &= \sum_{k=1}^{K} \left(a_{jk} \cos(\omega_f k t) + b_{jk} \sin(\omega_f k t) \right), \\ \ddot{q}_j(t) &= \omega_f \sum_{k=1}^{K} \left(k b_{jk} \cos(\omega_f k t) - k a_{jk} \sin(\omega_f k t) \right). \end{aligned} \tag{3.10}$$

Substituting (3.9) and (3.10) in (3.8) for $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$, yields the optimization problem for generation of persistent excitation trajectory

$$\begin{aligned} \min_{\boldsymbol{\nu}} \quad & \kappa(\boldsymbol{\nu}) \\ \text{s.t.} \quad \mathbf{q}_{\min} \leq{} & \mathbf{D}_q \boldsymbol{\nu} \leq \mathbf{q}_{\max} \\ -\dot{\mathbf{q}}_{\max} \leq{} & \mathbf{D}_v \boldsymbol{\nu} \leq \dot{\mathbf{q}}_{\max} \\ -\ddot{\mathbf{q}}_{\max} \leq{} & \mathbf{D}_a \boldsymbol{\nu} \leq \ddot{\mathbf{q}}_{\max} \\ & \mathbf{D}_f \boldsymbol{\nu} = \mathbf{0} \\ & \mathbf{g}_q(\boldsymbol{\nu}) \leq \mathbf{0}, \end{aligned} \tag{3.11}$$

with the vector $\boldsymbol{\nu} \in \mathbb{R}^{n(1+2K)}$ containing all the Fourier coefficients and the offset variables q_{j0}, $\forall j \in \{1, \dots, n\}$. The matrices $\mathbf{D}_q, \mathbf{D}_v$, and $\mathbf{D}_a$ contain corresponding Fourier terms specific to the linear inequality constraints from (3.8). The linear equality constraints represent conditions on the robot joint configuration, velocities and accelerations at the initial t_0 and terminal time point t_T, given by

$$\begin{aligned} \mathbf{q}(t_0) &= \mathbf{q}(t_T) \\ \dot{\mathbf{q}}(t_0) &= \dot{\mathbf{q}}(t_T) = \mathbf{0} \\ \ddot{\mathbf{q}}(t_0) &= \ddot{\mathbf{q}}(t_T) = \mathbf{0}. \end{aligned} \tag{3.12}$$

These constraints are included in the optimization problem to use the trajectory multiple times and ensure reliable behavior. The nonlinear inequality constraints represent additional joint restrictions to avoid collisions with the workspace and between the robot's links.

Due to the periodic and nonconvex nature of the functions in the observation Matrix $\mathbf{X}$, any considerable objective function is expected to yield many local minima and has to be treated by a global optimization approach. Using multiple starting points is necessary to have a considerable chance of finding a global minimum. Therefore, genetic algorithms are appropriate, but the known implementations [107–109] lack the advantages of gradient-based approaches. To this end, a memetic algorithm is proposed to combine the advantages of both methods, see [76], [110]. Memetic algorithms extend the allegory of genetic algorithms to cultural terms. "Memes" instead of only "genes" are transferred between individuals and generations. The difference is that features of the individuals are not only recombined or slightly mutated but undergo substantial improvement procedures until the result is passed on to the next generation. The applied procedures use information about the problem structure and the already known solutions [111].

The presented memetic algorithm basically consists of two repeated steps. Out of a predefined maximum number of generations $\mathtt{G}_{\max}$, $\mathtt{G}_{\mathtt{R}}$ are randomly generated with a population $\mathtt{P}$ each, which represents the number of individuals $\boldsymbol{\nu}$. The optimized individuals $\boldsymbol{\nu}^*$ of the generations are computed using an NLP-solver. The remaining generations $\mathtt{G}_{\mathtt{MA}}$ are then obtained by applying a recombination procedure introducing a fitness function, followed by a local search method. By using recombinations, the best individuals are used to create candidates for new generations, aiming to combine all successful features in one individual. The local search method aims at finding candidates for new starting points in the neighborhood of an already known local minimum $\boldsymbol{\nu}^*$. Depending on whether a new best solution is found or not by applying the NLP-solver, the optimization parameters used for recombination and local search are tightened or relaxed, providing more randomness and less reliance on the made assumptions, see [76].

3.3 Parameter estimation

For the considered robot manipulator UR5, the system model linear in the dynamic parameters $\boldsymbol{\varrho}$ is obtained by applying the recursive Newton-Euler equations (3.2). Applying symbolic regrouping techniques leads the vector $\boldsymbol{\varrho}_b \in \mathbb{R}^{\underline{n}_\varrho}$ consisting of $\underline{n}_\varrho = 52$ base parameters, which are estimated using optimized persistent excitation trajectories. For the trajectory generation problem (3.11), the joint constraints are chosen to be

$$\begin{bmatrix} -2\pi \\ 0 \\ -0.7\pi \\ -2\pi \\ -2\pi \\ -2\pi \end{bmatrix} \leq \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix} \leq \begin{bmatrix} 2\pi \\ \pi \\ 0.7\pi \\ 2\pi \\ 2\pi \\ 2\pi \end{bmatrix}, \quad |\dot{\mathbf{q}}| \leq 3.2\,\mathrm{rad/s}, \quad \text{and} \quad |\ddot{\mathbf{q}}| \leq 25\,\mathrm{rad/s^2}. \tag{3.13}$$

The constraints on q_2 and q_3 are essentially part of the non-collision constraints to prevent the robot from colliding with the table and avoid collisions between robot links. For further non-collision constraints, simple rules are used instead of a full collision detection

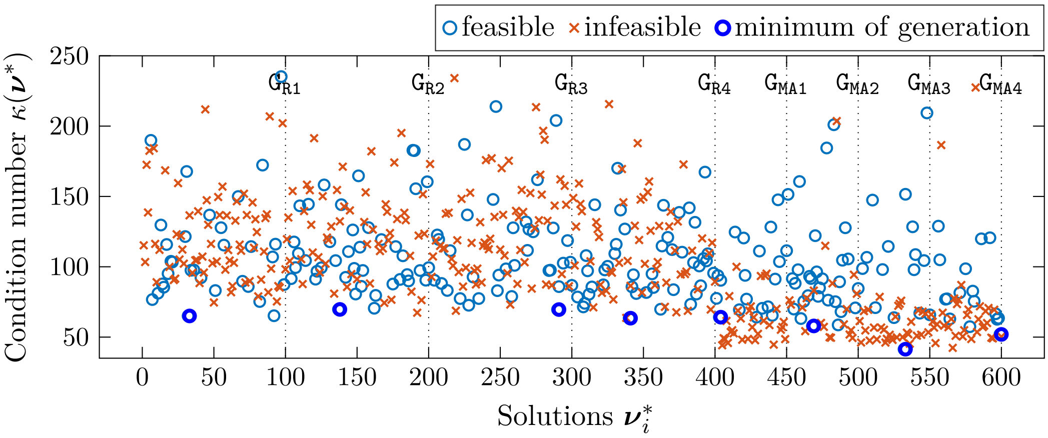


Figure 3.1: Solutions for the optimized persistent excitation trajectory. The vertical lines indicate the generation which the solution $\boldsymbol{\nu}_i^*$ belongs to. Note that the obtained minima utilizing random initialization are all worse than the ones obtained by the memetic algorithm. The best solution is obtained in the third MA-generation $\mathtt{G}_{\mathtt{MA3}}$. The fourth MA-generation $\mathtt{G}_{\mathtt{MA4}}$ did not yield any improvements, which caused the algorithm to stop. The solutions are widely scattered, and their quality depends on initial conditions. By finding better initial conditions for the NLP-solver, the MA is able to obtain better solutions. Numerical issues lead the NLP-solver to converge to infeasible solutions, which constitute a great proportion.

scheme to reduce computational effort by restricting the joints applying

$$\begin{aligned} |q_2 + \frac{\pi}{2} + q_3| &\leq \frac{3\pi}{4} \\ \sin(q_4)\ |\cos(q_5)| &\leq 0.15\,. \end{aligned} \tag{3.14}$$

To generate persistent excitation trajectories, the memetic algorithm is executed multiple times for a maximum number of generations $\mathtt{G}_{\max} = 10$ of which $\mathtt{G}_{\mathtt{R}} = 4$ are random generations with a generation size of $\mathtt{P} = 100$. The resulting nonlinear optimization problem is solved using the MATLAB's *fmincon* solver. As previously mentioned, for the cost function the L^2 condition number is used, i. e.,

$$\kappa(\boldsymbol{\nu}) = \operatorname{cond}(\mathbf{X}(\boldsymbol{\nu})) = \frac{\sigma_{\max}(\boldsymbol{\nu})}{\sigma_{\min}(\boldsymbol{\nu})}\,, \tag{3.15}$$

where $\sigma_{\max}(\boldsymbol{\nu})$ and $\sigma_{\min}(\boldsymbol{\nu})$ are the largest and smallest singular values of $\mathbf{X}(\boldsymbol{\nu})$. For the optimization, 20 time samples over a time period of $T = 10\,\mathrm{s}$ and a fundamental frequency $\omega_f = \pi/T$ are used. The optimization progress of the proposed method over generations of initial conditions is shown in Figure 3.1. The best solution using only randomly generated initial conditions is $\kappa(\boldsymbol{\nu}^*) = 67$. Applying the memetic algorithm, an improvement is achieved reducing the value of the cost function to $\kappa(\boldsymbol{\nu}^*) = 41$, obtained in the third MA-generation $\mathtt{G}_{\mathtt{AM3}}$.

The computed optimized position, velocity, and acceleration trajectories are forwarded to the local robot controller using ROS. The robot follows the planned trajectories sufficiently accurately, also resulting in only minor deviations from the condition number of the planned trajectory. The identification trajectory in joint space is given in Figure 3.2, with the resulting end-effector path in the operational space shown in Figure 3.3.

The robot's internal control unit takes measurements at a rate of 125 Hz, resulting in $n_s = 25000$ samples. Relevant measurements for the parameter estimation, which can be

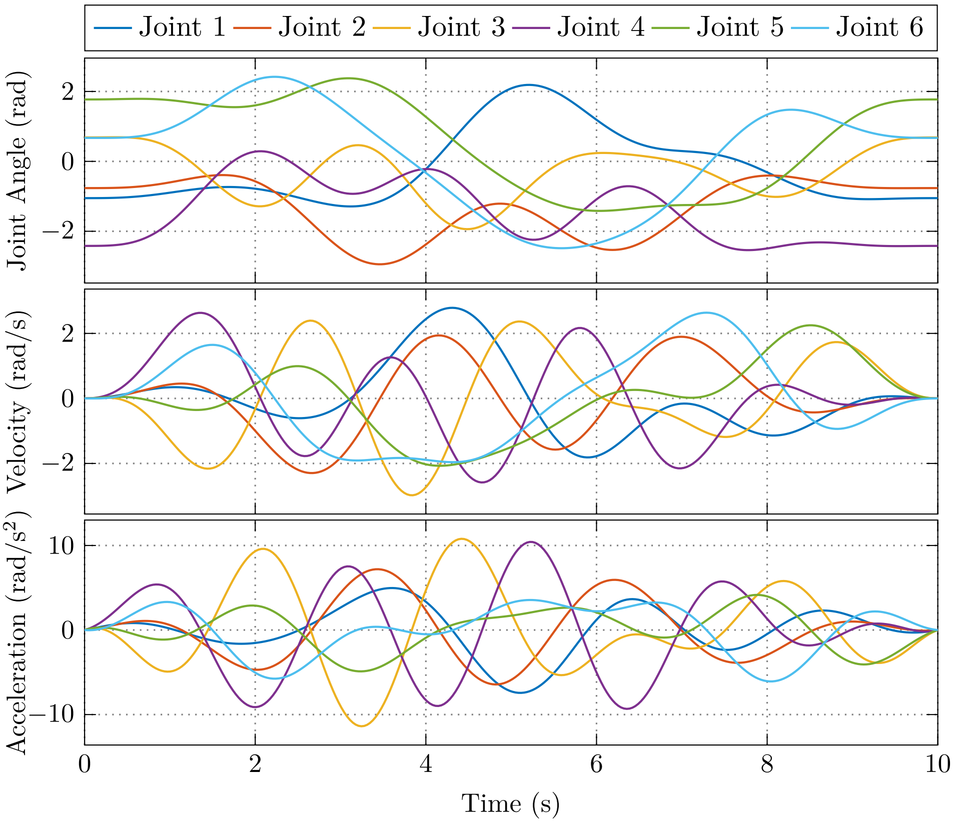


Figure 3.2: Joint position, velocity and acceleration of the optimized persistently exciting trajectory over one period.

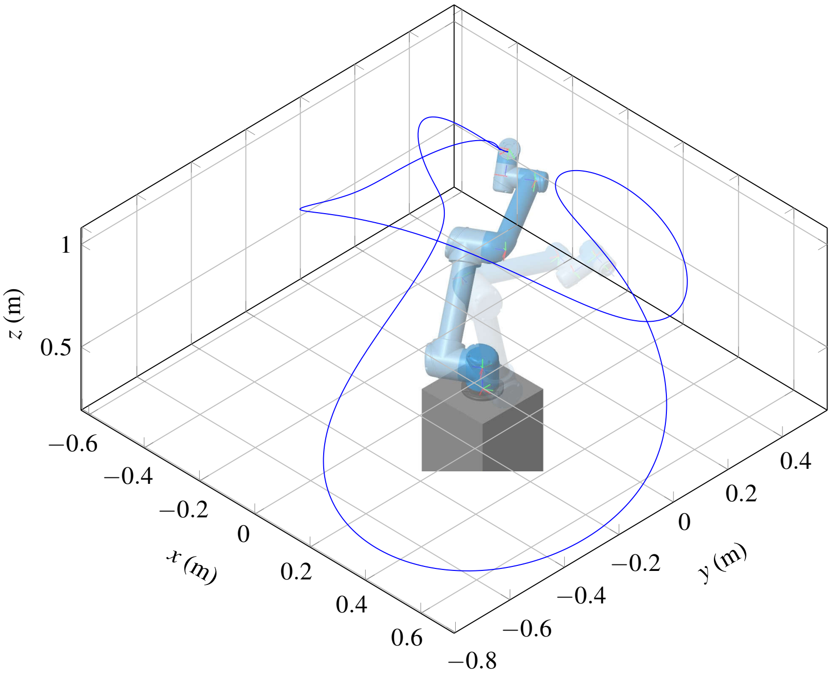


Figure 3.3: End-effector path over one period of the optimized persistently exciting trajectory.

obtained directly, are the actual joint positions, velocities, and electric motor currents. Due to measurement noise, the velocities and currents are filtered using a robust local regression to obtain smooth signals. Using the filtered velocities, the accelerations are calculated by applying the central difference derivation method.

With the smoothed actual currents i_{aj}, it is possible to estimate the actual joint torques τ_{aj}. The control unit also gives access to data of the internal controller, such as targeted motor currents i_{tj} and targeted joint torques τ_{tj}. For the relationship between currents and torques, a linear dependency is assumed. For static configurations, actual and target currents are equal, and it can be assumed that targeted and actual torques are also equal for such cases, i. e.,

$$i_{aj}\frac{\tau_{tj}}{i_{tj}} = i_{aj}\zeta_j = \tau_{aj}\,, \quad \forall j \in \{1, \ldots, n\}\,. \tag{3.16}$$

The robot configurations are chosen such that the static torque acting on the observed joint is close to its maximum. The current-torque constants ζ_j for every joint are then obtained by using average values of measurement data collected for multiple static configurations, yielding the motor constant vector

$$\boldsymbol{\zeta} = \begin{bmatrix} 11.4376 & 11.4376 & 11.4376 & 8.3238 & 8.3238 & 8.3238 \end{bmatrix}^{\mathrm{T}}, \tag{3.17}$$

which is used in the upcoming estimations to compute actual joint torques from measured actuator current. The considered robot manipulator has two types of actuators, bigger ones for the first three joints and three smaller ones for the small joints in the wrist area. This is also reflected in the obtained motor constant values.

With the collected joint measurement data for $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$, the observation matrix $\mathbf{X} \in \mathbb{R}^{n_s \times \underline{n}_\varrho}$ is generated, yielding, along with the measured torque vector $\boldsymbol{\tau}_b$, the overdetermined linear parameter estimation problem (3.5). To provide the opportunity to compare the obtained results with existing ones found in [124], the same statistical quality criteria is used by computing the standard deviation for the i-parameter

$$\sigma_i = \sqrt{(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}_{i,i}}\,, \tag{3.18}$$

and the normalized error of the estimated torques over the whole regression

$$\rho_N = \frac{1}{n_s}\sqrt{\boldsymbol{\rho}^{\mathrm{T}}\boldsymbol{\rho}}\,, \tag{3.19}$$

with the residual error vector $\boldsymbol{\rho} \in \mathbb{R}^{\underline{n}_\varrho}$. The normalized errors are also calculated for each single joint

$$\rho_{Nj} = \frac{1}{n_s}\sqrt{\boldsymbol{\rho}_j^{\mathrm{T}}\boldsymbol{\rho}_j}, \tag{3.20}$$

where $\boldsymbol{\rho}_j$ is the vector of residuals for the corresponding joint $j \in \{1, \ldots, n\}$.

The parameters are estimated using ordinary Least Square regression (OLS), which provides a good fit. Because of the chosen kinematic model, some of the base parameters $\boldsymbol{\varrho}_b$ were expected to be negative. However, negative values result in a non-symmetric or not positive definite mass matrix for the rotor inertia, which is physically not valid. The same result was found in [124]. Therefore, a constraint Least Squares (CLS) approach is also used, forcing positive rotor inertia values. Since the values for the rotor inertia are small anyway, the constraint fit deteriorates negligibly in quality. The estimated parameters obtained from both methods are validated by testing how the calculated models predict

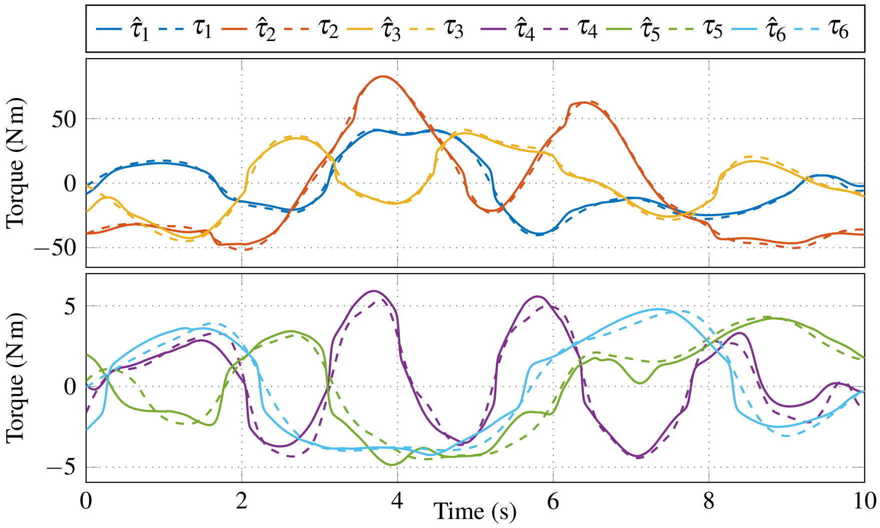
Table 3.1: Normalized estimation errors.

ρ_{Nj}	Est. OLS	Est. CLS	Val. OLS	Val. CLS
1	0.0122	0.0123	0.0102	0.0094
2	0.0143	0.0146	0.0172	0.0199
3	0.0134	0.0132	0.0151	0.0170
4	0.0034	0.0043	0.0050	0.0058
5	0.0037	0.0038	0.0068	0.0061
6	0.0024	0.0033	0.0025	0.0036
ρ_N	0.0040	0.0040	0.0044	0.0049

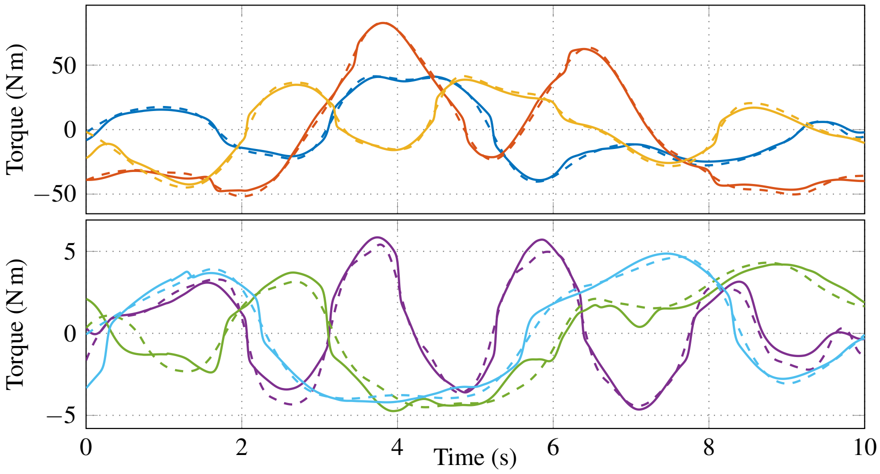
the torques for another excitation trajectory. An overview of the estimation ("Est.") and validation ("Val.") errors is shown in Table 3.1. It is shown that for both regression methods, the normalized errors are small when using the identification and validation trajectories. Referring once more to Table 3.2, it can be seen that the standard deviations σ_i for all parameters are sufficiently small. Figure 3.4 shows how close the predictions $\hat{\tau}_j$ are to the actual measurements of the joint torques τ_j. Note that the predictions using the OLS and CLS parameters are nearly indistinguishable. Therefore, and since we are mostly interested in validating the sufficiency of the achieved condition number, the torque results obtained when using the validation trajectory, shown in Figure 3.4c), are presented only for the parameters obtained by applying the OLS regression method.

The presented results show a very good matching of the computed torques using identified parameters with measured joint torques for both the identification and validation trajectory. Especially the predicted torques of the bigger robot joints $j \in \{1, 2, 3\}$ are very close to the measured ones, indicating a very good estimation of dynamic parameters related to the bigger robot links, including the actuators. Overall, the obtained normalized errors for estimation and validation shown in Table 3.1 are close to six, respectively, two times smaller compared to results presented in [124]. Since the approach presented in this work with the one from [124] uses similar solving techniques by applying OLS to solve the overdetermined parameter estimation problem, it is to be expected that the small values for the normalized torque prediction errors are related to the improved condition number of the optimized excitation trajectory. The standard deviations for the inertial parameters presented in Table 3.2 also suggest this. The exciting trajectory is well suited to excite the dynamics, granting minor standard deviations for the parameters, and the subsequent regressions yield small normalized errors for the predicted joint torques.

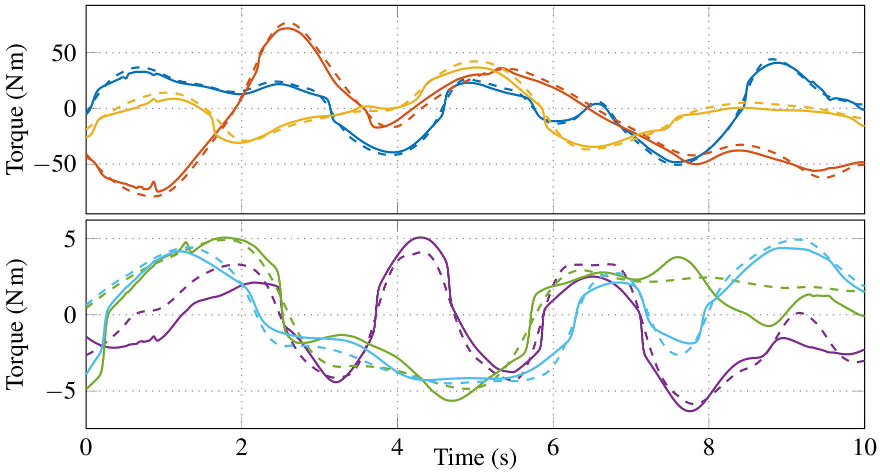
The condition number, chosen as an objective function for the trajectory optimization, depends on the selection of base parameters and is not invariant to reparameterization. To show the validity of the proposed methods in broader generality, invariant objective functions like D-optimality could be used. Further improvements can be made by introducing a nonlinear estimation model, which allows to consider more sophisticated friction models and also introducing additional constraints, for instance, to consider joint flexibility or enforce a positive definite mass matrix, see, e. g., [123], [110]. In addition, since there exist dynamic aspects of the robot model which can not be expressed linear in inertial parameters, an advantage of the nonlinear method can be that it breaks free from the limitations of representing the system dynamics linear in the base parameter vector. By applying global nonlinear optimization techniques, the physical robot parameters can be directly estimated without the need of regrouping. However, linear regression models reduce the computation effort and are also suitable for online parameter estimation.



(a) CLS on identification trajectory.



(b) OLS on identification trajectory.



(c) Validation trajectory with parameters obtained by OLS regression.

Figure 3.4: Predicted torques $\hat{\tau}_j$ compared to measured torques τ_j. (a) CLS regression method on the optimized trajectory, (b) uses the OLS regression method on the same trajectory. (c) Torque predictions using a validation trajectory with estimated parameters obtained by OLS regression. Note that (b) OLS and (a) CLS look very similar. The results are shown separately for the large (Joint 1 - Joint 3) and small (Joint 4 - Joint 6) joints. In all cases, the predicted joint torques are close to the measured values.

Table 3.2: Regrouped parameters and standard deviations.

$\boldsymbol{\varrho}_b$	Symbolic terms	σ_i
1	$(m_3+m_4+m_5+m_6)a_3^2+(m_4+m_5+m_6)d_4^2+2d_4P_{z_4}+(m_4+m_5+m_6)a_4^2+\Theta_{yy_3}+\Theta_{yy_4}+\Theta_{zz_1}+\Theta_{yy_2}+\Theta_{A1}$	0.0094
2	$(m_4+m_5+m_6)d_4^2+(m_4+m_5+m_6)a_4^2+\Theta_{xx_2}-\Theta_{yy_2}-(a_4^2+d_4^2)(m_4+m_5+m_6)-(m_3+m_4+m_5+m_6)a_3^2$	0.0077
3	Θ_{xy_2}	0.0042
4	$\Theta_{xz_2}+a_3(P_{z_3}+P_{z_4}+(m_4+m_5+m_6)d_4)$	0.0055
5	Θ_{yz_2}	0.0047
6	$\Theta_{zz_2}+(m_3+m_4+m_5+m_6)a_3^2+\Theta_{A2}$	0.0070
7	$(-m_3-m_4-m_5-m_6)a_3+P_{x_2}$	0.0019
8	P_{y_2}	0.0014
9	$(m_4+m_5+m_6)d_4^2+\Theta_{xx_3}-\Theta_{yy_3}-(a_4^2+d_4^2)(m_4+m_5+m_6)$	0.0070
10	Θ_{xy_3}	0.0035
11	$((m_4+m_5+m_6)d_4+P_{z_4})a_4+\Theta_{xz_3}$	0.0038
12	Θ_{yz_3}	0.0041
13	$\Theta_{zz_3}+(m_4+m_5+m_6)a_4^2$	0.0045
14	$(-m_4-m_5-m_6)a_4+P_{x_3}$	0.0013
15	P_{y_3}	0.0012
16	$(m_5+m_6)d_5^2+2P_{z_5}d_5+\Theta_{yy_5}+\Theta_{xx_4}-\Theta_{yy_4}$	0.0095
17	Θ_{xy_4}	0.0047
18	Θ_{xz_4}	0.0054
19	Θ_{yz_4}	0.0046
20	$\Theta_{zz_4}+\Theta_{yy_5}+2P_{z_5}d_5+(m_5+m_6)d_5^2$	0.0059
21	P_{x_4}	8.88e-4
22	$(-m_5-m_6)d_5+P_{y_4}-P_{z_5}$	0.0013
23	$\Theta_{xx_5}+\Theta_{yy_6}-\Theta_{yy_5}$	0.0048
24	Θ_{xy_5}	0.0020
25	Θ_{xz_5}	0.0030
26	Θ_{yz_5}	0.0045
27	$\Theta_{zz_5}+\Theta_{yy_6}$	0.0057
28	P_{x_5}	0.0013
29	$P_{z_6}+P_{y_5}$	0.0010
30	$\Theta_{xx_6}-\Theta_{yy_6}$	0.0028
31	Θ_{xy_6}	0.0013
32	Θ_{xz_6}	0.0019
33	Θ_{yz_6}	0.0013
34	Θ_{zz_6}	0.0032
35	P_{x_6}	9.15e-4
36	P_{y_6}	9.24e-4
37	F_{c1}	0.0131
38	F_{c2}	0.0139
39	F_{c3}	0.0139
40	F_{c4}	0.0137
41	F_{c5}	0.0127
42	F_{c6}	0.0142
43	F_{v1}	0.0124
44	F_{v2}	0.0127
45	F_{v3}	0.0104
46	F_{v4}	0.0099
47	F_{v5}	0.0113
48	F_{v6}	0.0102
49	Θ_{A3}	0.0035
50	Θ_{A4}	0.0030
51	Θ_{A5}	0.0059
52	Θ_{A6}	0.0047

CHAPTER 4

Task and Trajectory Planning

Robot manipulators cover in various industrial sectors a wide range of different applications, whether as stand-alone robotic cells or in combination with other machines, equipment, or classic feeding, assembly, and packaging stations. The variety of tasks combined with increasing demands for efficiency and productivity motivates the computation of optimal robot task processing sequences, with the main goal of achieving predefined performance criteria. One of these criteria is the robot cycle time, which defines the time it takes for a robot to complete one entire cycle of its assigned tasks. Although the tasks may be different and diverse, like picking, packing, and handling various goods and products or the automated execution of specific tasks such as spot welding, painting, quality inspection, etc., they have in common that for their execution mostly a point-to-point trajectory planning has to be performed, from the current to the desired robot position. The problem becomes challenging when multiple task points are considered. Motivated by the increase of efficiency and productivity, in such cases, the execution of optimal discrete sequences is typically of interest, this resulting in a difficult problem due to the excessively large number of involved combinatorial motion scenarios. In this case, optimization techniques can be used to optimize and coordinate the robot's workflow and balance the workload between robots in a multi-robot system.

The problems of robot task allocation and trajectory planning are mostly and traditionally treated separately and, in many cases, are also solved offline. In this chapter, optimization-based algorithms that address the problem of robot task allocation and trajectory planning are introduced. The problem formulation consists of a robot manipulator that has to pick up a set of objects and place them in a set of slots according to their classification. Therefore, task scheduling is performed first followed by point-to-point trajectory planning.

By introducing a two-layer hierarchical control structure, task scheduling is decoupled from trajectory planning, deploying two separate optimization problems: a binary one for task scheduling and a continuous optimization problem for trajectory planning. In a further step, both planning layers are integrated into a monolithic layer for online task and trajectory planning within the framework of a hybrid model predictive controller. The resulting recursive mixed-integer nonlinear programming (MINLP) problem is transformed into a relaxed mixed-integer quadratically constrained programming (MIQCP) problem, suitable to generate feasible robot trajectories online. The proposed algorithms are implemented and validated on an experimental setup using Robot Operating System (ROS) and a robot manipulator performing multiple pick-and-place tasks.

4.1 Problem formulation

Let $\mathcal{P}$ denote a set of multiple task points consisting of a subset of objects $\mathcal{O}$ and slots $\mathcal{S}$, i. e., $\mathcal{P} = \mathcal{O} \cup \mathcal{S}$. Objects and slots can belong to different classes $b \in \mathcal{B}$, with $\mathcal{O}_b \subseteq \mathcal{O}$ and $\mathcal{S}_b \subseteq \mathcal{S}$ denoting the respective objects and slots of class b. In order to be able to allocate each object to a slot, it is assumed that the number of objects and slots is the same, i. e., $|\mathcal{O}| = |\mathcal{S}| = N \in \mathbb{N}$. It is further assumed that for each class, the number of objects and slots is the same, and there can be at most N different classes, i. e., $|\mathcal{O}_b| = |\mathcal{S}_b|, \forall b \in \mathcal{B}$ where $B = |\mathcal{B}| \leq N$ and $\sum_{\forall b \in \mathcal{B}} |\mathcal{O}_b| = N$.

For the considered test cases, the objects and slots are distributed as shown in Figure 4.1, consisting of a set of six objects and slots of the same class and a set of six objects and slots of three different classes $\mathcal{B} = \{\text{"red"}, \text{"green"}, \text{"blue"}\}$. Each task point has, in addition to its position, also a desired orientation that defines the orientation of the robot gripper when picking up an object or placing it in a slot. If different classes exist, an object can be placed only into a slot of the same class. Therefore, task scheduling has to be performed to define a grasping and placing sequence, i. e., in which order the objects should be picked and in which slots they should be placed. Task scheduling is followed by an online trajectory planning to execute the assigned tasks.

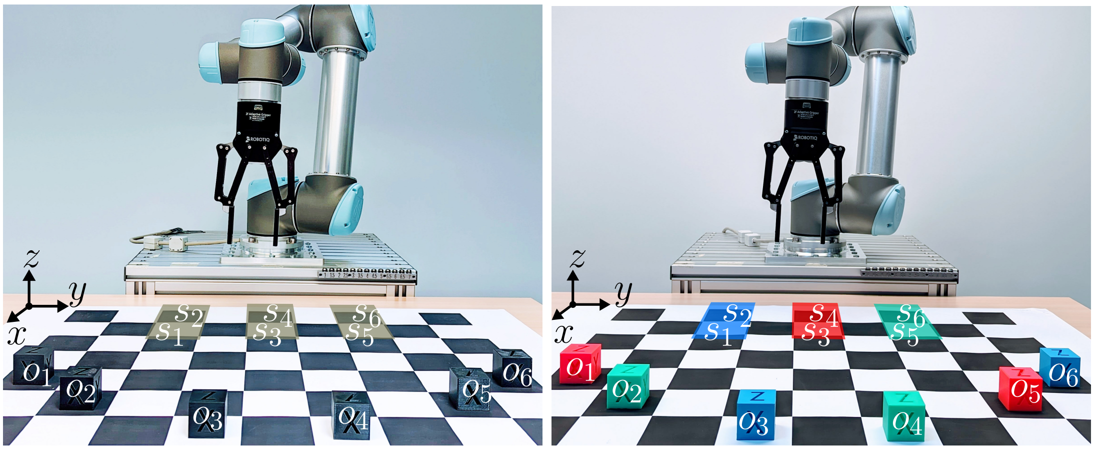


Figure 4.1: Experimental setup with a collaborative robot manipulator placing the objects $o \in \mathcal{O}$ in the slots $s \in \mathcal{S}$. The task points can belong to the same class (left) of three different classes $\mathcal{B} = \{\text{"red"}, \text{"green"}, \text{"blue"}\}$ (right). The figure shows the initial robot pose and the initial distribution of objects and slots.

In the following, two optimization-based algorithms that address the problem of robot task allocation and trajectory planning are introduced, and their performance is analyzed in terms of robot cycle time. The first algorithm represents a hierarchical control structure consisting of two-layer control policies. The upper layer, scheduling, performs task planning and generates a task sequence defining the order in which the tasks should be executed. The generated task sequence is provided as input to the lower layer, which plans robot trajectories online from the actual robot position to the assigned target point. This approach decouples task planning from trajectory planning, considering two separate optimization problems. The second algorithm represents a hybrid controller for inherent robot task and trajectory planning by integrating both planning layers into one monolithic layer in the form of a mixed-integer optimization problem.

4.2 Hierarchical control structure

The schematic illustration of the proposed hierarchical control structure is shown in Figure 4.2. The scheduling layer gets information about the position and orientation of the task points, and the actual robot position as denoted by the vector $\mathbf{q} = [\theta_1, \ldots, \theta_n]^\mathrm{T}$ of the joint angular positions. After computing an optimal task sequence, the scheduling layer provides information to the trajectory planning layer about the upcoming task by forwarding the corresponding final robot joint configuration $\mathbf{q}_f \in \mathbb{R}^n$. Using an algorithm based on model predictive control (MPC), the lower layer recursively performs online robot trajectory planning from the current position to the assigned target point. The computed optimal position $\mathbf{q}^*$, velocity $\dot{\mathbf{q}}^*$, and acceleration $\ddot{\mathbf{q}}^*$ trajectories are forwarded to ROS Control [125], which uses a Velocity Controller interface to finally send a reference velocity profile to the low-level robot controller. The implementation and experimental section will provide more information regarding ROS Controller. In the following, the scheduling algorithm and the time-optimal trajectory planning layer will be discussed in more detail.

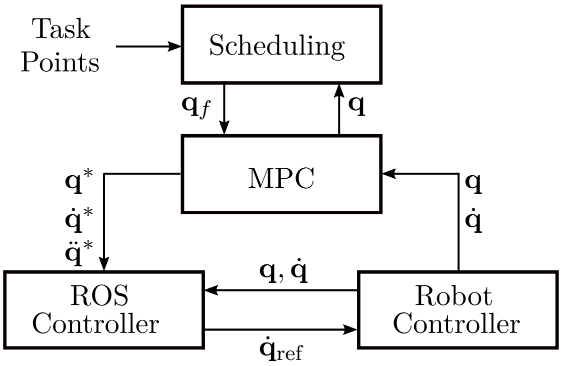


Figure 4.2: Hierarchical control structure.

4.2.1 Task scheduling

According to the problem formulation in Section 4.1, the goal of the scheduling layer is to compute an optimal picking sequence for the objects $o \in \mathcal{O}$ and filling sequence for the slots $s \in \mathcal{S}$. To solve the scheduling problem, two types of binary variables are introduced, $\delta_{oi} \in \{0, 1\}$, $\forall o \in \mathcal{O}, \forall i \in \{1, \ldots, |\mathcal{O}|\}$, for the objects and $\delta_{si} \in \{0, 1\}$, $\forall s \in \mathcal{S}$, $\forall i \in \{1, \ldots, |\mathcal{S}|\}$, for the slots. The iteration index i defines the object's gripping and slot's filling order, respectively. $\delta_{o1} = 1$ and $\delta_{s1} = 1$ would mean, for example, that object o is picked up first, and slot s is filled first. As a consequence, object o is placed in slot s. Given the same number of objects and slots, i. e., $|\mathcal{O}| = |\mathcal{S}| = N$, the binary decision variables can be written in matrix form as follows

$$\boldsymbol{\delta}_o = \begin{bmatrix} \delta_{o_1 1} & \cdots & \delta_{o_1 N} \\ \vdots & & \vdots \\ \delta_{o_N 1} & \cdots & \delta_{o_N N} \end{bmatrix}, \quad \boldsymbol{\delta}_s = \begin{bmatrix} \delta_{s_1 1} & \cdots & \delta_{s_1 N} \\ \vdots & & \vdots \\ \delta_{s_N 1} & \cdots & \delta_{s_N N} \end{bmatrix}. \tag{4.1}$$

The optimization variables related to the object assignment must satisfy the following equality constraints

$$\sum_{\forall o \in \mathcal{O}} \delta_{oi} = 1\,,\, \forall i \in \mathcal{N} = \{1, \ldots, N\}\,, \tag{4.2}$$

to ensure that exactly one object has to be selected at each picking iteration i, and the constraints

$$\sum_{\forall i \in \mathcal{N}} \delta_{oi} = 1\,,\, \forall o \in \mathcal{O} = \{o_1, \ldots, o_N\}\,, \tag{4.3}$$

to guarantee that across all picking iterations, each object has to be selected only once. Similarly, for the slot variables, the constraints

$$\sum_{\forall s \in \mathcal{S}} \delta_{si} = 1\,,\, \forall i \in \mathcal{N} = \{1, \ldots, N\} \tag{4.4}$$

and

$$\sum_{\forall i \in \mathcal{N}} \delta_{si} = 1\,,\, \forall s \in \mathcal{S} = \{s_1, \ldots, s_N\} \tag{4.5}$$

have to be applied to ensure that exactly one slot is being filled at each placing iteration, and across all iterations each slot has to be filled only once.

Since the objects and slots can belong to different classes, the equality constraints

$$\sum_{\forall o \in \mathcal{O}_b} \delta_{oi} = \sum_{\forall s \in \mathcal{S}_b} \delta_{si}\,, \quad \forall b \in \mathcal{B} \quad \text{and} \quad \forall i \in \mathcal{N} = \{1, \ldots, N\} \tag{4.6}$$

are applied to ensure that an object is placed only into a slot of the same class. If in the picking iteration i an object $o \in \mathcal{O}_b$ is selected, the following slot s at the placing iteration i has to belong to the same class, i. e., $s \in \mathcal{S}_b$. Note that for $|\mathcal{B}| = 1$, all objects and slots belong to the same class. In this case, constraints (4.6) correspond to (4.2)-(4.4)=0 and are trivially satisfied.

This work presents two methods for computing the optimal task execution sequence. The first one, minimum-distance scheduling, is based on the idea presented in [79] and minimizes the total Euclidean distance covered by the robot end-effector. The second method, minimum-time scheduling, is considering the scheduling problem in the robot's configuration space and aims to minimize the cycle time needed by the robot to execute all pick-and-place tasks.

4.2.1.1 Minimum-distance scheduling

The scheduling problem is considered in the three-dimensional workspace of the robot manipulator. The inputs to the scheduling layer are the position vectors ${}_0\mathbf{p}_o$ and ${}_0\mathbf{p}_s$ of all objects and slots and the position vector ${}_0\mathbf{p}_\mathrm{e}$ of the robot end-effector. With all the vectors given in Cartesian coordinates relative to the robot base, the Euclidean distances between the robot end-effector and all objects, and between all objects and all slots are computed as

$$\begin{aligned} d_o &= \|{}_0\mathbf{p}_o - {}_0\mathbf{p}_\mathrm{e}\|_2\,, \quad \forall o \in \mathcal{O} = \{o_1, \ldots, o_N\}\,, \\ d_{os} &= \|{}_0\mathbf{p}_o - {}_0\mathbf{p}_s\|_2\,, \quad \forall o \in \mathcal{O} = \{o_1, \ldots, o_N\}\,, \forall s \in \mathcal{S} = \{s_1, \ldots, s_N\}\,. \end{aligned} \tag{4.7}$$

The total path of the robot's end-effector while executing pick-and-place tasks can be divided into an initial movement from the actual robot position to the first object, the movement from the object to its assigned slot of the same class, and the movement from a slot to the next object. The minimum-distance scheduling problem results thus in an integer bilinear programming (IBLP) problem of the following form

$$\min_{\boldsymbol{\delta}_o,\boldsymbol{\delta}_s} \quad \underbrace{\sum_{\forall o\in\mathcal{O}} d_o\,\delta_{o1}}_{\text{Initial robot movement}} + \underbrace{\sum_{\forall b\in\mathcal{B}}\left(\sum_{\forall i\in\mathcal{N}}\sum_{\forall s\in\mathcal{S}_b}\sum_{\forall o\in\mathcal{O}_b} d_{os}\,\delta_{oi}\,\delta_{si}\right)}_{\text{Movement from the objects to the slots}} \tag{4.8}$$

$$+ \underbrace{\sum_{\forall i\in\mathcal{N}\setminus\{N-1\}}\ \sum_{\forall s\in\mathcal{S}}\sum_{\forall o\in\mathcal{O}} d_{os}\,\delta_{o\,i+1}\,\delta_{si}}_{\text{Movement from the slots to the objects}}$$

$$\text{s.t.} \quad \sum_{\forall o\in\mathcal{O}} \delta_{oi} = 1\,,\ \forall i\in\mathcal{N} = \{1,\dots,N\} \tag{4.8a}$$

$$\sum_{\forall i\in\mathcal{N}} \delta_{oi} = 1\,,\ \forall o\in\mathcal{O} = \{o_1,\dots,o_N\} \tag{4.8b}$$

$$\sum_{\forall s\in\mathcal{S}} \delta_{si} = 1\,,\ \forall i\in\mathcal{N} = \{1,\dots,N\} \tag{4.8c}$$

$$\sum_{\forall i\in\mathcal{N}} \delta_{si} = 1\,,\ \forall s\in\mathcal{S} = \{s_1,\dots,s_N\} \tag{4.8d}$$

$$\sum_{\forall o\in\mathcal{O}_b} \delta_{oi} = \sum_{\forall s\in\mathcal{S}_b}\delta_{si}\,,\quad \forall b\in\mathcal{B} \quad \text{and} \quad \forall i\in\mathcal{N} = \{1,\dots,N\}\,. \tag{4.8e}$$

The initial movement depends on which object is selected first and is expressed as a linear term of the overall cost function. The modeling of the movements from the objects to the slots and back to the objects involves both optimization variables resulting in bilinear terms of the cost function. While the robot can only move from a selected object to a slot of the same class, there are no restrictions for the vice versa motion as the robot can select any object after filling a slot with a corresponding object. The optimization is subject to the equality constraints (4.2)-(4.6) to ensure that all objects are placed in the slots while taking into account the class association.

Since task scheduling is performed in the Euclidean space, the orientations of the task points are not taken into account and thus do not influence the planning results. The shortest end-effector path does not necessarily lead to the shortest task execution time. Moreover, the trajectory planning is performed in the robot configuration space and does not result in a straight-line motion of the end-effector. At least not without additional constraints forcing such a motion, which is required only for specific tasks. Therefore, the following will transform the scheduling problem in the configuration space by considering multiple solutions of the inverse kinematics problem, resulting in a minimum-time scheduling algorithm.

4.2.1.2 Minimum-time scheduling

By transforming the scheduling problem from the workspace $\mathcal{W}$ into the configuration space $\mathcal{C}$ of the robot, it is possible to also incorporate the orientation of the task points in the scheduling model. Furthermore, it can be considered that due to the non-uniqueness of the inverse kinematics (IK) solutions, the robot can reach a certain task point in different joint configurations.

For each task point, $p \in \mathcal{P}$, multiple inverse kinematics solutions are computed, leading to c possible robot configurations per task point. Considering all possible joint angles for each configuration results, for the used robotic arm with six rotational joints and eight possible configurations ($c = 8$), in a set $\mathcal{C}_p$ of 512 possible joint configurations per task point p, see Section 2.1.2. Let $p, \bar{p} \in \mathcal{P}$ denote two task points and $\mathcal{C}_p, \mathcal{C}_{\bar{p}}$ the corresponding sets containing all feasible IK solutions. In order to find the fastest transition time between the two task points, the set

$$\mathcal{T}_{p\bar{p}}(\mathbf{q}_c, \mathbf{q}_{\bar{c}}) = \Bigg\{ t_{p\bar{p}}(\mathbf{q}_c, \mathbf{q}_{\bar{c}}) \,|\, t_{p\bar{p}} = \max \left| \frac{\theta_{j,c} - \theta_{j,\bar{c}}}{\dot{\theta}_{j,\max}} \right| , \forall \mathbf{q}_c = [\theta_{1,c}, \dots, \theta_{n,c}]^{\mathrm{T}} \in \mathcal{C}_p \,, \\ \forall \mathbf{q}_{\bar{c}} = [\theta_{1,\bar{c}}, \dots, \theta_{n,\bar{c}}]^{\mathrm{T}} \in \mathcal{C}_{\bar{p}} \Bigg\}, \quad (4.9)$$

is used, including the maximum transition times between all feasible robot configurations corresponding to the respective task points. With $\dot{\theta}_{j,\max}$ representing the maximum velocity of robot joint $j \in \{1, \dots, n\}$, equation (4.9) denotes that the slowest robot joint determines the transition time between two robot configurations. Since, for the considered robot manipulator, all joints have the same speed characteristics, the slowest robot joint is also the farthest from its target. The fastest transition time between the task points p and $\bar{p}$ can be found by

$$t^*_{p\bar{p}} = \min_{\mathbf{q}_c, \mathbf{q}_{\bar{c}}} \mathcal{T}_{p\bar{p}}(\mathbf{q}_c, \mathbf{q}_{\bar{c}}) \,, \quad (4.10)$$

representing the minimum time needed by the robot to move between the corresponding joint configurations $\mathbf{q}_p$ and $\mathbf{q}_{\bar{p}}$, with

$$\{\mathbf{q}_p, \mathbf{q}_{\bar{p}}\} = \arg\min_{\mathbf{q}_c, \mathbf{q}_{\bar{c}}} \mathcal{T}_{p\bar{p}}(\mathbf{q}_c, \mathbf{q}_{\bar{c}}) \,. \quad (4.11)$$

The goal of the minimum-time scheduling is to compute an optimal task execution sequence by minimizing the robot cycle time. Similar to the total Euclidean distance of the robot end-effector, the total cycle time can also be divided into three parts. The initial part expressed as a linear term of the cost function is modeling the transition time the robot needs to move from its current position to the first task points, i. e., to the objects. The second part of the cost function is modeling the transition time from the objects to the slots of the same class, followed by the transition time from the slots back to the objects, represented each as bilinear terms of the cost function. The cost function is minimized by imposing the equality constraints (4.2)-(4.6), resulting in the minimum-time IBLP problem of the form

$$\min_{\boldsymbol{\delta}_o, \boldsymbol{\delta}_s} \quad \sum_{\forall o \in \mathcal{O}} t^*_o \, \delta_{o1} + \sum_{\forall b \in \mathcal{B}} \left(\sum_{\forall i \in \mathcal{N}} \sum_{\forall s \in \mathcal{S}_b} \sum_{\forall o \in \mathcal{O}_b} t^*_{os} \, \delta_{oi} \, \delta_{si} \right) \quad (4.12)$$

$$+ \sum_{\forall i \in \mathcal{N} \setminus \{N-1\}} \sum_{\forall s \in \mathcal{S}} \sum_{\forall o \in \mathcal{O}} t^*_{os} \, \delta_{o\,i+1} \, \delta_{si}$$

$$\text{s.t.} \quad \sum_{\forall o \in \mathcal{O}} \delta_{oi} = 1 \,, \, \forall i \in \mathcal{N} = \{1, \dots, N\} \quad (4.12a)$$

$$\sum_{\forall i \in \mathcal{N}} \delta_{oi} = 1 \,, \, \forall o \in \mathcal{O} = \{o_1, \dots, o_N\} \quad (4.12b)$$

$$\sum_{\forall s \in \mathcal{S}} \delta_{si} = 1\,,\ \forall i \in \mathcal{N} = \{1, \ldots, N\} \tag{4.12c}$$

$$\sum_{\forall i \in \mathcal{N}} \delta_{si} = 1\,,\ \forall s \in \mathcal{S} = \{s_1, \ldots, s_N\} \tag{4.12d}$$

$$\sum_{\forall o \in \mathcal{O}_b} \delta_{oi} = \sum_{\forall s \in \mathcal{S}_b} \delta_{si}\,, \quad \forall b \in \mathcal{B} \quad \text{and} \quad \forall i \in \mathcal{N} = \{1, \ldots, N\}\,. \tag{4.12e}$$

Here,

$$t_o^* = \min_{\mathbf{q}_{c_o}} \mathcal{T}_o(\mathbf{q}_{c_o}, \mathbf{q}_0)\,, \quad \forall o \in \mathcal{O} \tag{4.13}$$

denote the minimum transition times from the actual robot position $\mathbf{q}_0$ to all objects $o \in \mathcal{O}$, and

$$t_{os}^* = \min_{\mathbf{q}_{c_o}, \mathbf{q}_{c_s}} \mathcal{T}_{os}(\mathbf{q}_{c_o}, \mathbf{q}_{c_s})\,, \quad \forall o \in \mathcal{O}, \forall s \in \mathcal{S}, \tag{4.14}$$

represent the minimum transition times between all objects $o \in \mathcal{O}$ and all slots $s \in \mathcal{S}$. The corresponding joint configurations $\mathbf{q}_o$ and $\mathbf{q}_s$, which lead to the minimum times (4.13) and (4.14), are obtained from (4.11).

4.2.2 Trajectory planning

As opposed to the standard applications of the MPC algorithms for path following or reference tracking, this work considers MPC-based time-optimal trajectory planning with terminal constraints. Let $\mathbf{x}(t) = [\mathbf{q}^{\mathrm{T}}(t), \dot{\mathbf{q}}^{\mathrm{T}}(t)]^{\mathrm{T}}$ denote the vector of robot joint positions and velocities. The goal is to generate feasible trajectories online so that the robot reaches its assigned target in the shortest possible time t_f i. e.,

$$\lim_{t \to t_f} ||\mathbf{x}(t) - \mathbf{x}_f|| = 0\,,\ t \in [t_0, t_f]. \tag{4.15}$$

Here, $\mathbf{x}_f = [\mathbf{q}_f^{\mathrm{T}}, \mathbf{0}^{\mathrm{T}}]^{\mathrm{T}}$ denotes the targeted final robot joints position and velocities, t_0 the current time point and t_f the final time to be minimized. The trajectory planning problem is thus formulated as a time-optimal optimization problem of the form

$$\min_{\mathbf{u}(t),\, t_f} \quad \int_{t_0}^{t_f} \mathrm{d}t \tag{4.16}$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{4.16a}$$

$$\underline{\mathbf{x}} \le \mathbf{x}(t) \le \bar{\mathbf{x}}\,, \quad t \in [t_0, t_f] \tag{4.16b}$$

$$\underline{\mathbf{u}} \le \mathbf{u}(t) \le \bar{\mathbf{u}}\,, \quad t \in [t_0, t_f] \tag{4.16c}$$

$$\mathbf{x}(t_f) = \mathbf{x}_f\,, \tag{4.16d}$$

where (4.16a) denotes the dynamic system representing the robot with the states $\mathbf{x}(t)$ and the inputs $\mathbf{u}(t)$, with their upper and lower bounds (4.16b) and (4.16c). The robot should reach the assigned target $\mathbf{x}_f$ at the end of the time horizon, as represented by the terminal constraint (4.16d). The final time t_f represents the a priori unknown time needed for the robot to reach the desired target point starting from its current position. This results in a free terminal time trajectory planning problem, which is transformed into a fixed terminal time problem by introducing the time scaling

$$\tau := \frac{t - t_0}{t_f - t_0}\,, \tag{4.17}$$

to map the time interval $t \in [t_0,\, t_f]$ to $\tau \in [0,\, 1]$.

