Optimal task planning and predictive online motion control for a multi-manipulator system

Vom Fachbereich Maschinenbau und Verfahrenstechnik

der Rheinland-Pfälzischen Technischen Universität Kaiserslautern-Landau

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte

Dissertation

von

Frau

M.Sc. Nigora Gafur

aus Duschanbe, Tadschikistan

Tag der mündlichen Prüfung: 11.12.2024

Dekan:	Prof. Dr. rer. nat. Roland Ulber

Promotionskommision:

Vorsitender:	Prof. Dr.-Ing. Kristin de Payrebrune
1. Berichterstatter:	Prof. Dr.-Ing. Martin Ruskowski
2. Berichterstatter:	Prof. Dr.-Ing. Daniel Görges

D 386

Acknowledgments

For me, a doctorate is less of a destination and more of a journey—shaped by learning experiences, highs and lows, and invaluable scientific and personal encounters. As I reflect on this time, I do so with great joy and gratitude, though it is somewhat bittersweet to see this chapter come to an end.

My deepest gratitude goes to Prof. Dr.-Ing. Martin Ruskowski, who gave me the freedom to choose my own research topic and pursue a PhD in robotics. The trust and autonomy he granted me were pivotal to the success of this work and consistently motivated me to implement my own ideas. I especially appreciate the opportunity to test my algorithms on real robots and work with cutting-edge hardware. I would also like to sincerely thank Prof. Dr.-Ing. Daniel Görges for serving as the second examiner of my dissertation, as well as for his exceptional lectures in optimal control, which have significantly contributed to my academic development during my PhD. Additionally, I am grateful to Prof. Dr.-Ing. Kristin de Payrebrune for chairing the examination committee. My gratitude extends to PD Dr. Achim Wagner for the valuable discussions and insightful feedback over the years, which have greatly enriched this research. His willingness to engage in discussions and his enthusiasm for the research topic have always been a source of motivation.

I would like to express my heartfelt thanks to my colleagues at WSKL, who made my PhD years truly unforgettable. Despite the challenges posed by the COVID-19 pandemic and my later move to Karlsruhe, we managed to create lasting memories together. For me, WSKL represents team spirit, cheerful gatherings over cake, Flammkuchen, and barbecue evenings—memories I will always cherish. A special thank you to Tatjana, Magnus, Simon, and Vassilios for the excellent teamwork on projects and in teaching. I would like to express a special thank you to Vassilios, with whom I've had the privilege of sharing this PhD journey. We've always supported each other, both professionally and personally, and I'm truly grateful for his friendship. Our collaboration on projects, co-supervision of courses, and joint publications have been some of the most rewarding aspects of this journey. My sincere thanks also go to my colleagues at the Innovative Factory Systems Group at DFKI and SmartFactoryKL for the valuable collaboration and deep insights into Industry 4.0. I am also grateful to my students, Leo and Gaja, for their trust in me as a mentor throughout their bachelor's, research, master's, and diploma theses. The countless hours we spent at the UR remain among my fondest memories of my PhD years.

From the bottom of my heart, I thank my family – especially my parents and my siblings Ilhom and Nargis – who have always encouraged and supported me throughout this journey. My deepest gratitude goes to my mother, who was my constant source of support, love, and strength. Her unwavering belief in me and her boundless encouragement have shaped me in ways words can scarcely express. My deepest appreciation also goes to my closest friends – Nadine, Corinna, Patrick, Andreas, Anna, and Julie – for their unwavering support, interest, and trust. I am equally grateful to my Kaiserslautern friends, Larissa and Christopher – our PhD and flat-sharing trio made this time so much more enjoyable with their humor and zest for life.

My greatest thanks go to my husband, Sebastian. Thank you for supporting me all these years, even while pursuing your own PhD, and for always standing by my side with advice and encouragement. You have been, and continue to be, my rock, and for that, I thank you from the bottom of my heart.

Finally, my deepest gratitude goes to my two joyful and lively little wonders, Amilia and Charlotte. During the final phase of my dissertation, you patiently waited until I had completed my practical work before making your grand entrance into the world. Even after that, your cheerfulness and patience made it easier for me to complete my thesis, and you have enriched my life beyond measure.