4.2.2.1 Prediction model

For optimization-based online robot trajectory generation, the nonlinear robot dynamic model can be considered resulting in a nonlinear and non-convex optimization problem, which, in addition to position, velocity, and acceleration constraints, can account for torque limitations as well. However, considering that these optimization problems require a high computational effort and the update rate of time-optimal robot trajectories should be in the range of tens of milliseconds, one can reconsider using a simplified dynamic model. In addition, the robots usually offer a lower-level controller that can be used to control the robot via the corresponding interfaces by forwarding the joint position, velocity, or acceleration. The robot considered in this work can be controlled by using a position or velocity interface. Therefore, it is convenient to consider a chain of n double integrators

$$\ddot{\mathbf{q}}(t) = \mathbf{u}(t) \tag{4.18}$$

as a prediction model for online robot trajectory generation, by which also constraints in joint angular position $\mathbf{q}(t) \in \mathbb{R}^n$, velocity $\dot{\mathbf{q}}(t) \in \mathbb{R}^n$, and acceleration $\mathbf{u}(t) \in \mathbb{R}^n$ can be incorporated in the planning algorithm. The use of a simplified kinematic prediction model can also be motivated by assuming a local robot controller based on dynamic inversion (Computed Torque), cf., e. g., [87], given by

$$\boldsymbol{\tau}(t) = \hat{\mathbf{M}}(\mathbf{q}(t))\mathbf{u}(t) + \hat{\boldsymbol{\eta}}(\mathbf{q}(t), \dot{\mathbf{q}}(t)) = \mathbf{W}_b(\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))\hat{\boldsymbol{\varrho}}_b\,, \tag{4.19}$$

with the input vector $\mathbf{u}(t) \in \mathbb{R}^n$ and the estimated parameter vector $\hat{\boldsymbol{\varrho}}_b$. Substituting (4.19) in (2.76) leads the closed loop dynamics

$$\mathbf{u}(t) = \ddot{\mathbf{q}}(t) + \hat{\mathbf{M}}^{-1}(\mathbf{q}(t)) \left[\left(\mathbf{M}(\mathbf{q}(t)) - \hat{\mathbf{M}}(\mathbf{q}(t)) \right) \ddot{\mathbf{q}}(t) + \boldsymbol{\eta}(\mathbf{q}(t), \dot{\mathbf{q}}(t)) - \hat{\boldsymbol{\eta}}(\mathbf{q}(t), \dot{\mathbf{q}}(t)) \right]\,, \tag{4.20}$$

which can be expressed in the form

$$\mathbf{u}(t) = \ddot{\mathbf{q}}(t) + \hat{\mathbf{M}}^{-1}(\mathbf{q}(t))\mathbf{W}_b(\mathbf{q}(t), \dot{\mathbf{q}}(t), \ddot{\mathbf{q}}(t))\Delta\boldsymbol{\varrho}_b\,. \tag{4.21}$$

Here, $\Delta\boldsymbol{\varrho}_b$ represents the difference between real and estimated robot dynamic parameters. For small estimation errors, the closed loop dynamics can be approximated by $\mathbf{u}(t) \approx \ddot{\mathbf{q}}(t)$ yielding (4.18).

Using the state space vector $\mathbf{x}(t)$ and the input vector $\mathbf{u}(t)$ the dynamic model can be written as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0\,, \tag{4.22}$$

with

$$\mathbf{A} = \begin{bmatrix} \mathbf{0}^{n\times n} & \mathbf{I}^{n\times n} \\ \mathbf{0}^{n\times n} & \mathbf{0}^{n\times n} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0}^{n\times n} \\ \mathbf{I}^{n\times n} \end{bmatrix}, \quad \text{and} \quad \mathbf{x}(t) = \begin{bmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \end{bmatrix}. \tag{4.23}$$

The solution of the linear time-invariant system (4.22) by successively applying Picard's iteration method is given in the form

$$\mathbf{x}(t) = \boldsymbol{\Phi}(t - t_0)\mathbf{x}_0 + \int_{t_0}^{t_f} \boldsymbol{\Phi}(t - \tilde{t})\mathbf{B}\mathbf{u}(\tilde{t})\,\mathrm{d}\tilde{t}\,, \tag{4.24}$$

with the state-transition matrix

$$\boldsymbol{\Phi}(t) = \sum_{i=0}^{\infty} \mathbf{A}^i \frac{t^i}{i!} = \mathbf{I} + \mathbf{A}t\,. \tag{4.25}$$

With the time transformation $\tau = (t - t_0)/\Delta t_f$, the joint trajectories $\mathbf{q}(t) = \mathbf{q}(\tau)$, and the time derivatives

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}(t) = \frac{1}{\Delta t_f}\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbf{q}(\tau), \quad \frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{q}(t) = \mathbf{u}(t) = \frac{1}{(\Delta t_f)^2}\frac{\mathrm{d}^2}{\mathrm{d}\tau^2}\mathbf{q}_r(\tau)\,, \tag{4.26}$$

where $\Delta t_f = t_f - t_0$, the solution of the scaled linear time-invariant system for the time interval $\tau \in [k\Delta\tau,\ (k+1)\Delta\tau)$ can be written as

$$\mathbf{x}(k+1) = \mathbf{\Phi}(\Delta\tau)\mathbf{x}(k) + (\Delta t_f)^2 \int_{k\Delta\tau}^{(k+1)\Delta\tau} \mathbf{\Phi}((k+1)\Delta\tau - \tilde{\tau})\mathbf{B}\mathbf{u}(t_0 + \tilde{\tau}\Delta t_f)\,\mathrm{d}\tilde{\tau} \tag{4.27}$$

Applying the substitution $\tilde{t} = t_0 + \tilde{\tau}\Delta t_f$ yields

$$\mathbf{x}(k+1) = \mathbf{\Phi}(\Delta\tau)\mathbf{x}(k) + \Delta t_f \int_{t_0+k\Delta\tau\Delta t_f}^{t_0+(k+1)\Delta\tau\Delta t_f} \mathbf{\Phi}\left((k+1)\Delta\tau - \frac{(\tilde{t} - t_0)}{\Delta t_f}\right)\mathbf{B}\mathbf{u}(k)\,\mathrm{d}\tilde{t}\,, \tag{4.28}$$

with the input vector $\mathbf{u}(\tilde{t}) = \mathbf{u}(k)$ for $\tilde{t} \in [t_0 + k\Delta\tau\Delta t_f, t_0 + (k+1)\Delta\tau\Delta t_f)$. Finally, the discrete time dynamics in the scaled time τ is given by

$$\mathbf{x}(k+1) = (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}(k) + (\Delta t_f)^2\Delta\tau\left(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\right)\mathbf{B}\mathbf{u}(k), \tag{4.29}$$

with the identity matrix $\mathbf{I} \in \mathbb{R}^{2n\times 2n}$ and the input vector $\mathbf{u}(k)$, which is constant over the time interval $\tau \in [k\Delta\tau,\ (k+1)\Delta\tau)$, i. e., $t \in [t_k,\ t_k + \Delta\tau\Delta t_f)$, see [82]. In the following, without loss of generality, $t_0 = 0$, i. e., $\Delta t_f = t_f$, is assumed.

4.2.2.2 MPC-based trajectory planning

After transforming the trajectory planning problem (4.16) into a discrete fixed terminal time optimization problem, the resulting MPC algorithm for recurrent time-optimal robot trajectory planning can be written as a static optimization problem in the form

$$\min_{\mathbf{u}(\cdot),\, t_f} \quad t_f \tag{4.30}$$

$$\text{s.t.} \quad \mathbf{x}(j+1\,|\,k) = (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}(j\,|\,k) + \Delta\tau t_f^2\left(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\right)\mathbf{B}\mathbf{u}(j\,|\,k) \tag{4.30a}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}(j\,|\,k) \leq \bar{\mathbf{u}} \tag{4.30b}$$

$$\mathrm{diag}(\mathbf{I}, t_f\mathbf{I})\,\underline{\mathbf{x}} \leq \mathbf{x}(j+1\,|\,k) \leq \mathrm{diag}(\mathbf{I}, t_f\mathbf{I})\,\bar{\mathbf{x}} \tag{4.30c}$$

$$\mathbf{x}(0\,|\,k) = \mathrm{diag}(\mathbf{I}, t_f\mathbf{I})\,\mathbf{x}_0(k) \tag{4.30d}$$

$$\mathbf{x}(N_T\,|\,k) = \mathbf{x}_f(k)\,, \tag{4.30e}$$

with the iteration index $j \in \{0, \ldots, N_T - 1\}$, the prediction horizon N_T and the sampling time of the scaled discrete dynamics (4.30a) $\Delta\tau = 1/N_T$. In each optimization step k the robot trajectories are recursively planned from the actual measured state $\mathbf{x}_0^{\mathrm{T}}(k) = \left[\mathbf{q}_0^{\mathrm{T}}(k), \dot{\mathbf{q}}_0^{\mathrm{T}}(k)\right]$ to the desired state $\mathbf{x}_f^{\mathrm{T}}(k) = \left[\mathbf{q}_f^{\mathrm{T}}(k), \mathbf{0}^{\mathrm{T}}\right]$, as expressed by the initial and terminal constraints (4.30d) and (4.30e), respectively. The desired final robot configuration $\mathbf{q}_f$ is given by the scheduling layer, as schematically shown in Figure 4.2.

The input variables $\mathbf{u}(k)$, representing the joint accelerations, are limited to the maximum $\bar{\mathbf{u}}$ respective minimum values $\underline{\mathbf{u}}$. Similarly, the robot joint position and velocity are limited in (4.30c) to the maximum $\bar{\mathbf{x}}$ and minimum $\underline{\mathbf{x}}$ bounds, to satisfy the kinematic limitations of the robot and avoid self-collisions between the robot links and collisions with the static working environment. Note that the joint velocities $\dot{\mathbf{q}}(k)$ in (4.30c) and (4.30d) are scaled by t_f to transform them in the scaled time τ since the velocity upper and lower bounds and the measured robot state are given in time t. Time scaling for the terminal constraints is not applied since the desired final robot speed is zero. Due to time scaling, the $\Delta\tau$ intervals are mapped to increasingly tighter Δt-intervals as time t propagates, because computed optimal value t_f^* reduces with each optimization step. The closer the robots approach the target, the smaller the solution t_f^* gets leading to a finer prediction towards the end. In early iterations, the time resolution of the computed trajectories is lower and increases after each MPC iteration, resulting in sequentially finer robot trajectories. This has the advantage that, towards the end, when the robot approaches the target, the time resolution of the trajectories is higher, leading to more accurate planning required to reach the target points.

4.3 Hybrid control structure

As opposed to the two-layer hierarchical control structure, a hybrid controller represents a co-design approach which integrates both task and trajectory planning into a monolithic framework, as shown in Figure 4.3. The inherent incorporation of binary optimization variables for task scheduling and continuous optimization variables for trajectory planning leads to an MINLP problem for time-optimal robot tasks and trajectory planning. First, a single-step hybrid controller is presented, which consists of selecting the nearest possible task point out of a set of task points and planning the trajectory to it in the minimum possible time, see [77]. In a further step, the algorithm is extended in [78] to a multiple-step hybrid controller by considering the future task sequences in the scheduling since a task sequence resulting from successively selecting the nearest task points does not necessarily lead to the shortest overall cycle time. Binary constraints similar to (4.6) ensure that objects are assigned to slots of the same class.

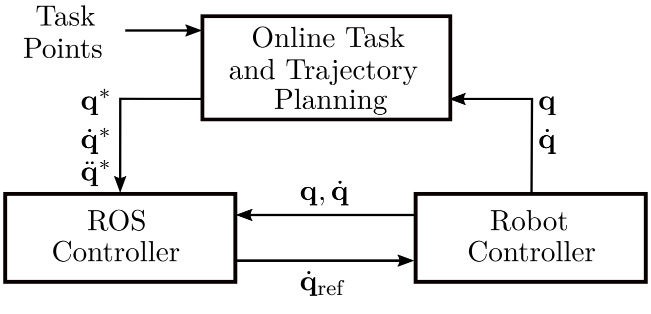


Figure 4.3: Hybrid controller for online task and trajectory planning.

In the problem formulation in Section 4.1, a set $\mathcal{P}$ of task points containing objects $\mathcal{O}$ and slots $\mathcal{S}$ is defined. Let $\mathcal{P}_p^b \subset \mathcal{P}$ denote a set of task points $p \in \mathcal{P}$ of class $b \in \mathcal{B}$. The task points $p \in \mathcal{P}_p^b$ can be either objects ($\mathcal{P}_p^b \subseteq \mathcal{O}_b$) or slots ($\mathcal{P}_p^b \subseteq \mathcal{S}_b$), i. e., $\mathcal{P}_p^b \cap \mathcal{P}_{\bar{p}}^b = \emptyset$, so the robot continuously alternates between objects and slots. Suppose the trajectory planning is performed from a previously reached slot to the objects. In that case, $\mathcal{P}_p^b \subseteq \mathcal{O}_b$

and $\mathcal{P}_{\bar{p}}^b \subseteq \mathcal{S}_b$. And vice versa, if the current trajectory is planned from the previously reached object, then $\mathcal{P}_p^b \subseteq \mathcal{S}_b$ and $\mathcal{P}_{\bar{p}}^b \subseteq \mathcal{O}_b$. Furthermore, the sets $\mathcal{P}_p$ and $\mathcal{P}_{\bar{p}}$ are defined as follows

$$\mathcal{P}_p = \bigcup_{\forall b \in \mathcal{B}} \mathcal{P}_p^b \quad \text{and} \quad \mathcal{P}_{\bar{p}} = \bigcup_{\forall b \in \mathcal{B}} \mathcal{P}_{\bar{p}}^b \,, \tag{4.31}$$

where $\mathcal{P}_p^b \cap \mathcal{P}_{\bar{p}}^b = \emptyset$ holds true. The binary parameter

$$\tilde{\delta}_{\mathcal{P}} = \begin{cases} 1, & \text{if } \mathcal{P}_p \subseteq \mathcal{S} \\ 0, & \text{if } \mathcal{P}_p \subseteq \mathcal{O} \end{cases} \tag{4.32}$$

is introduced to distinguish between the cases when the subsequent task point is a slot or an object.

4.3.1 Single-step hybrid controller

The final robot configuration leading to a specific task point $p \in \mathcal{P}_p$ is not known a priori and is subject to the optimization problem along with the online trajectory planning. Combining the robot task scheduling with the trajectory planning into a monolithic MPC framework results in a time optimal mixed-integer optimization problem as follows

$$\min_{\mathbf{u}(\cdot),\, \delta_1,\ldots,\delta_{|\mathcal{P}_p|},\, t_f} \quad J(t_f) = t_f \tag{4.33}$$

$$\text{s.t.} \quad \mathbf{x}(j+1\,|\,k) = (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}(j\,|\,k) + \Delta\tau t_f^2 \left(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\right)\mathbf{B}\mathbf{u}(j\,|\,k) \tag{4.33a}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}(j\,|\,k) \leq \bar{\mathbf{u}} \tag{4.33b}$$

$$\operatorname{diag}(\mathbf{I}, t_f\mathbf{I})\,\underline{\mathbf{x}} \leq \mathbf{x}(j+1\,|\,k) \leq \operatorname{diag}(\mathbf{I}, t_f\mathbf{I})\,\bar{\mathbf{x}} \tag{4.33c}$$

$$\mathbf{x}(0\,|\,k) = \operatorname{diag}(\mathbf{I}, t_f\mathbf{I})\,\mathbf{x}_0(k) \tag{4.33d}$$

$$\mathbf{x}(N_T\,|\,k) = \sum_{\forall p \in \mathcal{P}_p} \mathbf{x}_{f,p}(k)\delta_p \,, \quad \sum_{\forall p \in \mathcal{P}_p} \delta_p = 1 \tag{4.33e}$$

$$\tilde{\delta}_{\mathcal{P}} \sum_{\forall \bar{p} \in \mathcal{P}_{\bar{p}}^b} \delta_{\bar{p}}^* = \tilde{\delta}_{\mathcal{P}} \sum_{\forall p \in \mathcal{P}_p^b} \delta_p \,, \quad \forall b \in \mathcal{B} \,. \tag{4.33f}$$

Here, $j \in \{0, \ldots, N_T - 1\}$ is the iteration index, N_T the prediction horizon and $\Delta\tau = 1/N_T$ the sampling time of the scaled discrete dynamics (4.33a). The idea is to recursively schedule the robot's tasks and replan its joint trajectories in each optimization step k, from the actual measured state $\mathbf{x}_0^{\mathrm{T}}(k) = \left[\mathbf{q}_0^{\mathrm{T}}(k), \dot{\mathbf{q}}_0^{\mathrm{T}}(k)\right]$ to the desired state $\mathbf{x}_d^{\mathrm{T}}(k) = \left[\mathbf{q}_d^{\mathrm{T}}(k), \mathbf{0}^{\mathrm{T}}\right]$, as expressed by the initial and terminal constraints (4.33d) and (4.33e), respectively. Since the desired final robot configuration $\mathbf{q}_d(k)$ is subject to the optimization problem, it can be expressed as

$$\mathbf{q}_d(k) = \sum_{\forall p \in \mathcal{P}_p} \mathbf{q}_{f,p}(k)\delta_p \,, \tag{4.34}$$

with the possible final configurations $\mathbf{q}_{f,p}(k)$ computed using equation (4.11) $\forall p \in \mathcal{P}_p$. The discrete decision variables $\delta_p \in \{0, 1\}$ must satisfy the constraint

$$\sum_{\forall p \in \mathcal{P}_p} \delta_p = 1 \,, \tag{4.35}$$

to force the robot manipulator to go to exactly one of the $|\mathcal{P}_p|$ task points, leading to the constraints (4.33e) with $\mathbf{x}_{f,p}^{\mathrm{T}}(k) = \left[\mathbf{q}_{f,p}^{\mathrm{T}}(k),\, \mathbf{0}^{\mathrm{T}}\right]$. The constraints (4.33f) apply for $\mathcal{P}_p \subseteq \mathcal{S}$ implying that a previously selected object of class $b \in \mathcal{B}$, denoted by the optimal values $\delta_{\tilde{p}}^{*}$, can be placed only in a slot of the same class. If the robot moves from a slot to an object, i. e., $\tilde{\delta}_{\mathcal{P}} = 0$, the constraints are trivially satisfied. The optimization problem (4.33) is, similar to the trajectory planning problem (4.30), subject to additional inequality constraints (4.33b) and (4.33c), representing minimum / maximum bounds for the control inputs $\mathbf{u}$ and the state vector $\mathbf{x}$.

The mixed-integer optimization problem (4.33) is nonlinear and nonconvex due to the cubic equality constraints (4.33a) and thus falls into the class of NLMIP problems, which are computationally demanding and time-intensive to solve. However, the resulting NLMIP problem can be relaxed into an MIQCP problem based on the McCormick envelopes approach [126] by introducing the auxiliary variables

$$\varsigma = t_f^2\,, \quad \text{and} \quad \mathbf{w}(k) = \varsigma\mathbf{u}(k)\,, \tag{4.36}$$

with the upper and lower bounds

$$t_f \in [0,\, \bar{t}_f]\,, \quad \varsigma \in [0,\, \bar{t}_f^2]\,, \quad \text{and} \quad \mathbf{u}(k) \in [\underline{\mathbf{u}},\, \bar{\mathbf{u}}]. \tag{4.37}$$

The under- and overestimators of the auxiliary variables are represented by

$$\begin{gathered} 0 \le \varsigma \le t_f \bar{t}_f\,, \quad \varsigma \ge 2t_f\bar{t}_f - \bar{t}_f^2\,, \\ \varsigma\underline{\mathbf{u}} \le \mathbf{w}(k) \le \varsigma\bar{\mathbf{u}}\,, \\ \bar{t}_f^2\mathbf{u}(k) + \varsigma\bar{\mathbf{u}} - \bar{t}_f^2\bar{\mathbf{u}} \le \mathbf{w}(k) \le \bar{t}_f^2\mathbf{u}(k) + \varsigma\underline{\mathbf{u}} - \bar{t}_f^2\underline{\mathbf{u}}\,. \end{gathered} \tag{4.38}$$

With (4.36) and the substitutions

$$\mathbf{A}_d = \mathbf{I} + \Delta\tau\mathbf{A}\,, \quad \mathbf{B}_d = \Delta\tau\left(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\right)\mathbf{B}\,, \tag{4.39}$$

the dynamic equality constraints (4.33a) can be written as

$$\mathbf{x}(k+1) = \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d\mathbf{w}(k)\,, \tag{4.40}$$

with the new input $\mathbf{w}(k) \in \mathbb{R}^n$. Besides the additional linear inequality constraints (4.38), the convex relaxation also enforces the quadratic and bilinear constraints (4.36). The quadratic equality constraint can be written as a quadratic inequality constraint

$$t_f^2 \le \varsigma \le t_f^2\,. \tag{4.41}$$

The bilinear constraints

$$\varsigma u_j(k) - w_j(k) = 0 \quad j \in \{1, \cdots n\} \tag{4.42}$$

can be transformed by introducing the auxiliary variables

$$y_{1,j}(k) = \frac{1}{2}(\varsigma + u_j(k))\,, \quad y_{2,j}(k) = \frac{1}{2}(\varsigma - u_j(k)) \tag{4.43}$$

into the separable form

$$y_{1,j}^2(k) - y_{2,j}^2(k) - w_j(k) = 0\,, \quad j \in \{1, \cdots n\}\,, \tag{4.44}$$

with the lower and upper bounds

$$\begin{aligned} \frac{1}{2}\underline{u}_j \leq y_{1,j}(k) &\leq \frac{1}{2}(\bar{t}_f^2 + \bar{u}_j) \\ -\frac{1}{2}\bar{u}_j \leq y_{2,j}(k) &\leq \frac{1}{2}(\bar{t}_f^2 - \underline{u}_j)\,. \end{aligned} \tag{4.45}$$

The non-convex equality constraints (4.44) can then be written as

$$\boldsymbol{\xi}_j^{\mathrm{T}}(k)\mathbf{Q}_{\xi_j}\boldsymbol{\xi}_j(k) + \mathbf{c}_{w_j}^{\mathrm{T}}\boldsymbol{\xi}_j(k) = 0\,, \tag{4.46}$$

with $\boldsymbol{\xi}_j^{\mathrm{T}}(k) = [y_{1,j}(k),\, y_{2,j}(k),\, w_j(k)]$ and a symmetric positive semi-definite matrix $\mathbf{Q}_{\xi_j}$. Finally, they can be transformed into convex inequality constraints of the form

$$\begin{aligned} y_{1,j}^2(k) - y_{2,j}^2(k) - w_j(k) &\leq 0 \\ -y_{1,j}^2(k) + y_{2,j}^2(k) + w_j(k) &\leq 0\,, \end{aligned} \tag{4.47}$$

with $j \in \{1, \ldots n\}$.

For smooth robot motions, the cost function of the optimization problem (4.33) is extended to minimize the time derivative of the robot accelerations along the prediction horizon N_T, yielding

$$J(t_f, \mathbf{w}) = \alpha t_f + \beta \sum_{j=1}^{N_T-1} ||\mathbf{w}(j\,|\,k) - \mathbf{w}(j-1\,|\,k)||^2\,, \tag{4.48}$$

with the weights α and β.

To solve the MPC-based problem (4.33) in each optimization step k (sampling step t_k) all binary and continuous variables, including the auxiliary variables, are merged into a vector $\mathbf{z}(k)$ of optimization variables

$$\begin{aligned} \mathbf{z}(k) = [\mathbf{x}^{\mathrm{T}}(0|k) \quad \cdots \quad \mathbf{x}^{\mathrm{T}}(N_T|k) \quad \mathbf{u}^{\mathrm{T}}(0|k) \quad \cdots \quad \mathbf{u}^{\mathrm{T}}(N_T-1|k) \\ \mathbf{w}^{\mathrm{T}}(0|k) \quad \cdots \quad \mathbf{w}^{\mathrm{T}}(N_T-1|k) \quad \mathbf{y}_1^{\mathrm{T}}(0|k) \quad \cdots \quad \mathbf{y}_1^{\mathrm{T}}(N_T-1|k) \\ \mathbf{y}_2^{\mathrm{T}}(0|k) \quad \cdots \quad \mathbf{y}_2^{\mathrm{T}}(N_T-1|k) \quad \varsigma \quad t_f \quad \delta_1 \quad \cdots \quad \delta_{|\mathcal{P}_p|}]^{\mathrm{T}}\,, \end{aligned} \tag{4.49}$$

leading the MIQCP problem for minimum-time minimum-jerk robot task and trajectory planning

$$\begin{aligned} \min_{\mathbf{z}(k)} \quad & \mathbf{z}^{\mathrm{T}}(k)\mathbf{Q}\mathbf{z}(k) + \mathbf{c}^{\mathrm{T}}\mathbf{z}(k) \\ \text{s.t.} \quad & \mathbf{A}_{\mathrm{eq}}\mathbf{z}(k) = \mathbf{0} \\ & \mathbf{A}_{\mathrm{in}}\mathbf{z}(k) \leq \mathbf{a}(\bar{t}_f, \underline{\mathbf{u}}, \bar{\mathbf{u}}) \\ & \mathbf{z}^{\mathrm{T}}(k)\mathbf{Q}_{\mathtt{m}}\mathbf{z}(k) + \mathbf{c}_{\mathtt{m}}^{\mathrm{T}}\mathbf{z}(k) \leq 0\,, \end{aligned} \tag{4.50}$$

with $\mathtt{m} \in \{1, \ldots, 12N_T+2\}$. The quadratic cost function (4.48) is being minimized subject to the linear equality constraints (4.40), (4.33d), (4.33e), and (4.33f) the linear inequality constraints (4.33b), (4.33c), (4.38) and (4.45), and the quadratic inequality constraints (4.41) and (4.47).

4.3.2 Multiple-step hybrid controller

The single-step hybrid controller results in a robot task execution sequence where always the fastest possible task point is selected. However, subsequent execution of the fastest possible task does not necessarily result in a minimum robot cycle time. Therefore, an extension of the hybrid controller in the form of a multi-step controller is introduced that considers the future task sequence in the optimization problem.

To systematically model the overall transition time of the robot depending on the considered future task sequence, the two auxiliary parameters

$$\tilde{\delta}_{p\bar{p}} = \begin{cases} 1, & \text{if } \mathcal{P}_p \subseteq \mathcal{S} \vee \left(\mathcal{P}_p^b \subseteq \mathcal{O}_b \wedge p \in \mathcal{O}_b \wedge \bar{p} \in \mathcal{S}_b,\, \forall b \in \mathcal{B}\right) \\ 0, & \text{otherwise} \end{cases} \tag{4.51}$$

and

$$\tilde{\delta}_{\bar{p}p} = \begin{cases} 1, & \text{if } \mathcal{P}_p \subseteq \mathcal{O} \vee \left(\mathcal{P}_p^b \subseteq \mathcal{S}_b \wedge p \in \mathcal{S}_b \wedge \bar{p} \in \mathcal{O}_b,\, \forall b \in \mathcal{B}\right) \\ 0, & \text{otherwise} \end{cases} \tag{4.52}$$

are introduced. The robot cycle time is then modeled by three parts

$$J(t_f, \boldsymbol{\delta}) = t_f + \sum_{\forall p \in \mathcal{P}_p} \sum_{\forall \bar{p} \in \mathcal{P}_{\bar{p}}} \sum_{i=1}^{N_S} \tilde{\delta}_{p\bar{p}}\, t^*_{p\bar{p}}\, \delta_{pi}\, \delta_{\bar{p}i} + \sum_{\forall p \in \mathcal{P}_p} \sum_{\forall \bar{p} \in \mathcal{P}_{\bar{p}}} \sum_{i=1}^{N_S-1} \tilde{\delta}_{\bar{p}p}\, t^*_{p\bar{p}}\, \delta_{p\,i+1}\, \delta_{\bar{p}i}\,, \tag{4.53}$$

with the vector $\boldsymbol{\delta}$ containing all occurring binary optimization variables. The first part, t_f, is modeling the transition time to the first task points $\mathcal{P}_p$. The second part is modeling the time between $\mathcal{P}_p$ and $\mathcal{P}_{\bar{p}}$, where $\tilde{\delta}_{p\bar{p}}$ is used to exclude the times between objects and slots of different classes if the trajectory is planned from an object $\mathcal{P}_p \subseteq \mathcal{O}$ to a slot $\mathcal{P}_{\bar{p}} \subseteq \mathcal{S}$. The last part is modeling the robot motion from $\mathcal{P}_{\bar{p}}$ back to the task points $\mathcal{P}_p$. Here, the parameter $\tilde{\delta}_{\bar{p}p}$ is used to exclude transition times between objects and slots of different classes in case $\mathcal{P}_p \subseteq \mathcal{S}$ and $\mathcal{P}_{\bar{p}} \subseteq \mathcal{O}$. The scheduling horizon N_S indicates the extend to which the future planning sequence is included in the current task planning. The cost function is extended by the finite time derivatives of the robot joint accelerations for smooth trajectory planning. The optimization is further subject to trajectory planning and task scheduling constraints leading the hybrid controller

$$\begin{aligned} \min_{\mathbf{u}(\cdot),\, \boldsymbol{\delta},\, t_f} \quad & \alpha t_f + \alpha \sum_{\forall p \in \mathcal{P}_p} \sum_{\forall \bar{p} \in \mathcal{P}_{\bar{p}}} t^*_{p\bar{p}} \left(\tilde{\delta}_{p\bar{p}} \sum_{i=1}^{N_S} \delta_{pi}\, \delta_{\bar{p}i} + \tilde{\delta}_{\bar{p}p} \sum_{i=1}^{N_S-1} \delta_{p\,i+1}\, \delta_{\bar{p}i} \right) \\ & \quad + \beta \sum_{j=1}^{N_T-1} ||\mathbf{u}(j\,|\,k) - \mathbf{u}(j-1\,|\,k)||^2 \end{aligned} \tag{4.54}$$

$$\text{s.t.} \quad \mathbf{x}(j+1\,|\,k) = (\mathbf{I} + \Delta\tau \mathbf{A})\mathbf{x}(j\,|\,k) + \Delta\tau t_f^2 \left(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\right) \mathbf{B}\mathbf{u}(j\,|\,k) \tag{4.54a}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}(j\,|\,k) \leq \bar{\mathbf{u}} \tag{4.54b}$$

$$\mathrm{diag}(\mathbf{I}, t_f\mathbf{I})\, \underline{\mathbf{x}} \leq \mathbf{x}(j+1\,|\,k) \leq \mathrm{diag}(\mathbf{I}, t_f\mathbf{I})\, \bar{\mathbf{x}} \tag{4.54c}$$

$$\mathbf{x}(0\,|\,k) = \mathrm{diag}(\mathbf{I}, t_f\mathbf{I})\, \mathbf{x}_0(k) \tag{4.54d}$$

$$\mathbf{x}(N_T\,|\,k) = \sum_{\forall p \in \mathcal{P}_p} \mathbf{x}_{f,p}(k) \delta_{p1} \tag{4.54e}$$

$$\sum_{\forall p \in \mathcal{P}_p} \delta_{pi} = 1 \ \wedge \sum_{\forall \bar{p} \in \mathcal{P}_{\bar{p}}} \delta_{\bar{p}i} = 1, \quad \forall i \in \{1, \dots, N_S\} \tag{4.54f}$$

$$\sum_{i=1}^{N_S} \delta_{pi} = 1 \ \wedge \sum_{i=1}^{N_S} \delta_{\bar{p}i} = 1, \quad \forall p \in \mathcal{P}_p, \forall \bar{p} \in \mathcal{P}_{\bar{p}} \tag{4.54g}$$

$$\tilde{\delta}_{\mathcal{P}} \sum_{\forall \bar{p} \in \mathcal{P}^b_{\bar{p}}} \delta_{\bar{p}\, i-1} + (1 - \tilde{\delta}_{\mathcal{P}}) \sum_{\forall p \in \mathcal{P}^b_p} \delta_{pi} = \tilde{\delta}_{\mathcal{P}} \sum_{\forall p \in \mathcal{P}^b_p} \delta_{pi} + (1 - \tilde{\delta}_{\mathcal{P}}) \sum_{\forall \bar{p} \in \mathcal{P}^b_{\bar{p}}} \delta_{\bar{p}i}\,, \qquad \forall b \in \mathcal{B}, \forall i \in \{1, \dots, N_S\}\,. \tag{4.54h}$$

Since the task scheduling is subject to optimization, the robot recursively selects a task point p at each optimization step k and plans the trajectory to it. This is achieved by applying the terminal constraint (4.54e), with the binary variables δ_{p1}, $\sum_{\forall p \in \mathcal{P}_p} \delta_{p1} = 1$, and the final robot configurations $\mathbf{x}^{\mathrm{T}}_{f,p}(k) = \left[\mathbf{q}^{\mathrm{T}}_{f,p}(k),\, \mathbf{0}^{\mathrm{T}}\right]$, denoting that, out of $|\mathcal{P}_p|$ task points with the respective joint configurations $\mathbf{q}_{f,p}$, only one has to be selected and reached in the minimum possible time at the end of the prediction horizon N_T, i. e., at $\tau = 1$. The desired final robot configurations $\mathbf{q}_{f,p}$ satisfy equation (4.11), i. e., $\forall p \in \mathcal{O}$ and $\forall \bar{p} \in \mathcal{S}$

$$\{\mathbf{q}_{f,p}, \mathbf{q}_{f,\bar{p}}\} = \{\mathbf{q}_o, \mathbf{q}_s\} = \arg\min_{\mathbf{q}_{c_o}, \mathbf{q}_{c_s}} \mathcal{T}_{os}(\mathbf{q}_{c_o}, \mathbf{q}_{c_s})\,, \quad \forall o \in \mathcal{O}, \forall s \in \mathcal{S}\,. \tag{4.55}$$

For $N_S \geq 1$, the future task sequence is also considered in the optimization by including the binary variables $\delta_{pi}, \forall p \in \mathcal{P}_p$ and $\forall i \in \{2, \dots, N_S\}$, and $\delta_{\bar{p}i}$, $\forall \bar{p} \in \mathcal{P}_{\bar{p}}$ and $\forall i \in \{1, \dots, N_S\}$. Thus, the binary constraints (4.54f) and (4.54g) also apply. They imply that, at each iteration along the scheduling horizon, exactly one task point should be selected and that each task point has to be selected only once across all iterations. From the predicted future task sequence only the first one, i. e., δ^*_{p1}, is used for trajectory planning.

The binary equality constraints (4.54h) are introduced to ensure that an object $o \in \mathcal{O}_b$ of class b can be only placed into a slot $s \in \mathcal{S}_b$ of the same class. For $\mathcal{P}_p \subseteq \mathcal{S}$, it should be considered that out of $|\mathcal{P}_p|$ slots the one should be selected next which belongs to the same class as the previously picked object δ^*_{p1}. The same applies also for the future slot sequence along the scheduling horizon N_S. With $\delta_{\bar{p}0} = \delta^*_{p1}$ denoting the previous optimal task planning solution, the constraints (4.54h) obtained for $\tilde{\delta}_{\mathcal{P}} = 1$, see (4.32), read

$$\sum_{\forall \bar{p} \in \mathcal{P}^b_{\bar{p}}} \delta_{\bar{p}\, i-1} = \sum_{\forall p \in \mathcal{P}^b_p} \delta_{pi}\,, \quad \forall b \in \mathcal{B}, \forall i \in \{1, \dots, N_S\}\,. \tag{4.56}$$

Since the robot can move from a previously filled slot to any object, the previous scheduling results are not considered if the subsequent task is an object, i. e., $\tilde{\delta}_{\mathcal{P}} = 0$. In this case, the constraints

$$\sum_{\forall p \in \mathcal{P}^b_p} \delta_{pi} = \sum_{\forall \bar{p} \in \mathcal{P}^b_{\bar{p}}} \delta_{\bar{p}i}\,, \quad \forall b \in \mathcal{B}, \forall i \in \{1, \dots, N_S\}\,. \tag{4.57}$$

are applied to ensure that the object-slot assignment along the scheduling horizon is within the same class $b \in \mathcal{B}$.

The convexification techniques presented in Section 4.3.1 are again applied to transform the MINLP problem (4.54) into a relaxed MIQCP problem, suitable for online task and trajectory planning of robot manipulators. In the following section, some implementation details followed by experimental results will be presented to demonstrate, validate and compare the performance of the proposed algorithms.

4.4 Implementation

The experimental setup consists of a UR5 collaborative robotic arm from Universal Robots equipped with gripper, and a Host-PC where the planning algorithms are implemented. The planned time-optimal trajectories are sent to the robot controllers via TCP / IP (Transmission Control Protocol / Internet Protocol) communication protocol using ROS melodic [127]. The resulting system and implementation architecture is shown in Figure 4.4. The computed minimum-time joint trajectories consisting of position $\mathbf{q}^*(\cdot) = [\mathbf{q}^*(0), \ldots, \mathbf{q}^*(N_T)]$, velocity $\dot{\mathbf{q}}^*(\cdot) = [\dot{\mathbf{q}}^*(0), \ldots, \dot{\mathbf{q}}^*(N_T)]$, and acceleration $\mathbf{u}^*(\cdot) = [\mathbf{u}^*(0), \ldots, \mathbf{u}^*(N_T)]$ data points are forwarded, along with a time vector $t^* = [0, \ldots, t_f^*]$, to ROS Control to finally generate reference points for the low-level robot controller. In this work, *joint velocity interface* in conjunction with *joint velocity controller* is used by ROS Control to send and receive control commands to the underlying robot controller. More specifically, *joint trajectory controller* is used for executing joint-space trajectories on a group of joints. Since the trajectories are planned iteratively, the *action interface* is used, which grants the possibility of replacing trajectories at each optimization step when the new ones are available.

Given a set of position, velocity, and acceleration data points to be reached at specific time instants, the controller performs a spline interpolation using quintic polynomials. As shown in Figure 4.4 the used velocity controller provides a closed-loop control structure using a proportional integral derivative (PID) controller in combination with a feedforward term, given by

$$\dot{\mathbf{q}}_{\text{ref}}(t) = \mathbf{K}_F \dot{\mathbf{q}}_d(t) + \mathbf{K}_P \mathbf{e}_q(t) + \mathbf{K}_I \int_0^t \mathbf{e}_q\left(\tilde{t}\right) \mathrm{d}\tilde{t} + \mathbf{K}_D \dot{\mathbf{e}}_q(t), \tag{4.58}$$

with the diagonal matrices $\mathbf{K}_P$ for the proportional gain, $\mathbf{K}_I$ the integral gain, $\mathbf{K}_D$ the derivative gain and $\mathbf{K}_F$ the feedforward gain. The velocity controller generates a reference velocity profile which is forwarded to the underlying robot controller (UR controller) via a velocity interface.

To realize the feedback loops needed for MPC and the ROS velocity controller, fast reading of the current position and velocity of the robot joints is crucial. Measuring and reading the necessary feedback data requires time, depending on the update frequency

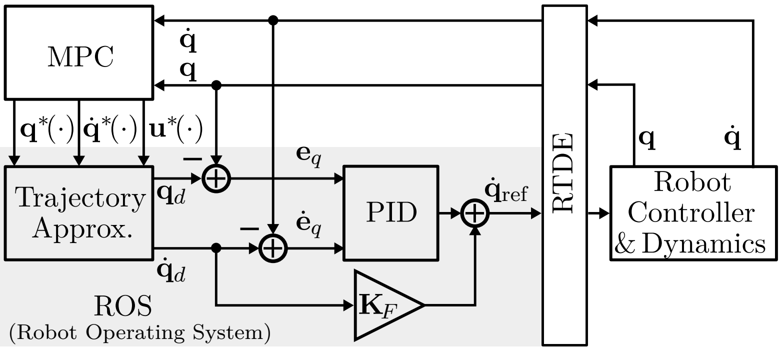


Figure 4.4: Schematic illustration of the hardware implementation using ROS. The generated minimum-time trajectories $\mathbf{q}^*(\cdot) = [\mathbf{q}^*(0), \ldots, \mathbf{q}^*(N_T)]$, $\dot{\mathbf{q}}^*(\cdot) = [\dot{\mathbf{q}}^*(0), \ldots, \dot{\mathbf{q}}^*(N_T)]$, and $\mathbf{u}^*(\cdot) = [\mathbf{u}^*(0), \ldots, \mathbf{u}^*(N_T)]$ are approximated by quintic polynomials and forwarded to the low-level robot controller via a PID-feedback controller in combination with a feedforward term using ROS and Velocity Control Interface.

of the UR controller and ROS itself. The control and update frequency of the robot controller is 125 Hz, providing sufficient performance for online robot trajectory planning. However, when using ROS to access the measurement data via ROS messages, the performance of the reading time is poor, causing time delays between 15 ms – 60 ms per robot, as shown in Figure 4.5. To overcome this problem, a direct synchronization for data exchange between the UR controller and the Host-PC is established using Real-Time Data Exchange (RTDE) interface over a standard TCP / IP connection, without breaking any real-time properties of the UR controller [128]. Although the ROS driver uses the RTDE interface for data exchange, the processing overhead when using ROS messages causes significant delays far greater than the delays resulting from RTDE, see Figure 4.5.

The parameters of the feedforward PID controller are tuned using sinusoidal reference trajectories following some basic guidelines described in [87]. The sinusoidal reference trajectories were created considering the position and velocity limits of the robots in order not to exceed physical limitations and to avoid self-collision between the links. Therefore, optimization-based trajectories can also be considered, similar to those introduced in [76] for robot dynamic parameter estimation. To evaluate the tracking performance of the tuned parameters shown in Table 4.1, pick-and-place tasks are performed to track trajectories other than those used during the tuning process. As can be seen in Figure 4.6, the chosen parameters result in a very good tracking performance with nonsignificant tracking errors in the range of single-digit milli-radian for the position and double-digit milli-radian for the velocity. The position error is slightly smaller than the velocity error, given the fact that accurate positioning of the robots is crucial for the task execution.

The algorithms are implemented in PYTHON on a standard Host-PC with Intel Core i7-8700 Processor and a 3.20 GHz clock rate using Ubuntu 18.04.6 LTS with a real-time kernel. To model the optimization problems the CasADi framework [129] is used. The IBLP optimization problems (4.8) and (4.12) for task scheduling are solved with the GUROBI solver. The GUROBI solver is also applied to solve the mixed-integer optimization problems (4.33) (4.54), after applying convexification techniques and transform them into MIQCP problems of the form (4.50). The nonlinear optimization problem (4.30) for minimum-time robot trajectory planning is solved by applying the Interior Point OPTimizer (IPOPT) [130].

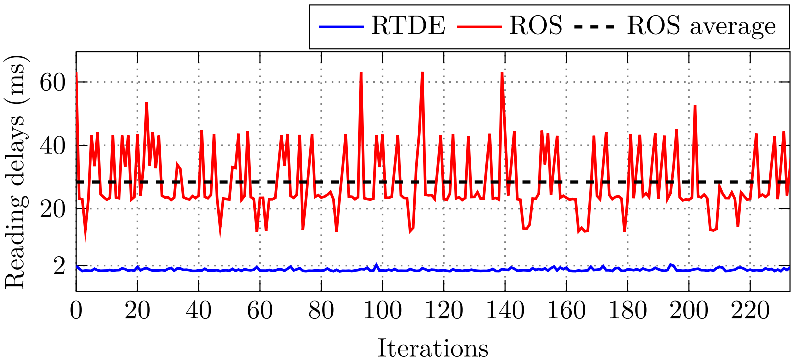


Figure 4.5: Direct comparison of reading time delays coming from RTDE and ROS while performing pick-and-place tasks. Evidently ROS delays far exceed the ones coming from RTDE. The mean value of ROS delay is approximately 28 ms as represented with the black dashed line.

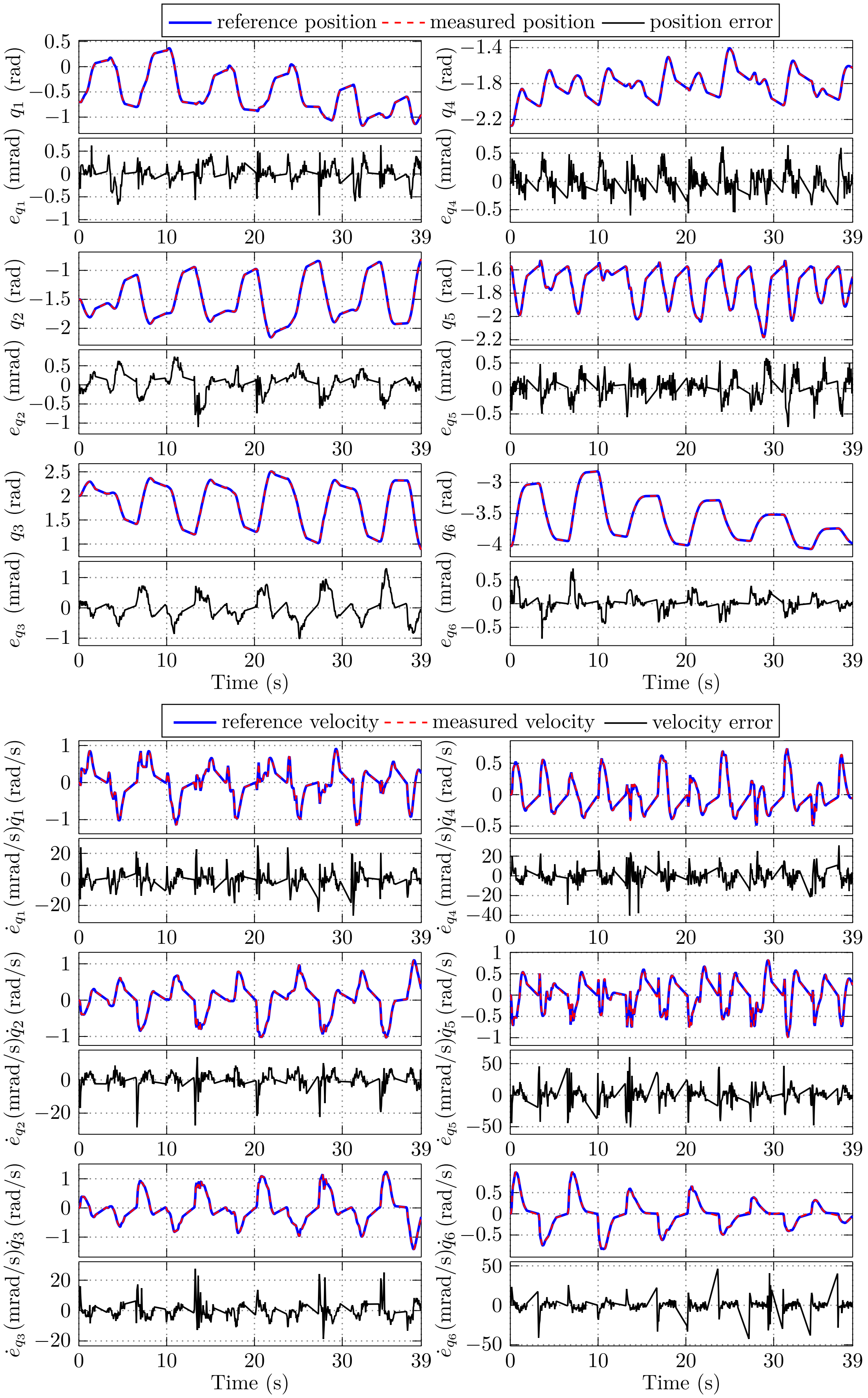


Figure 4.6: Comparison between the target and the measured position and velocity of the robot joints and depiction of the resulting tracking error.

Table 4.1: Tuned PID parameters.

Description	Parameter	Value
Proportional gain	$\mathbf{K}_P$	$5\mathbf{I}$
Integral gain	$\mathbf{K}_I$	$0.05\mathbf{I}$
Derivative gain	$\mathbf{K}_D$	$0.05\mathbf{I}$
Feedforward gain	$\mathbf{K}_F$	$\mathbf{I}$

4.5 Experimental results

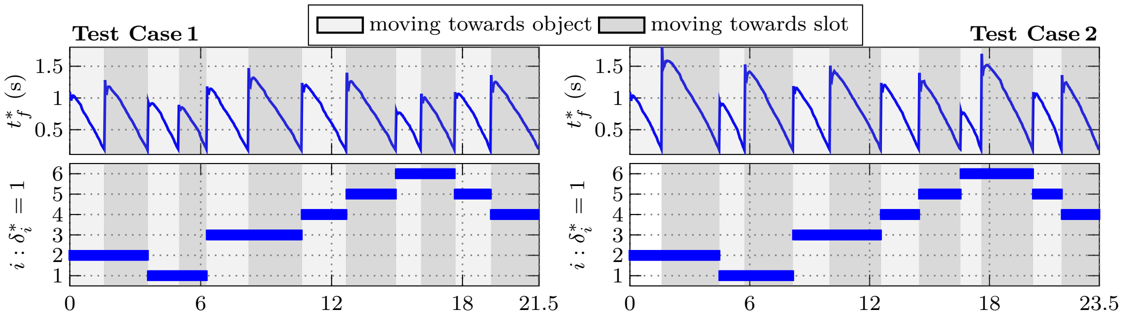
The experiment setup used to demonstrate the functionality and evaluate the performance of the proposed approaches is presented in Section 4.1. It involves a single robot manipulator, six objects $\mathcal{O} = \{o_1, \ldots, o_6\}$, and six slots $\mathcal{S} = \{s_1, \ldots, s_6\}$. The overall task is to place the objects in the slots in the shortest possible time. Therefore, three test cases are considered. In all cases, the distribution of the objects $o \in \mathcal{O}$ and slots $s \in \mathcal{S}$, as well as the initial position of the robot are chosen as shown in Figure 4.1. In Test Case 1 and 2 all objects and slots belong to the same class $\mathcal{B} = \{\text{"black"}\}$. The orientation of the gripper when picking up the objects and placing them in the slots in Test Case 1 is chosen to be the same for all objects and slots. It corresponds to the initial end-effector orientation shown in Figure 4.1. In Test Case 2, on the other hand, the orientation of the end-effector when grasping objects o_1, o_3, and o_5, and filling slots s_2, s_4, and s_6 is rotated by 90° about the z-axis compared to its initial orientation. In Test Case 3 the task points belong to three different classes $\mathcal{B} = \{\text{"red"}, \text{"green"}, \text{"blue"}\}$. Compared to its initial orientation, the robot gripper when picking object $\{o_1, o_2, o_3\}$ and filling slots $\{s_1, s_3, s_5\}$ is rotated by 90° about the z-axis.

Test Case 1 and Test Case 2

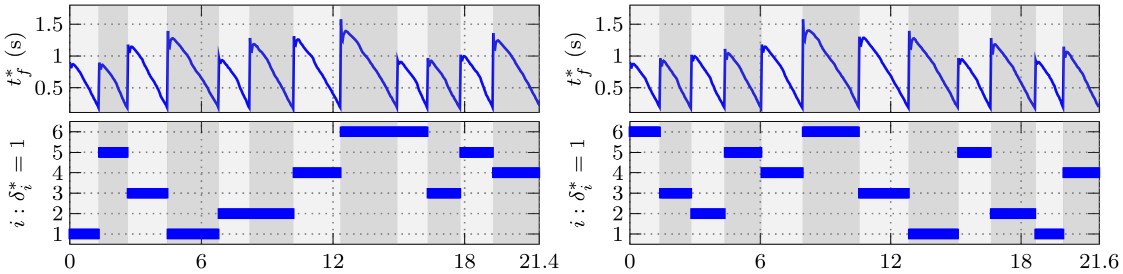
The computed optimal scheduling sequences and the resulting optimal final times t_f^* for both test cases are shown in Figure 4.7. Figure 4.7a) and Figure 4.7b) present the experimental results of the hierarchical control structure for minimum-distance and minimum-time scheduling. Figure 4.7c), Figure 4.7d), and Figure 4.7e) display the results of the hybrid MPC algorithm for one (1S-HMPC), two (2S-HMPC) and three (3S-HMPC) scheduling horizons, i. e., $N_S \in \{0, 1, 2\}$. Note that, the multi-step hybrid controller (4.54) is for $N_S = 0$ equal to the optimization problem (4.33), i. e., to the single-step hybrid controller. The prediction horizon for the trajectory planning is chosen $N_T = 10$; the robot joint velocity limits $\pm\pi/2\,\text{rad/s}$, and the joint acceleration limits $\pm 2\pi\,\text{rad/s}^2$.

The computed minimum final time t_f^* is shown for each test case and optimization algorithm separately. The time periods denoting the robot's motion to the respective objects and slots are denoted by light and dark gray areas, respectively. A peak in the optimal time course represents the beginning of the trajectory planning to lead the robot from its current position to an object or a slot. Accordingly, the minimum time is maximal at the beginning and decreases continuously as the robot approaches its target with each sampling step (MPC iteration). As can be seen, the optimal time course for the considered test cases has twelve maximum and twelve minimum peaks, representing each the beginning and the end of task execution (picking or placing an object), respectively. The higher the maximum peak, the longer the robot needs to finish the selected task.

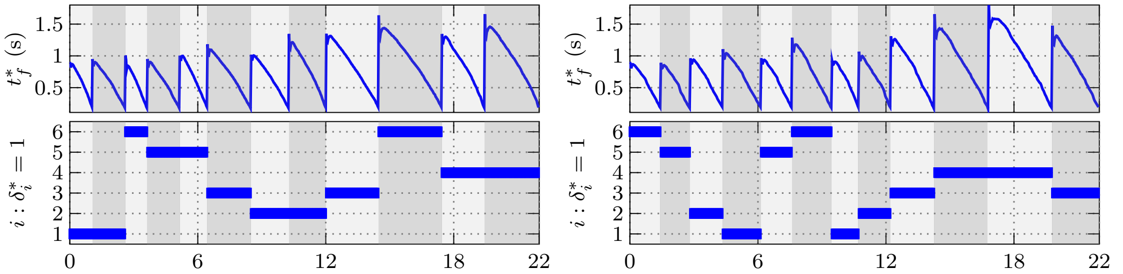
In Test Case 1, the robot will start from its initial position, pick and place all objects without changing the end-effector's orientation. Given the fact that, the gripper is



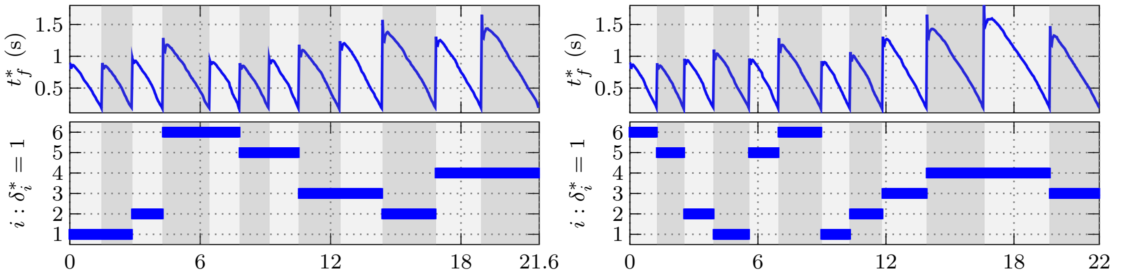
a) Hierarchic controller with minimum-distance scheduling.



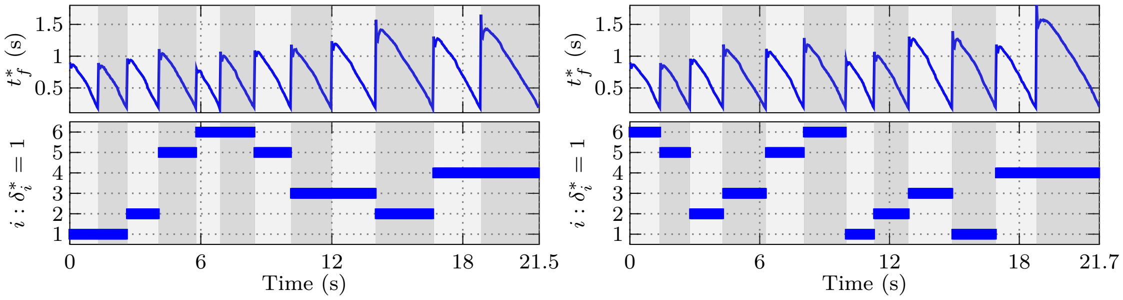
b) Hierarchic controller with minimum-time scheduling.



c) HMPC with one step scheduling horizon (1S-HMPC).



d) HMPC with two steps scheduling horizon (2S-HMPC).



e) HMPC with three steps scheduling horizon (3S-HMPC).

Figure 4.7: Minimum final time t_f^* and computed optimal values for the binary decision variables δ_i^* for Test Case 1 (left) and Test Case 2 (right). a) and b) show the results of the hierarchical control structure using minimum-distance (4.8) and minimum-time (4.12) scheduling, respectively. c), d) and e) show the results of the hybrid planning algorithm (4.54) with $N_S = 0$ (1S-HMPC), $N_S = 1$ (2S-HMPC) und $N_S = 2$ (3S-HMPC), respectively. The light gray areas represent the period when the robot is on the way to pick up an object, and the dark gray regions display the time span after picking up the object until placing it in the chosen slot.

grasping and placing the objects perpendicular to them, the orientation of the gripper at the target points is depending mainly on the sixth robot joint (θ_6), see Section 2.1. In this test case, the sixth robot joint only needs to compensate for the change in orientation caused by the rotation of the first (θ_1) robot joint to keep the gripper's direction constant. Considering that the objects are located on the xy-plane, joint two (θ_2) three (θ_3) and the first joint are the dominant ones since they are decisive for stretching the robot arm along the xy-plane. Thus, the problem formulation in the joint space is quite similar to the one in the robot workspace, where the Euclidean distance is minimized. Furthermore, the location of the objects is symmetrical, i. e., objects o_3 and o_4 have approximately the same distance from the robot base. The same also holds for objects o_2, o_5 and o_1, o_6. Since the minimum-distance scheduling (4.8) is performed in the workspace and the minimum-time scheduling (4.12) in the configuration space, the computed optimal sequences are different (Figure 4.7a) and 4.7b) left), but the scheduling performance in terms of overall task execution time is similar, with the minimum-time scheduler performing slightly better. The computed minimum-distance scheduling sequence is

$$\{o_2, s_2, o_1, s_1, o_3, s_3, o_4, s_5, o_6, s_6, o_5, s_4\},$$

meaning that, first Object o_2 will be placed in Slot s_2, then Object o_1 in Slot s_1 and so on. The minimum-time scheduling sequence, on the other side, reads

$$\{o_1, s_5, o_3, s_1, o_2, s_2, o_4, s_6, o_6, s_3, o_5, s_4\}.$$

Note that both scheduling algorithms consider all possible combinations while computing the optimal task sequences. According to the hierarchical control structure in Section 4.2, the computed optimal task sequences from the scheduling layer are forwarded to the lower layer for minimum-time online trajectory planning to finally execute the assigned tasks.

Next, focusing again on Test Case 1, the performance of the hybrid controller will be analyzed, starting with the single-step hybrid controller ($N_S = 0$) for online task and trajectory planning. The resulting optimal scheduling sequence, as shown in Figure 4.7c), reads

$$\{o_1, s_1, o_6, s_5, o_5, s_3, o_2, s_2, o_3, s_6, o_4, s_4\}.$$

To better understand the performance of the single-step hybrid controller, the maximum angular distances between the objects and slots in conjunction with resulting optimal task execution sequence are shown in Table 4.2. According to the computation of the fastest transition time between two task points (4.9), the maximum angular distance between two joint configurations denotes the joint farthest from its target, which, in case all robot joints have the same speed characteristics, determines the time needed by the robot to move from the current to the next task point, see also [77]. Starting from the initial robot position $\mathbf{q}_{\text{in}}$, the computed maximum angular distances to all objects, i. e., $\max|\mathbf{q}_{\text{in}} - \mathbf{q}_{o_i}|$ in Table 4.2, indicate that objects o_1 and o_6 are closest to the actual robot configuration. Considering that they are located at the same distance along the x-axis (see Figure 4.1), the maximum angular distance is the same for both objects. The resulting value corresponds to the distance of robot joint three (θ_3), which, compared to the other joints, has to cover the largest angular distance on the way from the initial robot position to the objects. From the two nearest possible objects, the robot decides to pick up Object o_1, i. e., $\delta^*_{o_1 1} = 1$ and place it in Slot s_1 ($\delta^*_{s_1 1} = 1$), which is the

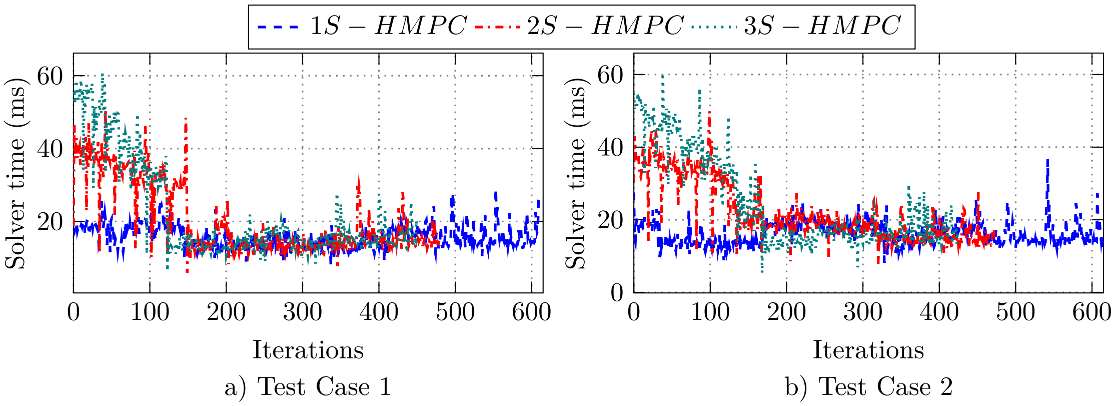
Table 4.2: Maximum angular distances for Test Case 1 when applying 1S-HMPC.

Object o_i / Slot s_i	1	2	3	4	5	6
$\max \lvert \mathbf{q}_{\text{in}} - \mathbf{q}_{o_i} \rvert$	**0.903**	1.096	1.342	1.342	1.096	0.903
$\max \lvert \mathbf{q}_{o_1} - \mathbf{q}_{s_i} \rvert$	**0.855**	1.015	0.924	1.085	0.855	1.015
$\max \lvert \mathbf{q}_{s_1} - \mathbf{q}_{o_i} \rvert$		1.048	1.294	1.294	1.048	**0.928**
$\max \lvert \mathbf{q}_{o_6} - \mathbf{q}_{s_i} \rvert$		1.059	0.924	1.085	**0.855**	1.015
$\max \lvert \mathbf{q}_{s_5} - \mathbf{q}_{o_i} \rvert$		1.048	1.294	1.294	**1.048**	
$\max \lvert \mathbf{q}_{o_5} - \mathbf{q}_{s_i} \rvert$		1.207	**1.116**	1.277		1.207
$\max \lvert \mathbf{q}_{s_3} - \mathbf{q}_{o_i} \rvert$		**1.116**	1.362	1.362		
$\max \lvert \mathbf{q}_{o_2} - \mathbf{q}_{s_i} \rvert$		**1.207**		1.277		1.207
$\max \lvert \mathbf{q}_{s_2} - \mathbf{q}_{o_i} \rvert$			**1.453**	1.453		
$\max \lvert \mathbf{q}_{o_3} - \mathbf{q}_{s_i} \rvert$				1.524		**1.453**
$\max \lvert \mathbf{q}_{s_6} - \mathbf{q}_{o_i} \rvert$				**1.453**		
$\max \lvert \mathbf{q}_{o_4} - \mathbf{q}_{s_i} \rvert$				**1.524**		

nearest task point according to the maximum angular distances $\max |\mathbf{q}_{o_1} - \mathbf{q}_{s_i}|$ shown in Table 4.2. From there, the robot will go to Object o_6 ($\delta^*_{o_6 2} = 1$) and place it in Slot s_5 ($\delta^*_{s_5 2} = 1$), continue further to Object o_5, place it in Slot s_3, and so on. It can be seen that towards the end, the task execution time becomes larger, which is consistent with the proposed algorithm aiming to choose the fastest possible task point and plan the trajectory to it. This is also evident from the values in Table 4.2, where the objects and slots corresponding to the smallest angular distance are always selected (represented by bold values). The objects o_3 and o_4 will be selected last, as their execution requires more time compared to the other objects. The empty cells in the table represent task points that are no longer available.

When using the single-step hybrid control strategy, the resulting optimal task sequence does not necessarily lead to an overall time-optimal robot task execution. As previously presented in Figure 4.7c) and Table 4.2, always selecting the fastest possible task point results in an overall task execution time of about 22 s, which is higher compared to the performance of the hierarchic controllers shown in Figure 4.7a) and Figure 4.7b). To improve the performance of the hybrid controller, the scheduling horizon N_S can be increased to consider more possible task point combinations while recursively deciding which task point to execute next. Increasing the scheduling horizon by one ($N_S = 1$) results in an hybrid MPC algorithm with two steps scheduling horizon (2S-HMPC), which, as shown in Figure 4.7d) for Test Case 1, leads to a different task sequence and an improvement on the overall task execution time. A further increase of the scheduling horizon to $N_S = 2$ in the form of a three-step hybrid MPC (3S-HMP) results in slightly different task execution order and shorter overall task execution time, see Figure 4.7e) for Test Case 1.

Comparing the performance of all presented optimization approaches for Test Case 1, it can be seen that, the hierarchic controller with the minimum-time scheduling performs slightly better. This is to be expected, since the problem is modeled in the configuration space and it considers all possible task point combinations when generating the optimal task sequence. In general, the co-design approach in form of the multi-step hybrid controller should perform best, if the scheduling horizon is chosen to consider all possible task point combinations in the recursive task and trajectory planning. However, increasing



a) Test Case 1 b) Test Case 2

Figure 4.8: Solver time over MPC iterations.

the scheduling horizon also increases the number of binary optimization variables resulting, as shown in Figure 4.8, in a larger computation time. The computation time is significantly higher at the beginning, where more task points are available but never exceeds 60 ms at any iteration. When comparing the performance of the hierarchic and hybrid controller one should consider also that the hybrid controller beside the time, which is parameterized depending on the scheduling horizon, minimizes also the robot joint jerk to generate smooth robot trajectories.

In Test Case 2, where the gripper orientation at the task points $\{o_1, o_3, o_5, s_2, s_4, s_6\}$ is rotated by 90° about the z-axis compared to its initial orientation, the sixth robot joint (θ_6) significantly influences the optimization results. However, this is not the case for the hierarchic structure with minimum-distance scheduling since the orientation of the task points can not be incorporated into the scheduling model when considering the problem in the workspace. Given that the distribution of the task points in Test Case 1 and Test Case 2 remains the same, and only the orientation of some task points changes, the minimum-distance scheduling computes the same scheduling sequence in both cases, see Figure 4.7a). However, in comparison to the results of the other proposed algorithms, minimum-distance scheduling delivers the poorest performance in terms of overall task execution time. The best performance is again given by the hierarchic controller with minimum-time scheduling and the computed optimal sequence

$$\{o_6, s_3, o_2, s_5, o_4, s_6, o_3, s_1, o_5, s_2, o_1, s_4\},$$

resulting in an overall task execution time of about 21.6 s.

The scheduling results for the single-step hybrid controller (1S-HMPC) and Test Case 2 are shown to the right of Figure 4.7c) in conjunction with Table 4.3. Starting form its initial position $\mathbf{q}_{\text{in}}$ the robot will pick up Object o_6 which is with an angular distance of $\Delta\theta_3 = 0.903$ rad nearest to the actual robot configuration. Because of the 90°-orientation of the end-effector while picking up Object o_1, this object is with an angular distance $\Delta\theta_6 = 1.842$ rad the farthest to the actual robot configuration. Note that, the angular distance of robot joint six, is larger than $\pi/2$, although the end-effector orientation at object o_1 is rotated by 90° about the z-axis compared to its orientation at the initial position, see Figure 4.1. This is because the sixth robot joint also has to compensate for the end-effector rotation over the negative z-axis, which is imposed by the displacement of the first robot joint (θ_1), moving the end-effector along the negative y-axis. The angular distance $\Delta\theta_3 = 1.096$ rad also underpins this fact, as the maximum distance comes from

Table 4.3: Maximum angular distances for Test Case 2 when applying 1S-HMPC.

Object o_i / Slot s_i	1	2	3	4	5	6
$\max\lvert\mathbf{q}_{\text{in}} - \mathbf{q}_{o_i}\rvert$	1.842	1.096	1.467	1.342	1.096	**0.903**
$\max\lvert\mathbf{q}_{o_6} - \mathbf{q}_{s_i}\rvert$	0.928	2.630	0.924	2.214	**0.855**	1.727
$\max\lvert\mathbf{q}_{s_5} - \mathbf{q}_{o_i}\rvert$	2.300	**1.048**	1.925	1.294	1.517	
$\max\lvert\mathbf{q}_{o_2} - \mathbf{q}_{s_i}\rvert$	**1.048**	1.864	1.116	1.447		1.207
$\max\lvert\mathbf{q}_{s_1} - \mathbf{q}_{o_i}\rvert$	1.556		1.294	1.294	**1.048**	
$\max\lvert\mathbf{q}_{o_5} - \mathbf{q}_{s_i}\rvert$		1.207	1.126	1.277		**1.207**
$\max\lvert\mathbf{q}_{s_6} - \mathbf{q}_{o_i}\rvert$	**1.015**		1.453	1.453		
$\max\lvert\mathbf{q}_{o_1} - \mathbf{q}_{s_i}\rvert$		**1.015**	1.910	1.085		
$\max\lvert\mathbf{q}_{s_2} - \mathbf{q}_{o_i}\rvert$			**1.453**	2.288		
$\max\lvert\mathbf{q}_{o_3} - \mathbf{q}_{s_i}\rvert$			1.534	**1.524**		
$\max\lvert\mathbf{q}_{s_4} - \mathbf{q}_{o_i}\rvert$				**1.871**		
$\max\lvert\mathbf{q}_{o_4} - \mathbf{q}_{s_i}\rvert$			**1.362**			

robot joint three, although the end-effector orientation while picking up Object o_5 is also rotated by $\pi/2$ over the z-axis. Joint number six is not the dominant one in this case, since to move the end-effector along the positive y-axis, robot's basis (θ_1) is rotating also over the positive z-axis. By this, the end-effector is being rotated towards the desired final orientation and the sixth robot joints needs, in this case, to cover only the difference between the desired final gripper-orientation and the one imposed by the movement of the robot base. Following the single-step hybrid controller where the fastest possible task point is always executed next, Object o_6 is placed in Slot s_5, Object o_2 in Slot s_1, and so on, resulting in the task sequence

$$\{o_6, s_5, o_2, s_1, o_5, s_6, o_1, s_2, o_3, s_4, o_4, s_3\}\,.$$

Increasing the scheduling horizon by one (2S-HMPC) results in the same task sequence as when applying 1S-HMPC without reducing the overall task execution time. A slight improvement is achieved by a further increase of the scheduling horizon (3S-HMPC), which, compared to 1S-HMPC and 2S-HMPC, results in a change in the sequence order, where Object o_2 is placed in Slot s_3 instead of s_1, Object o_3 is placed in Slot s_1 instead of Slot s_4, and Object o_4 is placed in Slot s_4 instead of Slot s_3, (see right of Figure 4.7c)-4.7d))

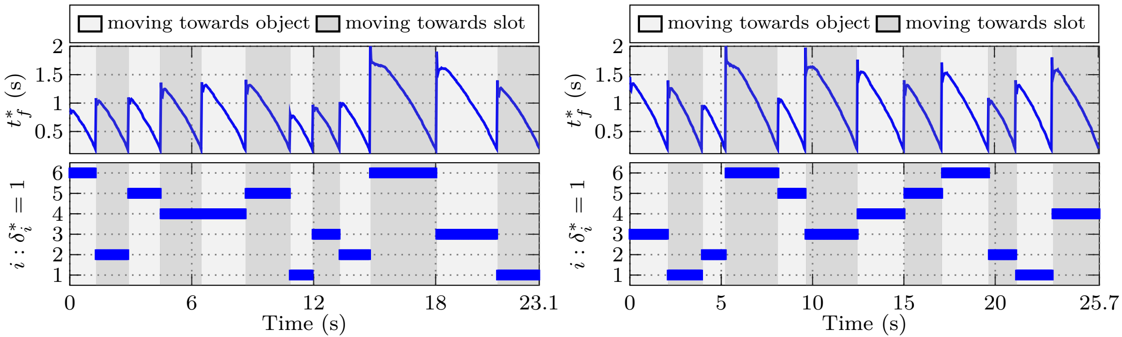
Test Case 3

In Test Case 3, there exist three different object and slot types distributed as follows

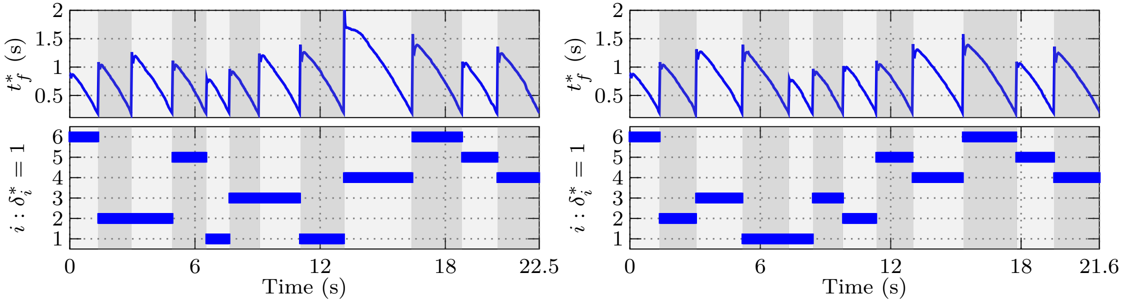
$$\begin{aligned} &\mathcal{O}_{\text{red}} = \{o_1,\, o_5\}\,, \quad \mathcal{O}_{\text{green}} = \{o_2,\, o_4\}\,, \quad \mathcal{O}_{\text{blue}} = \{o_3,\, o_6\}\,, \quad \text{and} \\ &\mathcal{S}_{\text{red}} = \{s_3,\, s_4\}\,, \quad \mathcal{S}_{\text{green}} = \{s_5,\, s_6\}\,, \quad \mathcal{S}_{\text{blue}} = \{s_1,\, s_2\}\,. \end{aligned}$$

The experimental results for both control structures and the hybrid controller's computation time are jointly displayed in Figure 4.9. The orientation of the robot end-effector for the task points $\{o_1, o_2, o_3, s_1, s_3, s_5\}$ is rotated by 90° about the z-axis compared to its initial orientation. The hierarchic controller with minimum-time scheduling, shown in Figure 4.9e), delivers again the best performance resulting in the task execution sequence

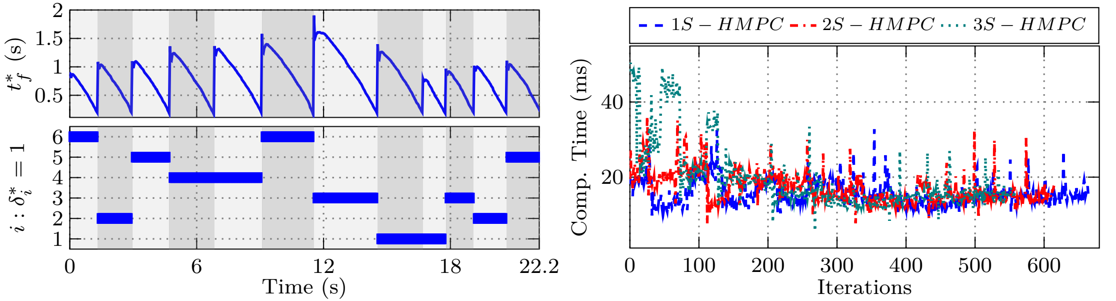
$$\{o_6, s_2, o_3, s_1, o_1, s_3, o_2, s_5, o_4, s_6, o_5, s_4\}\,.$$



a) HMPC with one step scheduling horizon (1S-HMPC). d) Minimum-distance scheduling.



b) HMPC with two steps scheduling horizon (2S-HMPC). e) Minimum-time scheduling.



c) HMPC with three steps scheduling horizon (3S-HMPC). f) Computation time of the hybrid controller.

Figure 4.9: Task and trajectory planning results for Test Case 3. a)-c) show the experimental results of the hybrid controller for $N_S \in \{0, 1, 2\}$ with the resulting computation time shown in f). d) and e) display the experimental results of the hierarchic structure with minimum-distance and minimum-time scheduling, respectively.

As expected, the hierarchic controller with minimum-distance scheduling in Figure 4.9d) performs poorest, resulting in an overall robot cycle time being about 4 s slower compared to the hierarchic structure with minimum-time scheduling.

Applying the hybrid controller with one step scheduling horizon (1S-HMPC) leads to the task execution sequence

$$\{o_6, s_2, o_5, s_4, o_4, s_5, o_1, s_3, o_2, s_6, o_3, s_1\},$$

which corresponds to selecting the nearest possible feasible task point, see Table 4.4. Starting from the initial robot configuration, Object o_6 is selected first and placed in Slot s_2 of the same class. Then Object o_5 is picked and placed in Slot s_4. In Table 4.4, it can be seen that, when moving from a slot to an object, always the object with the smallest angular distance is selected. When moving from an object to a slot, the slot that belongs to the same class as the previously selected object and has the minimum angular distance to it, i. e., the fastest to reach, is chosen.

Increasing the scheduling horizon by one, respectively two, leads to the optimal task sequences presented in Figure 4.9b) and Figure 4.9c), which improve the overall robot

Table 4.4: Maximum angular distances for Test Case 3 when applying 1S-HMPC.

Object o_i / Slot s_i	1	2	3	4	5	6
$\max \lvert \mathbf{q}_{\text{in}} - \mathbf{q}_{o_i} \rvert$	1.842	1.964	1.467	1.342	1.096	**0.903**
$\max \lvert \mathbf{q}_{o_6} - \mathbf{q}_{s_i} \rvert$	2.499	**1.059**	2.146	1.085	1.755	1.015
$\max \lvert \mathbf{q}_{s_2} - \mathbf{q}_{o_i} \rvert$	1.425	1.277	1.453	1.453	**1.207**	
$\max \lvert \mathbf{q}_{o_5} - \mathbf{q}_{s_i} \rvert$	2.368		2.015	**1.227**	1.624	1.207
$\max \lvert \mathbf{q}_{s_4} - \mathbf{q}_{o_i} \rvert$	1.842	1.693	1.524	**1.524**		
$\max \lvert \mathbf{q}_{o_4} - \mathbf{q}_{s_i} \rvert$	2.157		1.804		**1.413**	1.453
$\max \lvert \mathbf{q}_{s_5} - \mathbf{q}_{o_i} \rvert$	**0.855**	1.048	1.294			
$\max \lvert \mathbf{q}_{o_1} - \mathbf{q}_{s_i} \rvert$	0.855		**0.924**			2.328
$\max \lvert \mathbf{q}_{s_3} - \mathbf{q}_{o_i} \rvert$		**1.116**	1.362			
$\max \lvert \mathbf{q}_{o_2} - \mathbf{q}_{s_i} \rvert$	1.048					**2.180**
$\max \lvert \mathbf{q}_{s_6} - \mathbf{q}_{o_i} \rvert$			**1.953**			
$\max \lvert \mathbf{q}_{o_3} - \mathbf{q}_{s_i} \rvert$	**1.294**					

task execution time. A larger scheduling horizon results in an increased computation time of the hybrid controller as displayed in Figure 4.9f). Compared to Test Case 1 and Test Case 2, the computation time is slightly shorter since the additional class-assignment constraints reduce the number of potential combinatorial decisions.

The presented control structures and optimization-based approaches have each their benefits and drawbacks. The hierarchic control structure consists of two decoupled optimization problems, out of which only the robot trajectory planning problem is solved online. The scheduling layer is performed offline as it considers all possible robot task combinations and is thus computationally demanding with a computation time varying between 100 ms-250 ms for the considered test cases with six objects and six slots. The minimum-distance scheduling approach is easier to implement as it requires only the computation of the Euclidean distances between the task points. The inverse kinematics is needed after scheduling is performed to compute the robot joint configuration corresponding to the allocated task. Therefore, a numerical IK-solver can also be used, which converges into a single IK solution. However, minimizing the Euclidean distance provides good results when the robot end-effector motions are rectilinear and when the orientation of the end-effector does not matter when performing tasks. If the aim is to minimize the robot cycle time, the problem should be transferred to the configuration space considering multiple IK solutions. The drawback hereby is the necessity to compute all feasible IK solutions for each task point.

The co-design approach in the form of a hybrid controller is expected to perform better the larger the scheduling horizon is chosen. Since the algorithm inherently plans robot tasks and trajectories, it should be performed online, which poses a limitation, particularly for high-dimensional MIQCP problems. We can see in the results that, for Test Case 2 and Test Case 3, even a scheduling horizon of one performs better than minimum-distance scheduling in terms of overall robot cycle time. The algorithm is particularly well suited when considering a few non-stationary task points, as it constantly recalculates the optimal task sequence.

CHAPTER 5

Cooperative Task and Trajectory Planning

Advanced assembly, production, and packaging tasks enhance the need to employ two or more robot manipulators working in the same or highly overlapping physical areas. Efficient deployment of multiple cooperating robots brings the promise of shorter robot cycle times and faster task execution. In a shared workspace, multiple robots can perform a wider variety of tasks, either as a composition of different subtasks into a more complex task or by extending the operating space, e. g., by passing a tool or workpiece from one robot to another. Typically, in such settings, the robots can communicate with each other, share sensor information and coordinate their actions and motion to jointly perform the assigned tasks. The complexity of these systems poses new computational challenges, especially when the robot arrangement leads to a significant overlap of robot working areas and, thus, an increased collision risk. Therefore, the setup, control, and operation of multi-robot systems is still primarily an active field of research, see, e. g., [131–135].

The efficient and safe use of cooperating robots in a shared workspace relies on balancing the robots' workload and ensuring certain safety aspects for safe and collision-free robot operation. Analogous to the case of a single robot manipulator, these challenges can be addressed by introducing two optimization-based control policies, a discrete one for task scheduling and a continuous one for point-to-point trajectory planning. Following this approach, task scheduling can be decoupled from trajectory planning, resulting in a hierarchic control structure, or they can be combined into a single hybrid controller for inherent task and trajectory planning. Multi-robot systems in a confined working environment are subject to challenging manipulation tasks where moving robots appear to each other as dynamic obstacles of complex geometries. This poses additional challenges to optimization-based trajectory planning as it involves the incorporation of highly nonlinear collision avoidance constraints. Moreover, robot trajectories must be computed online to update the planned robot motions continuously. Consequently, the hybrid controller leads to an MINLP problem that cannot be relaxed into a less computationally intensive class of mixed-integer optimization problems suitable for online applications.

This chapter introduces an extension of the proposed hierarchic controller for a multi-robot system with overlapping robot working areas. To this end, a novel collision avoidance approach is presented based on robot geometry approximation by smooth Bézier curves and spherical objects. Velocity restrictions as constraints of a time-optimal MPC-based trajectory planning algorithm are introduced to enforce the geometry approximating spheres slide along tangent separating planes and ensure collision-free robot operation. The performance of the proposed control scheme is demonstrated by experiments with a robotic system consisting of two robot manipulators performing pick-and-place tasks.

5.1 Problem formulation

Let $\mathcal{R}$ represent a set of robots that have to execute a set of task points $\mathcal{P} = \mathcal{O} \cup \mathcal{S}$ consisting of picking objects $o \in \mathcal{O}$ and placing them in the slots $s \in \mathcal{S}$. Objects and slots can belong to different classes $b \in \mathcal{B}$, i. e., $\mathcal{O} = \bigcup_{\forall b \in \mathcal{B}} \mathcal{O}_b$ and $\mathcal{S} = \bigcup_{\forall b \in \mathcal{B}} \mathcal{S}_b$. It is assumed that all task points are reachable for both robots and each class contains an equal number of objects and slots, i. e., $|\mathcal{O}_b| = |\mathcal{S}_b|, \forall b \in \mathcal{B}$, with $\sum_{\forall b \in \mathcal{B}} |\mathcal{O}_b| = |\mathcal{O}|$ and $\sum_{\forall b \in \mathcal{B}} |\mathcal{S}_b| = |\mathcal{S}|$. It is also assumed that, depending on the number of robots, sufficient tasks points are available for simultaneous task completion.

Arranging the robot such that both can reach all task points results in a significant overlap of their working area, as shown in Figure 5.1 for the considered demonstrator with two robot manipulators. The position of the task points is given relative to the coordinate frame attached to the base of Robot 1. In addition, each object and each slot is assigned a picking, respectively filling gripper orientation.

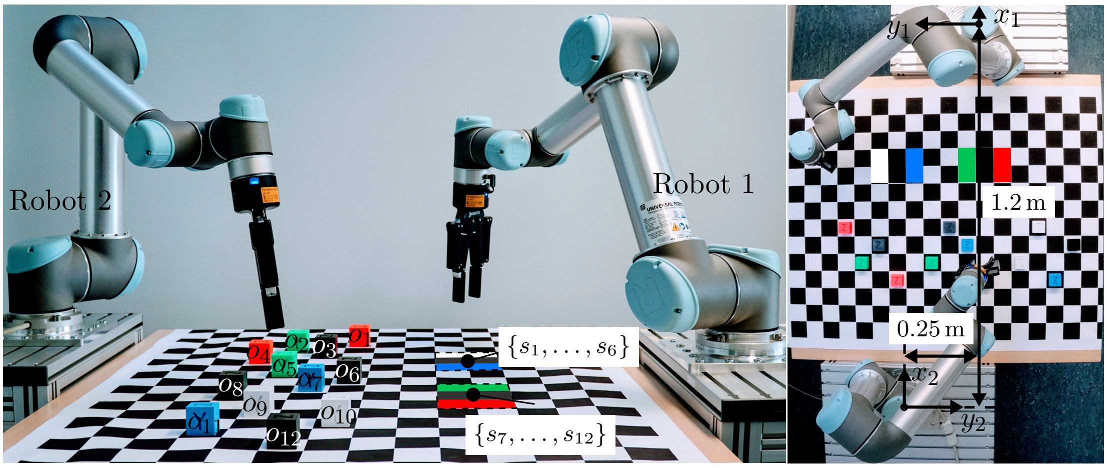
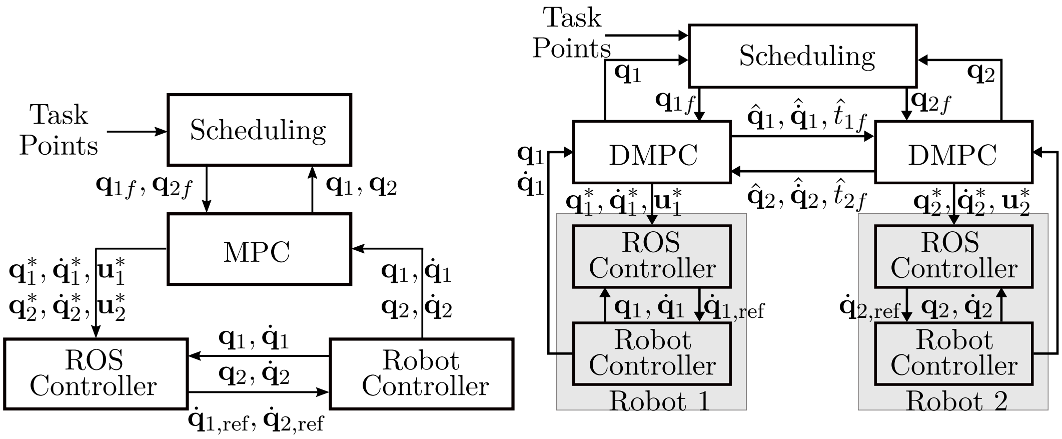


Figure 5.1: Two robot manipulators performing pick-and-place tasks with twelve objects $o \in \mathcal{O}$ and slots $s \in \mathcal{S}$, which belong to the classes $\mathcal{B} = \{"\text{black}", "\text{red}", "\text{green}", "\text{blue}", "\text{white}"\}$. The chosen distribution of the task points, with $\mathcal{S}_{\text{white}} = \{s_1, s_2\}, \mathcal{S}_{\text{black}} = \{s_3, s_4, s_9, s_{10}\}, \mathcal{S}_{\text{blue}} = \{s_5, s_6\}, \mathcal{S}_{\text{red}} = \{s_{11}, s_{12}\}$, and the robot arrangement relative to each other lead to challenging robot motions with high collision potential.

The robots should simultaneously pick one object each and place them in the corresponding slots of the same class. The task points belong to four different classes $\mathcal{B} = \{"\text{black}", "\text{red}", "\text{green}", "\text{blue}", "\text{white}"\}$. For the considered test case, placing objects according to their class often leads to robot motions with intersecting end-effector paths. Therefore, when computing the optimal gripping and filling sequences, it is essential to ensure that the resulting final robot configurations are feasible and do not lead to collisions between the robot arms. Feasible and collision-free initial and final robot configurations do not guarantee collision-free robot motions along the entire trajectories between the start and target points. Consequently, both layers of the hierarchic controller, which address task scheduling and trajectory planning, must consider safety-related constraints to guarantee collision-free robot task execution.

5.2 Hierarchical control structure

The hierarchical control structure is similar to the control architecture for task and trajectory planning of a single robot manipulator presented in Section 4.2. It consists of a discrete upper layer for task scheduling followed by an underlying trajectory planning layer based on model predictive control. In a multi-robotic system, the trajectory planning layer can be organized as a centralized algorithm [79] common to all robot arms or as a distributed planning layer [80] where each robot has its local planning algorithm. A schematic illustration of the hierarchical control structure with a centralized and distributed MPC-based trajectory planning layer for the considered robotic setup with two robot manipulators is shown in Figure 5.2.


Task Points
Scheduling
$\mathbf{q}_{1f}, \mathbf{q}_{2f}$
$\mathbf{q}_1, \mathbf{q}_2$
MPC
$\mathbf{q}_1^*, \dot{\mathbf{q}}_1^*, \mathbf{u}_1^*$
$\mathbf{q}_2^*, \dot{\mathbf{q}}_2^*, \mathbf{u}_2^*$
$\mathbf{q}_1, \dot{\mathbf{q}}_1$
$\mathbf{q}_2, \dot{\mathbf{q}}_2$
ROS Controller
Robot Controller
$\mathbf{q}_1, \dot{\mathbf{q}}_1$
$\mathbf{q}_2, \dot{\mathbf{q}}_2$
$\dot{\mathbf{q}}_{1,\mathrm{ref}}, \dot{\mathbf{q}}_{2,\mathrm{ref}}$
Task Points
Scheduling
$\mathbf{q}_1$
$\mathbf{q}_2$
$\mathbf{q}_{1f}$
$\mathbf{q}_{2f}$
$\hat{\mathbf{q}}_1, \hat{\dot{\mathbf{q}}}_1, \hat{t}_{1f}$
DMPC
DMPC
$\hat{\mathbf{q}}_2, \hat{\dot{\mathbf{q}}}_2, \hat{t}_{2f}$
$\mathbf{q}_1$
$\dot{\mathbf{q}}_1$
$\mathbf{q}_2$
$\dot{\mathbf{q}}_2$
$\mathbf{q}_1^*, \dot{\mathbf{q}}_1^*, \mathbf{u}_1^*$
$\mathbf{q}_2^*, \dot{\mathbf{q}}_2^*, \mathbf{u}_2^*$
ROS Controller
ROS Controller
$\mathbf{q}_1, \dot{\mathbf{q}}_1$
$\dot{\mathbf{q}}_{1,\mathrm{ref}}$
$\dot{\mathbf{q}}_{2,\mathrm{ref}}$
$\mathbf{q}_2, \dot{\mathbf{q}}_2$
Robot Controller
Robot Controller
Robot 1
Robot 2

a) Centralized MPC b) Distributed MPC

Figure 5.2: Hierarchical control structure with centralized (a) and distributed (b) trajectory planning layer.

The scheduling layer involves a centralized binary optimization problem that gets information about the distribution of the task points and the position of the robot arms and computes an optimal task execution for the robots to balance their workload and ensure fast and safe robot task execution. Task scheduling is performed offline, providing a sequence of feasible task points to the underlying trajectory planning layer, which computes robot trajectories online by minimizing the task execution time. In the centralized trajectory planning scheme, a single MPC algorithm is deployed to plan time-optimal trajectories for both robots. The computed optimal trajectories are forwarded to the local robot controllers using ROS Control via a velocity control interface, see Section 4.2. Each robot has its local MPC-based planning algorithm in the distributed planning architecture, resulting in a layer of communicating distributed model predictive controllers (DMPC). The DMPC-based planners communicate with each other and share information about the optimal value of their cost functions $(\hat{t}_{1f}, \hat{t}_{2f})$ and the computed robot position $(\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2)$ and velocities $(\hat{\dot{\mathbf{q}}}_1, \hat{\dot{\mathbf{q}}}_2)$, which are considered in the next planning iteration to coordinate the motions of the robots. The following will discuss the two layers of the hierarchical control structure in more detail.

5.2.1 Task scheduling

The goal of the task scheduling is to compute an optimal task sequence for the robots $r \in \mathcal{R}$ so that they simultaneously pick objects $o \in \mathcal{O}$ and place them in the slots $s \in \mathcal{S}$. Therefore, two types of binary variables are introduced. The variables $\delta_{r_o i} \in \{0,1\}$, $\forall o \in \mathcal{O}$ and $\forall r \in \mathcal{R}$, model the robot-object assignment at each picking iteration $i \in \mathcal{N} = \{1,\dots,N\}$. To model the robot-slot assignment at each filling iteration $i \in \mathcal{N} = \{1,\dots,N\}$, the binary variables $\delta_{r_s i} \in \{0,1\}$, $\forall s \in \mathcal{S}$ and $\forall r \in \mathcal{R}$ are used. By computing the optimal values of the binary variables, the robot task sequences are uniquely defined. $\delta_{r_o 1} = 1$ and $\delta_{r_s 1} = 1$ would mean, for instance, that robot r will pick up object o first and place it in slot s. The number of the required maximum scheduling iterations $N = |\mathcal{N}|$ depends on the number of the available task points $|\mathcal{P}| = |\mathcal{O}| + |\mathcal{S}|$ and robots $|\mathcal{R}|$. According to the problem formulation in Section 5.1, it is assumed that for $|\mathcal{O}| = |\mathcal{S}|$, $N = |\mathcal{P}|/(2|\mathcal{R}|) \in \mathbb{N}$.

In order to balance the workload between the robots, it is required that at each picking and placing iteration $i \in \mathcal{N}$ each robot has to pick an object $o \in \mathcal{O}$ and fill a slot $s \in \mathcal{S}$. This is imposed by applying the binary equality constraints

$$\sum_{\forall o \in \mathcal{O}} \delta_{r_o i} = 1 \quad \text{and} \quad \sum_{\forall s \in \mathcal{S}} \delta_{r_s i} = 1\,, \quad \forall i \in \mathcal{N}, \quad \forall r \in \mathcal{R}\,. \tag{5.1}$$

The constraints

$$\sum_{\forall r \in \mathcal{R}} \sum_{\forall i \in \mathcal{N}} \delta_{r_o i} = 1 \quad \text{and} \quad \sum_{\forall r \in \mathcal{R}} \sum_{\forall i \in \mathcal{N}} \delta_{r_s i} = 1\,, \quad \forall o \in \mathcal{O}, \quad \forall s \in \mathcal{S} \tag{5.2}$$

imply that across all scheduling iterations each object is picket only once and each slot is filled only once by one of the robots $r \in \mathcal{R}$. Further, the equality constraints

$$\sum_{\forall o \in \mathcal{O}_b} \delta_{r_o i} = \sum_{\forall s \in \mathcal{S}_b} \delta_{r_s i}, \quad \forall b \in \mathcal{B}, \quad \forall r \in \mathcal{R}, \quad \forall i \in \mathcal{N} \tag{5.3}$$

are used, ensuring that an object $o \in \mathcal{O}_b$ of class $b \in \mathcal{B}$ that is selected by robot $r \in \mathcal{R}$ can be placed only in a slot $s \in \mathcal{S}_b$ of the same class.

Two simultaneously selected task points executed each by one robot have to lead to feasible, i. e., collision-free robot configurations. Let $p, p' \in \mathcal{P}$ denote two simultaneously chosen task points, which can be either objects or slots. The vectors

$${}_r\mathbf{p}_{pp'} = {}_r\mathbf{p}_p - {}_r\mathbf{p}_{p'}, \quad \forall p, p' \in \mathcal{O} \quad \text{or} \quad \forall p, p' \in \mathcal{S}, \quad p \neq p', \quad r \in \mathcal{R}, \tag{5.4}$$

represent the relative position vector from task point p to p' expressed in the base coordinate frame of robot r. Here, ${}_r\mathbf{p}_p$ and ${}_r\mathbf{p}_{p'}$ denote the position vectors from the base frame of robot r to the respective task points, i. e., to the robot end-effector. If robot r is executing task point p at the scheduling iteration $i \in \mathcal{N}$ ($\delta_{r_o i} = 1$), and robot r' is executing task point p' at the same scheduling iteration i ($\delta_{r'_{o'} i} = 1$), it is required that the Euclidean distance between the task points $d_{pp'} = \|{}_r\mathbf{p}_{pp'}\|_2$, does not undercut a minimum distance value, i. e.,

$$d_{pp'} \geq d_{\min}\, e^{\left(\mathbf{e}_x^{\mathrm{T}}\, {}_r\mathbf{p}_{pp'}\right)}\,. \tag{5.5}$$

The function argument $\mathbf{e}_x^{\mathrm{T}}\, {}_r\mathbf{p}_{pp'}$, with $\mathbf{e}_x^{\mathrm{T}} = [1, 0, 0]$, represents an overlapping measure for the given robot arrangement in Figure 5.1. A larger $\mathbf{e}_x^{\mathrm{T}}\, {}_r\mathbf{p}_{pp'} > 0$ value results in a stronger overlapping of the robot working areas, increasing thus the value of the applied safety minimum distance. Therefore, the binary inequality constraints for the objects

$$d_{oo'} + \left(2 - \delta_{r_o i} - \delta_{r'_{o'} i}\right) d_{\min} \, e^{\left(\mathbf{e}_x^{\mathrm{T}} \, {}_r\mathbf{p}_{oo'}\right)} \geq d_{\min} \, e^{\left(\mathbf{e}_x^{\mathrm{T}} \, {}_r\mathbf{p}_{oo'}\right)} \,, \\ \forall o, o' \in \mathcal{O}, o \neq o', \quad \forall r, r' \in \mathcal{R}, r \neq r', \quad \forall i \in \mathcal{N} \tag{5.6}$$

and slots

$$d_{ss'} + \left(2 - \delta_{r_s i} - \delta_{r'_{s'} i}\right) d_{\min} \, e^{\left(\mathbf{e}_x^{\mathrm{T}} \, {}_r\mathbf{p}_{ss'}\right)} \geq d_{\min} \, e^{\left(\mathbf{e}_x^{\mathrm{T}} \, {}_r\mathbf{p}_{ss'}\right)} \,, \\ \forall s, s' \in \mathcal{S}, s \neq s', \quad \forall r, r' \in \mathcal{R}, r \neq r', \quad \forall i \in \mathcal{N} \tag{5.7}$$

are applied to ensure that two simultaneously selected task points are not located close to each other. If the task points p and p' are not selected both at the same scheduling iteration i, the constraints are trivially satisfied.

The binary optimization problem for task scheduling, analogous to the case of a single robot manipulator, can be modeled in the robot working $\mathcal{W}$ or configuration $\mathcal{C}$ space. The problem formulation in the robot joint space leads to a time-optimal IBLP problem of the form

$$\min_{\delta_{r_o}, \delta_{r_s}} \quad \underbrace{\sum_{\forall r \in \mathcal{R}} \sum_{\forall o \in \mathcal{O}} t^*_{ro} \, \delta_{r_o 1}}_{\text{Initial robot movement}} + \underbrace{\sum_{\forall b \in \mathcal{B}} \left(\sum_{\forall i \in \mathcal{N}} \sum_{\forall r \in \mathcal{R}} \sum_{\forall s \in \mathcal{S}_b} \sum_{\forall o \in \mathcal{O}_b} t^*_{r_o r_s} \, \delta_{r_o i} \, \delta_{r_s i} \right)}_{\text{Movement from the objects to the slots}} \tag{5.8}$$

$$+ \underbrace{\sum_{\forall i \in \mathcal{N} \setminus \{N-1\}} \sum_{\forall r \in \mathcal{R}} \sum_{\forall s \in \mathcal{S}} \sum_{\forall o \in \mathcal{O}} t^*_{r_o r_s} \, \delta_{r_o \, i+1} \, \delta_{r_s i}}_{\text{Movement from the slots to the objects}}$$

$$\text{s.t.} \quad \sum_{\forall o \in \mathcal{O}} \delta_{r_o i} = 1 \wedge \sum_{\forall s \in \mathcal{S}} \delta_{r_s i} = 1 \,, \quad \forall i \in \mathcal{N}, \quad \forall r \in \mathcal{R} \tag{5.8a}$$

$$\sum_{\forall r \in \mathcal{R}} \sum_{\forall i \in \mathcal{N}} \delta_{r_o i} = 1 \wedge \sum_{\forall r \in \mathcal{R}} \sum_{\forall i \in \mathcal{N}} \delta_{r_s i} = 1 \,, \quad \forall o \in \mathcal{O}, \quad \forall s \in \mathcal{S} \tag{5.8b}$$

$$\sum_{\forall o \in \mathcal{O}_b} \delta_{r_o i} = \sum_{\forall s \in \mathcal{S}_b} \delta_{r_s i}, \quad \forall b \in \mathcal{B}, \quad \forall r \in \mathcal{R}, \quad \forall i \in \mathcal{N}. \tag{5.8c}$$

$$d_{oo'} + \left(2 - \delta_{r_o i} - \delta_{r'_{o'} i}\right) d_{\min} \, e^{\left(\mathbf{e}_x^{\mathrm{T}} \, {}_r\mathbf{p}_{oo'}\right)} \geq d_{\min} \, e^{\left(\mathbf{e}_x^{\mathrm{T}} \, {}_r\mathbf{p}_{oo'}\right)} \,, \\ \forall o, o' \in \mathcal{O}, o \neq o', \quad \forall r, r' \in \mathcal{R}, r \neq r', \quad \forall i \in \mathcal{N} \tag{5.8d}$$

$$d_{ss'} + \left(2 - \delta_{r_s i} - \delta_{r'_{s'} i}\right) d_{\min} \, e^{\left(\mathbf{e}_x^{\mathrm{T}} \, {}_r\mathbf{p}_{ss'}\right)} \geq d_{\min} \, e^{\left(\mathbf{e}_x^{\mathrm{T}} \, {}_r\mathbf{p}_{ss'}\right)} \,, \\ \forall s, s' \in \mathcal{S}, s \neq s', \quad \forall r, r' \in \mathcal{R}, r \neq r', \quad \forall i \in \mathcal{N} \,. \tag{5.8e}$$

The first part of the cost function is modeling the transition time from the initial robot positions to all objects, with

$$t^*_{ro} = \min_{\mathbf{q}_{cr_o}, \mathbf{q}_{cr}} \mathcal{T}_o(\mathbf{q}_{cr_o}, \mathbf{q}_{r0}) \,, \quad \forall o \in \mathcal{O}, \quad \forall r \in \mathcal{R} \tag{5.9}$$

representing the minimum transition times between the initial robot configurations $\mathbf{q}_{r0}$ and the robot configurations $\mathbf{q}_{cr_o}$ corresponding to objects $o \in \mathcal{O}$. The second part of the cost function is modeling the overall transition time required by the robots to move from the objects to the slots, and the last part represents the vice-versa motion from the slots to the objects. Here,

$$t^*_{r_o r_s} = \min_{\mathbf{q}_{cr_o}, \mathbf{q}_{cr_s}} \mathcal{T}_{r_o r_s}(\mathbf{q}_{cr_o}, \mathbf{q}_{cr_s}) , \quad \forall o \in \mathcal{O}, \quad \forall s \in \mathcal{S}, \quad \forall r \in \mathcal{R} \tag{5.10}$$

represent the minimum transition times needed by robot $r \in \mathcal{R}$ to move between an object $o \in \mathcal{O}$ and a slot $s \in \mathcal{S}$, see Section 4.2.1.2.

The task scheduling optimization problem in the robot operating space $\mathcal{W}$ can be modeled by simply replacing the transition times t^*_{ro} and $t^*_{r_o r_s}$ by the Euclidean distances

$$\begin{aligned} d_{ro} &= \|{}_0\mathbf{p}_o - {}_0\mathbf{p}_{re}\|_2 , \quad \forall o \in \mathcal{O}, \quad \forall r \in \mathcal{R} \\ d_{os} &= \|{}_0\mathbf{p}_o - {}_0\mathbf{p}_s\|_2 , \quad \forall o \in \mathcal{O}, \quad \forall s \in \mathcal{S} , \end{aligned} \tag{5.11}$$

resulting in a minimum-distance IBLP problem, see Section 4.2.1.1.

5.2.2 Centralized trajectory planning

Similar to the problem formulation in Section 4.2.2, the goal of trajectory planning is to generate feasible robot trajectories online so that they safely reach the assigned targets in the shortest possible time t_f i. e.,

$$\lim_{t \to t_f} ||\mathbf{x}(t) - \mathbf{x}_f(t)|| = 0 \text{ , with } \mathbf{x}(t) \in \chi_{\text{free}} \text{ , } t \in [t_0, t_f]. \tag{5.12}$$

Both robots are represented by the generalized coordinates $\mathbf{q}_1(t) \in \mathcal{C}_1$ and $\mathbf{q}_2(t) \in \mathcal{C}_2$, which along with the respective joint velocities span the state space

$$\mathbf{x}(t) = [\mathbf{x}_1^{\mathrm{T}}(t), \mathbf{x}_2^{\mathrm{T}}(t)]^{\mathrm{T}} = [\mathbf{q}_1^{\mathrm{T}}(t), \dot{\mathbf{q}}_1^{\mathrm{T}}(t), \mathbf{q}_2^{\mathrm{T}}(t), \dot{\mathbf{q}}_2^{\mathrm{T}}(t)]^{\mathrm{T}} . \tag{5.13}$$

The trajectory planning problem is then formulated as a time-optimal optimization problem of the form

$$\min_{\mathbf{u}(t),\, t_f} \quad \int_{t_0}^{t_f} \mathrm{d}t \tag{5.14}$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{5.14a}$$

$$\mathbf{x}(t) \in \chi_{\text{free}} , \qquad t \in [t_0, t_f] \tag{5.14b}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}(t) \leq \bar{\mathbf{u}} , \quad t \in [t_0, t_f] \tag{5.14c}$$

$$\mathbf{x}(t_f) = \mathbf{x}_f(t_f) . \tag{5.14d}$$

Here, t_0 denotes the current time point, t_f the final time to be minimized subject to the dynamic system (5.14a), representing the robots with the states $\mathbf{x}(t)$ and inputs $\mathbf{u}(t) = [\mathbf{u}_1^{\mathrm{T}}(t), \mathbf{u}_2^{\mathrm{T}}(t)]^{\mathrm{T}}$. The latter are confined by the component-wise constrains (5.14c). The robots should simultaneously reach the assigned tasks at the end of the time horizon, as represented by the terminal constraints (5.14d) for a desired task point $\mathbf{x}_f = [\mathbf{q}_{1f}^{\mathrm{T}}, \dot{\mathbf{q}}_{1f}^{\mathrm{T}}, \mathbf{q}_{2f}^{\mathrm{T}}, \dot{\mathbf{q}}_{2f}^{\mathrm{T}}]^{\mathrm{T}}$. The computed trajectories should be collision-free and respect the physical robot limitations, which is ensured by applying the constraints (5.14b). χ_{free} describes the feasible safe region (i. e., free of collisions) in the state space. To solve the trajectory planning problem the time scaling (4.17) is applied transforming the free terminal time optimization problem into a fixed terminal time problem.

5.2.2.1 Prediction model

Following the same idea as presented in Section 4.2.2.1, a simplified kinematic prediction model of n double integrators for each robot, i. e.,

$$\ddot{\mathbf{q}}_r(t) = \mathbf{u}_r(t)\,, \quad r \in \{1, 2\}\,, \tag{5.15}$$

is used to predict robot trajectories while accounting for position $\mathbf{q}_r(t)$, velocity $\dot{\mathbf{q}}_r(t)$, and acceleration $\mathbf{u}_r(t)$ constraints. Obviously, using a kinematic model does not allow direct constraints to be applied to the robot's torque. In this case, a dynamic robot model must be considered, which is computationally intensive and is more suitable if direct torque control is applied to control the robots.

Considering a centralized minimum-time MPC-based planning algorithm for both robot manipulators represented by the state space $\mathbf{x}(t) \in \mathbb{R}^{4n}$ from (5.13) yields the overall prediction model

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)\,, \quad \mathbf{x}(t_0) = \mathbf{x}_0\,, \tag{5.16}$$

with the input vector $\mathbf{u}(t) \in \mathbb{R}^{2n}$, state matrix $\mathbf{A} \in \mathbb{R}^{4n\times 4n}$, and input matrix $\mathbf{B} \in \mathbb{R}^{4n\times 2n}$

$$\mathbf{A} = \operatorname{diag}\left(\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\right), \; \mathbf{B} = \operatorname{diag}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}, \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\right), \tag{5.17}$$

where $\mathbf{0} \in \mathbb{R}^{n\times n}$ and $\mathbf{I} \in \mathbb{R}^{n\times n}$ denote the zero and identity matrix, respectively.