Contents

List of Abbreviations

APF	Artificial Potential Field
BV	Bounding Volumes
CBF	Control Barrier Function
CHOMP	Covariant Hamiltonian Optimization for Motion Planning
CMPC	Centralized Model Predictive Control
CSP	Constraint Satisfaction Problem
DMPC	Distributed Model Predictive Control
dRRT	Discrete Rapidly-Exploring Random Tree
ELS	Ellipsoid-Line Segment
GA	Genetic Algorithm (GA)
GJK	Gilbert–Johnson–Keerthi
HRI	Human-Robot Interaction
LSS	Line Swept Sphere
MAS	Multi-Agent Systems
MIBLP	Mixed-Integer Bilinear Programming
MIQCP	Mixed-Integer Quadratically Constrained Programming
MPC	Model Predictive Control
MRS	Multi-Robot Systems
MRTA	Multi-Robot Task Allocation
mTSP	Multiple Traveling Salesman Problem
NLP	Nonlinear Programming
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
OMPL	Open Motion Planning Library
PRM	Probabilistic RoadMaps
PRM*	Optimal Probabilistic RoadMaps
QP	Quadratic Programming
ROS	Robot Operating System
RRT	Rapidly-Exploring Random Tree
TAMP	Task and Motion Planning
TrajOpt	Trajectory Optimization for Motion Planning
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

Kurzfassung

Diese Arbeit befasst sich mit einem Ansatz zur Aufgaben- und Bewegungsplanung (TAMP), welcher eine autonome Kooperation zwischen mehreren Manipulatoren in Echtzeit ermöglicht. Eine simultane und echtzeitfähige Ausführung von Aufgaben durch eine Gruppe von Manipulatoren stellt immer noch eine große Herausforderung dar, da eine komplexe Koordination zwischen den Robotern erforderlich ist, um Kollisionen und Deadlocks zu verhindern. Der vorgeschlagene optimierungsbasierte TAMP-Ansatz ist hierarchisch aufgebaut und besteht aus mehreren Ebenen, die jeweils einen individuellen Aspekt des TAMP-Ansatzes realisieren. Um die Aufgaben optimal zwischen den Robotern zu verteilen, wird ein gemischt-ganzzahliges lineares Optimierungsproblem formuliert. Um ein reaktives Verhalten eines jeden Roboters auf instantane Änderungen in dessen Umgebung und während dessen Bewegung zu ermöglichen, wird die DMPC-ELS-Methode als Trajektorienplaner vorgeschlagen. Die vorgeschlagene DMPC-ELS-Methode basiert auf einer verteilten modellprädiktiven Regelung, bei der das Trajektorienplanungsproblem jedes Roboters parallel gelöst wird, wobei die prädizierten Trajektorien seiner Nachbarn berücksichtigt werden. Dadurch wird sichergestellt, dass keine Kollisionen zwischen den Robotern auftreten. Im Gegensatz zu den bestehenden Methoden beruht die neu entwickelte Methode zur Kollisionsvermeidung nicht auf der Berechnung von Entfernungen zwischen kollisionsgefährdeten Körpern, sondern stellt sicher, dass deren Schnittmenge leer ist, wodurch das Problem verschachtelter logischer Bedingungen nicht besteht. Im Weiteren wird eine neu entwickelte Methode zur reaktiven Auflösung von Deadlocks vorgeschlagen, um Zielkonflikte zwischen Manipulatoren, die von Deadlocks betroffen sind, reaktiv aufzulösen, ohne den Rest des Roboterteams, der nicht in den Deadlock involviert ist, zu unterbrechen. Umfangreiche simulationsbasierte und experimentelle Studien einschließlich mehrerer Benchmark-Analysen werden durchgeführt, um die Vorteile des vorgeschlagenen TAMP-Ansatzes aufzuzeigen. Eine zusätzliche Skalierbarkeitsanalyse zeigt die Grenzen des DMPC-ELS-Ansatzes auf.

Abstract

This thesis proposes an online task and motion planning (TAMP) approach, enabling an autonomous cooperation between multiple manipulators. Online execution of tasks by a group of manipulators poses challenges as an intricate inter-robot coordination is critical to keep the robotic system collision- and deadlock-free. The proposed optimization-based approach is hierarchically structured and consists of several layers, each considering an individual aspect of the TAMP. To optimally distribute tasks between the robots, a task allocation problem is formulated as a mixed-integer programming problem. To allow a reactive behavior of each robot upon environmental changes during motion, an online motion planning approach, named DMPC-ELS, is proposed. The DMPC-ELS method is based on distributed model predictive control, where the trajectory planning problem of each robot is solved in parallel, taking the predicted trajectories of its neighbors into account. This ensures that no inter-robot collisions occur. Unlike existing methods, the newly developed collision avoidance approach ELS does not rely on computing distances between collision-prone bodies, but rather checks the intersection of sets, overcoming the problem with nested logical conditions. Further, a newly developed online resolution procedure for treating deadlocks is proposed to resolve conflicting goals between deadlock-affected manipulators without interrupting the rest of the robotic team not involved in the deadlock. Extensive simulation-based and experimental studies including several benchmark analyses are carried out to show the advantages of the proposed TAMP approach. An additional scalability analysis demonstrates the limitations of the DMPC-ELS approach.