Applying the time transformation (4.17) and the Picard's iteration method (4.24) the continuous time system is discretized yielding the discrete time dynamics

$$\mathbf{x}(k+1) = (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}(k) + \Delta\tau t_f^2\left(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\right)\mathbf{B}\mathbf{u}(k), \tag{5.18}$$

with the identity matrix $\mathbf{I} \in \mathbb{R}^{4n\times 4n}$ and the input $\mathbf{u}(k)$ being constant within the time interval $\tau \in [k\Delta\tau,\, (k+1)\Delta\tau)$. Here, without loss of generality, $t_0 = 0$ is assumed, see Section 4.2.2.1.

5.2.2.2 Collision avoidance

In this section, a feasible set χ_{free} of robot trajectories is defined to ensure that the robots on their way to the assigned targets do not collide with each other and the working environment, and there is also no self-collision between the links of a robot.

Performing cooperative tasks in a shared workspace with highly overlapping operating space between the robots significantly increases the potential for inter-robot collisions and imposes more complexity on trajectory planning. When formulating collision avoidance constraints, the entire robot geometry must be considered to exclude the possibility of collisions occurring between any robot parts. Therefore, each robot link is usually approximated by convex polyhedral objects, mostly spheres and ellipsoids. Collision avoidance conditions are then defined to prohibit intersection between all possible inter-robot links. Assuming five to six convex objects per robot for two robots with six joints each results in 25 to 36 constraints in the worst case. Moreover, in optimization-based planning algorithms with a rolling horizon, such as MPC, the constraints are applied along the horizon, which can increase their multiplicity to several hundred. This significantly increases the computational complexity of the underlying optimization problem and poses

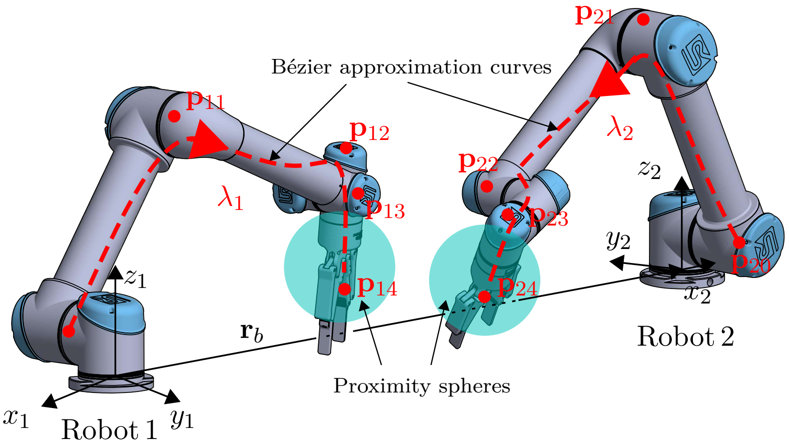


Figure 5.3: Robot geometry approximation with smooth polynomial Bézier curves. Collision avoidance conditions are defined using two sliding spheres, whose positions are updated according to the minimum distance between the corresponding linkage Bézier curves.

a substantial barrier to the fast solution of the optimization problem and, thus, to the online planning of collision-free robot trajectories. To overcome this problem, we have presented in [81] and [82] a novel approach based on a smooth approximation of the robot geometry that significantly reduces the number of collision avoidance constraints. For the approximation of the robot linkage, Bernstein basis polynomials

$$b_i^{n_r}(\lambda_r) := \begin{cases} \binom{n_r}{i} \lambda_r^i (1-\lambda_r)^{n_r-i}, & \text{for } 0 \le i \le n_r \\ 0\,, & \text{otherwise}\,, \end{cases} \tag{5.19}$$

are used, with a sequence of n_r+1 control points $\mathbf{p}_{ri}(\mathbf{q}_r) \in \mathbb{R}^3$ per robot $r \in \mathcal{R}$, resulting in a parametric curve of the form

$$\mathbf{p}_r(\lambda_r, \mathbf{q}_r) = \sum_{i=0}^{n_r} b_i^{n_r}(\lambda_r)\mathbf{p}_{ri}(\mathbf{q}_r)\,,\ 0 \le \lambda_r \le 1\,, \tag{5.20}$$

known as Bézier functions [136]. Here and in the following, we suppress the time dependence of the joint variables and other related functions for better readability until further notice.

A schematic illustration of the Bézier approximation using five control points per robot is shown in Figure 5.3. The coordinate frame $(o_1x_1y_1z_1)$ attached to the base of Robot 1 is also assumed to represent the inertial frame of reference $(o_0x_0y_0z_0)$. Robot 2 with the base frame $(o_2x_2y_2z_2)$ is displaced by $\mathbf{r}_b$ relative to the first robot and rotated by 90 degrees around the $z-$axis. For further calculations the control points $\mathbf{p}_{1i}(\mathbf{q}_1)$ and $\mathbf{p}_{2i}(\mathbf{q}_2)$ are represented in the inertial coordinate frame by

$$\begin{aligned} {}_0\mathbf{p}_{1i}(\mathbf{q}_1) &= \mathbf{p}_{1i}(\mathbf{q}_1) \ \text{ and} \\ {}_0\mathbf{p}_{2i}(\mathbf{q}_2) &= {}_0\mathbf{r}_b + \mathbf{R}_{z,\frac{\pi}{2}}\mathbf{p}_{2i}(\mathbf{q}_2)\,, \quad \forall i \in \{0, \cdots n_r\} \end{aligned} \tag{5.21}$$

using the $SO(3)$ rotation matrix $\mathbf{R}_{z,\frac{\pi}{2}}$. Thus for $n_1 = 4$, the Bézier curve of Robot 1 is given as

$$\begin{aligned} {}_0\mathbf{p}_1(\lambda_1, \mathbf{q}_1) =& (1-\lambda_1)^4 \, {}_0\mathbf{p}_{10}(\mathbf{q}_1) + 4\lambda_1(1-\lambda_1)^3 \, {}_0\mathbf{p}_{11}(\mathbf{q}_1) \\ &+ 6\lambda_1^2(1-\lambda_1)^2 \, {}_0\mathbf{p}_{12}(\mathbf{q}_1) + 4\lambda_1^3(1-\lambda_1) \, {}_0\mathbf{p}_{13}(\mathbf{q}_1) \\ &+ \lambda_1^4 \, {}_0\mathbf{p}_{14}(\mathbf{q}_1) \, , \quad 0 \leq \lambda_1 \leq 1. \end{aligned} \tag{5.22}$$

As opposed to the standard approach in the literature, we do not apply a static convex approximation of the robot geometry but introduce a dynamic approximation scheme based on the Bézier curves by continuously computing the minimum distance between them. According to the computed minimum distance, which reflects the collision potential, parts of the robots are locally approximated by spheres followed by collision-avoiding constraints, see Figure 5.3. The position of the spheres is not directly linked to the robot links but rather to the approximation curves, and their position is continuously updated depending on the robot configurations, i. e., collision potential along the robots. This results in sliding collision spheres along the robot geometries following the minimal distance between the robots. Here, the center of the spheres coincides with the points where the distance between the Bézier curves is minimal.

Minimum distance computation

The computation of maximum and minimum distance between geometric models in the Euclidean space is widely used not only in robotics for path planning and collision avoidance but also in various other fields, including computer-aided design and manufacturing (CAD / CAM), computer graphics, computer games, physical simulations, and geometric modeling [137]. Existing approaches for calculating the minimum distance between two parametric curves and surfaces can be classified into root-finding and culling-based methods [138].

An overview of different global solution techniques that can be applied to find all roots of a nonlinear system of polynomial equations is given in [139] where also a global solution method is introduced combining a subdivision algorithm with a local searching method to numerically guarantee that certain subdomains do not contain solutions. The presented iterative solution method is based on the *Projected Polyhedron* (PP) algorithm [140], which is used to isolate the roots of the polynomials. After isolating the roots, the *Newton-Raphson* method is applied to finally compute the roots to high precision.

The culling-based approach is a geometric rather than a numerical method and was originally introduced in [137] as a hierarchical framework for computing minimum distances between geometric objects. The framework includes an estimate of the lower and upper bounds of the minimum distance between nodes, followed by a subdivision of the nodes, with a node representing a curve segment or surface section resulting from the subdivision. These steps are adopted in [138] to compute the minimum distance between Bézier curves and surfaces. By using the convex hull property of the Bézier curves, a dynamic scheme for node subdivision is introduced, which can improve the convergence compared to the standard binary subdivision approach. A sweeping sphere clipping method for minimum distance computation between two Bézier curves is proposed in [141]. The squared distance function is computed in Bézier form, and the minimum distance problem is formulated as an intersection testing problem between a sphere and a curve. The geometric approaches is reported to be more robust and have a higher computational efficiency, especially when computing the minimum distance between high-order surfaces [137].

Given that information for the initial guess can be extracted from the control points of the curves, the computation of the minimum distance is formulated as an optimization problem

$$\begin{aligned} \boldsymbol{\lambda}^* = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}} \quad & \|{}_0\mathbf{p}_1(\lambda_1, \mathbf{q}_1) - {}_0\mathbf{p}_2(\lambda_2, \mathbf{q}_2)\|_2 \\ \text{s.t.} \quad & 0 \leq \boldsymbol{\lambda} \leq 1, \end{aligned} \tag{5.23}$$

with $\boldsymbol{\lambda} = [\lambda_1, \lambda_2]^{\mathrm{T}}$. To provide a good initial guess for the minimum distance, we use the heuristic approach that the closest point pair of the control points is a good approximation of the closest point pair of the underlying Bézier curves. However, this assumption does not always hold, such as when at least one edge of the control polygon defined by connecting the control points of a Bézier curve is not an edge of the convex hull formed by those control points. For a better approximation the minimum distance between the two convex hulls can also be computed by applying the *Gilbert Johnson Keerthi* (GJK) algorithm [142] as used in [138] to dynamically compute the subdivision regions. Using the indices

$$\begin{aligned} \{i^*, j^*\} = \arg\min_{i,j}\{ \; & \|{}_0\mathbf{p}_{1i}(\mathbf{q}_1) - {}_0\mathbf{p}_{2j}(\mathbf{q}_2)\|_2 \mid \\ & i \in \{0, \dots, n_1\}, \, j \in \{0, \dots, n_2\}\} \end{aligned} \tag{5.24}$$

of the closest pair of control points, the initial guess for the optimization problem (5.23) is chosen as $\boldsymbol{\lambda}_0 = [i^*/n_1, j^*/n_2]^{\mathrm{T}}$. Finally, after solving the optimization problem the distance between the centers of the two spheres corresponds to the value of the cost function

$$d = \|{}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1) - {}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)\|_2 \,. \tag{5.25}$$

Inter-robot collision avoidance

To avoid collisions between the two robots, we define, following the idea of tangent separating planes presented in [41], velocity constraints prohibiting the geometry approximating spheres to intersect with each other at any time [81]. Let $\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)$ and $\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)$ denote the centers of the spheres, see Figure 5.4, and $\mathbf{v}_1(\lambda_1^*, \mathbf{q}_1), \mathbf{v}_2(\lambda_2^*, \mathbf{q}_2)$ the respective velocities, the collision avoidance constraint forcing the spheres to stay on their respective side of the separating plane read

$$\mathbf{n}^{\mathrm{T}}(\mathbf{v}_2(\lambda_2^*, \mathbf{q}_2) - \mathbf{v}_1(\lambda_1^*, \mathbf{q}_1)) \geq \epsilon, \tag{5.26}$$

with the vector $\mathbf{n} \in \mathbb{R}^3$ normal to a tangent separating plane, and sufficiently large $\epsilon \in \mathbb{R}_{>0}$. Using the translational component of the analytical Jacobian matrices

$$\begin{aligned} \mathbf{J}_1(\mathbf{q}_1) &= \frac{\partial_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)}{\partial\mathbf{q}_1} \\ \mathbf{J}_2(\mathbf{q}_2) &= \frac{\partial_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)}{\partial\mathbf{q}_2}, \end{aligned} \tag{5.27}$$

where ${}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)$ and ${}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)$ are the vectors from the center of the inertial frame to the points $\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)$ and $\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)$, the constraint (5.26) is transformed in the robot's configuration space

$$\mathbf{n}^{\mathrm{T}}(\mathbf{J}_2(\mathbf{q}_2)\dot{\mathbf{q}}_2 - \mathbf{J}_1(\mathbf{q}_1)\dot{\mathbf{q}}_1) \geq \epsilon\,, \tag{5.28}$$

and finally expressed in the state space

$$\begin{bmatrix}\mathbf{0}^{1\times n} & -\mathbf{n}^{\mathrm{T}}\mathbf{J}_1(\mathbf{x}) & \mathbf{0}^{1\times n} & \mathbf{n}^{\mathrm{T}}\mathbf{J}_2(\mathbf{x})\end{bmatrix}\mathbf{x} \geq \epsilon\,, \tag{5.29}$$

with the state space vector $\mathbf{x} \in \mathbb{R}^{4n\times 1}$ from (5.13) and the Jacobian matrices $\mathbf{J}_1(\mathbf{x}) \in \mathbb{R}^{3\times n}$, $\mathbf{J}_2(\mathbf{x}) \in \mathbb{R}^{3\times n}$. Applying the time transformation yields the collision avoidance constraints in the scaled time τ

$$\delta_c\epsilon \leq (1-\delta_c)\epsilon + \delta_c\frac{1}{t_f}\begin{bmatrix}\mathbf{0} & -\mathbf{n}^{\mathrm{T}}\mathbf{J}_1(\mathbf{x}(k)) & \mathbf{0} & \mathbf{n}^{\mathrm{T}}\mathbf{J}_2(\mathbf{x}(k))\end{bmatrix}\mathbf{x}(k) \tag{5.30}$$

with the switching parameter

$$\delta_c = \begin{cases}1, & \text{if } d \leq d_{\mathrm{in}}\\ 0, & \text{if } d > d_{\mathrm{in}} \vee t_f^* \leq t_{\mathrm{min}}.\end{cases} \tag{5.31}$$

By introducing δ_c, the constraint is activated only if the computed minimum distance d between the Bézier curves lies within the influence distance d_{in}. Otherwise, or if the robots are close to their targets, i. e., $t_f^* \leq t_{\mathrm{min}}$, the constraint is trivially satisfied. Here, t_f^* denotes the optimal final time computed at the previous sampling time $k-1$. This is motivated by the fact that, the chosen robot targets result in feasible, i. e., collision-free, configurations by applying a scheduling algorithm presented in Section 5.2.1.

To solve the optimization problem and compute robot trajectories $\mathbf{x}$ satisfying the inter-robot collision avoidance constraints (5.29), the normal vector $\mathbf{n}$ must be predefined at each optimization step k, as it is not subject to the optimization. Therefore, we introduce the direction vector $\mathbf{n}_v$ of the relative movement of the proximity spheres,

$$\mathbf{n}_v = \frac{\Delta_0\mathbf{p}_2(\lambda_2^*) - \Delta_0\mathbf{p}_1(\lambda_1^*)}{||\Delta_0\mathbf{p}_2(\lambda_2^*) - \Delta_0\mathbf{p}_1(\lambda_1^*)||_2}\,, \tag{5.32}$$

with

$$\begin{aligned}\Delta_0\mathbf{p}_1(\lambda_1^*) &= {}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_{1f}) - {}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1),\\ \Delta_0\mathbf{p}_2(\lambda_2^*) &= {}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_{2f}) - {}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2),\end{aligned} \tag{5.33}$$

denoting the vectors from the position of the proximity spheres at the actual robot configurations $\mathbf{q}_1$ and $\mathbf{q}_2$ to the position of the spheres at the target configurations $\mathbf{q}_{1f}$ and $\mathbf{q}_{2f}$, respectively. This vector provides information about the spheres' relative direction when placing the robots from the actual to their final configuration. In fact, $\mathbf{n}_v$ describes the tentative relative movement of the spheres in Figure 5.4 during a conflict resolution phase. Roughly speaking, we hereby invoke global information referring to the relative motion of the distance spheres over the prediction of the conflict resolution horizon. Extensive simulations and experiments indicate that this alternative over the strategy based on local, i. e., the instantaneous relative motion of the collision distance spheres is more efficient and stable in resolving the conflicts [81].

In the following, two methods for choosing the normal vector $\mathbf{n}$ are presented. The first one is based on a 2D projection of the problem and is more suitable for pick-and-place robot tasks. The second method describes an optimal normal vector calculation in 3D.

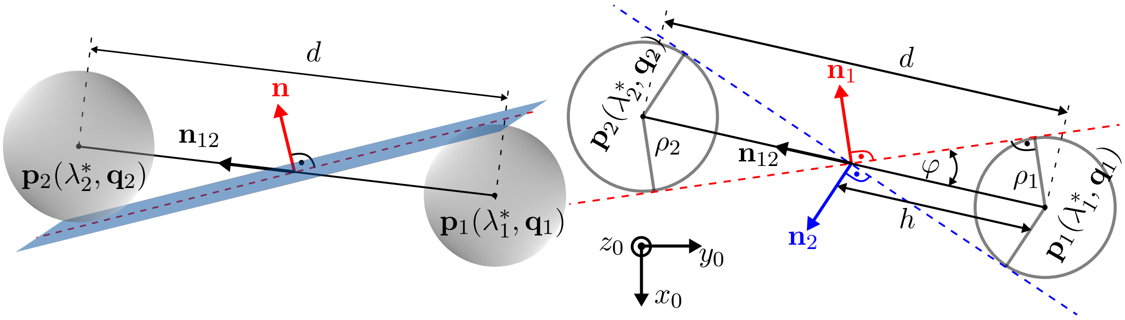


Figure 5.4: Collision spheres and a tangent separating plane. Schematic illustration of separating common tangents of the circles representing the spheres projected on the x_0y_0-plane.

1) 2D projection method

Considering the size and shape of robot manipulators in relation to the tasks to be performed, it is unlikely that feasible robot end configurations with overlapping robot arms will be found. Hence, one way of defining the normal vector $\mathbf{n}$ is to project the considered spheres onto 2D in the x_0y_0-plane, thus reducing the problem to the calculation of tangential separating lines, as shown in Figure 5.4. The vector normal to the separating tangent is then chosen to be equal to $\mathbf{n}_1$ or $\mathbf{n}_2$ depending on the relative robot movements. The collision avoidance constraints prevent the circles representing a 2D projection of the spheres from intersecting, which also results in motions that do not allow the robots to move on top of each other. As already stated, projecting the spheres onto 2D in the x_0y_0-plane results in two possible separating lines tangent to both circles, i. e., two possible perpendicular vectors $\mathbf{n} \in \{\mathbf{n}_1, \mathbf{n}_2\}$. To define the vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ perpendicular to the corresponding tangent separating lines, we conventionally use the direction of the vector between the centers of the circles

$$\mathbf{n}_{12} = \frac{{}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2) - {}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)}{d}\,, \tag{5.34}$$

as well as the length h and the resulting angle φ, defined by

$$h = \frac{\rho_1 d}{\rho_1 + \rho_2}\,, \quad \varphi = \arcsin\left(\frac{\rho_1}{h}\right), \tag{5.35}$$

according to Figure 5.4. The perpendicular vectors are then uniquely obtained by

$$\mathbf{n}_1 = \mathbf{R}^{\mathrm{T}}_{z,\frac{\pi}{2}-\varphi}\mathbf{n}_{12}\,, \quad \mathbf{n}_2 = \mathbf{R}^{\mathrm{T}}_{z,\varphi-\frac{\pi}{2}}\mathbf{n}_{12}\,. \tag{5.36}$$

The direction vector of the relative motions of the spheres (5.32) is then used to choose one of the computed perpendicular vectors and apply it to the collision avoidance constraint (5.29). If the scalar product of the vector $\mathbf{n}_v$ with one of the perpendicular vectors is positive, the vector $\mathbf{n}$ is chosen to be equal to this vector, i. e.,

$$\mathbf{n} = \mathbf{n}_1\frac{1}{2}(1 + \mathrm{sgn}(\mathbf{n}_1^{\mathrm{T}}\mathbf{n}_v)) + \mathbf{n}_2\frac{1}{2}(1 + \mathrm{sgn}(\mathbf{n}_2^{\mathrm{T}}\mathbf{n}_v))\,. \tag{5.37}$$

Otherwise, if

$$\mathrm{sgn}(\mathbf{n}_1^{\mathrm{T}}\mathbf{n}_v)\mathrm{sgn}(\mathbf{n}_2^{\mathrm{T}}\mathbf{n}_v) \geq 0, \tag{5.38}$$

holds true, the vector $\mathbf{n}$ is chosen to be equal to the vector with the smaller angle to the vector $\mathbf{n}_v$, i. e.,

$$\mathbf{n} = \mathbf{n}_1\frac{1}{2}(1 - \mathrm{sgn}(|\mathbf{n}_1^{\mathrm{T}}\mathbf{n}_v| - |\mathbf{n}_2^{\mathrm{T}}\mathbf{n}_v|)) + \mathbf{n}_2\frac{1}{2}(1 + \mathrm{sgn}(|\mathbf{n}_1^{\mathrm{T}}\mathbf{n}_v| - |\mathbf{n}_2^{\mathrm{T}}\mathbf{n}_v|))\,. \tag{5.39}$$

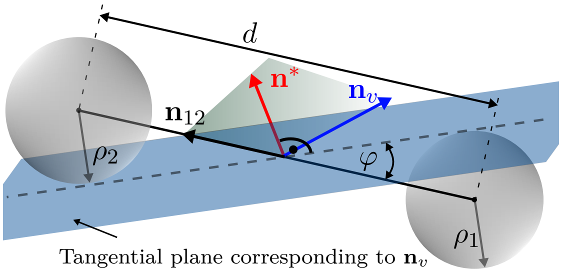


Figure 5.5: Schematic illustration in 3D: optimal vector perpendicular to a separating common tangential plane of both spheres.

2) 3D optimal perpendicular vector calculation

The proposed 2D projection method for choosing a separation tangent line between the collision avoiding objects leads to non-overlapping circles on the 2D plane. Since the circles represent the 2D projection of the collision spheres, it is evident that they can not cross over each other in a 3D space. This results in motions forcing the robots to move away from each other to avoid collisions instead of overcrossing each other. While this strategy may be intuitive for pick-and-place tasks, it can also be restrictive in general when other robot tasks are considered.

In the three-dimensional space there is an infinite number of separating planes tangential to both spheres. In this case, the direction of the relative velocity vector of the spheres (5.32) is used to define the corresponding tangential plane and compute the vector $\mathbf{n}$ perpendicular to this plane. The optimal normal $\mathbf{n}^*$ is then chosen by maximizing $\mathbf{n}_v^{\mathrm{T}}\mathbf{n}$ under the constraint of separation of solids ($\mathbf{n}_{12}^{\mathrm{T}}\mathbf{n} \geq d_{12}$) [41], i. e.,

$$\begin{aligned} \mathbf{n}^* = \underset{\mathbf{n}}{\operatorname{argmax}} \quad & \mathbf{n}_v^{\mathrm{T}}\mathbf{n} \\ \text{s.t.} \quad & \mathbf{n}^{\mathrm{T}}\mathbf{n} = 1 \\ & \mathbf{n}_{12}^{\mathrm{T}}\mathbf{n} \geq d_{12}, \end{aligned} \tag{5.40}$$

with $d_{12} = (\rho_1 + \rho_2)/d$. Using the Lagrangian function

$$L(\mathbf{n}, \mu_1, \mu_2) = \mathbf{n}_v^{\mathrm{T}}\mathbf{n} + \mu_1(\mathbf{n}^{\mathrm{T}}\mathbf{n} - 1) + \mu_2(\mathbf{n}_{12}^{\mathrm{T}}\mathbf{n} - d_{12}), \tag{5.41}$$

and applying the first optimality condition $\nabla_{\mathbf{n}} L(\mathbf{n}^*, \mu_1^*, \mu_2^*) = \mathbf{0}$, it is evident that the optimal normal $\mathbf{n}^*$ lies in the plane spanned by the vectors $\mathbf{n}_v$ and $\mathbf{n}_{12}$ (see Figure 5.5), yielding

$$\mathbf{n}^* = -\frac{1}{\mu_1^*}\mathbf{n}_v - \frac{\mu_2^*}{\mu_1^*}\mathbf{n}_{12}\,. \tag{5.42}$$

Here, μ_1^* and μ_2^* are the Langrange multipliers. Following the idea from [41], the optimal normal $\mathbf{n}^*$ is computed as

$$\mathbf{n}^* = d_{12}\mathbf{n}_{12} + \sqrt{1 - d_{12}^2}\,\frac{\mathbf{n}_v^{\perp}(\mathbf{n}_{12})}{\|\mathbf{n}_v^{\perp}(\mathbf{n}_{12})\|_2}, \tag{5.43}$$

with

$$\mathbf{n}_v^{\perp}(\mathbf{n}_{12}) = \mathbf{n}_v - \frac{\mathbf{n}_v^{\mathrm{T}}\mathbf{n}_{12}}{\mathbf{n}_{12}^{\mathrm{T}}\mathbf{n}_{12}}\mathbf{n}_{12}, \tag{5.44}$$

being the projection of the relative velocity vector $\mathbf{n}_v$ onto the plane normal to $\mathbf{n}_{12}$.

Self-robot collision avoidance

In addition to the constraint for collision avoidance between the robots, we need to further restrict the movement of the robots to avoid collisions with the static environment and self-collisions between the links of a robot. Considering pick-and-place tasks on a workbench, the constraints

$$\begin{aligned} &\mathbf{e}_3^{\mathrm{T}}\,{}_r\mathbf{p}_{r2}(\mathbf{q}_r) \geq d_{z1} \quad \text{and} \\ &\mathbf{e}_3^{\mathrm{T}}\,{}_r\mathbf{p}_{r4}(\mathbf{q}_r) \geq d_{z2}\,, \quad r \in \mathcal{R}\,, \end{aligned} \tag{5.45}$$

are introduced to limit the respective robot's working area to a certain minimum operating height and avoid collisions with the bench. Here, ${}_r\mathbf{p}_{r2}(\mathbf{q}_r)$ and ${}_r\mathbf{p}_{r4}(\mathbf{q}_r)$ denote the vectors from the origin of the respective base frame of the robots to the control points in the area of the robot wrist and end-effector, see Figure 5.3. $\mathbf{e}_3^{\mathrm{T}} = [0, 0, 1]$ is the unit vector, and d_{z1}, d_{z2} are distance parameters defining the minimum operating height along the $z-$axis. Furthermore, to avoid collisions between the gripper and the robot's shoulder, we restrict the distance between the control points of the basis and the end-effector to a minimum distance d_s by applying constraint

$$||{}_r\mathbf{p}_{r4}(\mathbf{q}_r) - {}_r\mathbf{p}_{r0}(\mathbf{q}_r)||_2 \geq d_s\,, \quad r \in \mathcal{R}\,. \tag{5.46}$$

Again, ${}_r\mathbf{p}_{r0}(\mathbf{q}_r)$ and ${}_r\mathbf{p}_{r4}(\mathbf{q}_r)$ denote the vectors from the origin of the base frame of the robots to the first and last control points modeling the robot kinematic chain. In order to avoid self-collisions on the wrist robot area, the motions of robot joints four and five are further restricted using the constraints

$$\sin(\mathbf{e}_4^{\mathrm{T}}\mathbf{q}_r)|\cos(\mathbf{e}_5^{\mathrm{T}}\mathbf{q}_r)| \leq \varepsilon\,, \quad r \in \mathcal{R}\,. \tag{5.47}$$

with the joint angles $\mathbf{e}_4^{\mathrm{T}}\mathbf{q}_r = q_{r4}$, $\mathbf{e}_5^{\mathrm{T}}\mathbf{q}_r = q_{r5}$, and a sufficiently large $\varepsilon \in \mathbb{R}_{>0}$. Additional linear constraints are considered in form of minimum and maximum bounds of the robot joints, i. e., $\underline{\mathbf{x}} \leq \mathbf{x} \leq \bar{\mathbf{x}}$.

5.2.2.3 Time-optimal MPC

The centralized trajectory planning algorithm (5.14) is transformed in the scaled time τ by applying the time transformation (4.26) and discretized using Picard's iteration method, yielding a discrete fixed terminal time optimization problem. Finally, the resulting MPC algorithm for recurrent time-optimal robot trajectory planning can be written as a static optimization problem in the form

$$\min_{\mathbf{u}(\cdot),\,t_f} \quad t_f \tag{5.48}$$

$$\text{s.t.} \quad \mathbf{x}(j+1\,|\,k) = (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}(j\,|\,k) + \Delta\tau t_f^2\Big(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\Big)\mathbf{B}\mathbf{u}(j\,|\,k) \tag{5.48a}$$

$$\operatorname{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\underline{\mathbf{x}} \leq \mathbf{x}(j+1\,|\,k) \leq \operatorname{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\bar{\mathbf{x}} \tag{5.48b}$$

$$\begin{aligned} &\underline{\mathbf{u}} \leq \mathbf{u}(j+1\,|\,k) \leq \bar{\mathbf{u}} \\ &\underline{\dot{\mathbf{u}}} \leq (\mathbf{u}(j+1\,|\,k) - \mathbf{u}(j\,|\,k))/(t_f\Delta\tau) \leq \bar{\dot{\mathbf{u}}} \end{aligned} \tag{5.48c}$$

$$\begin{aligned} &\mathbf{x}(0\,|\,k) = \operatorname{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\mathbf{x}_0(k) \\ &\mathbf{x}(N_T\,|\,k) = \operatorname{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\mathbf{x}_f(k) \end{aligned} \tag{5.48d}$$

$$\begin{aligned} &\mathbf{e}_3^{\mathrm{T}}\,{}_r\mathbf{p}_{r2}(\mathbf{x}(j+1|k)) \geq d_{z1} \\ &\mathbf{e}_3^{\mathrm{T}}\,{}_r\mathbf{p}_{r4}(\mathbf{x}(j+1|k)) \geq d_{z2} \end{aligned} \tag{5.48e}$$

$$||{}_r\mathbf{p}_{r4}(\mathbf{x}(j+1|k)) - {}_r\mathbf{p}_{r0}(\mathbf{x}(j+1|k))||_2 \geq d_s$$

$$\begin{aligned} &\sin\left(\mathbf{e}_4^{\mathrm{T}}\mathbf{x}(j+1|k)\right)|\cos\left(\mathbf{e}_5^{\mathrm{T}}\mathbf{x}(j+1|k)\right)| \leq \varepsilon \\ &\sin\left(\mathbf{e}_{16}^{\mathrm{T}}\mathbf{x}(j+1|k)\right)|\cos\left(\mathbf{e}_{17}^{\mathrm{T}}\mathbf{x}(j+1|k)\right)| \leq \varepsilon \end{aligned} \tag{5.48f}$$

$$\begin{aligned} \delta_c \epsilon \leq\, &(1-\delta_c) \\ &+ \delta_c \frac{1}{t_f}\left[\mathbf{0}\ \ -\mathbf{n}^{\mathrm{T}}\mathbf{J}_1(\mathbf{x}(\bar{j}+1\,|\,k))\ \ \mathbf{0}\ \ \mathbf{n}^{\mathrm{T}}\mathbf{J}_2(\mathbf{x}(\bar{j}+1\,|\,k))\right]\mathbf{x}(\bar{j}+1\,|\,k), \end{aligned} \tag{5.48g}$$

where $j \in \{0, \ldots, N_T - 1\}$ is the iteration index, N_T the prediction horizon, and $r \in \mathcal{R}$ the index denoting the robots. In each optimization step k, the time minimizing static optimization problem is solved to generate robot trajectories from the current measured state $\mathbf{x}_0(k)$ to a desired final state $\mathbf{x}_f(k)$, as expressed by the initial and terminal constraints (5.48d). Note that both robots share the same terminal constraints imposing a simultaneous task execution by forcing them to reach their targets at the same time $t = t_f$ at the end of the trajectory planning horizon N_T, which corresponds to the scaled time $\tau = 1$. The scaled discrete-time dynamic model (5.48a) with the sampling time $\Delta\tau = 1/N_T$ is used to predict robot trajectories along the prediction horizon. The optimization is also subject to minimum / maximum bounds on the states (5.48b), the inputs and their time derivatives (5.48c). The time transformation is applied to the upper and lower bounds of the joint's velocities, yielding

$$\operatorname{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\underline{\mathbf{x}} \leq \mathbf{x}(k) \leq \operatorname{diag}(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I})\bar{\mathbf{x}}\,, \tag{5.49}$$

with $\underline{\mathbf{x}}^{\mathrm{T}} = [\underline{\mathbf{q}}^{\mathrm{T}}, \underline{\dot{\mathbf{q}}}^{\mathrm{T}}, \underline{\mathbf{q}}^{\mathrm{T}}, \underline{\dot{\mathbf{q}}}^{\mathrm{T}}]$ and $\bar{\mathbf{x}}^{\mathrm{T}} = [\bar{\mathbf{q}}^{\mathrm{T}}, \bar{\dot{\mathbf{q}}}^{\mathrm{T}}, \bar{\mathbf{q}}^{\mathrm{T}}, \bar{\dot{\mathbf{q}}}^{\mathrm{T}}]$. Here, $\mathbf{I} \in \mathbb{R}^{n \times n}$ denotes the identity matrix, $\underline{\mathbf{q}} \in \mathbb{R}^n$, $\underline{\dot{\mathbf{q}}} \in \mathbb{R}^n$ the lower bounds, and $\bar{\mathbf{q}} \in \mathbb{R}^n$, $\bar{\dot{\mathbf{q}}} \in \mathbb{R}^n$ the upper bounds of the position and velocity of the robot joints in the time t. Similarly, the initial and terminal constraints are also transformed in the scaled time τ.

The constraints to avoid collisions with the workbench (5.48e) and between the robot's links (5.48f) are always active along the entire prediction horizon. Here, $\mathbf{e}_4^{\mathrm{T}}\mathbf{x}(\cdot) = q_{14}(\cdot)$, $\mathbf{e}_5^{\mathrm{T}}\mathbf{x}(\cdot) = q_{15}(\cdot)$, and $\mathbf{e}_{16}^{\mathrm{T}}\mathbf{x}(\cdot) = q_{24}(\cdot)$, $\mathbf{e}_{17}^{\mathrm{T}}\mathbf{x}(\cdot) = q_{25}(\cdot)$ represent the variables of the robot's joints four and five, see (5.47). The inter-robot collision avoidance constraints (5.48g), on the other side, are applied only along a part of the prediction horizon N_T, as denoted by the index $\bar{j} = j \in \{0, \ldots, N_C\}$ with $N_C < N_T$. This is motivated by the fact that robot trajectories are iteratively replanned at each optimization step k. Therefore, it is sufficient to ensure collision-free planning along a section of the prediction horizon that is sufficiently larger than the control horizon.

The scaled time interval $\tau \in [0, 1]$ is subdivided into N_T equidistant time intervals $\Delta\tau$, which are mapped by $\Delta t = \Delta\tau t_f^*$ into increasingly tighter Δt-intervals as time propagates, since the computed optimal final time t_f^* reduces with each optimization step. This results in robot trajectories with varying time resolution, which increases as the robots approach their targets while the prediction horizon remains the same.

5.2.3 Distributed trajectory planning

The proposed centralized MPC planning layer results in a synchronous robot task execution, with the robots reaching the assigned tasks simultaneously. While cooperatively executing pick-and-place tasks, a synchronous robot operation is useful in coordinating the robot motions, and it simplifies the scheduling problem since the rescheduling for both robots takes place simultaneously and, thus, less frequently. In addition, robot coordination and collision avoidance are less elaborate and require less computing power compared to an unsynchronized approach. On the other hand, synchronization entails a loss of performance since one of the robots may work slower than it eventually could. Theoretical and experimental analyses indicate that this does not significantly impair solution quality [143].

Time synchronization is not inherently given in the distributed case compared to the centralized MPC approach since each robot plans its trajectory locally. Therefore, coupling in the cost functions of the distributed optimization algorithms is introduced to synchronize the robot motions in order to reach their targets simultaneously. Additionally, coupling in the constraints is introduced to impose collision avoidance restrictions. This leads to DMPC algorithms of decoupled systems with coupling in the cost functions and the constraints. At each MPC iteration, the controllers share the calculated minimum time and their predicted trajectories.

The main feature of the proposed DMPC is that each of the local controllers optimizes its cost function, i. e., the local robot transition time $t_{rf}, r \in \mathcal{R}$, while additionally taking into account the previously achieved optimal value of the local cost function of the other controller. By coupling the cost functions, it is possible to coordinate the robots' motions in time so that they perform their tasks simultaneously following the synchronous approach. The time synchronization can be ensured by choosing the cost function for Robot 1 as

$$J_1(t_{1f}) = \left(t_{1f} - \hat{t}_{1f}\delta_t\left(\frac{\hat{t}_{2f}}{\hat{t}_{1f}}\right)\right)^2 , \tag{5.50}$$

with the switch function

$$\delta_t\left(\frac{\hat{t}_{2f}}{\hat{t}_{1f}}\right) = \frac{2}{1 + e^{-\tilde{\alpha}\left(\frac{\hat{t}_{2f}}{\hat{t}_{1f}} - 1\right)}} - 1 \, . \tag{5.51}$$

Here, t_{1f} describes the optimization variable, and $\hat{t}_{1f}, \hat{t}_{2f}$ are the previously calculated optimal values of the cost functions of the respective controllers. The cost function for Robot 2 is defined analogously. If Robot 1 requires less time to reach its target than Robot 2, i. e., $\hat{t}_{2f} > \hat{t}_{1f}$, then δ_t becomes positive for the Robot 1 and negative for Robot 2, see Figure 5.6. In the current optimization step, $\delta_t > 0$ implies that the minimum point of the cost function (5.50) is shifted to $\hat{t}_{1f}\delta_t$. Consequently, depending on δ_t, the faster robot is required to have a minimum time t^*_{1f} that is not far from the minimum time $\hat{t}_{1f}$ of the preceding optimization step. The faster robot is thus slightly slowed down in each optimization step as long as the optimal values of the cost functions do not converge toward the same point. On the other hand, $\delta_t < 0$ in the cost function $J_2(t_{2f})$ of the second robot does not influence the optimization. Therefore, Robot 2 continues its attempt to reach the target as quickly as possible.

Note that in the proposed distributed approach, the controllers exchange the optimal values of their cost functions and the resulting optimal trajectories at each optimization

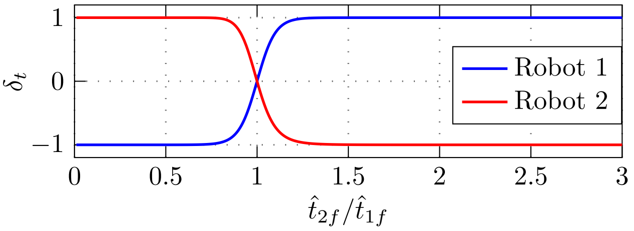


Figure 5.6: Switch functions for $\tilde{\alpha} = 20$ to coordinate the robots movements ensuring that they reach their targets simultaneously. The switch functions show a symmetrical behavior to each other since only one of the robots, the faster one, should be slowed down without any influence on the other.

step. While the calculated optimal times are needed in the cost functions, the predicted trajectories are used in the constraints to establish the conditions for avoiding collisions between the robots. This results in a DMPC algorithm of the decoupled systems with coupled cost functions and constraints. At each iteration k of the optimization, each controller optimizes its own set of inputs keeping the information needed from the other controller constant as previously received.

With the state space vectors $\mathbf{x}_1 \in \mathbb{R}^{2n}$, $\mathbf{x}_2 \in \mathbb{R}^{2n}$ from (5.13), and the input vector $\mathbf{u}_1 \in \mathbb{R}^n$, the trajectory planning problem for Robot 1 reads as follows

$$\min_{\mathbf{u}_1(\cdot),\, t_{1f}} \quad J_1(t_{1f}) \tag{5.52}$$

$$\text{s.t.} \quad \mathbf{x}_1(j+1\,|\,k) = (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}_1(j\,|\,k) + \Delta\tau t_{1f}^2\left(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\right)\mathbf{B}\mathbf{u}_1(j\,|\,k) \tag{5.52a}$$

$$\operatorname{diag}(\mathbf{I}, t_{1f}\mathbf{I})\underline{\mathbf{x}}_1 \leq \mathbf{x}_1(j+1\,|\,k) \leq \operatorname{diag}(\mathbf{I}, t_{1f}\mathbf{I})\bar{\mathbf{x}}_1 \tag{5.52b}$$

$$\begin{aligned} &\underline{\mathbf{u}}_1 \leq \mathbf{u}_1(j+1\,|\,k) \leq \bar{\mathbf{u}}_1 \\ &\underline{\dot{\mathbf{u}}}_1 \leq (\mathbf{u}_1(j+1\,|\,k) - \mathbf{u}_1(j\,|\,k))/(t_{1f}\Delta\tau) \leq \bar{\dot{\mathbf{u}}}_1 \end{aligned} \tag{5.52c}$$

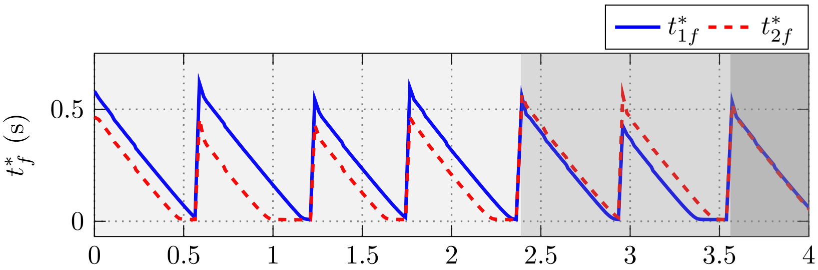
$$\begin{aligned} \mathbf{x}_1(0\,|\,k) &= \operatorname{diag}(\mathbf{I}, t_{1f}\mathbf{I})\mathbf{x}_{10}(k) \\ \mathbf{x}_1(N_T\,|\,k) &= \operatorname{diag}(\mathbf{I}, t_{1f}\mathbf{I})\mathbf{x}_{1f}(k) \end{aligned} \tag{5.52d}$$

$$\begin{aligned} &\mathbf{e}_3^{\mathrm{T}}\,{}_1\mathbf{p}_{12}(\mathbf{x}_1(j+1|k)) \geq d_{z1} \\ &\mathbf{e}_3^{\mathrm{T}}\,{}_1\mathbf{p}_{14}(\mathbf{x}_1(j+1|k)) \geq d_{z2} \end{aligned} \tag{5.52e}$$

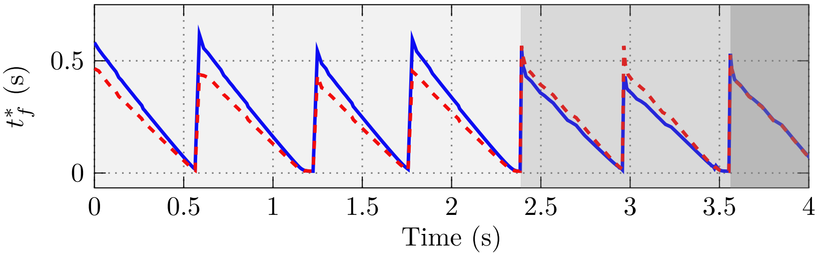
$$\begin{aligned} &||{}_1\mathbf{p}_{14}(\mathbf{x}_1(j+1|k)) - {}_1\mathbf{p}_{10}(\mathbf{x}_1(j+1|k))||_2 \geq d_s \\ &\sin(\mathbf{e}_4^{\mathrm{T}}\mathbf{x}_1(j+1|k))|\cos(\mathbf{e}_5^{\mathrm{T}}\mathbf{x}_1(j+1|k))| \leq \varepsilon \end{aligned} \tag{5.52f}$$

$$\begin{aligned} \delta_c\epsilon \leq{} & (1-\delta_c) \\ & + \delta_c \begin{bmatrix} \mathbf{0} & -\dfrac{\mathbf{n}^{\mathrm{T}}\mathbf{J}_1(\mathbf{x}_1(\bar{j}+1\,|\,k))}{t_{1f}} & \mathbf{0} & \dfrac{\mathbf{n}^{\mathrm{T}}\mathbf{J}_2(\hat{\mathbf{x}}_2(\bar{j}+1\,|\,k-1))}{\hat{t}_{2f}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1(\bar{j}+1\,|\,k) \\ \hat{\mathbf{x}}_2(\bar{j}+1\,|\,k-1) \end{bmatrix}. \end{aligned} \tag{5.52g}$$

Here, the prediction model (5.52a) corresponds to the prediction model (4.29) of a single robot manipulator. The optimization problem is similar to the centralized trajectory planning problem (5.48), which jointly plans trajectories for both robots. As previously mentioned, in the distributed scheme robots share the computed optimal values of the respective cost functions and the planned trajectories to coordinate their motions



(a) Unsynchronized trajectory planning ($\delta_t = 0$).



(b) Synchronized trajectory planning ($\delta_t \neq 0$).

Figure 5.7: Comparison of the calculated minimum times t_f^* of the two robots for both cases, unsynchronized (a) and synchronized (b). As a result of the time synchronization, the difference between the calculated optimal times decreases with each optimization step until they finally equalize. Due to the synchronization, the robots will reach their targets simultaneously.

for synchronous and collision-free task execution. Therefore, the inter-robot collision avoidance constraints (5.52g) involve the trajectory of Robot 2, denoted by $\hat{\mathbf{x}}_2(k-1)$, which represents the optimal robot trajectory computed at the previous planning step $k-1$. The local controller plans the trajectory for Robot 1 from its actual state $\mathbf{x}_{10}(k)$ to the desired final state $\mathbf{x}_{1f}(k)$, taking into account the collision avoidance constraints (5.52e) and (5.52g), and respecting the robot limitations expressed as lower and upper bounds on the state variables (5.52b), and the input variables including their time derivatives (5.52c). The trajectory planning problem for Robot 2 is defined analogously.

In order to analyze the performance of the proposed time synchronization, simulation results are presented next, with the robot arms performing pick-and-place tasks in a confined working place. At the beginning of the trajectory planning, the computed minimum time reaches a maximum value corresponding to the assigned task and decreases with each MPC step as the robot approaches its target with each applied control input. The minimum final time becomes nearly zero when the robot reaches the target, triggering the re-execution of the planning algorithm, and thus leading the robot to the next destination. As can be seen in Figure 5.7, this results in a sawtooth-like progression of the optimal time over the simulation duration. According to the results, Robot 1 needs more time to reach the first target than Robot 2, i. e., $t_{1f}^* > t_{2f}^*$.

In case the robots' trajectories are not synchronized, i. e., $\delta_t = 0$, Robot 2 will reach its first target faster than Robot 1. This is due to the fact that for the selected targets, the second robot has to cover a shorter distance compared to the first one. Robot 2 will reach the first task point after approximately 0.46 s and the following task point after 1 s, whereas Robot 1 after 0.58 s and 1.2 s, respectively, see Figure 5.7a). Since it is required that the robots perform their tasks simultaneously, the faster robot is gradually slowed down until the final times become equal. Synchronizing the trajectories in time, results in a lower slope of the optimal time curve of the slower robot compared to the

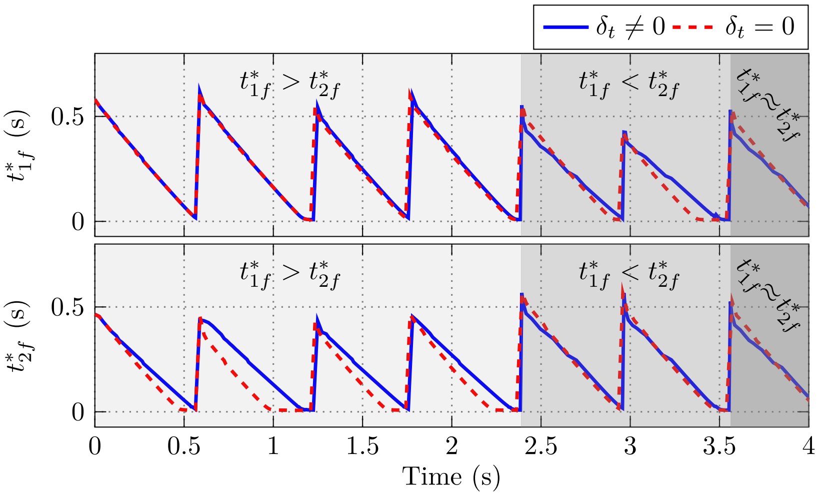


Figure 5.8: Comparison between the unsynchronized ($\delta_t = 0$) and synchronized ($\delta_t \neq 0$) case for each robot individually. For $t^*_{1f} > t^*_{2f}$, the time synchronization does not affect the slower Robot 1 and vise-versa in case of Robot 2 for $t^*_{2f} > t^*_{1f}$. The time synchronization ensures that the optimal time of the faster robot changes more slowly in each optimization step until it matches the time of the slower robot. For $t^*_{1f} \approx t^*_{2f}$, i. e., $\delta_t \approx 0$, both robots try to reach their targets as fast as possible (see also Figure 5.7).

uncoordinated case, see Figure 5.7b). As long as the calculated optimal times of the two robots differ, the faster robot is slowed down slightly in each step until the final times converge towards the same value. Nevertheless, this does not affect the slower robot, as it must not be slowed down any further. This can be seen by the comparison between the synchronized ($\delta_t \neq 0$) and the unsynchronized ($\delta_t = 0$) trajectory planning for both robots as shown in Figure 5.7 and Figure 5.8. For the slower robot (Robot 1 for $t^*_{1f} > t^*_{2f}$ and Robot 2 for $t^*_{2f} > t^*_{1f}$) the outcome of the optimization for the synchronized and the unsynchronized planning is the same. For the faster robot, on the other hand, the decrease of the final time becomes smaller due to the synchronization of the trajectory planning. The difference between the optimal final times (t^*_{1f}, t^*_{2f}) of the robots becomes smaller with each iteration until they finally approach to zero simultaneously. In case the robots need about the same time to reach their targets, i. e., $t^*_{1f} \approx t^*_{2f}$, the synchronization has no influence on the optimal final times, and both robots try to achieve their goals as fast as possible.

5.3 Experimental results

The system and control architecture of the experimental setup with two robotic manipulators used to validate the performance of the proposed hierarchical control structure is shown in Figure 5.9 for the case of the centralized trajectory planning layer. The scheduler gets information about the position and orientation of the task points and computes feasible robot targets, which are transformed into final robot joint positions $\mathbf{q}_{rf}$ and velocities $\dot{\mathbf{q}}_{rf}$. ROS Hardware Interface is used to control the grippers via the input / output robot interface. With the measured actual robot state and the given final target state, the trajectories are predicted along the prediction horizon using (5.18) and provided as an initial guess to the MPC algorithm to initialize the motion planning. The input vector $\mathbf{u}(k)$ needed to predict trajectories is approximated by a bang-bang

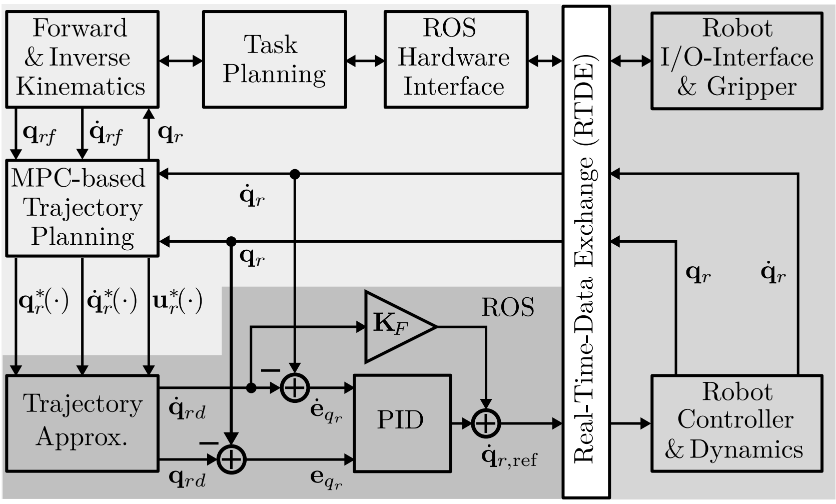


Figure 5.9: System architecture consisting of two robotic arms inclusive controllers, grippers, a real-time-communication unit, and a Host-PC. The minimum-time planning algorithms inclusive forward and inverse kinematics are implemented on the Host-PC. The generated trajectories $\mathbf{q}_r^*(\cdot) = [\mathbf{q}_r^*(0), \dots, \mathbf{q}_r^*(N_T)]$, $\dot{\mathbf{q}}_r^*(\cdot) = [\dot{\mathbf{q}}_r^*(0), \dots, \dot{\mathbf{q}}_r^*(N_T)]$, and $\mathbf{u}_r^*(\cdot) = [\mathbf{u}_r^*(0), \dots, \mathbf{u}_r^*(N_T)], r \in \mathcal{R}$, are approximated by quintic polynomials and forwarded to the low-level robot controller via a PID-feedback controller in combination with a feedforward term using ROS and a Velocity Control Interface.

acceleration / deceleration profile, typical for time-optimal planning. The CasADi [129] framework is used to model the optimization problems. The bilinear binary scheduling problem is solved applying the GUROBI solver with a feasibility tolerance of 10^{-6}. The nonlinear optimization problems for minimum-distance calculation (5.23) and MPC-based trajectory planning (5.48) are solved using IPOPT [130] with the linear solver MA57 [144] for the solution of the underlying linear system for step computations. The maximum number of iterations is limited to 100 with an overall acceptable relative convergence tolerance of 10^{-6}. Refer to Implementation Section 4.4 and [145] for more information about implementing the algorithms using ROS.

The parameters used during the experiments are listed in Table 5.1. For the presented results a prediction horizon of $N_T = 10$ is chosen, and the constraints for inter-robot collision avoidance (5.48g) are applied along its first half, i. e., $N_C = 5$. The velocity parameter ϵ is chosen to be updated dynamically using $\epsilon = 0.24\, t_f^*$, depending on the value of the minimum final time t_f^* (cost function) computed in the previous planning iteration. The value of the optimal final time may locally increase during a collision avoidance phase as the robots slow down their motions, leading to a higher time to reach the targets. In this case, a larger ϵ-value is beneficial to resolve a conflict situation. Contrary, in less critical scenarios with low collision potential, the final time decreases faster, also relaxing the inter-robot collision avoidance constraints by reducing ϵ. Considering the shape and size of robot's links, the radius of the geometry approximating spheres is chosen to be constant and equal for both robots. However, a varying sphere radius can also be considered as a function of the path variable λ.

The choice of the prediction horizons N_T is related to the expected maximum value for the cost function $t_{f,\max}$, which for the considered tasks lies between 0.6 s-1.2 s. In the scaled time τ, the prediction horizon covers the entire period until the final time point and is subdivided into N_T equidistant intervals $\Delta\tau$. The interval boundaries serve as

Table 5.1: Parameters used for experiments.

Description	Parameter	Value	Unit
Prediction horizon	N_T	10	
Collision-free horizon	N_C	5	
Influence distance	d_{in}	0.38	m
Minimum distance	d_{min}	0.30	m
Sphere radius	$\rho_1 = \rho_2$	0.11	m
Self collision angular parameter	ε	0.15	
Self collision distance parameter	d_{z1}	0.10	m
Self collision distance parameter	d_{z2}	0.35	m
Self collision distance parameter	d_s	0.18	m
Velocity parameter	ϵ	$0.24\,t_f^*$	m/s
Minimum time	t_{min}	0.2	s
Joint velocity limits	$\underline{\dot{q}}$, $\bar{\dot{q}}$	$\mp\pi$	rad/s
Joint acceleration limits	$\underline{u}$, $\bar{u}$	$\mp 3\pi/2$	rad/s^2
Joint jerk limits	$\underline{\dot{u}}$, $\bar{\dot{u}}$	∓ 5	rad/s^3

control points reshaping the trajectories after each MPC iteration. The aim is to achieve a control point distribution at $t_{f,\text{max}}$ in time steps of about 100 ms, i. e.,

$$\Delta t_{\text{max}} = \Delta\tau t_{f,\text{max}} = 100\,\text{ms}\,, \quad \text{with} \quad \Delta\tau = \frac{1}{N_T}\,. \tag{5.53}$$

Considering on average $t_{f,\text{max}} \approx 1\,\text{s}$, the prediction horizon is chosen as

$$N_T = \left\lfloor \frac{t_{f,\text{max}}}{\Delta t_{\text{max}}} \right\rfloor = 10\,, \tag{5.54}$$

with $\lfloor\cdot\rfloor$ denoting the integer (floor) operator. Since the time interval $\Delta t = \Delta\tau t_f^*$ is becoming shorter with decreasing t_f^*, the collision-free horizon N_C has to be sufficiently large to ensure collision-free operation along the entire control horizon t_c, which is mainly defined by the computation time. Here, it is referred to the minimum time $t_{\text{min}} = 200\,\text{ms}$ which denotes that for $t_f^* < t_{\text{min}}$ the robots are near their target points (approximately two control horizons far). Therefore, for $t_c \leq \Delta t_{\text{max}}$, it is sufficient to ensure that at least one control horizon is collision-free, i. e., $\Delta\tau t_{\text{min}} N_C \geq \Delta t_{\text{max}}$, yielding

$$N_C \geq \frac{\Delta t_{\text{max}}}{t_{\text{min}}} N_T\,. \tag{5.55}$$

Choosing $N_C = N_T/2$ satisfies equation (5.55) for the used time parameters.

For the considered test case with twelve objects and slots shown in Figure 5.1, the task points are chosen af follows

$$\begin{aligned}
&\mathcal{O}_{\text{red}} = \{o_1, o_4\}, \quad \mathcal{O}_{\text{green}} = \{o_2, o_5\}, \quad \mathcal{O}_{\text{blue}} = \{o_7, o_{11}\}, \\
&\qquad \mathcal{O}_{\text{white}} = \{o_9, o_{10}\}, \quad \mathcal{O}_{\text{black}} = \{o_3, o_6, o_8, o_{12}\} \\
&\mathcal{S}_{\text{red}} = \{s_{11}, s_{12}\}, \quad \mathcal{S}_{\text{green}} = \{s_7, s_8\}, \quad \mathcal{S}_{\text{blue}} = \{s_5, s_6\}, \\
&\qquad \mathcal{S}_{\text{white}} = \{s_1, o_2\}, \quad \mathcal{S}_{\text{black}} = \{s_3, s_4, s_9, s_{10}\}\,.
\end{aligned} \tag{5.56}$$

For simplicity, and since the effect of different grippe orientation is described in the previous chapter, the orientation of the gripper while picking and placing the objects is chosen to remain equal to the initial orientation. Applying the minimum-time scheduling algorithm (5.8), yields the optimal task execution sequence for Robot 1

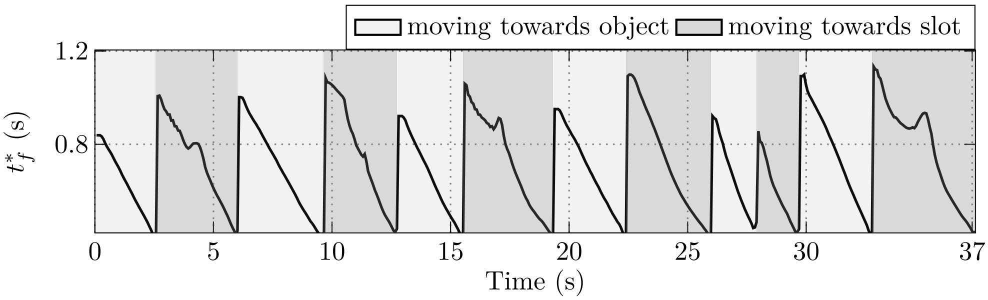
$$\{o_3, s_{10}, o_9, s_1, o_1, s_{11}, o_{10}, s_2, o_6, s_4, o_4, s_{12}\} \tag{5.57}$$

and Robot 2

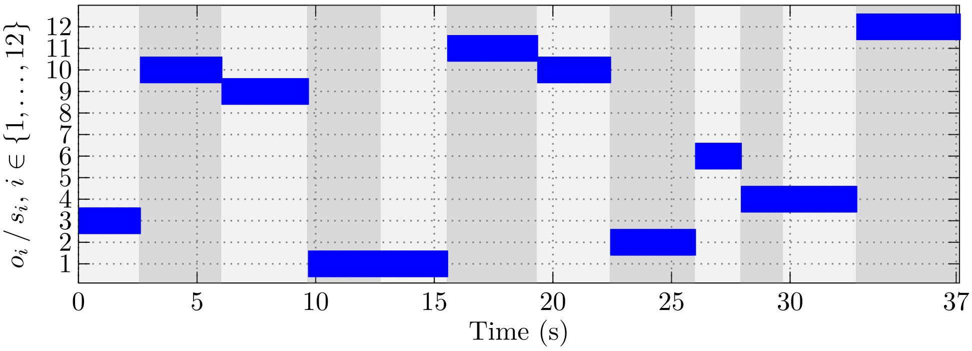
$$\{o_8, s_3, o_2, s_7, o_7, s_5, o_5, s_8, o_{12}, s_9, o_{11}, s_6\}\,. \tag{5.58}$$

which are graphically displayed in Figure 5.10 along with the computed minimum final time t_f^*.

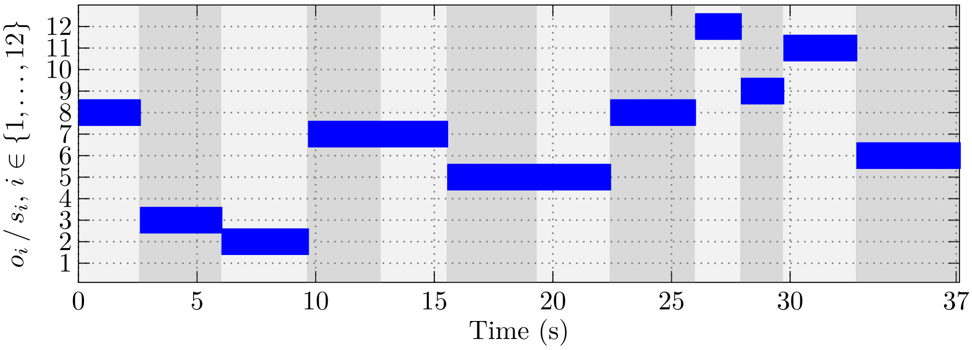
For the generation of collision-free robot trajectories, the centralized MPC approach (5.48) is used. Since the robots execute the tasks simultaneously, they share a common final time t_f^*. Starting from the initial robot position, Robot 1 will pick up first Object $o_3 \in \mathcal{O}_{\text{black}}$ and place it in Slot $s_{10} \in \mathcal{S}_{\text{black}}$. During the same time, Robot 2 will pick up Object $o_8 \in \mathcal{O}_{\text{black}}$ and place it in Slot $s_3 \in \mathcal{S}_{\text{black}}$. After placing the first objects,



a) Computed minimum final time t_f^*.



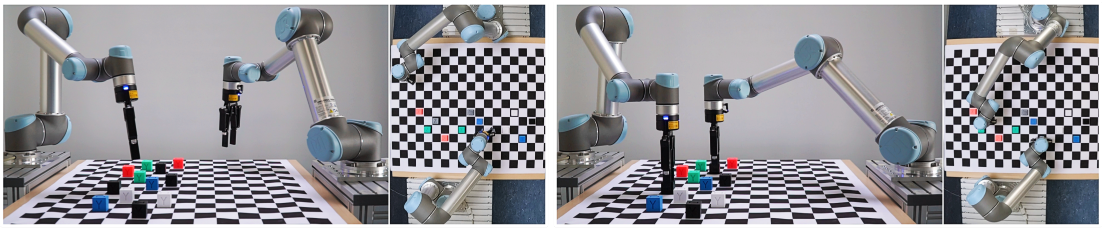
b) Task sequence for Robot 1: $\{o_3, s_{10}, o_9, s_1, o_1, s_{11}, o_{10}, s_2, o_6, s_4, o_4, s_{12}\}$.



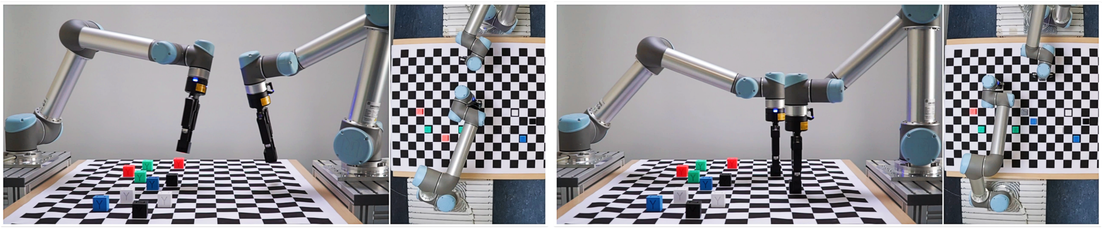
c) Task sequence for Robot 2: $\{o_8, s_3, o_2, s_7, o_7, s_5, o_5, s_8, o_{12}, s_9, o_{11}, s_6\}$.

Figure 5.10: Computed minimum final time t_f^*, and obtained time-optimal task sequences for Robot 1 and Robot 2.

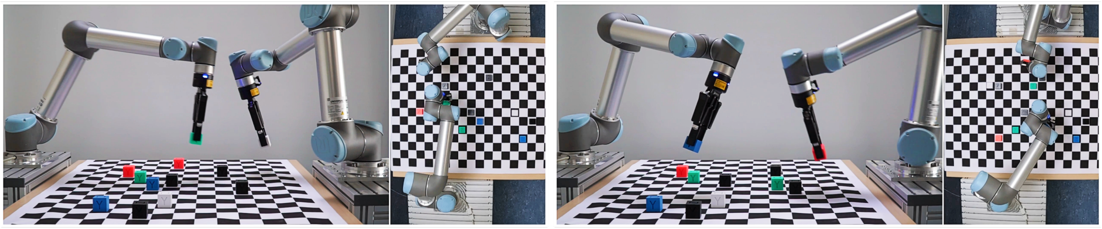
Robot 1 will pick up $o_9 \in \mathcal{O}_{\text{white}}$ and place it in $s_1 \in \mathcal{O}_{\text{white}}$, while Robot 2 will pick up $o_2 \in \mathcal{O}_{\text{green}}$ and place it in $s_3 \in \mathcal{S}_{\text{green}}$, see Figure 5.11. The objects are placed in the slots of the same class, resulting, for the chosen object and slot distribution, in motions where the robots have to avoid each other when moving from the objects to the slots to ensure collision-free operation. Therefore, the collision-avoidance constraints using the 2D projection method are applied, forcing the robots to move aside from each other.



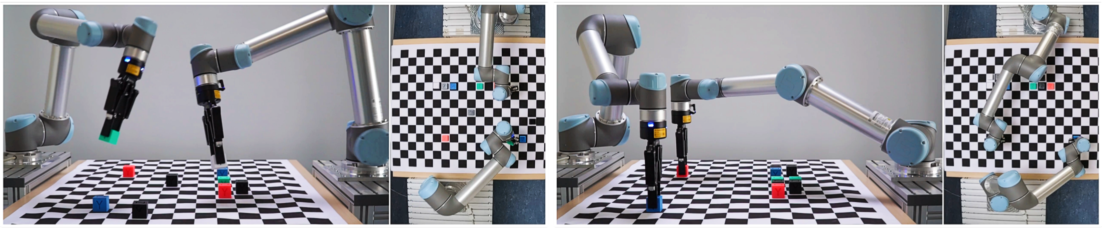
a) Initial robot position. b) Picking up first two objects $\{o_3, o_8\} \subset \mathcal{O}_{\text{black}}$.



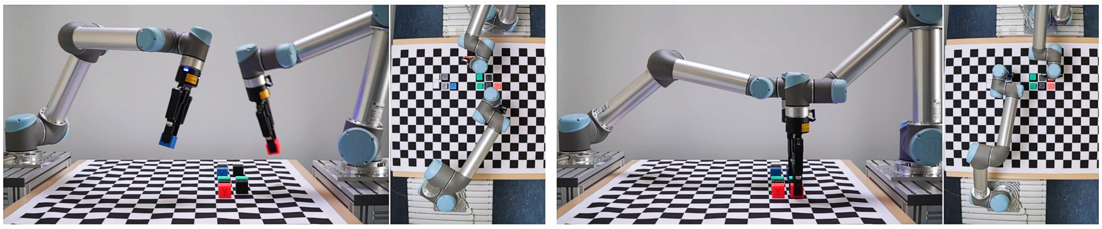
c) Collision avoidance on the way to the first slots. d) Placing objects in the slots $\{s_3, s_{10}\} \subset \mathcal{S}_{\text{black}}$.



e) Collision avoidance on the way to the next slots. f) Collision avoidance on the way to the slots.



g) Collision avoidance on the way to the slots. h) Picking up the last two objects $o_4 \in \mathcal{O}_{\text{red}}, o_{11} \in \mathcal{O}_{\text{blue}}$.



i) Collision avoidance on the way to the last slots. j) Placing last two objects in $s_6 \in \mathcal{S}_{\text{blue}}, s_{12} \in \mathcal{S}_{\text{red}}$.

Figure 5.11: The pictures show a side and top view of the experiment. Scenes a)-d) show an entire pick-and-place cycle starting from the initial point a), picking up the assigned objects b), avoiding collisions on the way to the slots c), and placing the objects in the slots d). The pictures e)-g) show different collision avoidance scenes when the robots are moving from the objects to the slots. The last pictures h)-j) show the completing final sequence of picking the objects $o_4 \in \mathcal{O}_{\text{red}}$ and $o_{11} \in \mathcal{O}_{\text{blue}}$, and placing them in the corresponding slots $s_6 \in \mathcal{S}_{\text{blue}}$, respectively $s_{12} \in \mathcal{S}_{\text{red}}$

Task scheduling is performed such that the robots always maintain a safety distance while picking and placing the objects, imposed by the constraints (5.6) and (5.7).

The computed optimal time t_f^* has twelve maximum and twelve minimum peaks, representing each the beginning and the end of task execution (picking or placing an object), respectively. This corresponds to picking six objects each and placing them in the respective slots. Accordingly, the minimum time is maximal at the beginning denoting the start of the trajectory planning, and decreases continuously as the robots approach their targets with each sampling step (MPC iteration). The slope of the decreasing cost function varies with time, and at some points, a local increase compared to the previous value can also be observed. This occurs especially during a conflict resolution and collision avoidance situation, which causes the robots to take more time to achieve their goals. A local change in the course of the computed minimum time is also reflected in the profile of the minimum distance between the robots, shown in Figure 5.12, and the robot trajectories, especially the velocity and acceleration, see Figure 5.13.

In order to analyze the performance of the collision avoidance constraints, the experiment is also conducted with deactivated constraints, i. e., $\delta_c = 0$. The dark gray area in Figure 5.12 marks the collision zone, and the light gray area denotes the influence distance where the collision avoidance constraints are active when enabled, i. e., $\delta_c = 1$. For $\delta_c = 0$, the resulting trajectories are not feasible since the distance between the spheres undercuts the safety distance, defined by the sum of the radii of the proximity spheres, which would result in collisions between the robots. Therefore, when experimenting with $\delta_c = 0$, the robots were arranged relative to each other so that no collisions occurred. Activating the collision avoidance constraints prohibits the minimum distance from falling below the critical value, ensuring safe robot operation during the entire experiment. It is evident that after entering the influence zone, the activation of the constraints imposes an immediate reaction on the robots' motions, enforcing a safe minimum distance between them.

It can be observed that whenever the slope of the decreasing cost function becomes smaller or locally positive, the distance between the robots usually increases as well. As a reaction to this, there are sometimes sudden changes in the velocity and acceleration in order to avoid collisions, but the robot trajectories remain smooth, as can be seen in Figure 5.13. The acceleration profile of at least one robot joint exhibits a time characteristic that is typical for time-optimal trajectory planning. In the trajectory planning problem, the terminal constraints impose that all robot joints reach their target angles simultaneously. Given that, for the considered robots, all robot joints have the same speed characteristics, the robot joints farthest from their targets or those who are more involved in avoiding collisions determine the time needed by the robots to reach the final configurations, i. e., the optimal value of the final time t_f^*. In Figure 5.13, it is shown that these robot joints exhibit bang-bang acceleration profiles.

For the trajectory planning algorithm, it is required to generate feasible robot trajectories online, with a computation time not exceeding 100 ms, in order to be able to replan and update trajectories during runtime. Since the planning algorithm contains two nonlinear optimization problems, namely minimum distance computation (5.23) and centralized MPC-based trajectory generation (5.48), the computation times for both problems obtained during the experiment are recorded separately and displayed in Figure 5.14. The computation time depends on the chosen parameters, especially the prediction N_T and the collision-free N_C horizon since they influence the dimension of the

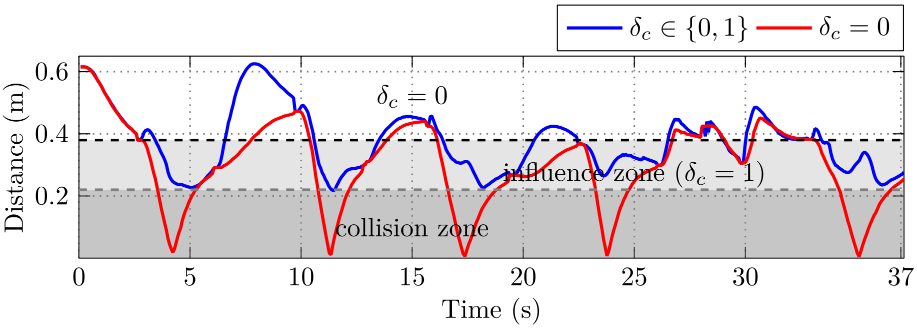
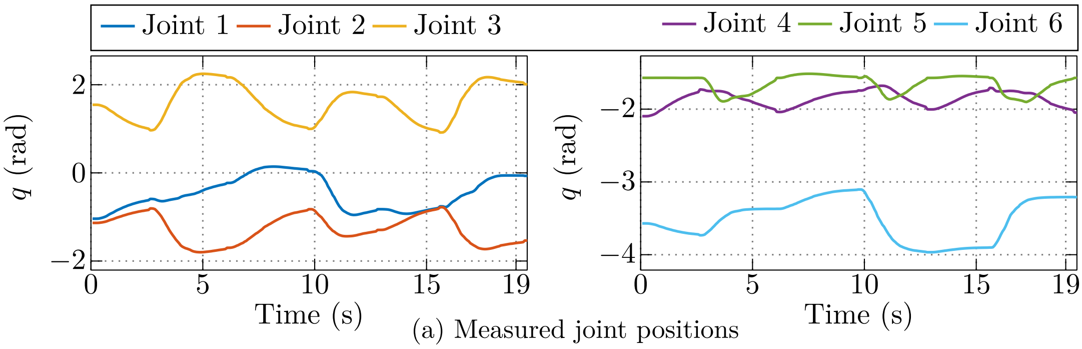
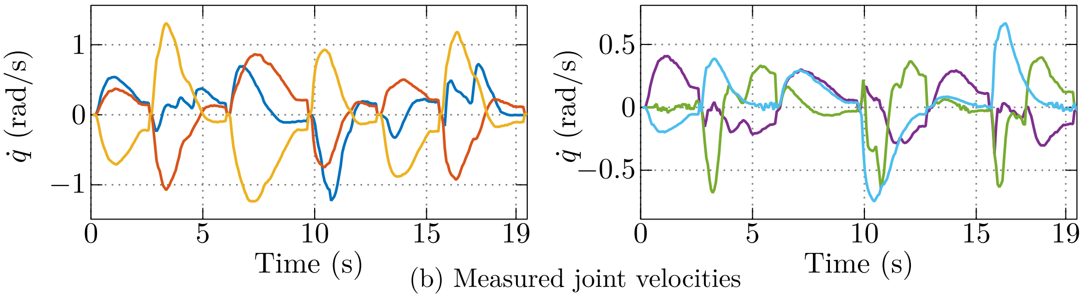


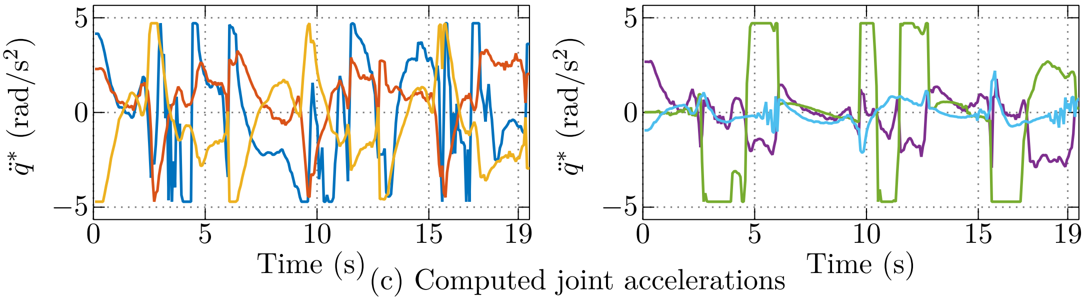
Figure 5.12: Minimum distance d between the robots computed during the experiment. For $\delta_c = 0$ the collision avoidance constraints are deactivated. In this case, the distance between the spheres undercuts the safety distance, defined by the sum of the radii of the proximity spheres, which would lead to collisions between the robots. In order to perform the experiment with $\delta_c = 0$, the robots were arranged outside the operational area of each-other so that no collisions occur. $\delta_c \in \{0,1\}$ represents the feasible case, where the collision avoidance constraints are active within the influence zone.



(a) Measured joint positions



(b) Measured joint velocities



(c) Computed joint accelerations

Figure 5.13: Measured joint positions, velocities, and computed optimal accelerations for Robot 1, displayed for half of experiment duration.

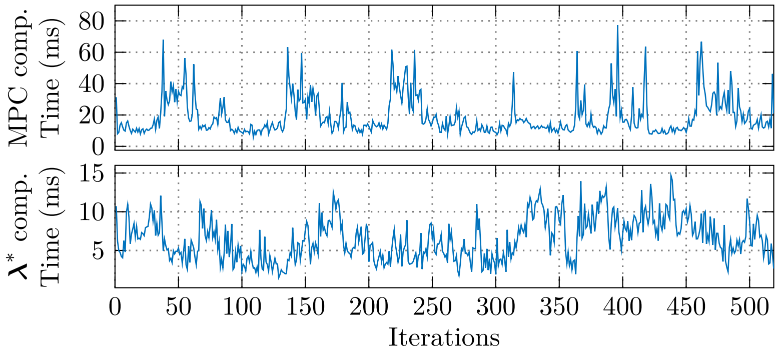


Figure 5.14: Computation time for the centralized MPC-based trajectory planning problem and the minimum distance calculation for $N_T = 10$ and $N_C = 5$.

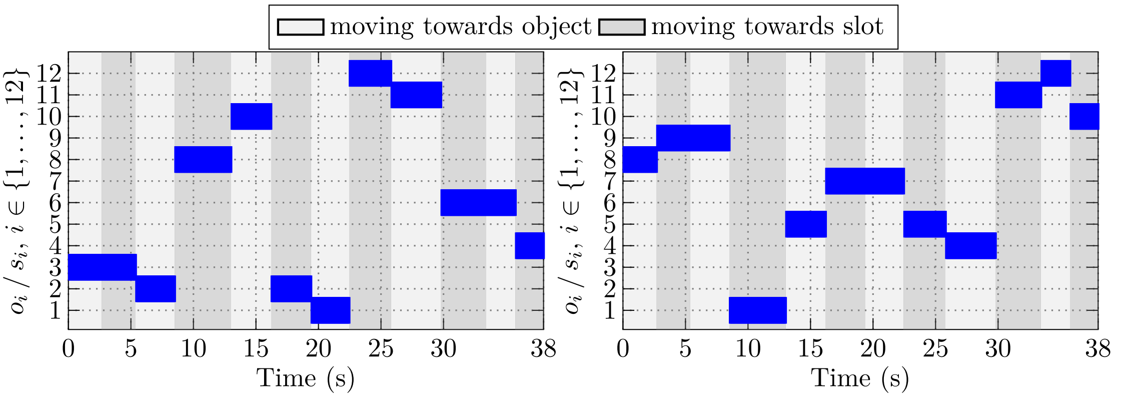
Table 5.2: Experimental results for $N_T = 10$ and $N_C = 5$.

	Two robots min. time scheduling	Two robots min. distance scheduling	Single robot min. time scheduling
$t^*_{f,\max}$ (s)	1.13	1.17	1.10
Max solver time (ms) (MPC / min. distance)	77 / 15	68 / 17	39 / -
Experiment duration (s)	37.14	38.12	66
Average solver iterations (MPC / min. distance)	11 / 6	10 / 6	8 / -

optimization problems. The peaks in the computation time of the MPC algorithm are strongly related to the nonlinear collision-avoidance constraints. In general, for the chosen parameters, the computation time remains below 100 ms, cf., Table 5.2. To analyze the influence of the horizon lengths and other optimization parameters on the computation time and the overall algorithm performance, in Section 7.2.3 experiments are performed for a more challenging test case using different values for the prediction horizon N_T, the collision-free N_C horizon, and the velocity parameter ϵ.

The same experiment setup with the object and slot distribution from (5.56) is used to conduct experiments applying minimum-distance scheduling algorithm, resulting in the task execution sequences presented in Figure 5.15. Since the orientation of the task points is chosen to be the same, there is not much difference in the overall task execution time when comparing minimum-distance scheduling results with the one obtained by applying minimum-time scheduling, with the latter one being around 1 s faster. Involving objects and slots with different picking, respectively, placing orientations demonstrates, in general, the efficiency of the minimum-time approach in terms of overall task execution time, see Section 4.5. However, the performance difference would not be that significant compared to the single robot solution since the task execution time mainly depends on whether the computed sequences lead to robot motions with high collision potential. The scheduling algorithm considers the collision risk only at the discrete task points and not along the entire robot paths. Hence, it may also occur that the computed minimum-distance task sequences are easier and, thus, faster to execute than the ones resulting from the minimum-time scheduling.

When deploying a single robot manipulator to perform the pick-and-place tasks applying the hierarchic controller presented in Section 4.2 with the minimum-time scheduling



a) $\{o_3, s_3, o_2, s_8, o_{10}, s_2, o_1, s_{12}, o_{11}, s_6, o_6, s_4\}$. b) $\{o_8, s_9, o_9, s_1, o_5, s_7, o_7, s_5, o_4, s_1 1, o_{12}, s_{10}\}$.

Figure 5.15: Computed minimum distance task sequences for Robot 1 a) and Robot 2 b).

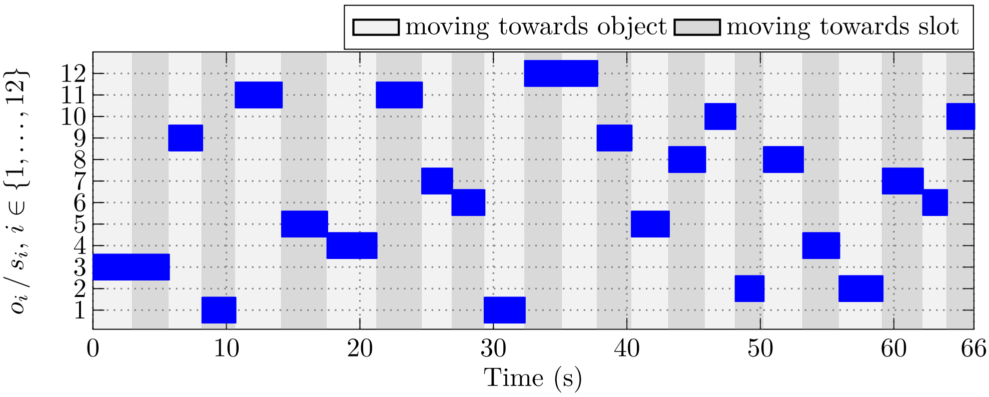


Figure 5.16: Computed minimum time task sequence when using a single robot manipulator to perform the pick-and-place tasks.

algorithm (4.12), which was performing best compared to other algorithms, leads the task execution sequence shown in Figure 5.16. Compared to the dual arm solution, a single robot requires about 66 s to perform the same experiment and place the objects in the slots according to their classification. An advantage of the single-arm solution is that it is easier to implement since it is not subject to collision avoidance between robots. Also, the solution space of the scheduling algorithms in this case is larger. Performing pick-and-place tasks with two robot manipulators requires excluding possible task points that lie close to each other from being executed simultaneously. This reduces the feasible region and, for some test cases, especially involving task points of different classes, might also lead to nonfeasible problems. However, efficient deployment of cooperating robots in shared working environments promises improvements in terms of overall cycle time and operation throughput. For the considered test case, a single robot takes nearly 29 s longer to finish the tasks than the dual manipulator system. Table 5.2 summarizes some key parameters to compare the three methods used during the experiments.

As optimization-based planners, the performance of the proposed approaches relies besides the problem formulation itself on the chosen parameters of the algorithm and the numerical solver. Task scheduling is performed offline, with computation times that mainly depend on the number of the considered task points and the involved object and slot classes. Experimenting with twelve objects and twelve slots of different classes, the encountered computation times range from a few hundred milliseconds to a few seconds. Reducing the number of classes increases the amount of feasible task combinations, often resulting in increased computation times needed to find the optimal task sequence.

For the trajectory planning algorithm, a longer prediction horizon is beneficial in terms of increased time resolution. However, this requires a longer collision-checking horizon, increasing the dimension of the optimization problem and, thus, the computation time. A longer computation time is reflected in increased control horizon and overall robot task execution time as it reduces the convergence rate of the cost function. Overall, the solver converges within a few iterations toward a feasible solution, but the iteration evaluation is computationally costly, cf., Table 5.2. The number of iterations varies between 6 to 40 for the MPC problem and 2 to 12 for the minimum distance computation. The experimental results obtained with different horizon lengths show that, although the average number of iterations does not considerably increase with the horizon, the computation time increases, see Section 7.2.3 and [82]. This comes from the collision avoidance formulation as nonlinear state-dependent constraints but also from the terminal constraints. The exact satisfaction of the equality constraints at the end of the prediction horizon may require high computational effort.

The success rate of the trajectory planner depends mainly on the difficulty of the tasks regarding inter-robot collisions, the horizon lengths, and the velocity parameter ϵ, which directly influences the collision avoidance constraints. Limiting the number of solver iterations to 100, experiments are performed considering the execution of 100 randomly chosen pick-and-place tasks subject to inter-robot collision avoidance. With ϵ varying between $0.1t_f^*$ to $0.35t_f^*$, for $N_T = 15$ and $N_C = 8$ the success planning rate varies from 86% at worst to 94% at best [82]. In all cases where the solver fails to find an optimal solution, the maximum number of iterations is reached, so the trajectory execution is aborted. For smaller ϵ, this is usually the case when the robots get very close to each other. Consequently, the robot's speed is very low, and the solver has difficulties resolving the conflict situation within a time span applicable for online trajectory planning. Contrary, a larger ϵ might bring the robots outside the influence zone defined by (5.31), deactivating the collision-avoidance constraints. Reentering the influence zone generates repulsive forces pushing the robots again outside the influence zone. Here, the trajectory planning results in high-velocity motions with robots leaving and entering the influence zone without being able to pass each other. The solver always converges to a feasible solution within a few iterations for pick-and-place tasks with low inter-robot collision risks since the trajectory planning problem in this case is simple.

CHAPTER 6

Predictive Path-Following Control for Mobile Robots

This section presents a path-following feedback controller based on model predictive control (MPC), driving a mobile robot to converge to a given geometric path and track it at the maximum possible speed. The MPC controller is formulated in the Frenet-Serret frame, and aims to minimize the error to a given parametrized reference path while maximizing the robot speed, i. e., the covered robot distance, by introducing a suitable time law for the dynamics of the path parameter. Since hyperparameters of the MPC formulation greatly affect the tradeoff between path-tracking error and maximum possible speed, a Bayesian optimization-based algorithm presented in [83] is used to automatically select suitable parameters that maximize the speed while keeping the tracking error low. The algorithm is implemented on an omnidirectional mobile robot and validated using different predefined geometric paths.

6.1 Problem formulation

Path-following robot control aims to design a feedback controller that makes the robot approach and follow a predefined geometric path. In order to derive the dynamics of the path-following error, a Frenet-Serret frame is considered attached to a reference point moving along a spatial path, as shown in Figure 6.1, cf. [63, 65]. The reference path $\mathbf{p}_{\mathrm{r}}(\varpi) \in \mathbb{R}^2$ is given as a continuous parameterized curve

$$\mathbf{p}_{\mathrm{r}}(\varpi) = \begin{bmatrix} x_{\mathrm{r}}(\varpi) \\ y_{\mathrm{r}}(\varpi) \end{bmatrix} : \varPi \rightarrow \mathbb{R}^2 \,, \tag{6.1}$$

with the path parameter ϖ, which is element of a nonempty set $\varPi \subseteq \mathbb{R}$, see, e. g., [146]. The orientation of the reference velocity vector relative to the inertial frame of reference $(o_0 x_0 y_0 z_0)$ is given by

$$\psi_{\mathrm{r}}(\varpi) = \arctan\left(\frac{y'_{\mathrm{r}}(\varpi)}{x'_{\mathrm{r}}(\varpi)}\right) , \tag{6.2}$$

with $x'_{\mathrm{r}}(\varpi) = \dfrac{\partial x_{\mathrm{r}}(\varpi)}{\partial \varpi}$ and $y'_{\mathrm{r}}(\varpi) = \dfrac{\partial y_{\mathrm{r}}(\varpi)}{\partial \varpi}$. By introducing a similar Frenet-Serret frame attached to the center point of the mobile robot, the alignment error between the robot velocity vector and the reference velocity vector is given by

$$e_{\psi}(\varpi) = \psi_{\mathrm{p}} - \psi_{\mathrm{r}}(\varpi) \,. \tag{6.3}$$

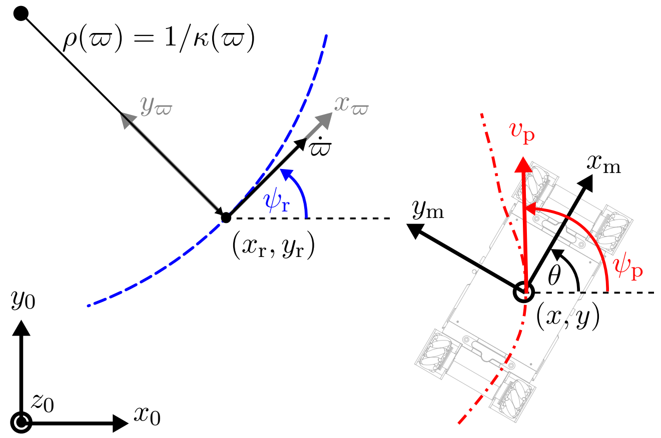


Figure 6.1: Path-following problem: (x_ϖ, y_ϖ) represents the two dimensional Frenet-Serret frame attached to a reference point, and $(x_\mathrm{m}, y_\mathrm{m})$ the reference frame attached to the mobile robot.

Using the reference path (6.1) and the actual robot position $\mathbf{p}_\mathrm{p} = [x, y]^\mathrm{T}$ relative to the inertial frame, the path error in Frenet coordinates is given by

$$ {}_\varpi\mathbf{e}_\mathrm{p}(\varpi) = \begin{bmatrix} e_x(\varpi) \\ e_y(\varpi) \\ e_\psi(\varpi) \end{bmatrix} = \mathbf{R}_z^\mathrm{T}(\psi_\mathrm{r}(\varpi)) \left(\begin{bmatrix} \mathbf{p}_\mathrm{p} \\ \psi_\mathrm{p} \end{bmatrix} - \begin{bmatrix} \mathbf{p}_\mathrm{r}(\varpi) \\ \psi_\mathrm{r}(\varpi) \end{bmatrix} \right) , \tag{6.4} $$

where $\mathbf{R}_z(\psi_\mathrm{r}(\varpi)) \in SO(3)$ represents a rotation matrix around the positive z_0-axis, i. e.,

$$ \mathbf{R}_z(\psi_\mathrm{r}) = \begin{bmatrix} \cos(\psi_\mathrm{r}) & -\sin(\psi_\mathrm{r}) & 0 \\ \sin(\psi_\mathrm{r}) & \cos(\psi_\mathrm{r}) & 0 \\ 0 & 0 & 1 \end{bmatrix} . \tag{6.5} $$

The time derivative of the path-following error equations yields the error dynamics

$$ {}_\varpi\dot{\mathbf{e}}_\mathrm{p} = \dot{\mathbf{R}}_z^\mathrm{T}(\psi_\mathrm{r}) \left(\begin{bmatrix} \mathbf{p}_\mathrm{p} \\ \psi_\mathrm{p} \end{bmatrix} - \begin{bmatrix} \mathbf{p}_\mathrm{r} \\ \psi_\mathrm{r} \end{bmatrix} \right) + \mathbf{R}_z^\mathrm{T}(\psi_\mathrm{r}) \left(\begin{bmatrix} \dot{\mathbf{p}}_\mathrm{p} \\ \dot{\psi}_\mathrm{p} \end{bmatrix} - \begin{bmatrix} \dot{\mathbf{p}}_\mathrm{r} \\ \dot{\psi}_\mathrm{r} \end{bmatrix} \right) \tag{6.6} $$

with the velocity vectors

$$ \begin{bmatrix} \dot{\mathbf{p}}_\mathrm{p} \\ \dot{\psi}_\mathrm{p} \end{bmatrix} = \mathbf{R}_z(\psi_\mathrm{p}) \begin{bmatrix} v_\mathrm{p} \\ 0 \\ \omega_\mathrm{p} \end{bmatrix} , \quad \begin{bmatrix} \dot{\mathbf{p}}_\mathrm{r} \\ \dot{\psi}_\mathrm{r} \end{bmatrix} = \mathbf{R}_z(\psi_\mathrm{r}) \begin{bmatrix} \dot{\varpi} \\ 0 \\ \omega_\mathrm{r} \end{bmatrix} . \tag{6.7} $$

Here, for notational convenience, the dependence on the path parameter ϖ is omitted. Using equations (6.3), (6.4), and (6.7) the error dynamics can be represented as

$$ {}_\varpi\dot{\mathbf{e}}_\mathrm{p} = \begin{bmatrix} 0 & \omega_\mathrm{r} & 0 \\ -\omega_\mathrm{r} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} e_x \\ e_y \\ e_\psi \end{bmatrix} + \mathbf{R}_z(e_\psi) \begin{bmatrix} v_\mathrm{p} \\ 0 \\ \omega_\mathrm{p} \end{bmatrix} - \begin{bmatrix} \dot{\varpi} \\ 0 \\ \omega_\mathrm{r} \end{bmatrix} , \tag{6.8} $$

with the angular velocities ω_p and $\omega_\mathrm{r} = \kappa(\varpi)\dot{\varpi}$, where $\kappa(\varpi) = \|\mathbf{p}_\mathrm{r}''(\varpi)\|_2$ denotes the curvature of the reference path.

Since the considered mobile robot has omnidirectional kinematics, its orientation, given by angle θ, does not depend on the orientation ψ_p of the velocity vector, cf., Figure 6.1.

Therefore, the error equations (6.4) are further extended by the error between the actual θ and a desired θ_d robot orientation, i. e., $e_\theta = \theta - \theta_d$, to consider the robot orientation as an additional degree of freedom. Finally, the resulting path error dynamics reads

$$\begin{aligned} \dot{e}_x(\varpi) &= \kappa(\varpi)\dot{\varpi}e_y(\varpi) + v_\mathrm{p}\cos\left(e_\psi(\varpi)\right) - \dot{\varpi} \\ \dot{e}_y(\varpi) &= -\kappa(\varpi)\dot{\varpi}e_x(\varpi) + v_\mathrm{p}\sin\left(e_\psi(\varpi)\right) \\ \dot{e}_\psi(\varpi) &= \omega_\mathrm{p} - \kappa(\varpi)\dot{\varpi} \\ \dot{e}_\theta(\varpi) &= \omega - \theta_d'(\varpi)\dot{\varpi}\,, \end{aligned} \tag{6.9}$$

with the robot angular velocity $\omega = \dot{\theta}$.

Different approaches exist to choose the time evolution of the path parameter ϖ. In the presented path-following problem, the idea is to design a feedback controller based on model predictive control so that the robot can track a predefined geometric path at the maximum possible speed. Projecting the robot speed on the reference path leads the path parameter dynamics

$$\dot{\varpi} = \frac{\rho(\varpi)v_\mathrm{p}\cos\left(e_\psi(\varpi)\right)}{\|[0, -\rho(\varpi)]^\mathrm{T} + [e_x(\varpi), e_y(\varpi)]^\mathrm{T}\|_2}\,, \tag{6.10}$$

with the path error (6.4) in Frenet coordinates and $\rho(\varpi) = 1/\kappa(\varpi)$, cf. [65]. An initial value for the path parameter can be found by computing the minimum distance between the actual robot position $\mathbf{p}_\mathrm{p}$ and the parameterized reference curve $\mathbf{p}_\mathrm{r}(\varpi)$, which requires solving the following optimization problem

$$\varpi^* = \arg\min_{\varpi\in\Pi}\|\mathbf{p}_\mathrm{p} - \mathbf{p}_\mathrm{r}(\varpi)\|_2\,. \tag{6.11}$$

For $\Pi \subset \mathbb{R}$ is a compact set, the solution of the optimization problem (6.11) is equivalent to solving the equality

$$\left(\mathbf{p}_\mathrm{p} - \mathbf{p}_\mathrm{r}(\varpi^*)\right)^\mathrm{T}\mathbf{p}_\mathrm{r}'(\varpi^*) = 0 \tag{6.12}$$

and inequality equation

$$\|\mathbf{p}_\mathrm{r}'(\varpi^*)\|_2^2 - \left(\mathbf{p}_\mathrm{p} - \mathbf{p}_\mathrm{r}(\varpi^*)\right)^\mathrm{T}\mathbf{p}_\mathrm{r}''(\varpi^*) > 0\,, \tag{6.13}$$

resulting from the first-order necessary and second-order sufficient condition for optimality [147]. Since $\mathbf{p}_\mathrm{r}'(\varpi^*)$ is tangent to the path at the reference point $\mathbf{p}_\mathrm{r}(\varpi^*)$, (6.12) implies that the error vector $\mathbf{p}_\mathrm{p} - \mathbf{p}_\mathrm{r}(\varpi^*)$ is orthogonal to the path at that point, i. e., $e_x(\varpi^*) = 0$. Using (6.10), the time evolution of the optimal path parameter reads

$$\dot{\varpi}^* = \frac{v_\mathrm{p}\cos\left(e_\psi(\varpi^*)\right)}{1 - e_y(\varpi^*)\kappa(\varpi^*)}\,. \tag{6.14}$$

6.2 Predictive-based path-following control

The goal of the predictive path-following control is to minimize the error to a given geometric path along a chosen prediction horizon N_p and simultaneously attempt to maximize the covered path distance, i. e., following the path at the highest possible speed. Introducing the error state vector $\mathbf{x}_e = [e_x, e_y, e_\psi, e_\theta]^\mathrm{T}$ and the angular velocity input vector

$\mathbf{u}_\omega = [\omega, \omega_\mathrm{p}]^\mathrm{T}$, the error dynamics (6.9) used to predict the time evolution of the path-following error can be written as $\dot{\mathbf{x}}_e(t) = \mathbf{f}_e(\mathbf{x}_e, v_\mathrm{p}, \mathbf{u}_\omega, \varpi, t)$. Let $\mathbf{x}_\mathrm{m} = [\mathbf{p}_\mathrm{p}^\mathrm{T}, \psi_\mathrm{p}, \theta, \varpi]^\mathrm{T} \in \mathbb{R}^5$ denote the state of the robot, $\mathbf{x} = [\mathbf{x}_\mathrm{m}^\mathrm{T}, \mathbf{x}_e^\mathrm{T}]^\mathrm{T} \in \mathbb{R}^9$ the full state (along with the path-following error) of the robot, and $\mathbf{u} = [v_\mathrm{p}, \mathbf{u}_\omega^\mathrm{T}] \in \mathbb{R}^3$ denote the input vector. Furthermore, let $\mathbf{x}_{N_\mathrm{p}} = [\mathbf{x}^\mathrm{T}(1), \dots, \mathbf{x}^\mathrm{T}(N_\mathrm{p})]^\mathrm{T} \in \mathbb{R}^{(N_\mathrm{p}-1)\times 9}$, $\mathbf{u}_{N_\mathrm{p}} = [\mathbf{u}^\mathrm{T}(0), \dots, \mathbf{u}^\mathrm{T}(N_\mathrm{p}-1)]^\mathrm{T} \in \mathbb{R}^{(N_\mathrm{p}-1)\times 3}$. Finally, the objective function $J(\varpi_0, \mathbf{u}_{N_\mathrm{p}}; \mathbf{x}_0(t), \varkappa)$ to minimize the error of the robot path with respect to the reference path is given as

$$
\begin{aligned}
J(\varpi_0, \mathbf{u}_{N_\mathrm{p}}; \mathbf{x}_0(t), \varkappa) = {} & \sum_{k=1}^{N_\mathrm{p}} \|\mathbf{x}_e(k)\|^2_{\mathbf{Q}_e} + \sum_{k=0}^{N_\mathrm{p}-2} \|\Delta \mathbf{u}_\omega(k)\|^2_{\mathbf{Q}_{du}} \\
& + \sum_{k=0}^{N_\mathrm{p}-1} \Big(\|\mathbf{u}_\omega(k)\|^2_{\mathbf{Q}_u} + \|\Delta \mathbf{x}(k)\|^2_{\mathbf{Q}_{dx}} - q_v v_\mathrm{p}(k) \Big) ,
\end{aligned}
\tag{6.15}
$$

where $\Delta \boldsymbol{v}(k) = \boldsymbol{v}(k+1) - \boldsymbol{v}(k)$, for $\boldsymbol{v} \in \{\mathbf{x}, \mathbf{u}_\omega\}$. Here, J is a function of the decision variables ϖ_0 and $\mathbf{u}_{N_\mathrm{p}}$, state feedback $\mathbf{x}_0(t) = [\mathbf{x}_\mathrm{m}(0)^\mathrm{T}, \mathbf{x}_e(0)^\mathrm{T}]^\mathrm{T}$ obtained at time t and some fixed parameters $\varkappa$. Based on this, the path-following controller reads

$$
\mathbf{u}^*_{N_\mathrm{p}}, \varpi_0^* = \underset{\mathbf{u}_{N_\mathrm{p}}, \varpi_0}{\operatorname{argmin}} J(\varpi_0, \mathbf{u}_{N_\mathrm{p}}; \mathbf{x}_0(t), \varkappa) \tag{6.16}
$$

$$
\begin{aligned}
\text{s.t.} \quad \mathbf{x}_e(k+1) &= \mathbf{x}_e(k) + T_s \mathbf{f}_e(\mathbf{x}_e, v_\mathrm{p}, \mathbf{u}_\omega, \varpi, k) && \text{(6.16a)} \\
\mathbf{x}_e(0) &= (\mathbf{x}_0(t))_e \\
\varpi(k+1) &= \varpi(k) + T_s \frac{v_\mathrm{p}(k) \cos\left(e_\psi(\varpi(k))\right)}{1 - e_y(\varpi(k))\kappa(\varpi(k))} && \text{(6.16b)} \\
\varpi(0) &= \varpi_0, \quad \varpi(k) \in \mathit{\Pi} \\
\underline{\mathbf{u}}_\omega &\le \mathbf{u}_\omega(k) \le \bar{\mathbf{u}}_\omega && \text{(6.16c)} \\
0 &\le v_\mathrm{p}(k) \le \bar{v}_\mathrm{p} && \text{(6.16d)} \\
\underline{\mathbf{x}}_e(\underline{\mathbf{x}}_\mathrm{m}) &\le \mathbf{x}_e(k) \le \bar{\mathbf{x}}_e(\bar{\mathbf{x}}_\mathrm{m}) , && \text{(6.16e)}
\end{aligned}
$$

with the iteration index $k \in \{0, \dots, N_\mathrm{p}-1\}$ along the prediction horizon N_p, and the initial conditions

$$
0 = (\mathbf{p}_{\mathrm{p}0} - \mathbf{p}_\mathrm{r}(\varpi_0))^\mathrm{T} \mathbf{p}'_\mathrm{r}(\varpi_0) \tag{6.16f}
$$

$$
0 < \|\mathbf{p}'_\mathrm{r}(\varpi_0)\|_2^2 - (\mathbf{p}_{\mathrm{p}0} - \mathbf{p}_\mathrm{r}(\varpi_0))^\mathrm{T} \mathbf{p}''_\mathrm{r}(\varpi_0) \tag{6.16g}
$$

$$
(\mathbf{x}_0(t))_e = \begin{bmatrix} \mathbf{R}_z^\mathrm{T}(\psi_\mathrm{r}(\varpi_0)) \left(\begin{bmatrix} \mathbf{p}_{\mathrm{p}0} \\ \psi_{\mathrm{p}0} \end{bmatrix} - \begin{bmatrix} \mathbf{p}_\mathrm{r}(\varpi_0) \\ \psi_\mathrm{r}(\varpi_0) \end{bmatrix} \right) \\ \theta_0 - \theta_d(\varpi_0) \end{bmatrix} . \tag{6.16h}
$$

Here, the diagonal positive definite matrices $\mathbf{Q}_e \in \mathbb{R}^{4\times 4}$, $\mathbf{Q}_u \in \mathbb{R}^{2\times 2}$, $\mathbf{Q}_{dx} \in \mathbb{R}^{9\times 9}$, and $\mathbf{Q}_{du} \in \mathbb{R}^{2\times 2}$ represent the weighting coefficients for the error state, angular input vector, difference of state, and input vectors, respectively. The coefficient $q_v > 0$ denotes the weight for the robot velocity, which should be maximized. The parameter vector $\varkappa \in \mathbb{R}^{11}_+$ consists of non-zero elements of weight matrices given by $\varkappa := [\operatorname{diag}(\mathbf{Q}_e), \operatorname{diag}(\mathbf{Q}_u), q_v, \operatorname{diag}(\mathbf{Q}_{du}), \operatorname{diag}(\mathbf{Q}_{dx}[:2, :2])]^\mathrm{T}$. The optimization is performed for the input variables $\mathbf{u}_\omega(k)$ and $v_\mathrm{p}(k)$ with the upper and lower bounds as expressed by the inequality constraints (6.16c) and (6.16d). The equality constraints (6.16a) and (6.16b) represent the discretized dynamic equations (6.9) and (6.14), with the upper and lower bounds expressed by (6.16e) as function of the bounds of the robot state $\mathbf{x}_\mathrm{m}$ and the set Π, and the respective initial conditions satisfying equations (6.16f), (6.16g), and (6.16h). Here, $\mathbf{p}_{\mathrm{p}0} \in \mathbb{R}^2$ and θ_0 denote the actual measured robot position and orientation, and $\psi_{\mathrm{p}0}$ the orientation of the current velocity vector, i. e.,

$$\psi_{\mathrm{p}0} = \theta_0 + \arctan\left(\frac{v_{y0}}{v_{x0}}\right), \tag{6.17}$$

with the measured robot velocities v_{x0} and v_{y0} in the x- respective y-axis, cf., Figure 6.1. The time discretization of the continuous time path error prediction model is performed using the forward Euler integration method with the sampling time T_s.

6.3 Experimental results

The path-following controller is implemented and validated on an omnidirectional mobile robot shown in Figure 6.6. The implementation architecture of the experimental setup follows the idea presented in Section 4.4 for the robot arm and consists of three parts, as schematically illustrated in Figure 6.2. The MPC algorithm is planning at each time step $t_k = kT_s$ optimal robot position, orientation, and velocity trajectories along the prediction horizon represented by the vectors $\mathbf{q}_\mathrm{m}^*(t_k) = [x^*(t_k),\, y^*(t_k),\, \theta^*(t_k)]^\mathrm{T}$ and $\dot{\mathbf{q}}_\mathrm{m}^*(t_k) = [v_x^*(t_k),\, v_y^*(t_k),\, \omega^*(t_k)]^\mathrm{T}$. The optimal values for the robot position and the orientation of the velocity vector are obtained using equation (6.4), i. e.,

$$\begin{bmatrix} x^* \\ y^* \\ \psi_\mathrm{p}^* \end{bmatrix} = \mathbf{R}_z^\mathrm{T}(\psi_\mathrm{r}(\varpi^*))_\varpi \mathbf{e}_\mathrm{p}^*(\varpi^*) + \begin{bmatrix} \mathbf{p}_\mathrm{r}(\varpi^*) \\ \psi_\mathrm{r}(\varpi^*) \end{bmatrix}. \tag{6.18}$$

Using angle ψ_p^* and the optimal robot orientation $\theta^* = e_\theta(\varpi^*) + \theta_d(\varpi^*)$ the robot velocity in the local frame is given by

$$v_x^* = v_\mathrm{p}^* \cos\left(\psi_\mathrm{p}^* - \theta^*\right), \quad v_y^* = v_\mathrm{p}^* \sin\left(\psi_\mathrm{p}^* - \theta^*\right). \tag{6.19}$$

The optimal trajectories are forwarded to the local robot controller (a programmable logic controller - PLC) using a velocity control interface. Optionally, the velocity interface can be used in combination with a proportional integral derivative (PID) controller with

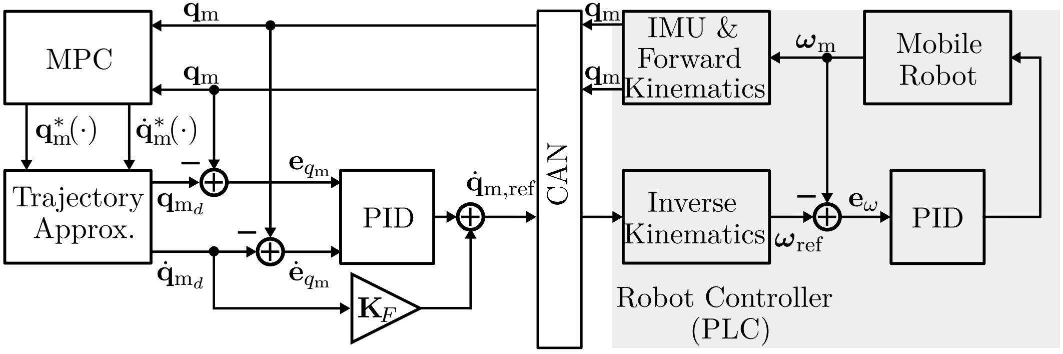

$\dot{\mathbf{q}}_\mathrm{m}$
MPC
$\mathbf{q}_\mathrm{m}$
$\mathbf{q}_\mathrm{m}^*(\cdot)$
$\dot{\mathbf{q}}_\mathrm{m}^*(\cdot)$
$-$
$\mathbf{e}_{q_\mathrm{m}}$
Trajectory Approx.
$\mathbf{q}_{\mathrm{m}d}$
PID
$\dot{\mathbf{q}}_\mathrm{m,ref}$
$-$
$\dot{\mathbf{e}}_{q_\mathrm{m}}$
$\dot{\mathbf{q}}_{\mathrm{m}d}$
$\mathbf{K}_F$
CAN
$\dot{\mathbf{q}}_\mathrm{m}$
$\mathbf{q}_\mathrm{m}$
IMU & Forward Kinematics
$\boldsymbol{\omega}_\mathrm{m}$
Mobile Robot
Inverse Kinematics
$-$
$\mathbf{e}_\omega$
$\boldsymbol{\omega}_\mathrm{ref}$
PID
Robot Controller (PLC)

Figure 6.2: Schematic illustration of the implementation architecture.

a feedforward term in the form of (4.58). A polynomial trajectory approximation can be performed in case acceleration variables are included in the path-following optimization problem, see, cf., Section 4.4. The reference linear and angular velocities $\dot{\mathbf{q}}_{\mathrm{m,ref}}$ are transformed to reference velocities $\boldsymbol{\omega}_{\mathrm{ref}}$ for the mecanum wheels by applying the inverse velocity kinematics (2.103). An internal PID controller is used to make the speed of the mecanum wheels $\boldsymbol{\omega}_{\mathrm{m}} = [\omega_1, \omega_2, \omega_3, \omega_4]^{\mathrm{T}}$ track the reference values. Using an inertial measurement unit (IMU) and forward velocity kinematics (2.105), information about the robot's position and velocity are forwarded back to the MPC controller as shown in Figure 6.2.

The MPC algorithm is implemented in PYTHON using the CasADi framework [129] on a CarPC with Intel Core i7-8700T Processor and a clock rate of 2.40 GHz running on Ubuntu 18.04.6 LTS. To solve the resulting optimization problem the IPOPT solver [130] with MA57 [144] is used. The communication between the PC and the robot controller is realized via CAN-Bus and is implemented in C++ [148].

The performance of the proposed approach is experimentally analyzed and validated on three test cases, representative of a circular, a lemniscate, and a pinched circular path. These are given by the following parametric curves

$$
\begin{aligned}
\mathbf{p}_{\mathrm{r}}(\varpi) &= \Big[\sin(\varpi),\ \cos(\varpi)\Big]^{\mathrm{T}}, \\
\mathbf{p}_{\mathrm{r}}(\varpi) &= \Big[1.5\sin(\varpi),\ 1.5\sin(\varpi)\cos(\varpi)\Big]^{\mathrm{T}}, \\
\mathbf{p}_{\mathrm{r}}(\varpi) &= \Big[1.5\sin(\varpi),\ 0.75\cos(\varpi)(1+\sin(\varpi))\Big]^{\mathrm{T}}, \varpi \in [0, 2\pi].
\end{aligned}
$$

The experimental results for the three paths are presented in Figure 6.3, Figure 6.4, and Figure 6.5. For each path, the reference and realized robot path are shown in plot a), with the path-following error displayed in plots c) and d). The computed normalized optimal velocities are shown in subfigure b). Following the idea presented in [83], for the parameter vector $\varkappa$ an automatic tuning procedure based on Bayesian optimization is used, yielding the offline tuned parameters

$$\varkappa = [3200, 5100, 7.1, 0.57, 26, 0.18, 0.63, 1.6, 0.226, 0.21, 0.343]^{\mathrm{T}}.$$

While performing the experiments, these parameters are online iteratively tuned during every MPC iteration. The maximum value of the velocity vector is chosen $\bar{v}_{\mathrm{p}} = 0.25\,\mathrm{m/s}$, and the robots state is upper and lower bounded by $\bar{\mathbf{x}}_{\mathrm{m}} = [2\,\mathrm{m}, 2\,\mathrm{m}, 2\pi, 2\pi, 2\pi]^{\mathrm{T}}$ and $\underline{\mathbf{x}}_{\mathrm{m}} = [-2\,\mathrm{m}, -2\,\mathrm{m}, -2\pi, -2\pi, 0]^{\mathrm{T}}$. For the path-following MPC, the prediction horizon $N_{\mathrm{p}} = 20$ and sampling time $T_s = 100\,\mathrm{ms}$ are used.

The presented path-following controller with the selected parameters results in a very good convergence toward the reference path and good overall tracking performance while maintaining maximum speed to the end of the path. The pinched circular path presented in Figure 6.5 involves a singular point for the omnidirectional robot, which, in general, poses a challenging path-following problem. Difficult paths usually require a readjustment of the weighting parameters of the underlying path-following optimization problem, which can result in a lengthy tuning process. In this case, automated tuning provides a remedy for fast, and even at runtime, updating of the weighting parameters. In combination with the proposed high-speed path-following strategy, the built-up momentum helps finding a suboptimal solution around the singular point making the robot continue following the path to the end.

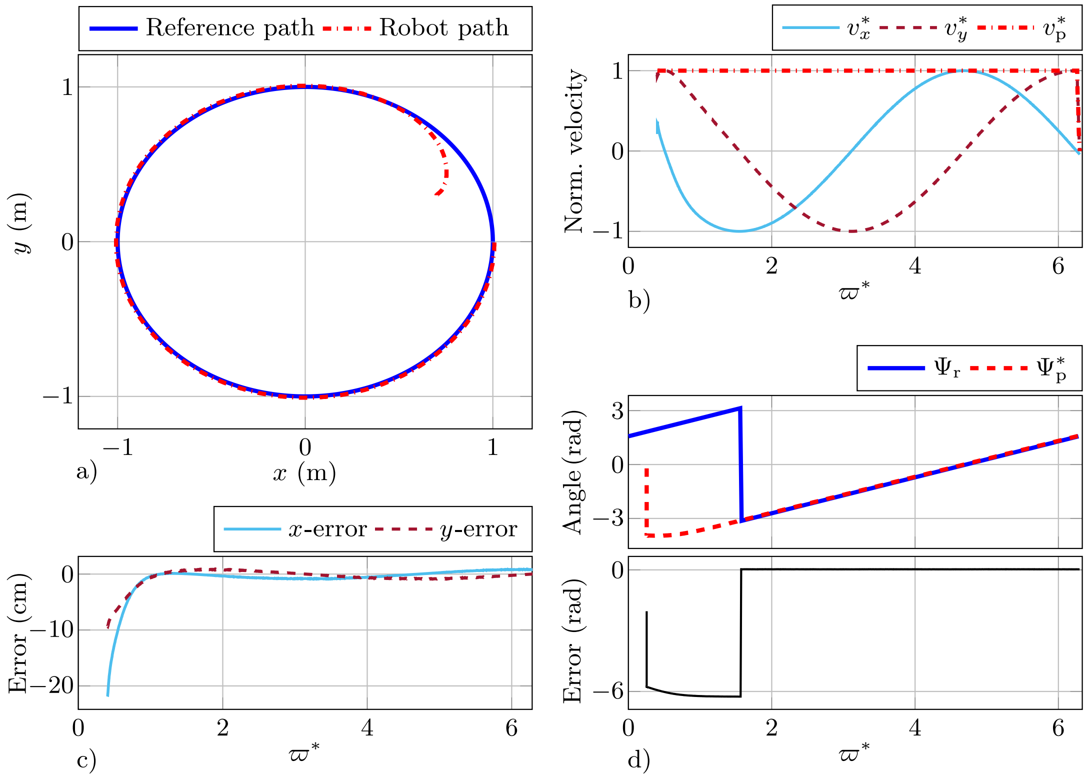


Figure 6.3: Experimental results of path-following control using a circular reference path.

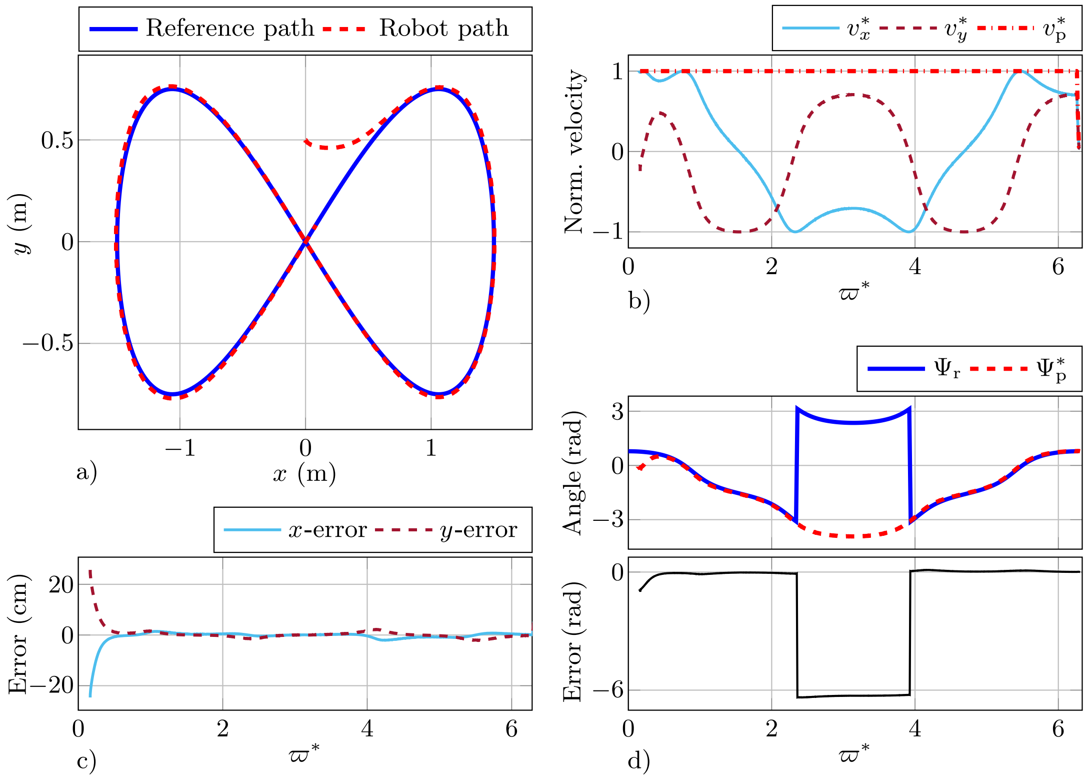


Figure 6.4: Experimental results of path-following control using a lemniscate reference path.

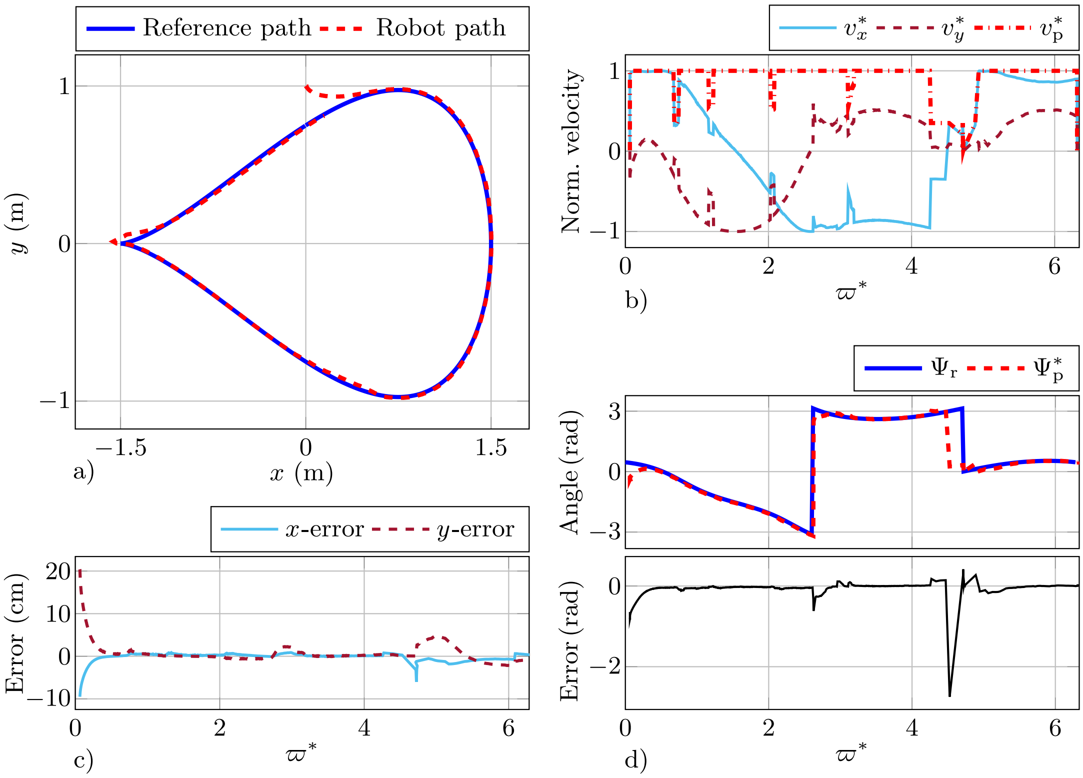


Figure 6.5: Experimental results of path-following control using a pinched circular reference path.

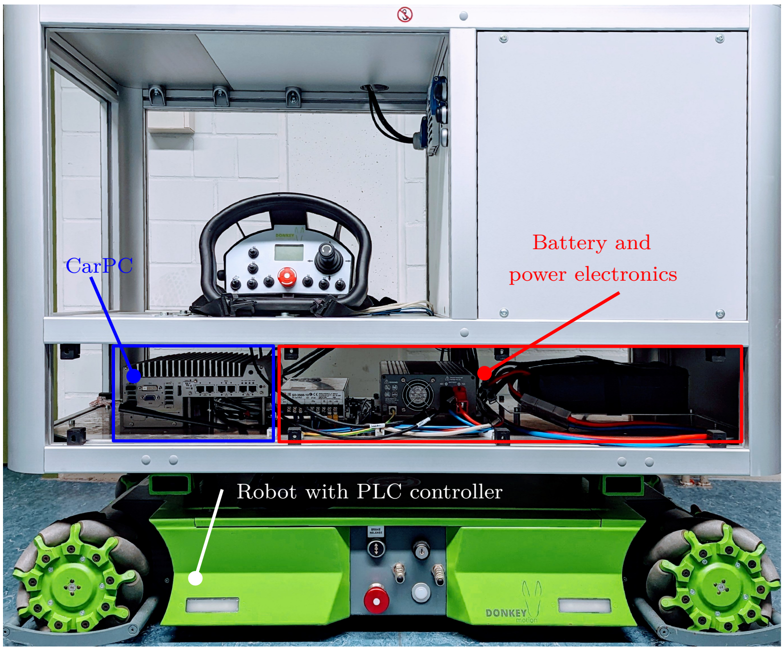


Figure 6.6: Mobile robot platform with omnidirectional wheels.

CHAPTER 7

Demonstrators

With increased automation in the food and beverage industry, multiple robots are often used for pick-and-place operations along sorting, packing, and processing lines, supplementary to high payload industrial robots usually deployed for palletizing [149]. For fast pick-and-place task execution, primarily non-collaborative SCARA (Selective Compliance Assembly Robot Arm) or Delta robots with three to four degrees of freedom (DoF) are used [150]. While these highly automated systems allow for high production throughput, they tend to be product-specific and usually require significant modifications to make changes and adjustments to production lines. With the increased diversity and personalization of goods and products driven by the latest developments in Industry 4.0, production in short batches becomes more attractive. Despite the highly automated food industry, manual methods are currently seen as the only cost-effective solution for production in short batches. To this end, this section deals with the integration of 6-DoF robots into existing processes to increase the level of automation on manual sorting and packaging stations, typical for small and seasonal producers or distributors. The robot integration can be either as static workplaces by deploying single or double robot manipulators along conveyors or as mobile platforms by mounting them on a mobile omnidirectional robot.

The demonstrators presented in this chapter are built within the scope of the research projects KORINS, CooPick, and KIMKO, supported by the German Federation of Industrial Research Associations (AiF) with the grant numbers ZF4335715DB9, ZF4335706DB7, and ZF4335711PO9, respectively.

7.1 Bottle sorting system

Due to the wide range of beverage products on the market, where each product is filled in different bottle designs in terms of size, color, and shape, it is common for the returned empty crates to be filled with inconsistent bottle brands, colors, or types. Existing bottle inspection systems perform sophisticated image processing, identify bottles that belong to the same brand as the inspected crate, and apply an exclusion algorithm to exclude bottles or at least classify those that do not belong to the same brand. Bottles whose type cannot be determined or whose condition is unclear are also identified. In addition to standardized bottle and crate pools, various manufacturers and fillers are increasingly using individual glass bottles and crates for marketing purposes. This makes sorting more difficult and leads to a steady increase in the number of incorrectly sorted bottles.

Automated inspection systems usually feature a manual inspection downstream to visually inspect and manually sort crates that are not clearly marked or are highly unlikely to contain bottles at certain slots, adding a manual workload to the otherwise highly automated bottle inspection and sorting system. Human inspection involves visually identifying each bottle in the crates, picking up and inspecting each bottle, replacing bottles that do not belong to the crate brand, and disposing of unidentifiable and damaged bottles. Workers must also keep track of the different types of bottles and swap bottles between crates to fill them with the correct type. A robotic system complementary to the automated inspection systems is presented to support the manual sorting process. It consists of a conveyor belt, a robot manipulator equipped with a pneumatic bottle gripper, and a camera system for individual bottle inspection, as shown in Figure 7.1.

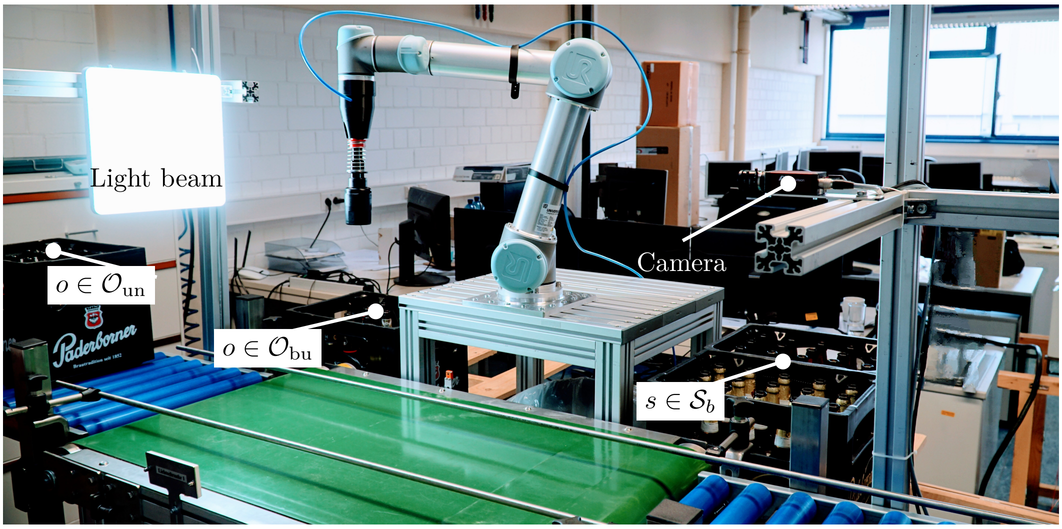
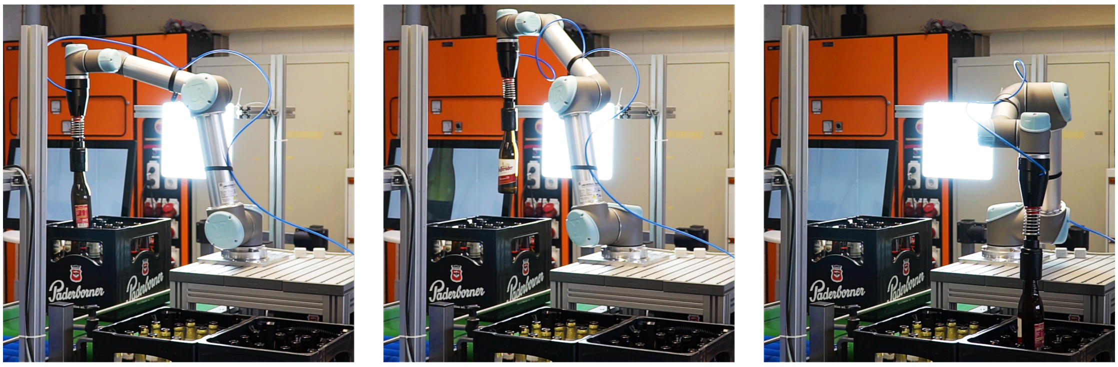

Light beam
$o \in \mathcal{O}_{\text{un}}$
$o \in \mathcal{O}_{\text{bu}}$
Camera
$s \in \mathcal{S}_b$

Figure 7.1: Experimental setup for bottle sorting. A robot manipulator has to pick up unknown bottles $o \in \mathcal{O}_{\text{un}}$ and place them in the respective slots $s \in \mathcal{S}_b$ according to their classification $b \in \mathcal{B}$, which is determined by a camera system. Empty slots in the inspected crate are filled with bottles of the same type from a pool of backup bottles $o \in \mathcal{O}_{\text{bu}}$.

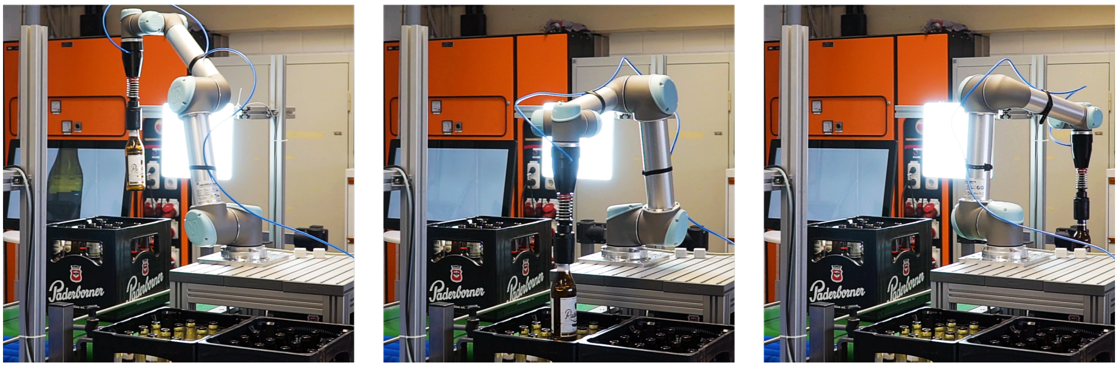
Beverage crates that the automatic system has not completely sorted are transported to the robotic system via the conveyor belt. The system receives information about the position of the bottles in the crates, which the supplementary robot system must check. These bottles are considered as objects of unknown class, i. e., $o \in \mathcal{O}_{\text{un}}$. An unknown object is first undergoing a visual inspection by the camera and needs, therefore, to be held in front of the light beam at a certain distance from the camera to detect bottle-specific features and labels required for identification. The identified bottle is then placed according to its classification $b \in \mathcal{B}$ either in a corresponding slot $s \in \mathcal{S}_b$ of the empty crates or back in the crate under inspection if it turns out that the bottle was in the right crate but not correctly identified by the automatic system. Empty slots in the inspected crate resulting from removing foreign bottles must be filled with bottles of the same class as the crate, which are taken from the pool of backup bottles $o \in \mathcal{O}_{\text{bu}}$.

The problem formulation is similar to the one described in Section 4.1 for the task and trajectory planning of a robot manipulator. Investigations in the beverage industry show that crates that are only partially sorted contain one to three foreign bottles to be examined by the robotic system since most bottles are detected and sorted by the automatic sorting system. Hence, the resulting planning problems are of comparably small dimension, which can be handled by the hybrid controller for inherent task and trajectory planning. During the inspection process, the crate remains stopped before the robot, resulting in pick-and-place tasks with static task points.

In the test case shown in Figure 7.1, the crate under inspection is of type "Paderborner" (Pa) and contains two bottles which are labeled as unknown, i. e., $|\mathcal{O}_{\text{un}}| = 2$. By applying the single-step hybrid controller (4.33), the robot will pick up the unknown object that is fastest to reach from its actual position and place it between the camera and the light beam. The bottle is identified as type "Hasseröder" (Ha) and is therefore placed in a slot $s \in \mathcal{S}_{\text{Ha}}$. Next, the robot can either pick up the next unknown object or an object $o \in \mathcal{O}_{\text{Pa}}, \mathcal{O}_{\text{Pa}} \subset \mathcal{O}_{\text{bu}}$ from the backup pool to fill the empty slot in the crate, resulting from removing the first foreign bottle. By selecting the fastest possible task point, the robot will pick up the second unknown object and place it in a slot $s \in \mathcal{S}_{\text{Ra}}$, since the bottle is identified as type "Radeberger" (Ra). Finally, the two empty slots in the inspected crate will be filled with bottles of the same type $o \in \mathcal{O}_{\text{Pa}}$, completing the sorting process. Some sequences of the pick-and-place tasks performed during the experiment are depicted in Figure 7.2.



a) Picking up first bottle. b) Bottle inspection. c) Placing first bottle.



d) Inspecting second bottle. e) Placing second bottle. f) Picking up first backup bottle.

Figure 7.2: Robot performing pick-and-place tasks on the bottle sorting system. Pictures a)-d) show the sequence from picking the first unknown object, bottle inspection, and placing the bottle in the slot according to its type. Picture d) shows the inspection of the second unknown bottle, which is then placed in the corresponding slot e). In f), the robot picks up the first backup bottle to fill the empty slots in the inspected crate.

7.2 Fruit packing station

The use of robots on existing assembly or packing lines aims to increase the flexibility of the production lines, reduce the production cost and to maximize throughput and machine utilization. In order to partially meet these requirements and to minimize the space required for the robot use or to enable cooperative tasks to be performed, multiple robot arms in the same workspace are increasingly used. As shown in Figure 7.3, the considered application of a fruit packing station consists of a conveyor, fruit feeder, a vision-based detection and inspection system, and two UR5 collaborative robot arms, which, by efficient deployment, can increase the packing throughput [82]. The conveyor belt has a special finger-shaped flexible band to avoid damaging the apples and prevent them from rolling along the conveyor belt. The packaging system is to be completed by a foil wrapping station, resulting in a confined robot setup with highly overlapping robot working areas. Sharing the same workspace by multiple robots can lead to collisions, so safe operation must be guaranteed. In addition to that, more challenges arise for the task assignments for randomly distributed objects transported by a conveyor belt and trajectory planning for grasping moving objects.

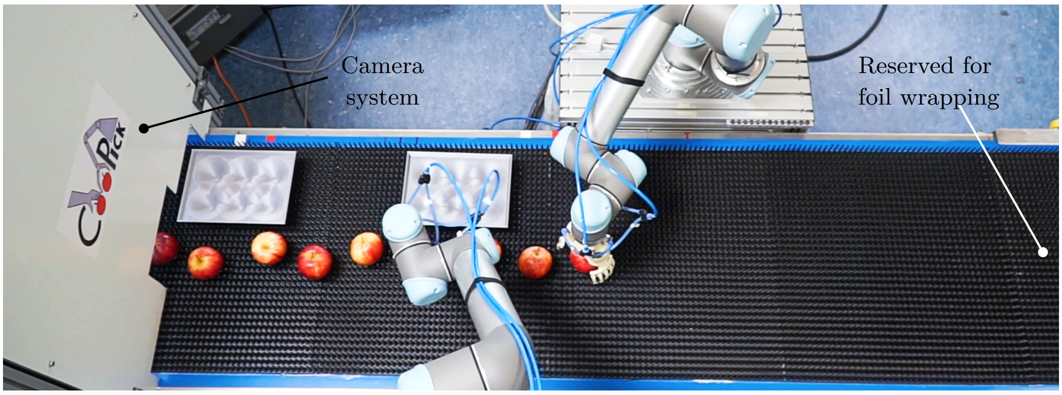


Figure 7.3: Experimental setup consisting of two robotic arms inclusive controllers and grippers, a conveyor belt with PLC, and a camera system for object detection and quality inspection.

The overall goal of the robotic system is to pick up dynamically moving objects, i. e., the fruits, and place them into dynamically moving slots in a tray in a time-optimal manner. In order to minimize the robot cycle time and increase the operating throughput, two main tasks are of fundamental importance and are addressed following the hierarchical control structure presented in Section 5.2. A resource allocation problem is raised by questioning which object is to be picked by which robot and in which slot of a tray it is to be placed. For this purpose, an iterative scheduling algorithm for the cooperative selection of non-stationary objects, while additionally considering safety requirements by adding some safety-related constraints, is used. Since the orientation of the objects does not play a significant role and fast remodeling is required for iterative scheduling, the minimum-distance scheduling approach is used to compute optimal task sequences for the robots. According to the proposed control structure, scheduling is followed by time-optimal MPC-based trajectory planning.

7.2.1 Task scheduling and trajectory planning

The goal of the task scheduler is to allocate a set of objects $o \in \mathcal{O} = \{o_1, \cdots, o_O\}$ to a set of slots $s \in \mathcal{S} = \{s_1, \cdots, s_S\}$ while minimizing the total distance covered by the robot end-effectors $r \in \mathcal{R} = \{r_1, \cdots, r_R\}$ to execute all necessary tasks. It is assumed that the number of slots is smaller than the number of objects $S \leq O$, and they are uniformly distributed on a set of trays (boxes) $b \in \mathcal{B} = \{b_1, \cdots, b_B\}$, where $\mathcal{S}_b \subset \mathcal{S}$, denotes the subset of slots located on tray b, i. e., $\mathcal{S}_1 = \{s_1, \cdots, s_{S/B}\}$ and $\mathcal{S}_2 = \{s_{S/B+1}, \cdots, s_{2\,S/B}\}$ denote the set of slots located to Tray 1 and Tray 2 respectively. Hence, it is necessary that both S and B are even numbers.

The scheduling problem is presented in [79] and is based on the algorithm proposed in Section 5.2.1, with a few application-specific assumptions made for simplicity. To each robot is assigned a tray and the sequence in which the slots are filled is fixed so as to eliminate the possibility of collisions during the placement of the objects into the trays. All slots filled at the same time as slot s belong to the subset $\mathcal{S}_s^c$. Slot $\hat{s}_b$ is the first one filled in tray b. Since the objects and trays are transported by a conveyor belt it is assumed that the considered objects and slots are within the workspace of the robots, i. e., they are reachable. After the first objects of the sequence have been placed into the trays, the corresponding slots are removed from the set $\mathcal{S}$, new objects are potentially added to the set $\mathcal{O}$ and the optimization is performed again in an iterative fashion. Therefore, and to be able to fast rebuild the scheduling model, its cost function and constraints are represented directly in matrix notation decoupled from any modeling frameworks.

The scheduling model includes two types of binary variables $\boldsymbol{\delta}_d^{\mathrm{T}} = \left[\boldsymbol{\delta}_{rb}^{\mathrm{T}}\ \boldsymbol{\delta}_{os}^{\mathrm{T}}\right]$, with

$$\boldsymbol{\delta}_{rb} = [\delta_{r_1 b_1} \cdots \delta_{r_1 b_B}\ \delta_{r_2 b_1} \cdots \delta_{r_R b_B}]^{\mathrm{T}}\,, \tag{7.1}$$

modeling the robot-tray assignment and the variables

$$\boldsymbol{\delta}_{os} = [\delta_{o_1 s_1} \cdots \delta_{o_1 s_S}\ \delta_{o_2 s_1} \cdots \delta_{o_O s_S}]^{\mathrm{T}} \tag{7.2}$$

for the object-to-slot allocation.

The robot-tray assignment must ensure that each robot serves a single tray and each tray can only be served by a single robot. Therefore, by considering the next R upcoming trays it is assumed that the number of trays equals the number of available robots, i. e., $B = R$. The robot-tray assignment can then be expressed by the following $B + R$ equality constraints

$$\sum_{\forall r \in \mathcal{R}} \delta_{rb} = 1,\ \forall b \in \mathcal{B}\,, \qquad \sum_{\forall b \in \mathcal{B}} \delta_{rb} = 1,\ \forall r \in \mathcal{R}\,, \tag{7.3}$$

which can be written as

$$\left[\left[\mathbf{I}^{B\times B} \oplus \mathbf{1}^{1\times R}\right]\ \mathbf{0}^{B\times O\cdot S}\right] \boldsymbol{\delta}_d = \mathbf{1}^{B\times 1} \tag{7.4}$$

$$\left[\left[\mathbf{1}^{1\times B} \oplus \mathbf{I}^{R\times R}\right]\ \mathbf{0}^{R\times O\cdot S}\right] \boldsymbol{\delta}_d = \mathbf{1}^{R\times 1}\,. \tag{7.5}$$

In order to meet the product requirements, each tray has to be filled with a specified amount of objects, i. e., each slot in the trays must contain exactly one object to not leave empty slots in the trays. However, since the number of objects can be greater than the number of slots, not every object has to be allocated to a slot. Nevertheless, each object can at most be allocated to one slot, but does not necessarily have to be allocated to a slot at all. The first condition is modeled by S equality and the second one by O inequality constraints

$$\sum_{\forall o \in \mathcal{O}} \delta_{os} = 1,\ \forall s \in \mathcal{S}\,, \quad \sum_{\forall s \in \mathcal{S}} \delta_{os} \leq 1,\ \forall o \in \mathcal{O}\,, \tag{7.6}$$

leading to

$$\left[\mathbf{0}^{S \times R \cdot B}\ \left[\mathbf{I}^{S\times S} \oplus \mathbf{1}^{1\times O}\right]\right] \boldsymbol{\delta}_d = \mathbf{1}^{S\times 1} \tag{7.7}$$

$$\left[\mathbf{0}^{O \times R \cdot B}\ \left[\mathbf{1}^{1\times S} \oplus \mathbf{I}^{O\times O}\right]\right] \boldsymbol{\delta}_d \leq \mathbf{1}^{O\times 1}\,. \tag{7.8}$$

The proposed synchronous approach to accomplish pick-and-place tasks involving dynamic objects implies that the robots simultaneously pick-and-place the objects. The sequence in which the slots are filled is fixed such that the robots maintain a safe distance from each other while placing the objects. Therefore, safety-related constraints need to be applied only for simultaneously picket objects to impose a minimum distance between the robots. Considering that the density of the objects on the conveyor is not high, the binary inequality constraints (5.6) are slightly modified, keeping the safety minimum distance equal $d_{\min}$, yielding

$$d_{oo'} + (2 - \delta_{os} - \delta_{o's'}) d_{\min} \geq d_{\min}, \quad \forall o, o' \in \mathcal{O}, o \neq o', s \in \mathcal{S}, s' \in \mathcal{S}_s^c. \tag{7.9}$$

$\mathcal{S}_s^c$ denotes the subset of slots being filled at the same time as slots $s \in \mathcal{S}$. Assuming that the first S/B and the next S/B slots from $\mathcal{S}$ are filled simultaneously, the constraint can be written as

$$\underbrace{\begin{bmatrix} \delta_{os_1} \\ \vdots \\ \delta_{os_S} \end{bmatrix}}_{\boldsymbol{\delta}_{oS}} + \underbrace{\left[\begin{bmatrix} \mathbf{0}^{\frac{S}{B}\times\frac{S}{B}} & \mathbf{I}^{\frac{S}{B}\times\frac{S}{B}} \\ & \\ \mathbf{I}^{\frac{S}{B}\times\frac{S}{2}} & \mathbf{0}^{\frac{S}{B}\times\frac{S}{B}} \end{bmatrix} \oplus \mathbf{I}^{(B-1)\times(B-1)}\right]}_{\mathbf{I}_c} \underbrace{\begin{bmatrix} \delta_{o's_1} \\ \vdots \\ \delta_{o's_S} \end{bmatrix}}_{\boldsymbol{\delta}_{o'S}} \leq \left(1 + \frac{d_{oo'}}{d_{\min}}\right) \mathbf{1}^{S\times 1}\,, \tag{7.10}$$

with $d_{oo'}$ denoting the Euclidean distance between two objects. For any $o \in \mathcal{O}$ and $\forall\, o' \in \mathcal{O}$, (7.10) can be written as

$$\left[\boldsymbol{\delta}_{oS} \oplus \mathbf{1}^{(O-1)\times 1}\right] + \left[\mathbf{I}_c \oplus \mathbf{I}_o\right] \boldsymbol{\delta}_b \leq \mathbf{1}^{(O-1)S\times 1} + \frac{1}{d_{\min}} \left[\mathbf{1}^{S\times 1} \oplus \mathbf{I}_o\right] \mathbf{d}_{oO}, \tag{7.11}$$

with $\mathbf{d}_{oO} = [d_{oo_1}\, d_{oo_2} \cdots d_{oo_O}]^{\mathrm{T}}$ and the matrix $\mathbf{I}_o^{(O-1)\times O}$ resulting from the identity matrix $\mathbf{I}^{O\times O}$ by removing the first row and shifting the first column vector to the column o. By this the case $o = o'$ is excluded. For instance, $o = o_2$ results in

$$\mathbf{I}_{o_2} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ \vdots & \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}.$$

Finally, with the identity vector $\mathbf{e}_o^{O\times 1}$ equation (7.11) can be represented as

$$\left[\mathbf{0}^{(O-1)S\times R\cdot B}\ \left[\left[\mathbf{I}^{S\times S} \oplus \mathbf{e}_o^{\mathrm{T}}\right] \oplus \mathbf{1}^{(O-1)\times 1} + \left[\mathbf{I}_c \oplus \mathbf{I}_o\right]\right]\right] \begin{bmatrix} \boldsymbol{\delta}_{rb} \\ \boldsymbol{\delta}_{os} \end{bmatrix}$$
$$\leq \mathbf{1}^{(O-1)S\times 1} + \frac{1}{d_{\min}} \left[\mathbf{1}^{S\times 1} \oplus \mathbf{I}_o\right] \mathbf{d}_{oO} \tag{7.12}$$

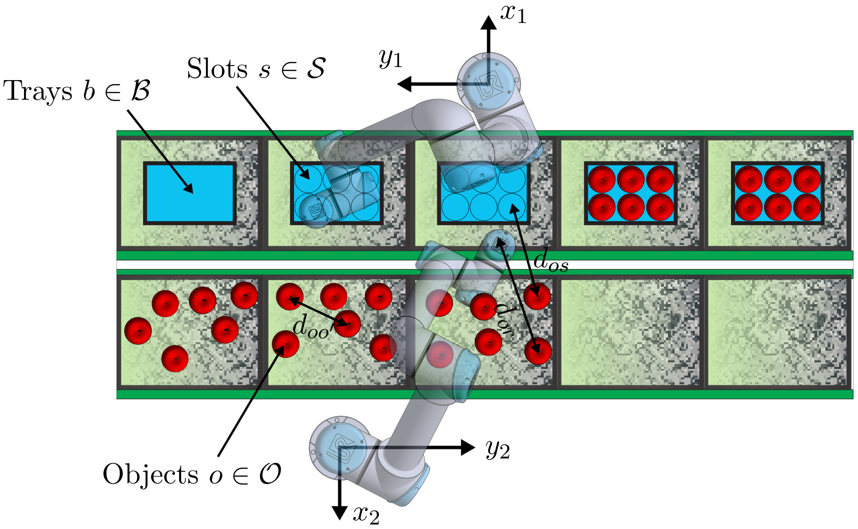

x_1
Trays $b \in \mathcal{B}$
Slots $s \in \mathcal{S}$
y_1
d_{os}
$d_{oo'}$
d_{or}
Objects $o \in \mathcal{O}$
y_2
x_2

Figure 7.4: Schematic illustration of the fruit packing station.

depending linearly on the binary optimization variables.

Since the objects and the trays move on the conveyor belt and their positions change with time, scheduling has to be performed online. As described in Section 4.2.1, minimum-time scheduling requires the transformation of the scheduling problem in the robot configuration space and the computation of multiple inverse kinematics solutions, which are computationally demanding in the case of moving target points. In contrast, minimum-distance scheduling is modeled in the robot workspace and requires only the computation of the Euclidean distances between all objects and slots d_{os}, all pair of objects $d_{oo'}$, and between all objects and robots d_{or}, $\forall o \in \mathcal{O}, \forall s \in \mathcal{S}, \forall r \in \mathcal{R}$, see Figure 7.4. Moreover, the Euclidean distances $d_{oo'}$ and d_{os} do not change as the objects and trays move since their relative position remains constant. Therefore, for the considered robotic system, the objective of the discrete optimization is to minimize the total Euclidean distance traversed by the robot's end-effectors. The paths of the end-effectors can be divided into the initial robot movement from their starting pose to the first picked object, the movement from the object's position to its assigned slot, and the movement from a slot to the next object. The total distance can be expressed as a bilinear function

$$\begin{aligned} f(\boldsymbol{\delta}_d) = & \underbrace{\sum_{\forall r \in \mathcal{R}} \sum_{\forall b \in \mathcal{B}} \delta_{rb} \left(\sum_{\forall o \in \mathcal{O}} \delta_{o\hat{s}_b} \, d_{or} \right)}_{\text{Initial robot movement to the objects}} + \underbrace{\sum_{\forall o \in \mathcal{O}} \sum_{\forall s \in \mathcal{S}} \delta_{os} \, d_{os}}_{\text{Movement from the objects to the slots}} \\ & + \underbrace{\sum_{\forall o' \in \mathcal{O}} \sum_{\forall b \in \mathcal{B}} \sum_{\forall s_i \in \mathcal{S}_b} \delta_{o's_{i+1}} \, d_{o's_i}}_{\text{Movement from the slots back to the objects}} . \end{aligned} \tag{7.13}$$

The initial movement of the robots is modeled by the bilinear part of the cost function and depends on the allocation of a robot to a tray and on the object allocated to the first slot of that tray. The movement from an object to its assigned slot is considered in the middle part of the cost function and can be derived from the binary object to slot allocation variable. Finally, the last part is modeling the path from the current slot s_i to the object o' assigned to the next slot s_{i+1}, as the sequence in which the slots are filled is fixed. The bilinear part of the cost function $f(\boldsymbol{\delta}_d)$ representing the initial movements of the robots can be written as

$$\boldsymbol{\delta}_{rb}^{\mathrm{T}} \underbrace{\begin{bmatrix} \mathbf{d}_{r_1O}^{\mathrm{T}} \oplus \mathbf{I}^{B\times B} \\ \vdots \\ \mathbf{d}_{r_rO}^{\mathrm{T}} \oplus \mathbf{I}^{B\times B} \\ \vdots \\ \mathbf{d}_{r_RO}^{\mathrm{T}} \oplus \mathbf{I}^{B\times B} \end{bmatrix}}_{\mathbf{D}_{RO}} \begin{bmatrix} \delta_{o_1\hat{s}_1} \\ \vdots \\ \delta_{o_O\hat{s}_1} \\ \delta_{o_1\hat{s}_2} \\ \vdots \\ \delta_{o_O\hat{s}_B} \end{bmatrix} = \boldsymbol{\delta}_d^{\mathrm{T}} \mathbf{Q} \boldsymbol{\delta}_d \,, \tag{7.14}$$

where

$$\mathbf{Q} = \begin{bmatrix} \mathbf{D}_{RO} \\ \mathbf{0}^{O\cdot S\times O\cdot B} \end{bmatrix} \begin{bmatrix} \left[\mathbf{0}^{O\times R\cdot B} \left[\mathbf{e}_{\hat{s}_1}^{\mathrm{T}} \oplus \mathbf{I}^{O\times O}\right]\right] \\ \vdots \\ \left[\mathbf{0}^{O\times R\cdot B} \left[\mathbf{e}_{\hat{s}_B}^{\mathrm{T}} \oplus \mathbf{I}^{O\times O}\right]\right] \end{bmatrix} ,$$

Here, $\hat{s}_b$ denotes the slot to be filled first in tray $b \in \mathcal{B}$, and $\mathbf{d}_{rO} = [d_{ro_1}\, d_{ro_2} \cdots d_{ro_O}]^{\mathrm{T}}$ the vector of the distances between robot $r \in \mathcal{R}$ and the objects $o \in \mathcal{O} = \{o_1, \cdots, o_O\}$. The linear part of the cost function representing the movement from the objects to the assigned slots and back to the next selected objects can be expressed as

$$\mathbf{c}^{\mathrm{T}} \boldsymbol{\delta}_d = \left[\left[\mathbf{0}^{1\times R\cdot B}\ \mathbf{d}_{OS}^{\mathrm{T}}\right] + \left[\mathbf{0}^{1\times R\cdot B}\ \mathbf{d}_{OS_B}^{\mathrm{T}}\right]\right] \boldsymbol{\delta}_d \tag{7.15}$$

with

$$\begin{aligned} \mathbf{d}_{OS} &= [d_{o_1s_1}\, d_{o_1s_2}\, \cdots\, d_{o_1s_S}\, d_{o_2s_1}\, \cdots\, d_{o_Os_S}]^{\mathrm{T}} \\ \mathbf{d}_{OS_B} &= [\mathbf{d}_{o_1S_B}^{\mathrm{T}}\, \mathbf{d}_{o_2S_B}^{\mathrm{T}}\, \cdots\, \mathbf{d}_{oS_B}^{\mathrm{T}}\, \cdots \mathbf{d}_{o_OS_B}^{\mathrm{T}}]\,, \quad \text{with} \\ \mathbf{d}_{oS_B} &= [0\, d_{os_1}\, d_{os_2}\, \cdots\, d_{oS/B-1}\, 0\, d_{oS/B+1}\, d_{oS/B+2}\, \cdots\, d_{oBS/B-1}]^{\mathrm{T}}\,. \end{aligned}$$

Finally, the IBLP problem can be given in this form

$$\begin{aligned} \min_{\boldsymbol{\delta}_d} \quad & \boldsymbol{\delta}_d^{\mathrm{T}} \mathbf{Q} \boldsymbol{\delta}_d + \mathbf{c}^{\mathrm{T}} \boldsymbol{\delta}_d \\ \text{s.t.} \quad & \mathbf{A}_{\mathrm{eq}} \boldsymbol{\delta}_d = \mathbf{1} \\ & \mathbf{A}_{\mathrm{in}} \boldsymbol{\delta}_d \leq \mathbf{1} \\ & \mathbf{A}_{\mathrm{sc}} \boldsymbol{\delta}_d \leq \mathbf{b}(d_{\min}) \end{aligned} \tag{7.16}$$

with the equality constraints (7.4) - (7.7), the inequality constraints (7.8) and the safety-related inequality constraints (7.12). The vector $\mathbf{b} \in \mathbb{R}^{O(O-1)S}$ represents the right hand side of the equation (7.12) $\forall\, o \in \mathcal{O}$.

The scheduler computes the optimal allocation of the objects and trays to the robots and provides the position set-points to the MPC layer, which computes the optimal trajectories of the robot manipulators by minimizing the execution time of the tasks, while accounting for potential collisions as described in Section 5.2.2. Let $\mathcal{P} = \mathcal{O} \cup \mathcal{S}$ denote the set of task points consisting of objects and slots. Moving task points require continuously updating the desired final robot state $\mathbf{x}_f(k)$ in the trajectory planning problem (5.48). Let ${}_0\mathbf{p}_p$ denote the position of a task point $p \in \mathcal{P}$ and ${}_0\mathbf{p}_p(k)$ its current from the camera system captured position value relative to the inertial frame. The final target point position is then continuously updated using the prediction

$$ {}_{0}\mathbf{p}_f(k) = {}_{0}\mathbf{p}_p(k) + {}_{0}\mathbf{v}_b(k)\, t_f^*(1-\Delta\tau)\,, \tag{7.17} $$

with ${}_{0}\mathbf{v}_b(k)$ representing the current measured speed vector of the conveyor relative to the inertial frame. Here, $t_f^*(1-\Delta\tau)$ models the remaining time period the robots need to reach the assigned targets with the optimal time value t_f^* computed at the previous planning iteration $k-1$. Applying inverse kinematics and inverse differential kinematics to (7.17) and ${}_{0}\mathbf{v}_b$, along with a desired end-effector orientation, the final position and velocities of the robot joints are calculated to update the desired final state vector $\mathbf{x}_f(k)$.

7.2.2 Simulation results

In the proposed robotics setup, the robots are arranged relative to each other at a distance of 0.98 m frontally and 0.3 m laterally displaced, as depicted in Figure 7.5. With a robot working radius of around 1 m each, this results in a high overlap in the robot working areas and, thus, a high potential for collisions. Therefore, the proposed hierarchical control structure is initially validated on simulations before implementing it in the real experimental setup. The simulation architecture shown in Figure 7.5 slightly differs from the implementation architecture in Figure 5.2 as it is implemented in MATLAB/SIMULINK without involving ROS. In order to provide high-quality visualization of the robots and the experimental setup, a dynamic simulation model in SIMSCAPE has been developed using the CAD data provided by the manufacturer. The SIMSCAPE model is mainly used for the simulation of the forward dynamics, whereas the analytically derived equations of motion are used in feedback control to implement a nonlinear dynamic inversion-based controller (Computed Torque) to compensate the nonlinear dynamics, resulting in a linear closed loop system of n double integrators per robot (see Section 4.2.2.1). For the dynamic model and the Computed Torque controller the robot dynamic parameters estimated in Chapter 3 are used. According to the proposed architecture in Figure 7.5 the computed optimal robot inputs are forwarded to the Computed Torque controller, which generates the torque inputs $\boldsymbol{\tau}_1$ and $\boldsymbol{\tau}_2$ for the robot dynamic model.

For evaluating the performance of the scheduling and trajectory planning method, a data set of randomly distributed points is generated, representing the fruits on the conveyor belt. The coordinate frame $(o_1x_1y_1z_1)$ attached to the base of Robot 1 represents the inertial frame of reference $(o_0x_0y_0z_0)$. Thus, the velocity vector of the conveyor belt

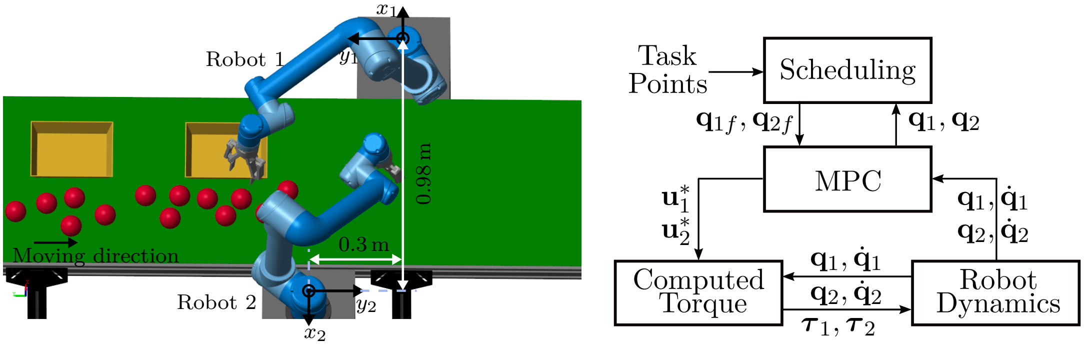
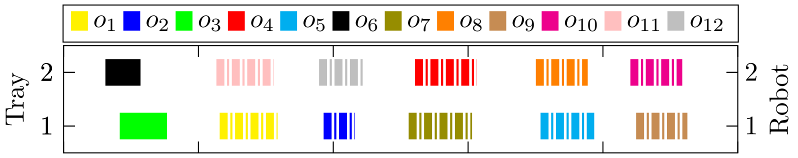
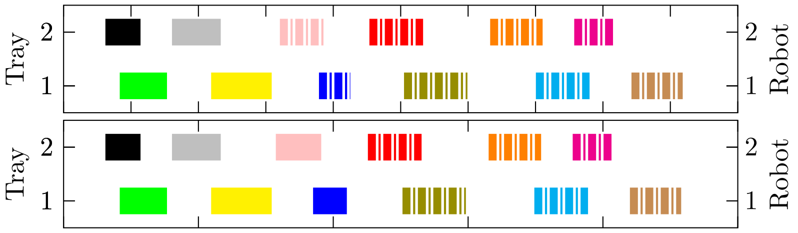

x_1
Robot 1
y_1
0.98 m
0.3 m
Moving direction
Robot 2
y_2
x_2
Task Points
Scheduling
$\mathbf{q}_{1f}, \mathbf{q}_{2f}$
$\mathbf{q}_1, \mathbf{q}_2$
MPC
$\mathbf{u}_1^*$
$\mathbf{u}_2^*$
$\mathbf{q}_1, \dot{\mathbf{q}}_1$
$\mathbf{q}_2, \dot{\mathbf{q}}_2$
Computed Torque
Robot Dynamics
$\mathbf{q}_1, \dot{\mathbf{q}}_1$
$\mathbf{q}_2, \dot{\mathbf{q}}_2$
$\boldsymbol{\tau}_1, \boldsymbol{\tau}_2$

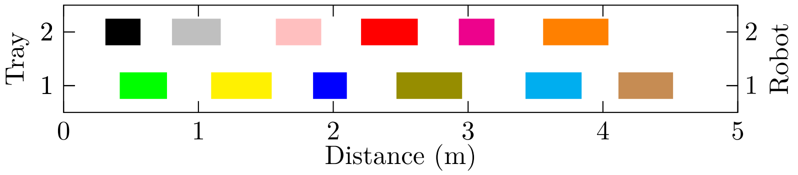
Figure 7.5: Dynamic simulation model along with the used hierarchical control structure with centralized trajectory planning layer.



(a) Initial schedule

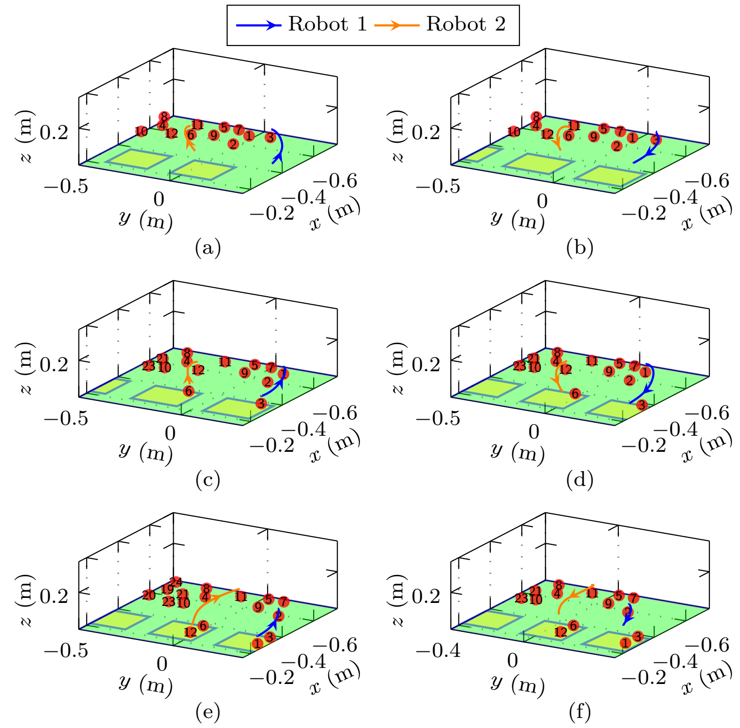


(b) Intermediate schedule



(c) Final schedule

Figure 7.6: Optimal scheduling for two robots, 12 objects and two trays relative to the traversed distance, which is minimized within the optimization problem. The dashed bars show the planed sequence, which has not been applied yet, since the optimization is performed iteratively after each placed object. The white space between the colored bars indicates the distance covered by the robot to pick up the next object. The final schedule shows the resulting sequence of the objects placed in the filled trays after six optimization iterations.



(a) (b) (c) (d) (e) (f)

Figure 7.7: Paths of the robots' end-effectors. (a): From the starting points to the first objects according to the scheduler results; (b), (d), (f): From the objects to the slots in the corresponding trays; (c), (e): From the slots to the next objects selected by the scheduler. As desired, two simultaneously chosen objects do not lie next to each other.

in (7.17) is given by ${}_0\mathbf{v}_b = [0, -v_b, 0]^{\mathrm{T}}$, with $v_b = 5\,\mathrm{cm/s}$. By providing the position of the first 12 objects and the robots to the scheduler, the resulting IBLP problem is solved, computing the initial scheduling results shown in Figure 7.6a). According to the initial schedule, the first tray and the object sequence $\{o_3, o_1, o_2, o_7, o_5, o_9\}$ are assigned to Robot 1, and the second tray along with the object sequence $\{o_6, o_{11}, o_{12}, o_4, o_8, o_{10}\}$ to Robot 2. From the sequences obtained, only the first element is selected. The remaining objects of the respective sequence, which are displayed as dashed bars in Figure 7.6a), are not further considered since the optimization is performed again after the first two objects have been placed. To grasp the first objects o_3 and o_6, the end-effectors will cover a respective distance of 0.31 m and 0.41 m. Once the objects have been grasped, the MPC for trajectory generation is carried out again, targeting to reach the first slots in the corresponding trays in the quickest possible time, see Figure 7.7. After placing the first two objects, the scheduler is executed again, while maintaining the previous robot-tray assignment. The decision of which robot will be filling which tray is only made when the current trays are filled.

As shown in Figure 7.6, the object sequence for Robot 1 is not changing compared to the first iteration. For Robot 2, there is a change in the object sequence resulting in a shorter overall traversed distance. Contrary to the initial plan, the next object to be picked up by Robot 2 is Object o_{12}, which is located at a shorter distance than object o_{11}. With the two selected objects, o_1 and o_{12}, MPC is performed again, leading the end-effectors from the current slots to the objects, see Figure 7.7c). In this way, MPC and scheduler are executed alternately until the trays are filled. The final sequence of selected objects after six optimization iterations is displayed in Figure 7.6c). In the subsequent optimization iteration, each robot is then assigned an object sequence and a tray.

The feasibility of the scheduling model depends mainly on the value of the minimum distance parameter $d_{\min}$ and the related constraints (7.9). Larger values for $d_{\min}$, as may be preferred for safety reasons, reduce the solution space and the performance of the optimization problem. However, the minimum distance is related to the geometry of the robots and cannot be chosen arbitrarily small. Hence, for larger $d_{\min}$ values, the number of slots $S = |\mathcal{S}|$ should be much smaller than the number of objects $O = |\mathcal{O}|$ to reduce the required amount of feasible combinations.

When performing task scheduling, the resulting execution sequence of the pick-and-place task does not usually result in intersecting paths of the robot end-effectors as shown in Figure 7.7. In Section 5.3, it was presented that path intersections are mainly due to the restrictions of placing objects in the slots of the same class. With all objects and slots belonging to the same class, motions with intersecting robot paths will rarely occur. Therefore, to analyze the performance of the collision avoidance constraints, scheduling is further constrained, resulting in pick-and-place tasks with high inter-robot collision potential. The sequence of these tasks is defined in such a way that without considering collision avoidance restrictions, the robots would always collide with each other. The focus of the simulation is to analyze the performance of the inter-robot collision avoidance constraints for the given experimental setup using both the 2D projection method and 3D optimal normal vector calculation as described in Section 5.2.2.2. The parameters used for simulations are the same used during the experiments in Section 5.3, presented in Table 5.1, and the trajectory planning problem is solved using IPOPT.

A schematic representation of the simulation results when using the 2D projection method is shown in Figure 7.8. Depending on the robot's relative position, the robot parts

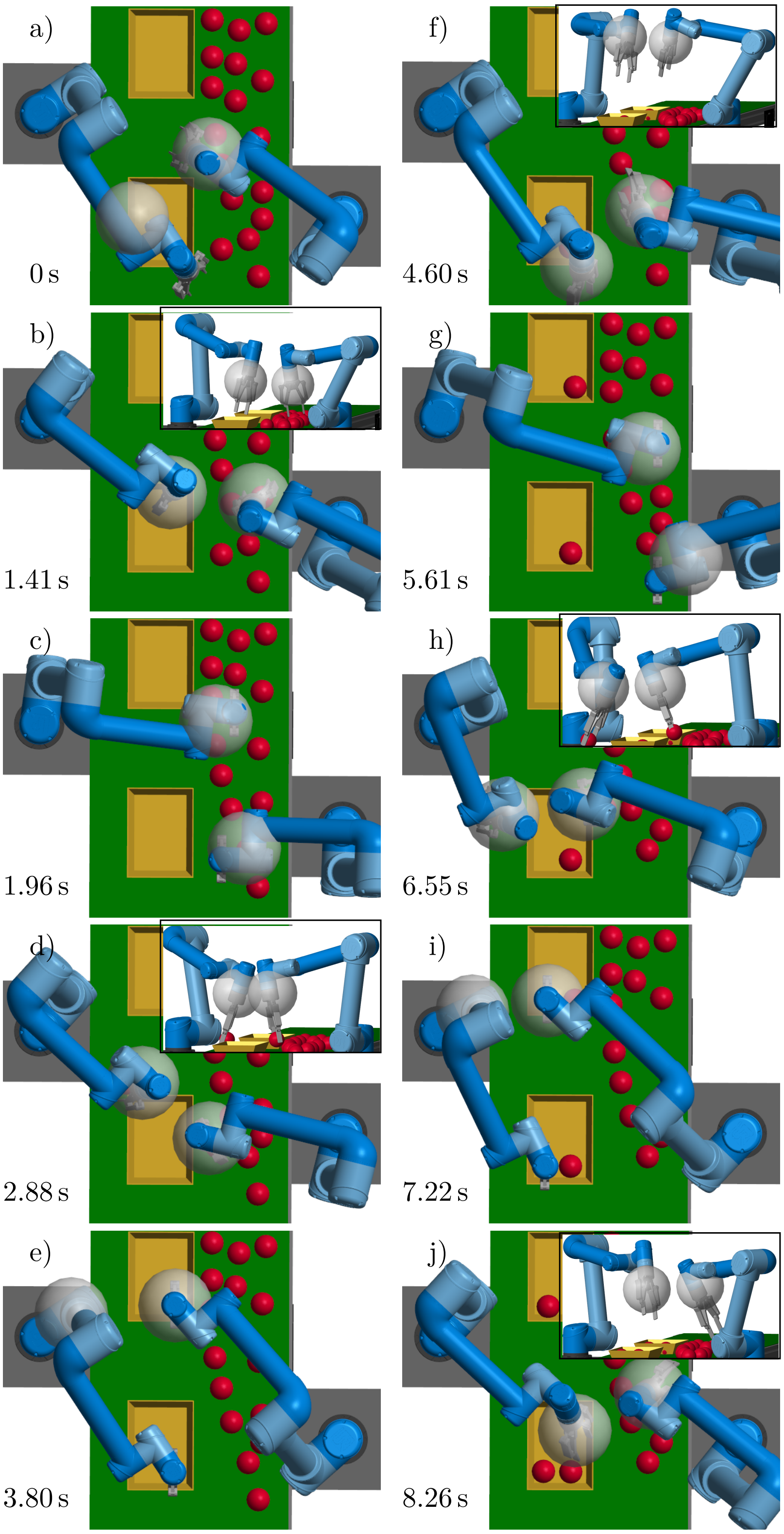


Figure 7.8: Two robots performing pick-and-place tasks are exposed to high collision potential. Collisions would always occur in the absence of their avoiding constraints. As shown, depending on the robot configurations, the collision spheres restrict the relative motion of the closest robots' parts. The illustrations show the effectiveness of the applied 2D projected constraints (cf. Figure 5.4). The following robot movements are shown: a) to c): from the initial positions to the first objects; c) to e): from the first objects to the slots; e) to g): from the slots to the next objects; and g) to j): from the objects to the next slots and on the way to the objects.

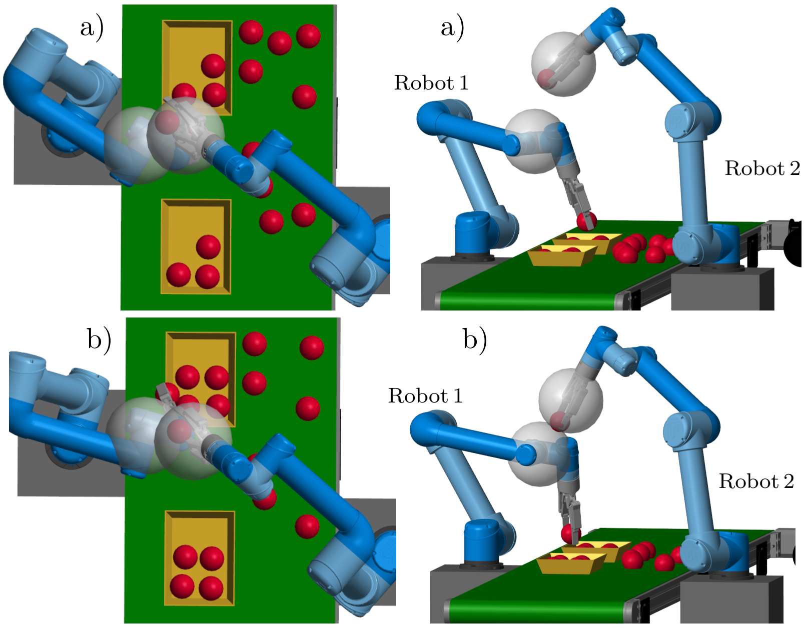
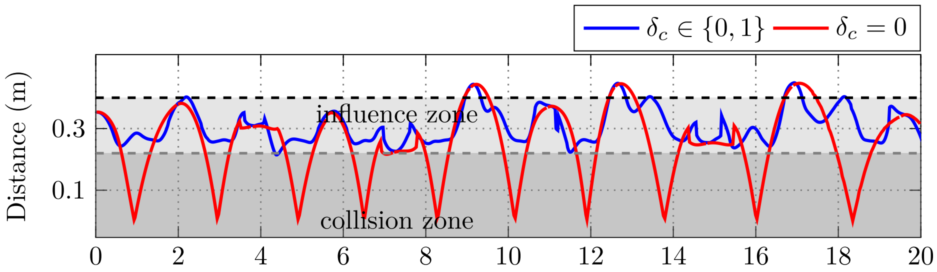
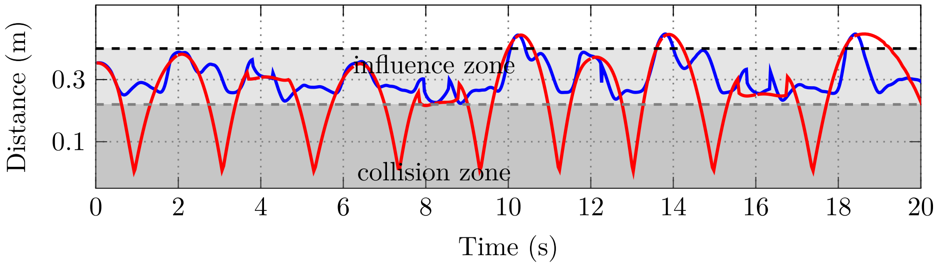


Figure 7.9: Collision avoidance when applying 3D collision avoidance constraints according to Figure 5.5. Robot 2 is, in both cases a) and b), selecting trajectories above Robot 1 on his way to placing an object into the corresponding slot.



(a) 2D projection method.



(b) 3D optimal normal vector.

Figure 7.10: Distance between the centers of the spheres. $\delta_c = 0$ denotes the case where the collision constraints are not active. As shown, the distance between the spheres' centers undercuts the safety distance, defined by the sum of the radii of the proximity spheres, resulting in collisions between the robots. For $\delta_c \in \{0, 1\}$, the collision avoiding constraints are active within the influence zone, i. e., $d \leq d_{\text{in}}$, in (5.31), resulting in no overlapping of the spheres at any time. Consequently, for both presented methods on choosing a vector normal to a tangential separating plane, the minimum distance between the approximating spheres always remains above the safety distance, ensuring safe robot operation.

closest to each other are geometrically approximated by spheres. As the robots move, the spheres' position is continuously updated following the minimum distance between the Bézier curves, which approximate the robot kinematics. The collision constraints force the spheres to slide along the separating tangents, thus preventing the robots from colliding. Figure 7.8 shows the collision-free robot motions from the initial position to the first objects (a - c) and back to the corresponding slots (c - e) to place the objects. Afterward, the robots go to the next objects and place them in the slots (e - i) and so forth. It can be seen that collision avoidance is performed in such a way that the robot arms do not move above each other. However, this is not always the case if a vector normal to a 3D separating plane is used. For the same pick-and-place tasks, it is shown in Figure 7.9 that sometimes, the robots decide to cross over each other to avoid collisions.

The distance between the centers of the spheres, for the robot motions illustrated in Figure 7.8 and Figure 7.9, is shown in Figure 7.10. The dark gray area marks the region where the spheres overlap, i. e., where the robots collide. It is evident that without collision conditions ($\delta_c = 0$), collisions would always occur between the robots. The proposed collision avoidance constraints, which are active ($\delta_c = 1$) within the defined influence area, ensure safe robot operation by preventing the minimum distance between the robots from falling below a certain safety distance.

Detailed results on collision avoidance and its effect on the robot trajectories and the algorithm's overall performance are presented in the following implementation section.

7.2.3 Experimental results

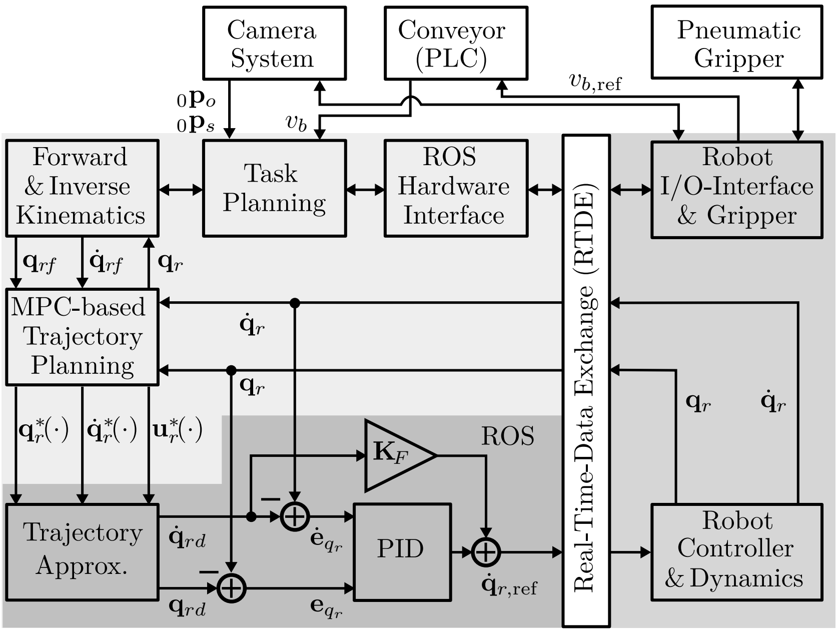


Figure 7.11: System architecture from Figure 5.9 extended by a conveyor belt, camera system for object detection and inspection, and a pneumatic unit to control the pneumatic robot grippers.

The implementation architecture is presented in Figure 7.11 and is similar to the architecture described in the Experimental Section 5.3, extended by a conveyor belt, camera system for object detection, and a pneumatic control unit to control the grippers.

The scheduler gets information about the position of objects ${}_0\mathbf{p}_o$, slots ${}_0\mathbf{p}_s$ (relative to the inertial frame), and the actual conveyor speed v_b, and computes feasible robot target points, which are transformed into final robot joint positions $\mathbf{q}_{rf}$ and velocities $\dot{\mathbf{q}}_{rf}$. ROS Hardware Interface along with the input / output robot interface is used to close or open the grippers, activate the camera system, and set the reference velocity $v_{b,\mathrm{ref}}$ for the PLC.

Experiments on two specific test cases validate the algorithm's performance. Test Case 1 is mainly constructed to reproduce the simulation results and confirm the effectiveness of the collision avoidance strategy in ensuring safe robot operation in an actual application. It is shown that, although the robotic setup with highly overlapping robot workspaces can lead to challenging manipulation tasks, even in the worst-case scenario, two robot manipulators achieve shorter cycle times than deploying a single robotic arm. Test Case 2, on the other hand, shows rather a normal operating mode of the robotic system involving task scheduling and online trajectory planning to pick and place moving objects on the conveyor.

Test Case 1

In Test Case 1, a set of twelve objects and two trays with six slots each is considered, distributed as shown in Figure 7.12. Performing task scheduling results in the task sequence shown in Figure 7.13b), with objects one to six and Tray 1 assigned to Robot 2, and objects seven to twelve along with Tray 2 assigned to Robot 1. To generate challenging robot motions and test the performance of the collision avoidance constraints, the scheduling algorithm is further constrained, resulting in the task sequence shown in Figure 7.13a), where the robot-object assignment remains the same with Robot 1 filling Tray 1 and Robot 2 filling Tray 2. In this case, the assigned object-tray combination leads to overlaps in the paths of the robot end-effectors and, thus, to motions with high inter-robot collision potential, similar to the test case considered in simulations.

A visual representation of the experimental results for Test Case 1a is shown in Figure 7.14. The scene succession a) to e) represents an entire pick-and-place sequence starting from the robot's initial position, picking up the assigned objects o_1 and o_7, and

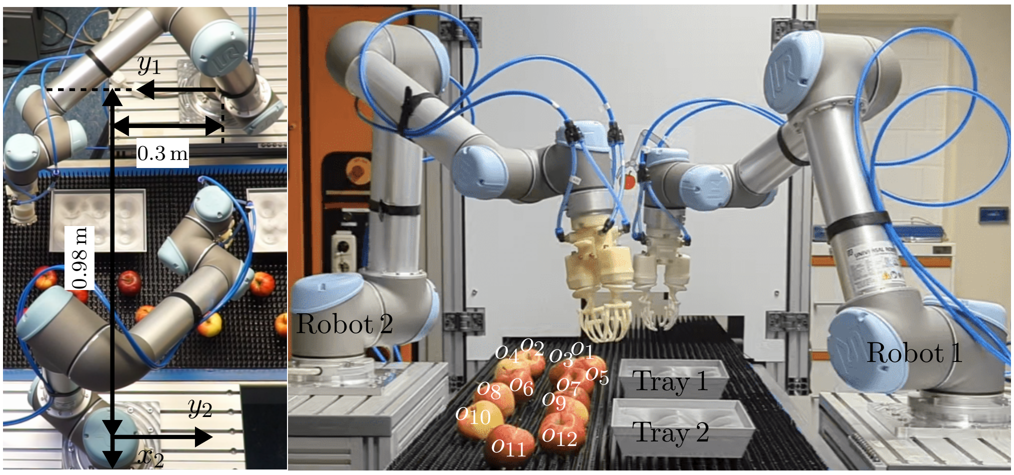


Figure 7.12: Experimental setup for Test Case 1: The robots have to pick up six objects each and place them in the trays.

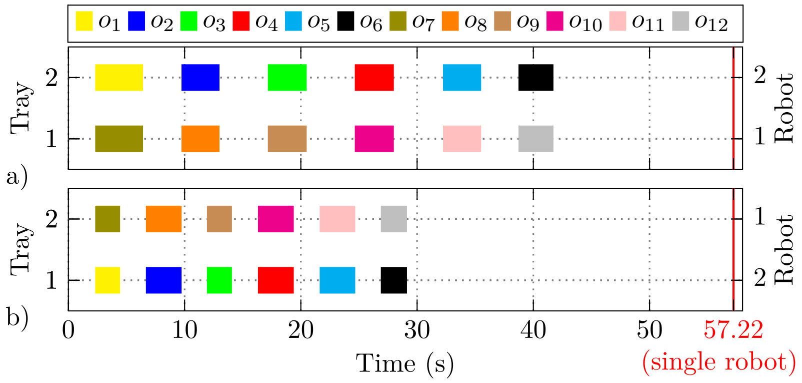
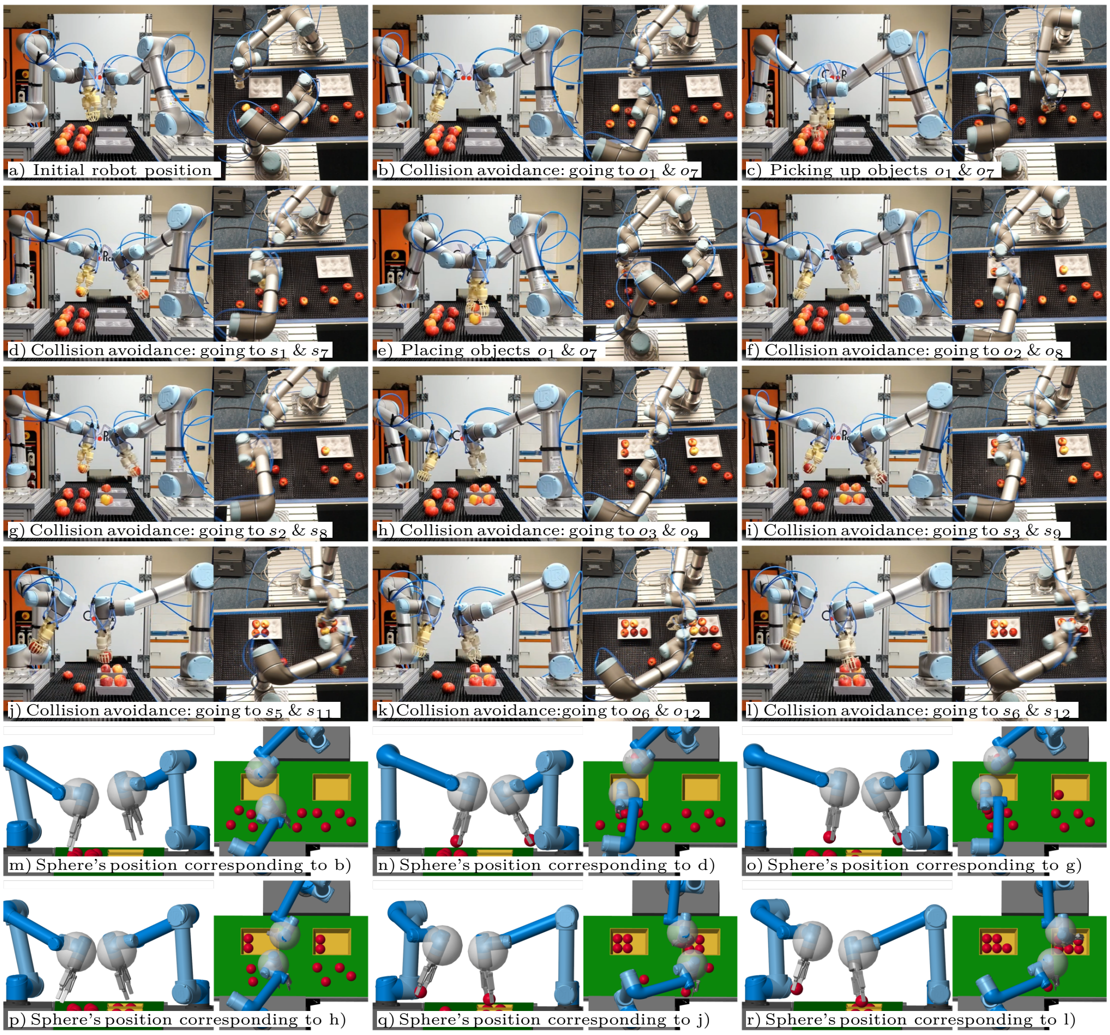


Figure 7.13: Test Case 1: Task scheduling for two robots, 12 objects, and two trays. a) Test Case 1a: The objects $\{o_7, o_8, o_9, o_{10}, o_{11}, o_{12}\}$ and Tray 1 are assigned to Robot$\tilde{1}$, and the objects $\{o_1, o_2, o_3, o_4, o_5, o_6\}$ along with Tray 2 to Robot 2, leading to tasks with high collision potential. b) Test Case 1b: The objects $\{o_7, o_8, o_9, o_{10}, o_{11}, o_{12}\}$ and Tray 2 are assigned to Robot 1, and the objects $\{o_1, o_2, o_3, o_4, o_5, o_6\}$ along with Tray 1 to Robot 2, leading to tasks with low collision potential. The red vertical line denotes the time it would take for a single robot to perform the pick-and-place tasks.

placing them in the respective slots. On their way to the objects and the respective slots, the robots sidestep each other to avoid collisions, as shown in the images b) and d). After placing the first objects, the robots go to the following assigned objects o_2 and o_8, according to the schedule shown in Figure 7.13a), and place them in the respective slots of the trays, and so forth. The other images, f) to l) represent snapshots of collision avoidance motions of the robots when they are on their way to the objects or slots. This clearly shows that collision avoidance is performed so that the robot arms do not move across each other. Subfigures m) to r) show the location of the geometry approximating spheres corresponding to the robot configurations in b), d), h), and l), respectively. Therefore, the dynamic simulation model of the experimental setup is used. In order to perform the same robot motions as in the experiment, a Computed Torque controller is used by applying the inverse robot dynamics and a PD controller with feedforward acceleration term, making the robots to track position, velocity, and acceleration trajectories recorded while performing the experiment. The position of the spheres is then visualized using recorded optimal curve parameters λ_1^* and λ_2^*, computed during the experiment by solving the minimum-distance optimization problem. It is shown that, similar to the presented simulation results, the spheres approximate the robot's geometry following the computed minimum distance between the robot arms. The computed minimum distance between the centers of the proximity spheres in Test Case 1 is shown in Figure 7.15.

To further evaluate the performance of the algorithm, the computed minimum final time t_f^* is shown in Figure 7.16, and the joint position, velocity, and acceleration trajectories for Robot 1 are shown in Figure 7.17. The trajectories are presented only for Robot 1 and half of the experiment duration since the trajectories for Robot 2 and the remaining tasks have similar profiles and do not provide additional information.

In Test Case 1b, starting from the initial position Robot 1 will pick up Object o_7, and Robot 2 Object o_1. This is the same as in Test Case 1a and corresponds to the image sequence a) - d) in Figure 7.14. However, Robot 1 then places its assigned object in Slot s_7 (Tray 2) and Robot 2 in Slot s_1 (Tray 1), see Figure 7.13b). Compared to Case 1a, this leads to non overlapping in the paths of the robot end-effectors and thus to low collision risk. Due to different scheduling, robots need to avoid each other only during the execution of the first tasks when moving from the initial points to the first objects. This can also be observed in the time course of the minimum distance and the optimal final time shown in Figure 7.15b) and Figure 7.16b), respectively. While executing the first task, the value of the minimum distance enters in the influence zone ($d \leq d_{\text{in}}$) activating the constraints and thus prohibiting the minimum distance from falling below the critical value. The computed optimal time for the first task is also higher and the decreasing slope is changing with time. The remaining task points do not lead to robot motions with high collision potential as the robots do not get close to each other. In this case the robots need less time to reach the assigned goals resulting in shorter transition times and



a) Initial robot position

b) Collision avoidance: going to o_1 & o_7

c) Picking up objects o_1 & o_7

d) Collision avoidance: going to s_1 & s_7

e) Placing objects o_1 & o_7

f) Collision avoidance: going to o_2 & o_8

g) Collision avoidance: going to s_2 & s_8

h) Collision avoidance: going to o_3 & o_9

i) Collision avoidance: going to s_3 & s_9

j) Collision avoidance: going to s_5 & s_{11}

k)Collision avoidance:going to o_6 & o_{12}

l) Collision avoidance: going to s_6 & s_{12}

m) Sphere's position corresponding to b)

n) Sphere's position corresponding to d)

o) Sphere's position corresponding to g)

p) Sphere's position corresponding to h)

q) Sphere's position corresponding to j)

r) Sphere's position corresponding to l)

Figure 7.14: The pictures show a side and top view of the experiment for Test Case 1a. Scenes a) - e) show an entire pick-and-place cycle starting from the start position a), picking up the assigned objects c), and placing them in the respective slots e). The robots avoid collisions on the way to the objects and slots, as shown by scenes c) and d), respectively. The other snapshots f) through l) also show conflict resolution and collision avoidance. The Simulation scenes m) to r) display the position of the spheres corresponding to the robot configurations shown in b), d), g), h), j) and l), respectively. The experiment lasted a total of approximately 41.8 s.

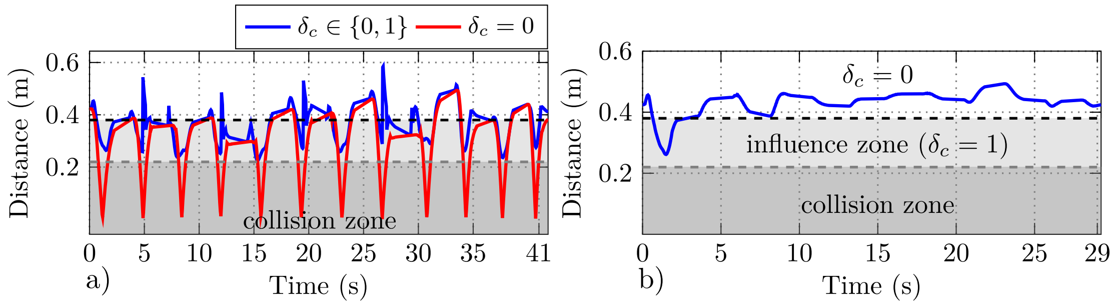


Figure 7.15: Distance between the centers of the spheres for Test Case 1. a) Test Case 1a: For $\delta_c \in \{0, 1\}$, the collision avoidance constraints are active within the influence zone, preventing the robots from colliding with each other. b) Test Case 1b: Robots perform tasks with low collision potential. The inter-robot collision avoidance constraints are only during the first task execution active and prohibit the minimum distance from undercutting the critical minimum value.

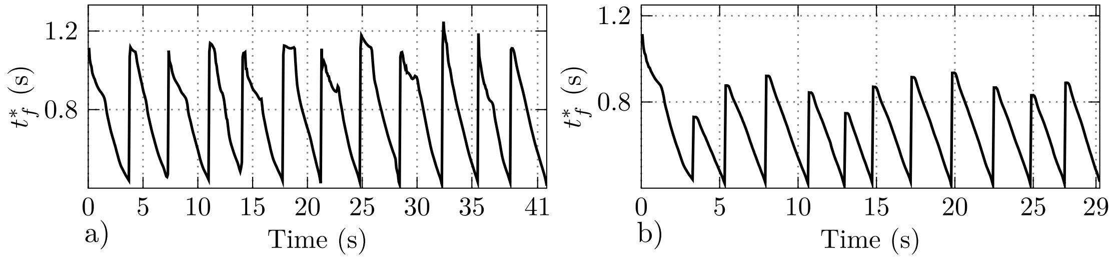


Figure 7.16: Minimum final time for Test Case 1a shown in a) and Test Case 1b shown in b).

faster overall mission completion, see Figure 7.16b). The resulting minimum final time t_f^* decreases steadily between a maximum (beginning of a task) and a minimum (end of a task). The entire experiment lasts about 29.3 s, which is 12.5 s faster compared to Test Case 1a. Deploying a single robot to execute the same tasks, i. e., fill both trays, results in an overall task execution time of about 57.2 s. Thus, depending on the task planning, two robots can perform the entire task between 15.4 s and 27.9 s faster than a single robot.

A correlation between the inter-robot collision avoidance constraints and the resulting time evolution of the trajectories in Figure 7.17 is evident. When avoiding collisions, the wrists of the robots are more involved in the trajectory planning, as the spheres are mainly located in the wrist and end-effector area, see Figure 7.14. In Figure 7.17, it is shown that these robot joints or the ones farthest from the target exhibit bang-bang acceleration profiles. While executing the first task in Test Case 1b, the fifth robot joint reacts quickly to avoid collisions, which is also reflected in the velocity and acceleration profile. During the execution of the remaining tasks with low collision potential, the fifth robot joint is not changing, keeping the orientation of the gripper perpendicular to the task points. In this case, the robot joints two and three are the dominant ones since they have to cover a larger distance and exhibit higher velocity and acceleration values.

The experimental results for Test Case 1 are summarized in Table 7.1 with the resulting computation times shown in Figure 7.18. The constraints for inter-robot collision avoidance are applied along the first half of the prediction horizon N_T, and the velocity parameter ϵ is chosen to be updated dynamically using $\epsilon = 0.24\, t_f^*$. For more information on the choice of the parameters is referred to Section 5.3. The computation time for the MPC-based trajectory planning problem and the minimum distance calculation increases

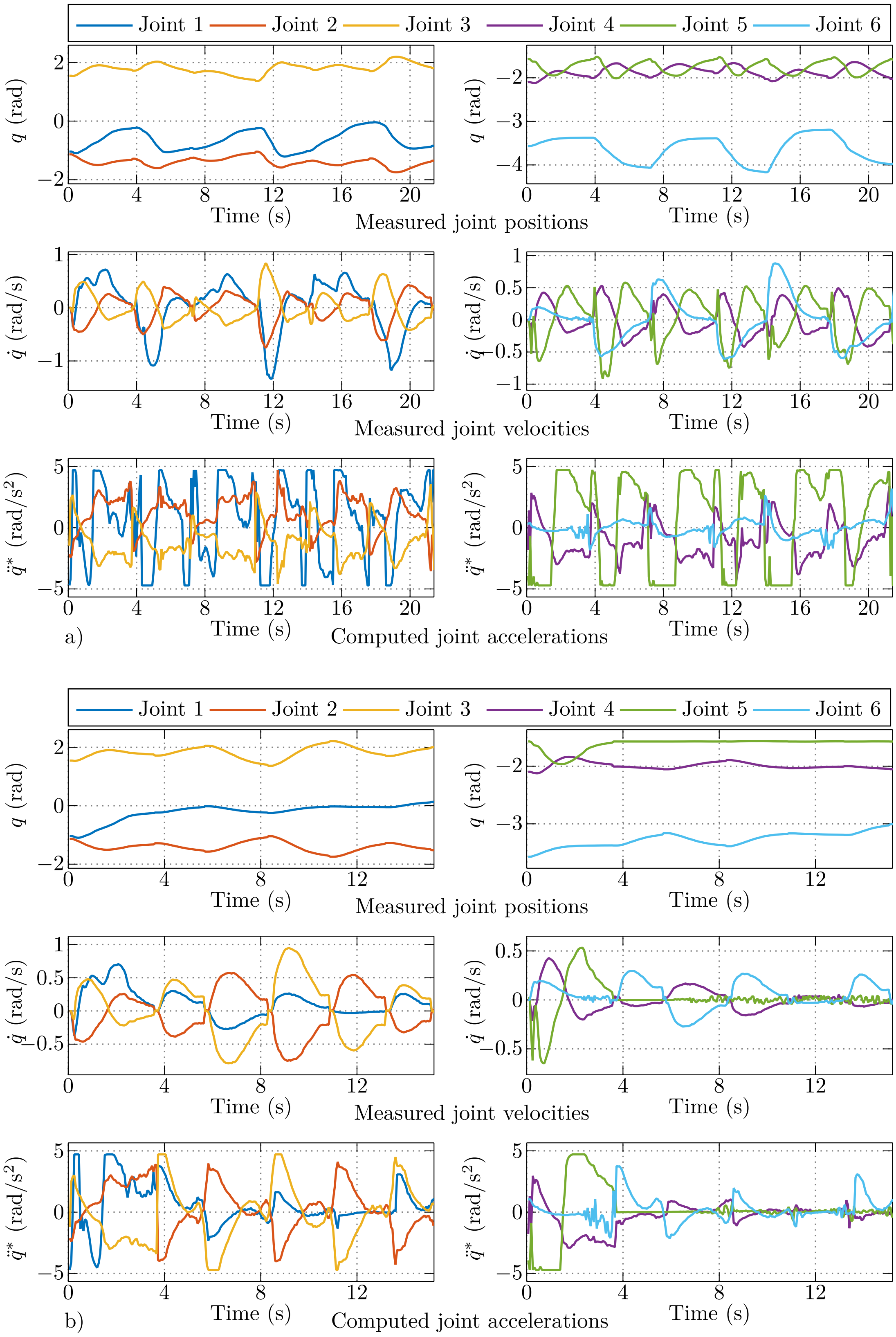


Figure 7.17: Measured joint positions, velocities, and computed optimal accelerations for Robot 1 in Test Case 1a shown in a) and Test Case 1b shown in b).

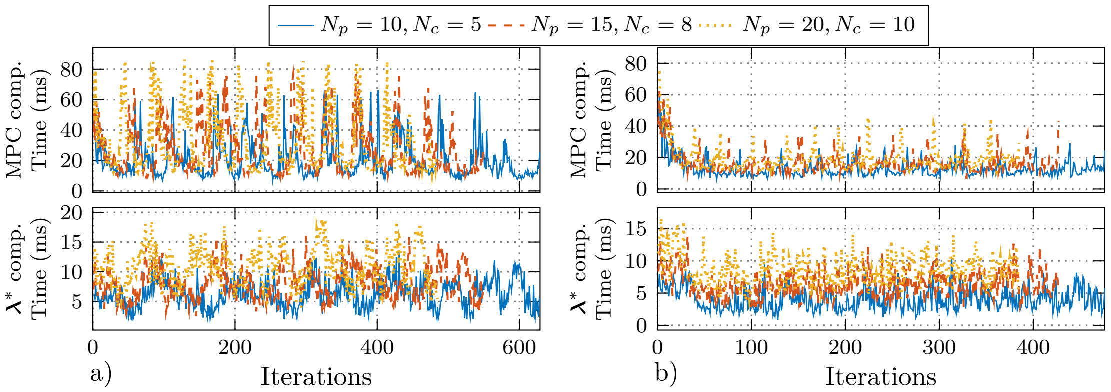


Figure 7.18: Computation time for the MPC-based trajectory planning problem and the minimum distance calculation for three different horizon lengths in Test Case 1a (a) and Test Case 1b (b).

Table 7.1: Experimental results for Test Case 1 using different horizon lengths.

Horizon	$N_T = 10$ $N_C = 5$	$N_T = 15$ $N_C = 8$	$N_T = 20$ $N_C = 10$
Max solver time (ms) (MPC / min. distance)	a) 66 / 13 b) 58 / 10	a) 77 / 16 b) 62 / 14	a) 86 / 19 b) 74 / 17
$t^*_{f,\max}$ (s)	a) 1.25 b) 1.12	a) 1.30 b) 1.14	a) 1.27 b) 1.17
Experiment duration (s)	a) 41.8 b) 29.3	a) 42.8 b) 29.7	a) 43.2 b) 29.9
Average solver iterations (MPC / min. distance)	a) 13 / 6 b) 8 / 3	a) 14 / 6 b) 8 / 3	a) 15 / 6 b) 8 / 3

with the increased horizon. In Test Case 1a, the computation time is higher since the collision-avoidance constraints significantly influence the computation time due to their complexity and nonlinearity. This can be seen Figure 7.18a), where high fluctuations in the MPC computation time are evident. For $N_T = 20$ and $N_C = 10$, the total computation time sometimes exceeds 100 ms being slightly larger than the requested maximum control horizon. In this case, $t_c > \Delta t_{\max}$, so $N_C > N_T/2$ should be chosen to ensure collision-free trajectory generation above $\Delta t_{\max}$ towards the end of the planning when $t^*_f \approx t_{\min}$, see Section 5.3. However, increasing N_C will also increase the computation time. In general, towards the end of the trajectory planning the computation time is shorter as the robots are before t^*_f approaches $t_{\min}$ outside the danger zone for inter-robot collisions ensured by the minimum-distance scheduling constraints. In Test Case 1b, the computation time remains below 100 ms and reaches its maximum value at the beginning. This is expected since the robots only need to avoid collisions in the first task and subsequently perform tasks with low collision potential.

To analyze the effect of the collision-free horizon N_C, experiments are performed with a fixed prediction horizon $N_T = 15$ and $N_C \in \{10, 12, 14\}$. The experimental results are summarized in Table 7.2 for the more critical Test Case 1a. Increasing N_C increases the number of the collision avoidance constraints resulting in higher computation times. For $N_C \geq 12$ the total computation time is above 100 ms. With the selected values for N_C also toward the end of the trajectory planning the collision-free horizon is longer than the maximum computation time, i. e., $\Delta\tau t_{\min} N_C \geq t_c$ holds true. A longer horizon N_C has

Table 7.2: Experimental results for Test Case 1a and $N_T = 15$.

Horizon	$N_C = 10$	$N_C = 12$	$N_C = 14$
max solver time (ms) (MPC / min. distance)	80 / 19	85 / 22	94 / 24
$t^*_{f,\max}$ (s)	1.25	1.25	1.26
Experiment duration (s)	44.1	45.6	46.1
average solver iterations (MPC / min. distance)	14 / 6	15 / 6	14 / 6

Table 7.3: Experimental results for Test Case 1a, with $N_T = 15$ and $N_C = 8$.

Velocity parameter ϵ	$0.10\,t^*_f$	$0.18\,t^*_f$	$0.36\,t^*_f$
max solver time (ms) (MPC / min. distance)	76 / 16	73 / 17	73 / 24
$t^*_{f,\max}$ (s)	1.20	1.22	1.22
Experiment duration (s)	45.6	44.42	44.8
average solver iterations (MPC / min. distance)	14 / 6	14 / 6	14 / 6

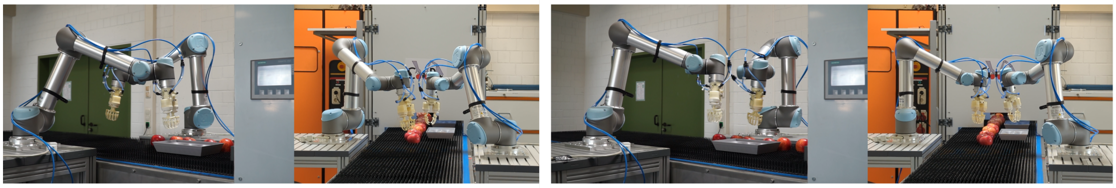
no effect on the obtained maximum value of the cost function $t^*_{f,\max}$. However, we see that with increased N_C the experiment duration increases by about $1\,\mathrm{s} - 2\,\mathrm{s}$. Applying the collision avoidance constraints along a longer horizon decreases the slope and the convergence rate of t^*_f, leading to slightly longer overall task execution times.

The velocity parameter ϵ is chosen by trial and error while performing the experiments. A smaller ϵ-value may lead to robots coming closer to each other, resulting in robots requiring more time to resolve conflict situations. A larger ϵ, on the other hand, pushes the robots farther from each other. Consequently, both increasing and decreasing ϵ can result in longer task execution times as shown in Table 7.3 when comparing to the results in Table 7.1 for $\epsilon = 0.24 t^*_f$ and $N_T = 15, N_C = 8$.

When collisions are not a concern, the interior point solver requires only few iterations to converge to a feasible solution. The number of iterations increases if inter-robot collisions pose an issue. In this case, the solver requires, on average, 14 iterations to converge to a feasible solution for the trajectory planning problem, respectively six iterations for the minimum distance calculation.

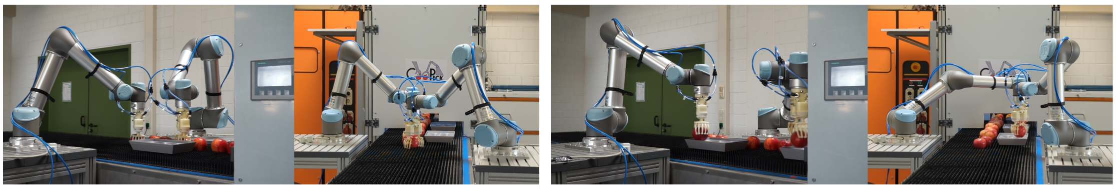
Test Case 2

Test Case 2 demonstrates a normal operating mode of the robotic application with the two manipulators performing pick-and-place to increase the level of automation of an existing manual packing process. For object detection and pose estimation the deep learning-based algorithm presented in [151] is used. To perform the pick-and-place tasks, scheduling is performed first to assign a tray and a sequence of objects to a robot. Since the task points are of the same class and each robot is assigned a tray, this rarely results in motions with intersecting end-effector paths and, thus, high inter-robot collision risks. Figure 7.19 shows a few sequences of the performed pick-and-place tasks. The picture sequences a) - d) display an entire pick-and-place sequence starting from the initial robot



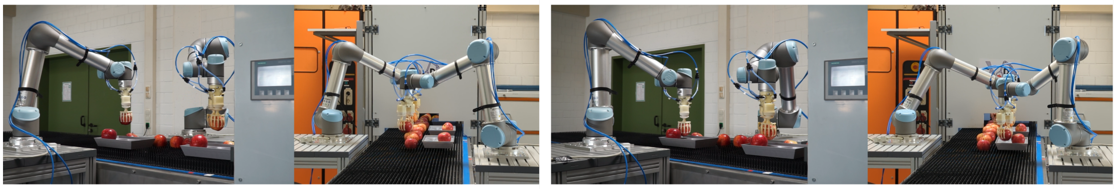
a) Initial robot position.

b) Collision avoidance on the way to the first objects.



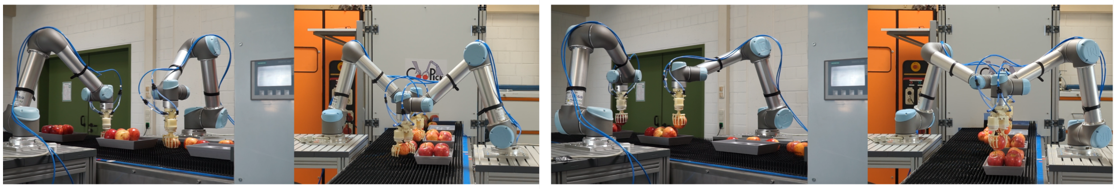
c) Picking up the first two assigned objects.

d) Placing first objects in the respective slots.



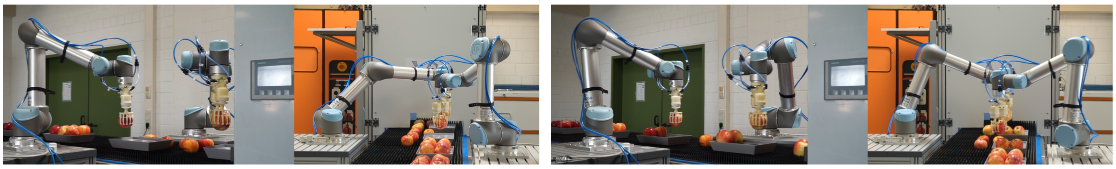
e) Picking up the next two assigned objects.

f) Placing next two objects in the respective slots.



g) Continue filling Tray 1 and Tray 2.

h) Placing last two objects in Tray 1 and Tray 2.



i) Start filling two next trays 3 and 4.

j) Continue filling Tray 3 and Tray 4.

Figure 7.19: Robots performing pick-and-place tasks on a conveyor belt in Test Case 2.

position a), collision-avoidance on the way to the first objects b), picking ap the first two objects c), and placing the two objects in the respective slots d). In Figure 7.19h) the robots complete the filling of the first trays and start filling two next upcoming trays i).

From the performance of the task execution, Test Case 2 is similar to Test Case 1b, as the robots usually perform tasks with low collision potential. The computed optimal final time t_f^* and the overall experimental results for a set of 24 objects and four trays are shown in Figure 7.20 and Table 7.4, respectively. A local increase of the optimal final time t_f^* towards the end of the execution of the first two tasks is related to the position prediction (7.17) of the task points. More precisely, this is due to the changes in the belt speed, which are higher at the beginning due to friction effects and decrease with time, so that the speed converges to a steady state and maintains the desired value of 5 cm/s nearly constant. The computation time shown in Figure 7.21 for the entire experiment duration and three different horizon lengths never exceeds 100 ms.

The entire experiment of filling four trays with two robots lasts about 48.1 s. The same experiment is performed deploying a single robot manipulator, resulting in an overall experiment duration of about 84.2 s. The double-arm solution being 36.1 s faster than the single-robot-arm solution is a promising indicator that efficient deployment of multiple robot arms can result in an improved performance in terms of robot cycle time and overall working throughout.

Table 7.4: Experimental results for Test Case 2.

Horizon	$N_T = 10$ $N_C = 5$	$N_T = 15$ $N_C = 8$	$N_T = 20$ $N_C = 10$
max solver time (ms) (MPC / min. distance)	53 / 11	64 / 15	74 / 18
max t_f^* (s)	0.93	0.97	1.04
Experiment duration (s)	48.1	48.4	48.6
average solver iterations (MPC / min. distance)	8 / 3	8 / 3	8 / 3

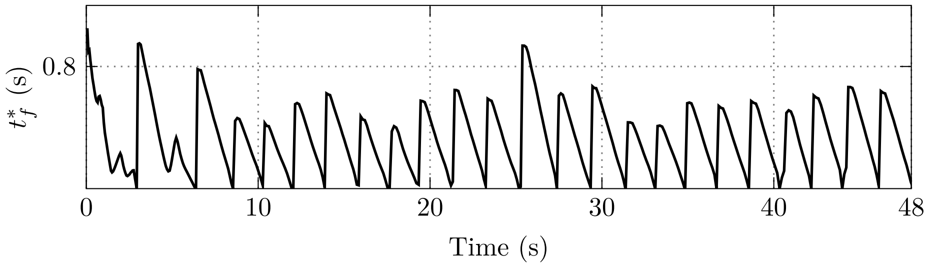


Figure 7.20: Computed minimum final time for Test Case 2 shown for a set of 24 objects and 4 trays.

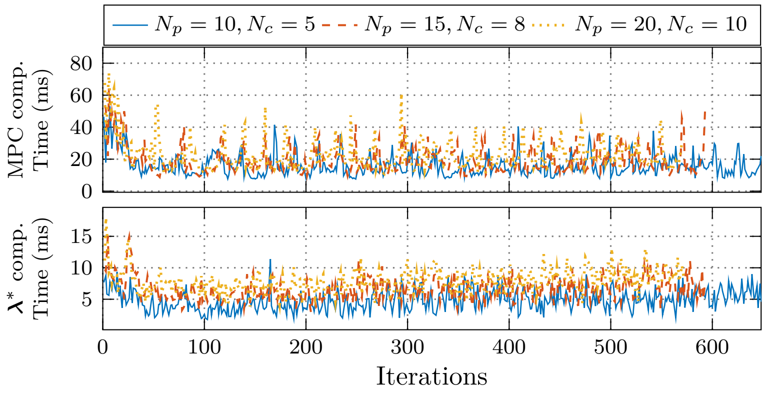


Figure 7.21: Computation time for the MPC-based trajectory planning problem and the minimum distance calculation for three different horizon lengths in Test Case 2.

7.3 Mobile robot platform

This section presents a mobile robot equipped with two collaborative robotic arms around a flexible robotic system for collecting, transporting, and distributing parts between workstations in industrial environments. The focus lies on the autonomous navigation of the robot in an indoor environment with static obstacles. Figure 7.22 shows the robot demonstrator along with the control architecture, which involves simultaneous localization and mapping (SLAM), global path planning, and MPC-based path-following control presented in Chapter 6. The MPC implementation builds upon the work presented in [83], see also Section 6.3.

Prior to implementing a global path planner, a map of the environment is generated using the ROS package *gmapping* for laser-based SLAM. Using point cloud data from two laser scanners and robot odometry data, a 2D occupancy grid map of the environment is created. The generated occupancy grid map is forwarded along with the localized robot position $\mathbf{p}_\mathrm{p}$ to a global path planner. For the global path planning between the start and a target robot position, a sampling-based probabilistic roadmap [20] planner is used. In the construction phase of the planner, the environment is randomly sampled, building a roadmap (graph). The occupancy map including an example of generated node graphs lying outside the occupied areas of the map is shown in Figure 7.23. In the query phase of the planner, an A*-based searching algorithm is applied to connect the start and goal point and obtain a feasible collision-free path for the robot. With the selected nodes as control points, the computed path is parametrized using basis splines (B-splines) [152]. The parametrized path $\mathbf{p}_\mathrm{r}(\varpi) = [x_\mathrm{r}(\varpi), y_\mathrm{r}(\varpi)]^\mathrm{T}, \varpi \in \Pi$, is then forwarded to the local MPC-based path-following controller, which generates optimal robot inputs $\mathbf{u}_\mathrm{m}^* = [v_x^*, v_y^*, \omega^*]^\mathrm{T}$ to follow the reference path at the highest possible speed, see Chapter 6.

For the performance evaluation of the proposed planning and control architecture, two paths are considered, as shown in Figure 7.24. The closed path consists of six path segments $\mathbf{p}_\mathrm{r}(\varpi), \varpi \in [\varpi_i, \varpi_{i+1}]$ for $i \in \{1, \ldots, 6\}$, where $\varpi_1 = \varpi_7$ denotes the beginning and the end of the path. The second path $\mathbf{p}_\mathrm{r}(\varpi), \varpi \in [\varpi_1, \varpi_2]$ represents an open path connecting the start ϖ_1 and the goal ϖ_2 robot positions.

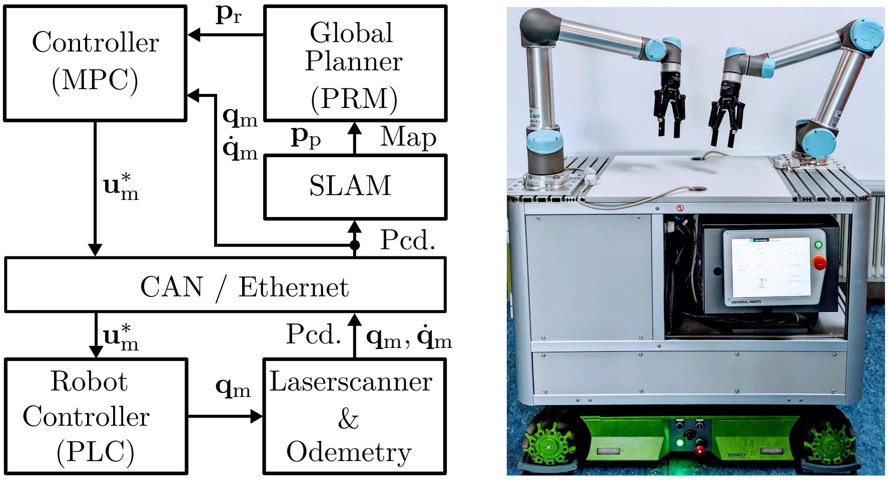


Figure 7.22: Mobile robot demonstrator along with the introduced control architecture.

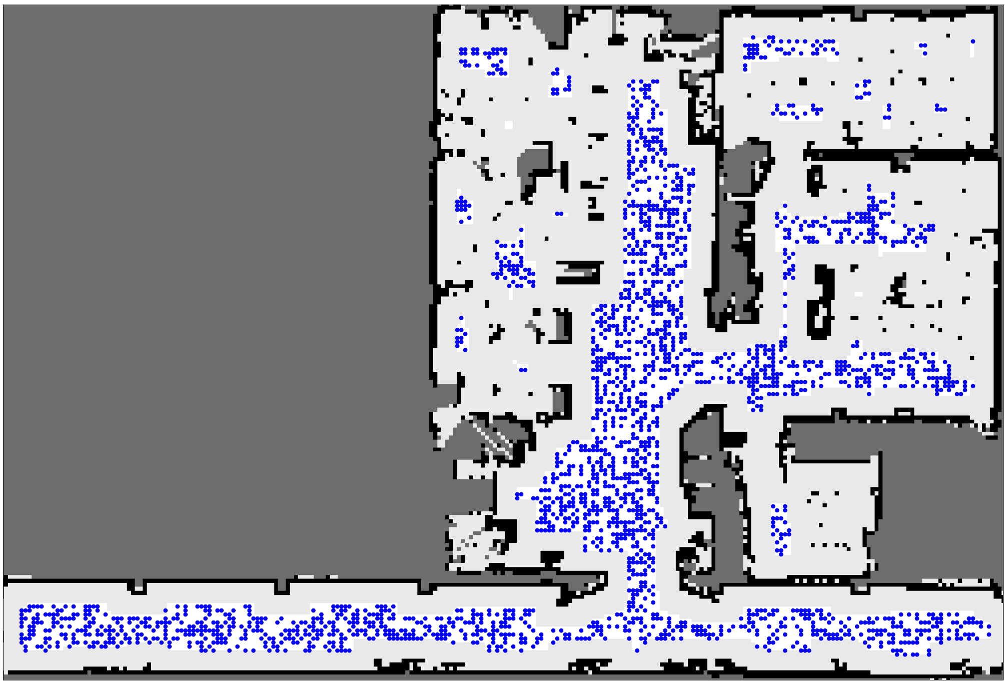
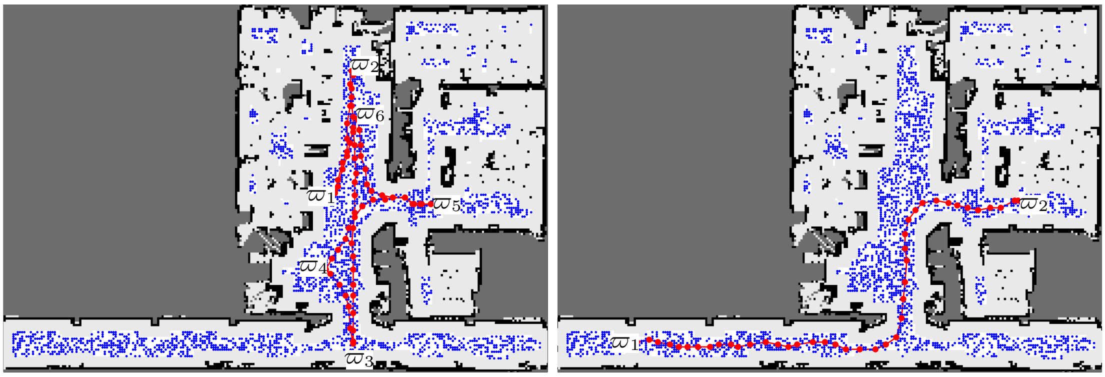


Figure 7.23: Occupancy map of the environment including PRM nodes.



a) Closed path

b) Open path

Figure 7.24: Control points of a closed and open path used for experimental validation.

The desired orientation of the robot $\theta_d(\varpi)$ along the reference path is chosen to be equal to the orientation of the reference velocity vector $\psi_{\mathrm{r}}(\varpi)$ (6.2), i. e.,

$$\theta_d(\varpi) = \arctan\left(\frac{y'_{\mathrm{r}}(\varpi)}{x'_{\mathrm{r}}(\varpi)}\right). \tag{7.18}$$

Due to the omnidirectional kinematics, for the robot orientation it is not required to exactly match the value of the reference angle to keep the robot heading tangential to the path. A robot orientation $\theta = \theta_d(\varpi) \pm \pi$ would lead to the same heading with the robot eventually driving in the other direction. To incorporate this in the path-following problem, the alignment error of the robot $e_\theta = \theta - \theta_d$ is modified to

$$e_\theta = 1 - \cos\left(\theta - \theta_d\right). \tag{7.19}$$

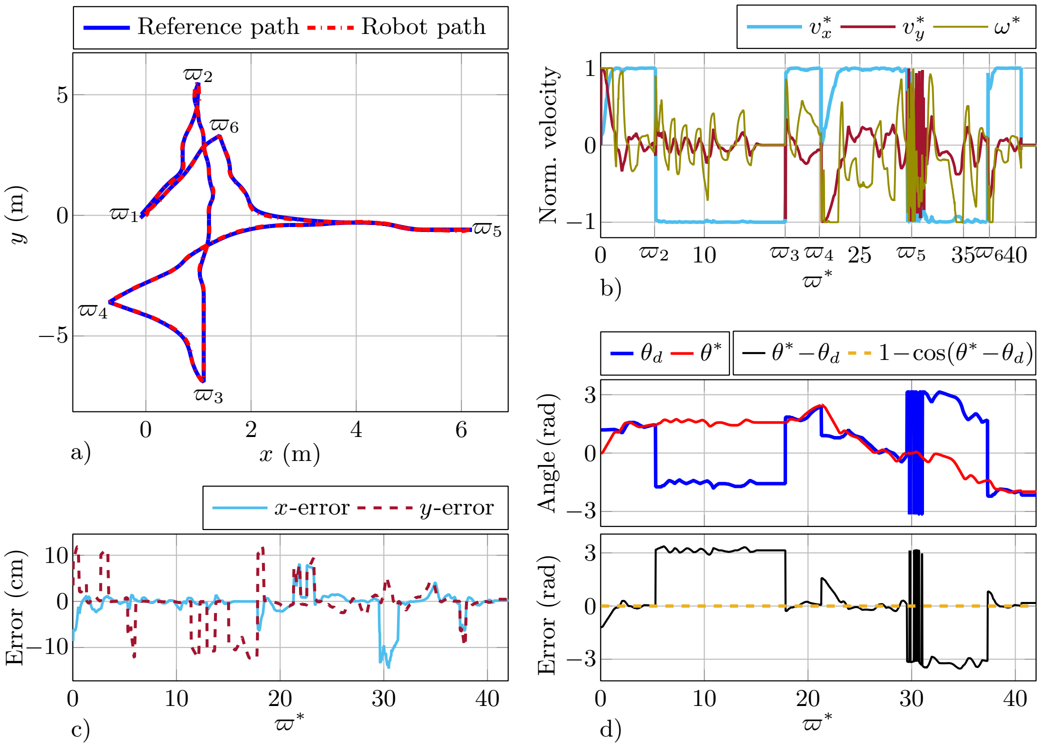


Figure 7.25: Experimental results of path-following control using a closed path composed of different path segments.

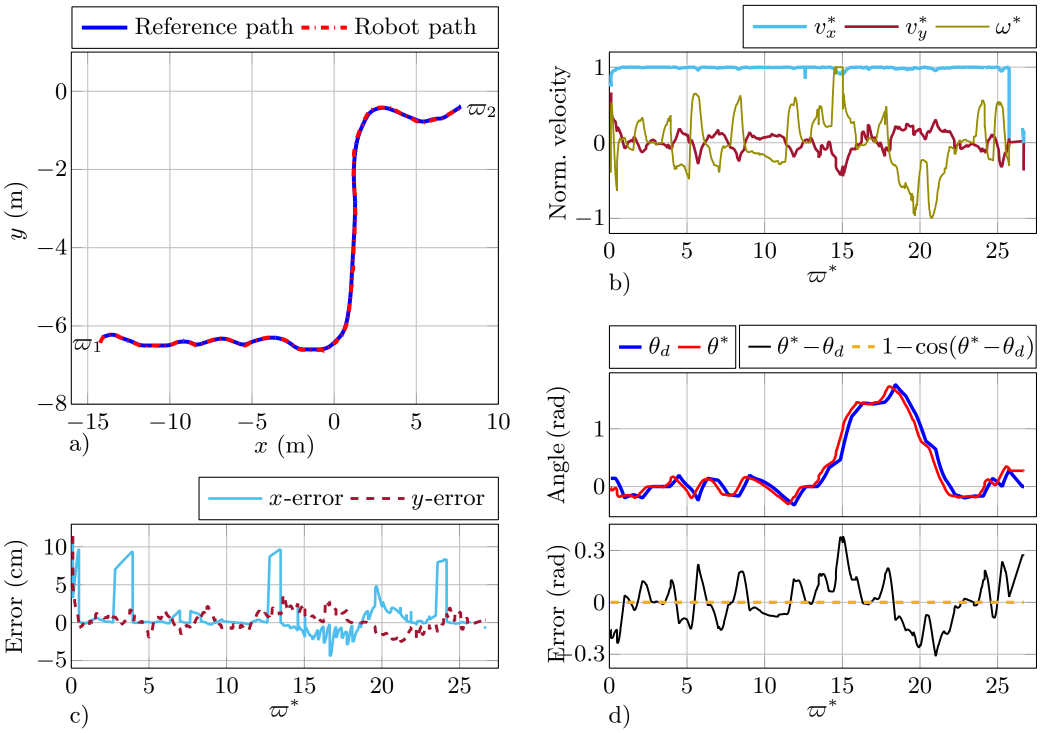


Figure 7.26: Experimental results of path-following control from a start to a desired final robot configuration.

The experimental results for both paths are shown in Figure 7.25 and Figure 7.26. Subfigures c) show a comparison between the reference and the driven robot path with the path-following errors displayed in subfigures c) and d), respectively. The optimal control inputs computed by the path-following controller (6.16) are shown in the subfigures Figure 7.25a) and Figure 7.26a). The parameters of the MPC-based controller are chosen as described in Section 6.3.

Overall, the path-following controller shows a very good tracking performance with a maximum path-following error around 10 cm at some points along the path, which at the target point is, however, reduced to about 1 cm showing a very good convergence of the robot position towards the final goal. According to the proposed maximum-speed path-following strategy, see Section 6.2, the robot will follow the reference path aiming to maximize its speed, i. e., the covered distance along the path. Since the robot heading is chosen to follow the reference speed vector, the robot will exhibit maximum speed along its x-axis, which is shown in the profile of the computed optimal speed $\mathbf{u}_\mathrm{m}^* = [v_x^*, v_y^*, \omega^*]^\mathrm{T}$. The velocity v_y^* oscillates around the zero value, compensating the path error along the robot y-axis, while v_x^* maintains most of the time a maximum value, being mainly responsible for the robot progression along the path. In Figure 7.25a) it can be seen that in the vicinity of a sharp corner at the end of the path segments, i. e., $\varpi \in \{\varpi_1, \dots, \varpi_6\}$, the robot sometimes is switching its direction of motion keeping the alignment error small by minimizing (7.19).

The closed reference path contains a singular point at the transition between the two path segments for $\varpi \approx \varpi_5$, resulting in the reference robot orientation (7.18) switching between $\theta_d(\varpi) \in \{-\pi/2, \pi/2\}$. Since, in this case, the optimality conditions (6.16f), (6.16g) are not satisfied, the solver requires more computation effort to converge to a suboptimal solution resulting in a slower convergence rate of the optimal path variable ϖ^*. The robot velocities exhibit high-frequency oscillations around the singularity point for a few iterations, yet the controller can overcome this point and advance the path following. Although the reference orientation $\theta_d(\varpi)$ is switching, the resulting robot orientation θ^* remains continuous according to (7.19).

The computation time of the MPC-based path-following controller is shown in Figure 7.27 for the more challenging closed path using different prediction horizons $N_\mathrm{p} \in \{10, 15, 30\}$. It is evident that the computation time depends on the chosen prediction horizon, as it influences the size of the optimization problem. A larger prediction horizon also affects the convergence rate of the path parameter, especially at singular points or transition points between path segments, resulting in an increased number of MPC iterations and longer computation times.

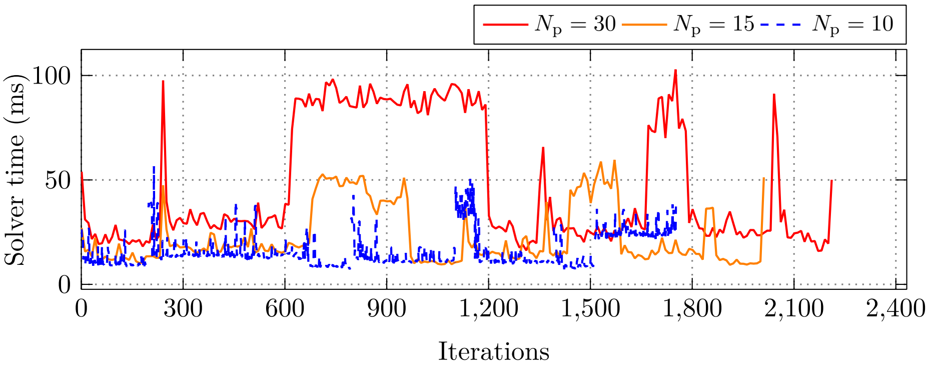


Figure 7.27: Computation time of the path-following controller over MPC iterations.

CHAPTER 8

Summary and Outlook

The goal of this work was to develop optimization methods for task scheduling and predictive-based trajectory planning for robot manipulators and path-following control for omnidirectional mobile robots. In order to increase the efficiency of robot deployment on working stations involving single or double robot manipulators, the overall robot cycle time was introduced as a performance criterion for the algorithm development. Therefore, task scheduling and online trajectory planning are performed directly in the configuration robot space $\mathcal{C}$, leading to joint position, velocity, and acceleration trajectories sent to the local robot controller without involving further planning or processing instances. For the task sequence planning, scheduling models in the operational robot space $\mathcal{W}$ are also introduced, resulting in minimum-distance scheduling without considering multiple solutions of the inverse kinematics of the robots.

The scheduling algorithms are based on the well-established symmetric TSP, with modifications involving inverse kinematics of the robots for minimum-time planning and the incorporation of task-specific and safety-related constraints. Online robot trajectory planning is based on the formulation of a model predictive controller as time-optimal optimization problem. Besides the robot kinematics and basic geometric information of the working environment, the planning algorithms do not require a time-consuming offline preprocessing stage, as is the case with sampling-based planners. Moreover, robot limitations to avoid collisions between the robot links, with the static working environment, and between the robots in a multi-robot operation are incorporated in the optimization problem as state-dependent constraints ensuring safe robot operation.

In addition to the algorithms for task and trajectory planning for robot manipulators, this work presented a predictive-based controller for high-speed path-following control of an omnidirectional mobile robot. All algorithms are validated on experimental setups and demonstrators involving single and double UR5 robot manipulators from UNIVERSAL ROBOTS and the omnidirectional mobile robot DONKEYmotion.

Since kinematic and dynamic models of the considered robots are required for algorithm development and performance validation, basic information on mathematical modeling of serial-linked robot manipulators and omnidirectional mobile robots was provided in Chapter 2. A dynamic parameter identification was performed in Chapter 3 for the robotic arms. Therefore, the system dynamics was expressed linearly in a set of basis parameters, which were then estimated by introducing optimized persistently excitation trajectories. The identification trajectory was computed by minimizing the condition number and solved by applying a memetic algorithm, resulting in a trajectory well suited to excite dynamics, granting small standard deviations for the estimated parameters.

Algorithms for task and trajectory planning of a robot manipulator involving multiple task points of different classes were introduced in Chapter 4. Therefore, two optimization-based approaches were presented, a two-layer hierarchical control structure, and a mixed-integer hybrid controller. The hierarchical structure enables an interaction of a discrete IBLP scheduling algorithm with a time-optimal MPC problem for trajectory generation. The hybrid controller represents a codesign approach that combines the two optimization-based layers into a single optimization problem for online robot tasks and trajectory planning. What appeared to necessarily result in a challenging MINLP problem was transformed into a larger MIQCP problem by applying convex relaxation techniques, implying real-time task and trajectory planning. Experiments have been conducted with a robotic manipulator using ROS, followed by a brief discussion of the advantages and disadvantages of the proposed approaches. Experimental validations showed that the hybrid controller in the form of a mixed-integer optimization problem has comparable performance to the time-optimal hierarchical controller despite its high computational cost and can recursively plan robot tasks and trajectories online.

Chapter 5 introduced an extension of the hierarchic controller for two robotic manipulators performing simultaneous pick-and-place tasks on a shared operational space with overlapping robot working areas. Both task scheduling and trajectory planning incorporate collision avoidance constraints for a safe robot operation. Whereas only a centralized approach was used for the scheduling layer, a distributed layer of communicating DMPC algorithms was proposed for the underlying trajectory planning layer in addition to that of a centralized MPC architecture. In the effort to generate time-optimal and real-time trajectories while avoiding collisions between the robots, a continuous approximation of the robot geometry by Bézier curves was introduced at the cost of a coarser geometry approximation by moving proximity spheres without compromising the safe robot use. To this end, an efficient collision avoidance strategy has been formulated by defining velocity restrictions along tangent separating planes as inequality constraints of an MPC problem. Thereby, the underlying predictive character of the algorithms is of eminent interest as potential collisions can be timely anticipated and, thus, avoided in advance. This fact underpins the key role and necessity of predictive control strategy in cooperative robotics. From the implementation perspective, however, this invokes online real-time calculations of trajectory adaptations, which also has been successfully demonstrated in this work using an experimental setup and demonstrator with two robotic arms performing pick-and-place tasks with static and moving task points. For the considered applications, it was shown that using two robot arms in a tight shared workspace leads, even in the worst case, to an improvement in cycle times compared to deploying a single robot for the execution of the same tasks.

A path-following model predictive controller for mobile robots was introduced in Chapter 6, formulated in the Frenet-Serret frame. For a given parametrized geometric reference path, dynamics of the path parameter were introduced and incorporated into the path-following optimization problem. Furthermore, an automatic parameter tuning approach was used to find a balance between path error minimization and robot speed maximization. The path-following controller was implemented on an omnidirectional mobile robot, and the performance was analyzed based on three different reference paths. The performed experiments demonstrated the effectiveness of the MPC controller in achieving accurate path following while maximizing the robot's speed.

In Chapter 7, demonstrators motivated by specific industrial applications were presented, addressing the need and possibility of integrating robot manipulators into existing manual processes, which, by an efficient robot deployment driven by the proposed algorithms, can result in increased productivity and flexibility. For a mobile robot platform, the introduced path-following controller was coupled with a sampling-based global path planner, showing the effectiveness of the controller for fast and accurate path following in indoor navigation scenarios.

Task planning and motion control are two fundamental problems in robotics that have dominated the research landscape in this field over the past few decades and remain an active field of research. The algorithms for task scheduling introduced in this work are based on symmetric TSP and do not take into account the dependency of the robot cycle time or the traversed distance on the motion direction. The motion cost is modeled based on geometric and kinematic parameters without considering the system's dynamics. A possible extension would be to adopt existing asymmetric approaches to consider, besides gravity, other dynamic phenomena by keeping the incorporation of multiple inverse kinematics solutions in the optimization problem. Another modification could involve the minimum-distance scheduling problem defined in the robot operational space to include, next to the Euclidean distance, the orientation of the task points in the scheduling model without enhancing the need to compute multiple solutions of the inverse kinematics.

Task planning can be tightly coupled to trajectory planning, as addressed in this work in the context of the hybrid controller. The introduced approaches do not consider robot environments where static obstacles or hurdles surround robots and task points. Other existing methods address these problems by combining task sequencing and path planning within the scope of computationally costly offline frameworks. Since the hybrid controller inherently includes task planning in the MPC-based online trajectory generation, it is possible to incorporate additional constraints and analyze to what extent these or similar controllers can be applied in obstacle-cluttered environments.

For the cooperative task and trajectory planning of multi-robots in a shared environment, only the hierarchic controller was used in this work. Future work could incorporate both planning layers into a single layer analogous to the hybrid controller introduced for a single robot manipulator, resulting, primarily due to the nonlinearity of the collision avoidance constraints, in a challenging MINLP problem. The proposed novel collision avoidance approach based on continuous robot geometry approximations involves the computation of the minimum distance between approximating curves, formulated as an optimization problem. Other approaches could be used to solve this problem, reducing further the computational effort. Furthermore, collision avoidance is mainly concerned with avoiding collisions between the robot arms without accounting for cluttered working environments, which can be subject to further improvements. The algorithm can also be extended to consider more robot manipulators using distributed optimization techniques, similar to the proposed architecture with a distributed trajectory planning layer.

The indoor navigation of mobile robots involves, besides path following, global path planning, perception, localization, etc., and remains an open research topic. Regarding the proposed predictive path-following controller, an obvious extension is incorporating further constraints in the MPC algorithm to avoid collisions with dynamic obstacles. To this end, the controller could be tightly coupled to the perception module involving additional sensors like cameras and LiDAR (Light Detection and Ranging). Recent research also includes data-driven approaches based on machine learning and artificial intelligence. An implementation of the introduced MPC controller using Deep Neural Network (DNN) provided promising results for further research in this direction.

Bibliography

[1] A. Jurkat, R. Klump, and F. Schneider, “Tracking the rise of robots: The IFR Database,” *Jahrbücher für Nationalökonomie und Statistik*, vol. 242, no. 5-6, pp. 669–689, 2022. [Online]. Available: `https://doi.org/10.1515/jbnst-2021-0059`.

[2] S. Berger and B. Armstrong, “The Puzzle of the Missing Robots,” *MIT Case Studies in Social and Ethical Responsibilities of Computing*, no. Winter 2022, Jan. 2022, https://mit-serc.pubpub.org/pub/puzzle-of-missing-robots.

[3] L. Sanneman, C. Fourie, and J. A. Shah, “The state of industrial robotics: Emerging technologies, challenges, and key research directions,” *Foundations and Trends® in Robotics*, vol. 8, no. 3, pp. 225–306, 2021, ISSN: 1935-8253.

[4] S. Alatartsev, S. Stellmacher, and F. Ortmeier, “Robotic Task Sequencing Problem: A Survey,” *Journal of Intelligent & Robotic Systems*, vol. 80, pp. 279–298, 2015. [Online]. Available: `https://api.semanticscholar.org/CorpusID:207174341`.

[5] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. USA: Princeton University Press, 2007, ISBN: 0691129932.

[6] Q. Zhang and M.-Y. Zhao, “Minimum time path planning of robotic manipulator in drilling/spot welding tasks,” *Journal of Computational Design and Engineering*, vol. 3, no. 2, pp. 132–139, 2016, ISSN: 22884300.

[7] L. L. Abdel-Malek and Z. Li, “The application of inverse kinematics in the optimum sequencing of robot tasks†,” *International Journal of Production Research*, vol. 28, no. 1, pp. 75–90, 1990.

[8] P. T. Zacharia and N. A. Aspragathos, “Optimal robot task scheduling based on genetic algorithms,” *Robotics and Computer-Integrated Manufacturing*, vol. 21, no. 1, pp. 67–79, 2005, ISSN: 07365845.

[9] E. K. Xidias, P. T. Zacharia, and N. A. Aspragathos, “Time-optimal task scheduling for articulated manipulators in environments cluttered with obstacles,” *Robotica*, vol. 28, no. 3, pp. 427–440, 2010.

[10] P. T. Zacharia, E. K. Xidias, and N. A. Aspragathos, “Task scheduling and motion planning for an industrial manipulator,” *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 6, pp. 449–462, 2013, ISSN: 0736-5845. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S0736584513000410`.

[11] E. K. Xidias, P. T. Zacharia, and N. A. Aspragathos, "Time-optimal task scheduling for two robotic manipulators operating in a three-dimensional environment," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 224, no. 7, pp. 845–855, 2010, ISSN: 0959-6518.

[12] S. Sutdhiraksa and R. Zurawski, "Scheduling robotic assembly tasks using petri nets," in *Proceedings of IEEE International Symposium on Industrial Electronics*, IEEE, 1996, pp. 459–465, ISBN: 0-7803-3334-9.

[13] K. Baizid, R. Chellali, A. Yousnadj, A. Meddahi, and T. Bentaleb, "Genetic algorithms based method for time optimization in robotized site," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 1359–1364.

[14] Y. Edan, T. Flash, U. Peiper, I. Shmulevich, and Y. Sarig, "Near-minimum-time task planning for fruit-picking robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 1, pp. 48–56, 1991.

[15] S. Dubowsky and T. Blubaugh, "Planning time-optimal robotic manipulator motions and work places for point-to-point tasks," *IEEE Transactions on Robotics and Automation*, vol. 5, no. 3, pp. 377–381, 1989.

[16] A. Kimmel and K. E. Bekris, "Scheduling pick-and-place tasks for dual-arm manipulators using incremental search on coordination diagrams," 2017. [Online]. Available: `https://api.semanticscholar.org/CorpusID:9630338`.

[17] J. K. Behrens, R. Lange, and M. Mansouri, "A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8705–8711.

[18] H. Touzani, H. Hadj-Abdelkader, N. Séguy, and S. Bouchafa, "Multi-robot task sequencing & automatic path planning for cycle time optimization: Application for car production line," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1335–1342, 2021.

[19] S. M. Lavalle, J. J. Kuffner, and Jr., "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, 2000, pp. 293–308.

[20] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[21] M. Gharbi, J. Cortés, and T. Siméon, "Roadmap composition for multi-arm systems path planning," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 2471–2476.

[22] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016.

[23] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. Bekris, "DRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning," English (US), *Autonomous Robots*, vol. 44, no. 3-4, pp. 443–467, Mar. 2020, ISSN: 0929-5593.

[24] S. Quinlan and O. Khatib, “Elastic bands: Connecting path planning and control,” in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 1993, 802–807 vol.2.

[25] S. LaValle and S. Hutchinson, “Optimal motion planning for multiple robots having independent goals,” *IEEE Transactions on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, 1998.

[26] D.-H. Lee, D.-H. Kim, J. Y. Lee, and C.-S. Han, “Collision-free coordination of two dual-arm robots with assembly precedence constraint,” in *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, 2014, pp. 515–520.

[27] P. A. O’Donnell and T. Lozano-Perez, “Deadlock-free and collision-free coordination of two robot manipulators,” in *Proceedings, 1989 International Conference on Robotics and Automation*, vol. 1, 1989, pp. 484–489.

[28] S. Akella and S. Hutchinson, “Coordinating the motions of multiple robots with specified trajectories,” in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, vol. 1, 2002, 624–631 vol.1.

[29] M. Otte and E. Frazzoli, “RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning,” *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 797–822, 2016.

[30] Y.-C. Tsai and H.-P. Huang, “Motion planning of a dual-arm mobile robot in the configuration-time space,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 2458–2463.

[31] W. Sun, S. Patil, and R. Alterovitz, “High-frequency replanning under uncertainty using parallel sampling-based motion planning,” *IEEE Transactions on Robotics*, vol. 31, no. 1, pp. 104–116, 2015.

[32] P. Leven and S. A. Hutchinson, “A framework for real-time path planning in changing environments,” *The International Journal of Robotics Research*, vol. 21, pp. 1030–999, 2002.

[33] A. Völz and K. Graichen, “Composition of dynamic roadmaps for dual-arm motion planning,” in *Proceedings IEEE International Conference on Advanced Intelligent Mechatronics*, 2017, pp. 1242–1248.

[34] A. Völz and K. Graichen, “A predictive path-following controller for continuous replanning with dynamic roadmaps,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3963–3970, Oct. 2019, ISSN: 2377-3766.

[35] T. Faulwasser, T. Weber, J. Zometa, and R. Findeisen, “Implementation of nonlinear model predictive path-following control for an industrial robot,” *IEEE Transactions on Control Systems Technology*, 2016.

[36] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, “Path-following NMPC for serial-link robot manipulators using a path-parametric system reformulation,” in *2016 European Control Conference (ECC)*, 2016, pp. 477–482.

[37] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 500–505.

[38] S. S. Ge and Y. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, pp. 207–222, 2002.

[39] C. Viturino, U. Junior, A. Scolari Conceicao, and L. Schnitman, "Adaptive artificial potential fields with orientation control applied to robotic manipulators," *IFAC-PapersOnLine*, vol. 53, pp. 9924–9929, Jan. 2020.

[40] P. Tournassoud, "A strategy for obstacle avoidance and its application to multi-robot systems," *in Proc. of the IEEE Int. Conference on Robotics and Automation, San Francisco*, pp. 1224–1229, Apr. 1986.

[41] P. Tournassoud, "On motion coordination," INRIA, Tech. Rep. RR-0549, Jul. 1986. [Online]. Available: `https://hal.inria.fr/inria-00076005`.

[42] B. Faverjon and P. Tournassoud., "A local based approach for path planning of manipulators with a high number of degrees of freedom. in proceedings.," *1987 IEEE International Conference on Robotics and Automation*, pp. 1152–1159, 1987.

[43] S. Lahouar, S. Zeghloul, and L. Romdhane, "Real-time path planning for multi-dof manipulators in dynamic environment," *International Journal of Advanced Robotic Systems*, vol. 3, Jun. 2006.

[44] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond, "A local collision avoidance method for non-strictly convex polyhedra," *in Proceedings of Robotics: Science and Systems IV*, Jun. 2008.

[45] P. Bosscher and D. Hedman, "Real-time collision avoidance algorithm for robotic manipulators," *in Proceedings of the 2009 IEEE International Conference on Technologies for Practical Robot Applications, Woburn, MA, USA*, vol. 38, pp. 113–122, Nov. 2009.

[46] S. M. LaValle, *Planning Algorithms.* Cambridge University Press, 2006.

[47] R. Ni, Z. Pan, and X. Gao, "Robust multi-robot trajectory optimization using alternating direction method of multiplier," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5950–5957, 2022.

[48] F. Kennel-Maushart, R. Poranne, and S. Coros, "Manipulability optimization for multi-arm teleoperation," in *IEEE International Conference on Robotics and Automation (ICRA) 2021*, Xi'an, China, 2021, pp. 3956–3962.

[49] M. Zucker *et al.*, "CHOMP: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013. [Online]. Available: `https://doi.org/10.1177/0278364913488805`.

[50] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, pp. 1251–1270, Aug. 2014.

[51] M. M. G. Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, "Real-time trajectory generation using model predictive control," in *2015 IEEE Int. Conference on Automation Science and Engineering (CASE)*, Q.-S. Jia, Ed., Piscataway, NJ: IEEE, 2015, pp. 942–948, ISBN: 978-1-4673-8183-3.

[52] L. van den Broeck, M. Diehl, and J. Swevers, "Time optimal MPC for mechatronic applications," in *2009 48th IEEE conference on Decision and Control*, I. Staff, Ed., Shanghai, China: IEEE, 2009, pp. 8040–8045, ISBN: 978-1-4244-3871-6.

[53] C. Rosmann, A. Makarow, F. Hoffmann, and T. Bertram, “Time-optimal nonlinear model predictive control with minimal control interventions,” in *First Annual IEEE Conference 2017*, 2017, pp. 19–24.

[54] G. P. Incremona, A. Ferrara, and L. Magni, “MPC for robot manipulators with integral sliding modes generation,” *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1299–1307, 2017.

[55] M. Rubagotti, T. Taunyazov, B. Omarali, and A. Shintemirov, “Semi-autonomous robot teleoperation with obstacle avoidance via model predictive control,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2746–2753, 2019.

[56] M. Krämer, C. Rösmann, F. Hoffmann, and T. Bertram, “Model predictive control of a collaborative manipulator considering dynamic obstacles,” *Optimal Control Applications and Methods*, vol. 41, no. 4, pp. 1211–1232, 2020. [Online]. Available: `https://onlinelibrary.wiley.com/doi/abs/10.1002/oca.2599`.

[57] P. S. Cisneros, A. Sridharan, and H. Werner, “Constrained predictive control of a robotic manipulator using quasi-LPV representations,” *IFAC-PapersOnLine*, vol. 51, no. 26, pp. 118–123, 2018, 2nd IFAC Workshop on Linear Parameter Varying Systems LPVS 2018, ISSN: 2405-8963.

[58] J. Fehr, P. Schmid, G. Schneider, and P. Eberhard, “Modeling, simulation, and vision-/MPC-based control of a powercube serial robot,” *Applied Sciences*, vol. 10, no. 20, 2020.

[59] Y. Wang, M. Leibold, J. Lee, W. Ye, J. Xie, and M. Buss, “Incremental model predictive control exploiting time-delay estimation for a robot manipulator,” *IEEE Transactions on Control Systems Technology*, pp. 1–16, 2022.

[60] S. A. Homsi, “Online generation of time-optimal trajectories for industrial robots in dynamic environments,” Theses, Université Grenoble Alpes, Mar. 2016.

[61] C. Zhou, M. Lei, L. Zhao, Z. Wang, and Y. Zheng, “TOPP-MPC-based Dual-Arm Dynamic Collaborative Manipulation for Multi-Object Nonprehensile Transportation,” in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 999–1005.

[62] N. Dehio, Y. Wang, and A. Kheddar, “Dual-Arm Box Grabbing With Impact-Aware MPC Utilizing Soft Deformable End-Effector Pads,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5647–5654, 2022.

[63] K. Kanjanawanishkul and A. Zell, “Path following for an omnidirectional mobile robot based on model predictive control,” in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3341–3346.

[64] A. Alessandretti, A. P. Aguiar, and C. N. Jones, “Trajectory-tracking and path-following controllers for constrained underactuated vehicles using model predictive control,” in *2013 European Control Conference (ECC)*, 2013, pp. 1371–1376.

[65] T. Weiskircher, Q. Wang, and B. Ayalew, “Predictive guidance and control framework for (semi-)autonomous vehicles in public traffic,” *IEEE Transactions on Control Systems Technology*, vol. 25, no. 6, pp. 2034–2046, 2017.

[66] P. Stano, U. Montanaro, and et al., “Model predictive path tracking control for automated road vehicles: A review,” *Annual Reviews in Control*, Dec. 2022.

[67] T. Faulwasser, B. Kern, and R. Findeisen, “Model predictive path-following for constrained nonlinear systems,” in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, 2009, pp. 8642–8647.

[68] S. Yu, X. Li, H. Chen, and F. Allgöwer, “Nonlinear model predictive control for path following problems,” *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 145–150, 2012, 4th IFAC Conference on Nonlinear Model Predictive Control, ISSN: 1474-6670.

[69] S. Yu, Y. Guo, L. Meng, T. Qu, and H. Chen, “MPC for Path Following Problems of Wheeled Mobile Robots,” *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 247–252, 2018, 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[70] S. Hiremath., P. Gummadi., A. Tika., P. Rama., and N. Bajcinca., “Learning based interpretable end-to-end control using camera images,” in *Proceedings of the 20th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO*, INSTICC, SciTePress, 2023, pp. 474–484, ISBN: 978-989-758-670-5.

[71] G. V. Raffo, M. G. Ortega, and F. R. Rubio, “Backstepping/nonlinear $\mathcal{H}_\infty$ control for path tracking of a quadrotor unmanned aerial vehicle,” *2008 American Control Conference*, pp. 3356–3361, 2008. [Online]. Available: `https://api.semanticscholar.org/CorpusID:28071838`.

[72] A. Banaszuk and J. Hauser, “Feedback linearization of transverse dynamics for periodic orbits,” *Systems & Control Letters*, vol. 26, no. 2, pp. 95–105, 1995, ISSN: 0167-6911.

[73] D. Lam, C. Manzie, and M. Good, “Model predictive contouring control,” in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 6137–6142.

[74] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, “Model predictive contouring control for collision avoidance in unstructured dynamic environments,” *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, Jul. 2019.

[75] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, “Model predictive contouring control for near-time-optimal quadrotor flight,” *CoRR*, vol. abs/2108.13205, 2021. [Online]. Available: `https://arxiv.org/abs/2108.13205`.

[76] A. Tika, J. Ulmen, and N. Bajcinca, “Dynamic parameter estimation utilizing optimized trajectories,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 7300–7307.

[77] A. Tika, F. Gashi, and N. Bajcinca, “Robot online task and trajectory planning using mixed-integer model predictive control,” *European Control Conference, London, UK*, pp. 2005–2011, July 2022.

[78] A. Tika and N. Bajcinca, “Optimization-based task and trajectory planning for robot manipulators,” in *2023 31st Mediterranean Conference on Control and Automation (MED)*, 2023, pp. 562–568.

[79] A. Tika, N. Gafur, V. Yfantis, and N. Bajcinca, "Optimal scheduling and model predictive control for trajectory planning of cooperative robot manipulators," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9080–9086, 2020, 21st IFAC World Congress, ISSN: 2405-8963. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2405896320327889`.

[80] A. Tika and N. Bajcinca, "Synchronous minimum-time cooperative manipulation using distributed model predictive control," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7675–7681.

[81] A. Tika and N. Bajcinca, "Model predictive control based cooperative robot manipulation and collision avoidance in shared workspaces," in *2021 European Control Conference (ECC)*, 2021, pp. 702–709.

[82] A. Tika and N. Bajcinca, "Predictive control of cooperative robots sharing common workspace," *IEEE Transactions on Control Systems Technology*, pp. 1–16, 2023.

[83] A. Tika, S. Hiremath, and N. Bajcinca, "Predictive path following control for mobile robots with automatic parameter tuning," in *2023 European Control Conference (ECC)*, 2023, pp. 1–8.

[84] J. J. Craig, *Introduction to robotics : mechanics & control.* Addison-Wesley Pub. Co., 1986, ISBN: 0201103265.

[85] B. Siciliano and O. Khatib, *Springer handbook of robotics.* Springer, 2016.

[86] W. Khalil and E. Dombre, *Modeling, identification and control of robots.* Butterworth-Heinemann, 2004.

[87] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control.* Hoboken, N.J.: Wiley, 2006, ISBN: 9780471649908.

[88] W. Khalil and M. Gautier, "On the derivation of the dynamic models of robots," in *85 ICAR, Tokyo, Japan*, 1985, pp. 243–250.

[89] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, 1st. Springer Publishing Company, Incorporated, 2008, ISBN: 1846286417.

[90] R. S. Hartenberg and J. Denavit, "A kinematic notation for lower pair mechanisms based on matrices," 1955.

[91] A. C. Reddy, "Difference between denavit-hartenberg (dh) classical and modified conventions for forward kinematics of robots with case study," in *International Conference on Advanced Materials and manufacturing Technologies (AMMT)*, JNTUH College of Engineering Hyderabad Chandigarh, India, 2014.

[92] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised.* Springer, 2017, vol. 118.

[93] W. Khalil and J. Kleinfinger, "A new geometric notation for open and closed-loop robots," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, Apr. 1986, pp. 1174–1179.

[94] K. R. Kozlowski, *Modelling and identification in robotics.* Springer Science & Business Media, 2012.

[95] K. P. Hawkins, "Analytic inverse kinematics for the universal robots UR5/UR10 arms," *Technical report, Georgia Institute of Technology*, 2013.

[96] R. Keating, “UR5 Inverse Kinematics,” *Technical report, Johns Hopkins University*, 2017.

[97] R. S. Andersen, “Kinematics of a UR5,” *Technical report, Aalborg University*, May 2018.

[98] J. Villalobos, I. Y. Sanchez, and F. Martell, “Alternative Inverse Kinematic Solution of the UR5 Robotic Arm,” in *Advances in Automation and Robotics Research*, Cham: Springer International Publishing, 2022, pp. 200–207, ISBN: 978-3-030-90033-5.

[99] M. Calkin, *Lagrangian and Hamiltonian Mechanics*. Allied Publishers, 1996, ISBN: 9788177640649. [Online]. Available: `https://books.google.de/books?id=5qbGFSQcBJcC`.

[100] H. Olsson, K. Åström, C. Canudas de Wit, M. Gäfvert, and P. Lischinsky, “Friction models and friction compensation,” *European Journal of Control*, vol. 4, no. 3, pp. 176–195, 1998, ISSN: 0947-3580. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S094735809870113X`.

[101] C. LANCZOS, *The Variational Principles of Mechanics*. University of Toronto Press, 1962, ISBN: 9781487581770. [Online]. Available: `http://www.jstor.org/stable/10.3138/j.ctvfrxpd0` (visited on 09/25/2023).

[102] F. Pfeiffer and T. Schindler, *Einführung in die Dynamik*, 3rd ed. Springer Vieweg Berlin, Heidelberg, Germany, 2014, ISBN: 978-3-642-41045-1.

[103] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*, 1st. USA: Cambridge University Press, 2017, ISBN: 1107156300.

[104] M. Neubauer, H. Gattringer, and H. Bremer, “A persistent method for parameter identification of a seven-axes manipulator,” *Robotica*, vol. 33, no. 5, pp. 1099–1112, 2015.

[105] K. Radkhah, D. Kulic, and E. Croft, “Dynamic parameter identification for the CRS A460 robot,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2007, pp. 3842–3847.

[106] W. Rackl, R. Lampariello, and G. Hirzinger, “Robot excitation trajectories for dynamic parameter estimation using optimized b-splines,” in *2012 IEEE international conference on robotics and automation*, IEEE, 2012, pp. 2042–2047.

[107] W. Wu, S. Zhu, X. Wang, and H. Liu, “Closed-loop dynamic parameter identification of robot manipulators using modified fourier series,” *International Journal of Advanced Robotic Systems*, vol. 9, no. 1, p. 29, 2012.

[108] G. Calafiore, M. Indri, and B. Bona, “Robot dynamic calibration: Optimal excitation trajectories and experimental parameter estimation,” *Journal of robotic systems*, vol. 18, no. 2, pp. 55–68, 2001.

[109] A. Zakharov and S. Halász, “Parameter identification of a robot arm using genetic algorithms,” *Periodica Polytechnica Electrical Engineering*, vol. 45, no. 3-4, pp. 195–209, 2003.

[110] J. Ulmen, “Dynamic Parameter Estimation of the UR5 Robot,” M.S. thesis, Technische Universität Kaiserslautern, 2020.

[111] M. Gendreau, J.-Y. Potvin, *et al.*, *Handbook of metaheuristics.* Springer, 2010, vol. 2.

[112] W. Khalil and J.-F. Kleinfinger, "Minimum operations and minimum parameters of the dynamic models of tree structure robots," *IEEE Journal on Robotics and Automation*, vol. 3, no. 6, pp. 517–526, 1987.

[113] P. K. Khosla and T. Kanade, "Parameter identification of robot dynamics," in *1985 24th IEEE Conference on Decision and Control*, IEEE, 1985, pp. 1754–1760.

[114] M. Gautier, "Numerical calculation of the base inertial parameters of robots," *Journal of robotic systems*, vol. 8, no. 4, pp. 485–506, 1991.

[115] H. Kawasaki, T. Shimizu, and K. Kanzaki, "Symbolic analysis of the base parameters for closed-chain robots based on the completion procedure," in *Proceedings of IEEE International Conference on Robotics and Automation*, IEEE, vol. 2, 1996, pp. 1781–1786.

[116] C. Kalker-Kalkman, "An implementation of buchbergers' algorithm with applications to robotics," *Mechanism and machine theory*, vol. 28, no. 4, pp. 523–537, 1993.

[117] M. Gautier and W. Khalil, "Direct calculation of minimum set of inertial parameters of serial robots," *IEEE Transactions on robotics and Automation*, vol. 6, no. 3, pp. 368–373, 1990.

[118] J. Wu, J. Wang, and Z. You, "An overview of dynamic parameter identification of robots," *Robotics and computer-integrated manufacturing*, vol. 26, no. 5, pp. 414–419, 2010.

[119] J. Jin and N. Gans, "Parameter identification for industrial robots with a fast and robust trajectory design approach," *Robotics and Computer-Integrated Manufacturing*, vol. 31, pp. 21–29, 2015.

[120] J. Swevers, C. Ganseman, D. B. Tukel, J. De Schutter, and H. Van Brussel, "Optimal robot excitation and identification," *IEEE transactions on robotics and automation*, vol. 13, no. 5, pp. 730–740, 1997.

[121] J. Swevers, W. Verdonck, and J. De Schutter, "Dynamic model identification for industrial robots," *IEEE control systems magazine*, vol. 27, no. 5, pp. 58–71, 2007.

[122] J. Swevers, C. Ganseman, J. De Schutter, and H. Van Brussel, "Experimental robot identification using optimised periodic trajectories," *Mechanical Systems and Signal Processing*, vol. 10, no. 5, pp. 561–577, 1996.

[123] A. Bahloul, S. Tliba, and Y. Chitour, "Dynamic parameters identification of an industrial robot with and without payload," *IFAC-PapersOnLine*, vol. 51, no. 15, pp. 443–448, 2018.

[124] N. Kovincic, A. Mueller, H. Gattringer, M. Weyrer, A. Schlotzhauer, and M. Brandstötter, "Dynamic parameter identification of the Universal Robots UR5," May 2019.

[125] M. Quigley *et al.*, "ROS: An open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, vol. 3, Jan. 2009.

[126] G. P. McCormick, "Computability of global solutions to factorable nonconvex programs: Part I- convex underestimating problems," *Mathematical Programming*, vol. 10, pp. 147–175, 1976.

[127] Stanford Artificial Intelligence Laboratory et al., *Robotic Operating System*, version ROS Melodic Morenia, May 23, 2018. [Online]. Available: `https://www.ros.org`.

[128] U. Robots, *RTDE: Real-Time Data Exchange*, 2019. [Online]. Available: `https://www.universal-robots.com/articles/ur/interface-communication/real-time-data-exchange-rtde-guide/` (visited on 06/14/2022).

[129] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[130] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, pp. 25–57, Mar. 2006.

[131] H. Ha, J. Xu, and S. Song, "Learning a decentralized multi-arm motion planner," in *Conference on Robotic Learning (CoRL)*, 2020.

[132] J. Chen *et al.*, *Cooperative task and motion planning for multi-arm assembly systems*, 2022. arXiv: `2203.02475 [cs.RO]`.

[133] M. Wagner, P. Heß, S. Reitelshöfer, and J. Franke, "Cooperative processing with multi-robot systems," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2017, pp. 663–669.

[134] C. Wong, S. Shackleford, D. Potter, J.-P. Richardson, L. McDermott, and J. Nolan, "Robotic task sequencing and motion coordination for multiarm systems," *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 6, pp. 5275–5286, 2022.

[135] S. Zhang and F. Pecora, "Online sequential task assignment with execution uncertainties for multiple robot manipulators," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6993–7000, 2021.

[136] P. Bézier, *The Mathematical Basis of the UNISURF CAD System.* London: Butterworth, 1986.

[137] D. Johnson and E. Cohen, "A framework for efficient minimum distance computations," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 4, 1998, 3678–3684 vol.4.

[138] J.-W. Chang, Y.-K. Choi, M.-S. Kim, and W. Wang, "Computation of the minimum distance between two Bézier curves/surfaces," *Computers & Graphics*, vol. 35, no. 3, pp. 677–684, 2011, ISSN: 0097-8493.

[139] N. M. Patrikalakis and T. Maekawa, *Shape Interrogation for Computer Aided Design and Manufacturing*, 1st. Springer Publishing Company, Incorporated, 2009, ISBN: 364204073X.

[140] E. C. Sherbrooke and N. M. Patrikalakis, "Computation of the solutions of nonlinear polynomial systems," *Comput. Aided Geom. Des.*, vol. 10, pp. 379–405, 1993.

[141] X.-D. Chen, L. Chen, Y. Wang, G. Xu, J.-H. Yong, and J.-C. Paul, "Computing the minimum distance between two Bézier curves," *Journal of Computational and Applied Mathematics*, vol. 229, no. 1, pp. 294–301, 2009, ISSN: 0377-0427.

[142] E. Gilbert, D. Johnson, and S. Keerthi, “A fast procedure for computing the distance between complex objects in three-dimensional space,” *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[143] R. Shome, K. Solovey, J. Yu, K. Bekris, and D. Halperin, “Fast, high-quality two-arm rearrangement in synchronous, monotone tabletop setups,” *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 888–901, 2021.

[144] I. S. Duff, “Ma57—a code for the solution of sparse symmetric definite and indefinite systems,” *ACM Trans. Math. Softw.*, vol. 30, pp. 118–144, 2004. [Online]. Available: `https://api.semanticscholar.org/CorpusID:7888447`.

[145] F. Gashi, “Real-Time Implementation of an Optimization based Hierarchical Control Algorithm for Robot Task Allocation and Online Trajectory Planning,” M.S. thesis, Technische Universität Kaiserslautern, 2021.

[146] V. A. Toponogov, *Differential Geometry of Curves and Surfaces: A Concise Guide*. Boston: Birkhäuser, 2006.

[147] D. Luenberger and Y. Ye, *Linear and Nonlinear Programming* (International Series in Operations Research & Management Science). Springer International Publishing, 2015, ISBN: 9783319188423.

[148] C. Huang, “Model Predictive Control for Trajectory Planning of Omnidirectional Autonomous Mobile Robot,” M.S. thesis, Technische Universität Kaiserslautern, 2022.

[149] H. J. Bückenhüskes and G. Oppenhäuser, “DLG trend report: Robots in the food and beverage industry,” *DLG Lebensmittel 9(6)*, 16-17 (2014).

[150] S. Han, S. Feng, and J. Yu, “Toward fast and optimal robotic pick-and-place on a moving conveyor,” *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, Dec. 2019.

[151] L. Giefer, J. D. Arango C., M. M. Babr, and M. Freitag, “Deep learning-based pose estimation of apples for inspection in logistic centers using single-perspective imaging,” *Processes*, vol. 7, no. 7, p. 424, 2019.

[152] I. J. Schoenberg, “Contributions to the problem of approximation of equidistant data by analytic functions,” in *I. J. Schoenberg Selected Papers*, C. de Boor, Ed. Boston, MA: Birkhäuser Boston, 1988, pp. 3–57, ISBN: 978-1-4899-0433-1. [Online]. Available: `https://doi.org/10.1007/978-1-4899-0433-1_1`.

Supervised Theses With Contribution to this Work

- J. Ulmen, “Dynamic Parameter Estimation of the UR5 Robot,” M.S. thesis, Technische Universität Kaiserslautern, 2020.
- F. Gashi, “Real-Time Implementation of an Optimization based Hierarchical Control Algorithm for Robot Task Allocation and Online Trajectory Planning,” M.S. thesis, Technische Universität Kaiserslautern, 2021.
- C. Huang, “Model Predictive Control for Trajectory Planning of Omnidirectional Autonomous Mobile Robot,” M.S. thesis, Technische Universität Kaiserslautern, 2022.

Lebenslauf

Dipl.-Ing. Argtim Tika

Ausbildung

2017 – 2023 **Ph.D., Rheinland-Pfälzische Technische Universität**
Fachbereich Maschinenbau und Verfahrenstechnik
Disertation: *Optimization based Algorithms for Task Planning and Predictive Motion Control in Robotics*

2013 – 2016 **Dipl.-Ing. Automatisierungstechnik, Technische Universität Wien**
Masterstudium, Fakultät für Elektrotechnik und Informationstechnik
Diplomarbeit: *Mathematische Modellierung von Permanentmagnet-Synchronmotoren mit Hilfe von Reluktanznetzwerken unter Berücksichtigung von Wicklungskurzschlüssen*

2007 – 2013 **BSc. Elektro- und Informationstechnik, Technische Universität Wien**
Bachelorstudium, Fakultät für Elektrotechnik und Informationstechnik
Bachelorarbeit am Institut für Automatisierungs- und Regelungstechnik
Bachelorarbeit am Institute of Telecommunications

2006 – 2007 **Österreichische Orient-Gesellschaft Hammer-Purgstall**
Deutschkurs zur Vorbereitung auf die Ergänzungsprüfung Deutsch

2002 – 2006 **Gymnasium, Mirko Mileski Kërçovë, Nordmazedonien**
Mathematisch-Naturwissenschaftliches Gymnasium

Berufserfahrung

Wissenschaftlicher Mitarbeiter

2016 – **Rheinland-Pfälzische Technische Universität (RPTU)**
Fachbereich Maschinenbau und Verfahrenstechnik
Lehrstuhl für Mechatronik in Maschinenbau und Fahrzeugtechnik

Lehre

2016 – 2023 **Betreuung von Veranstaltungen an der RPTU**

- Übung Maschinendynamik
- Übung Fahrdynamikregelung
- Übung und Labor Mechatronik