1 Introduction

The need for autonomous robots is rapidly growing due to flexibility requirements of customized production, where robots work alongside human workers [AC19; Rus+20]. Serial robots, also called manipulator, received an increasing interest in a wide variety of applications ranging from industry [MBF18], medicine [HLL19], aerospace [Ryb18] to construction [Pet+19]. A series of links connected by joints form the kinematic chain of a manipulator, where an end-effector is attached to its end. Robotic arm systems exhibit significant flexibility, attributed to the complexity of their kinematic chains, characterized by a high degree of precision and repeatability. Furthermore, they demonstrate a robust capability to bear heavy payloads, thereby underscoring their efficacy in diverse industrial applications. However, manipulators employed in industry still operate behind safety barriers to prevent any injuries to human workers. Besides, their application is generally limited to perform repetitive tasks, where the robots follow predefined paths in static and known environments [Vil+18]. Consequently, the interaction of manipulators with humans or other robotic systems in dynamic and cluttered environments is still very limited in real-world applications.

The definition of a *robot*, as it is defined by the International Organization for Standardization [Int21], states:

> *"Actuated mechanism programmable in two or more axes with a degree of autonomy, moving within its environment, to perform intended tasks."*

Besides, there are three fundamental laws of robotics, also known as Asimov's Laws, introduced by Isaac Asimov in 1942 [Asi42]:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

Asimov's Laws laid the foundation to define specifications for the design of a robot and gave an impulse for ethics in artificial intelligence [SS00].

Based on the aforementioned specifications and definition it can be inferred that safety and autonomy are the most important requirements for designing a robotic system. In real-world applications, the interaction of robots with humans or other robotic systems occur in dynamic, cluttered and uncertain environments. A robot should be able to anticipate unpredictable changes in its environment and initiate an appropriate response. In other words, a robot should guarantee safety of the system, i.e., causing no harm to human workers and its environment.

In the last decade, collaborative robots, also called cobots [CWP96], gained significant popularity both in industry and in the research world. Cobots are easily programmable and thus adaptable to their environment. Due to cobots' lightweight design, they are less likely to cause harm to humans as lower contact forces and torques are exerted compared to industrial robots. Primarily, cobots were developed to work alongside human workers in assembly lines to improve safety, quality and productivity of the overall system [Ake+99]. The idea of developing cobots emerged to serve as assistants to human workers carrying out more complex tasks. As a consequence, the field of human-robot interaction (HRI) started to emerge and is still an active research area. In 2016, Cherubini et al. [Che+16] proposed a solution that allowed a direct physical contact between a human and a robot in real industrial scenarios meeting safety standards.

Besides HRI, multirobot systems (MRS) represent another active research area, where multiple robots in cooperation solve a joint task. Cooperation between multiple robots can considerably increase efficiency and performance of the overall system compared to a single robot system. Multiple robots working together can accomplish several tasks simultaneously or carry out more complex tasks, such as transportation of heavy loads or assembly, which are not possible by utilizing only a single robot [DB17]. Additionally, reachability of the system can be increased by realizing hand-over tasks between the robots. MRS allows for modularization leading to a more robust and reliable system [FIN04]. Tasks that can be accomplished by multiple manipulators range from cooperative manipulation [DH20; YSV21], assembly tasks [Dog+19; Har+23] to spray painting and grit-blasting [HL17]. Notably, different types of robotic systems can be combined to achieve the desired system requirements. For instance, robotic arm systems can be combined with Unmanned Aerial Vehicle (UAV), called aerial robotic manipulator or with Unmanned Ground Vehicle (UGV), denoted as mobile manipulator. Ollero et al. [Oll+22] and Thakar et al. [Tha+22] provide excellent surveys of the two robotic systems.

Collision avoidance, coordination and task distribution play a crucial role in MRS. While well-developed solutions exist for MRS involving autonomous mobile robots in cluttered environments [BGL11; Sch+20], methods dealing with multiple manipulators are rare. This might be attributed to the manipulator's complex kinematic chain, where simplified models cannot be easily utilized as in the case of mobile robots [BGL11; RHB15; Sch+20].

Collision avoidance becomes a challenging task, especially considering cooperation between multiple manipulators. Moreover, works based on real-time optimization for robotic arm systems in dynamically changing environments are still limited [Mav+21]. Consequently, much more research in the field of safety and autonomy of robotic arm systems needs to be 'invested'.

The research on robotics is still lacking to provide a concept for high-level motion control, which is able to generate a path from an arbitrary initial to an arbitrary target state in an environment exposed to sudden changes. This becomes especially important in case the target state or obstacles change their pose, while the robot is already executing its previously planned trajectory. Being able to reach the (new) target state, a replanning of the initially planned trajectory is required, still guaranteeing safety of the robot and its environment at each time instant. There is a great number of state-of-the-art planners for manipulators [ŞMK12], which are limited to static environments. In the past decade, model predictive control (MPC) has been widely used in various number of robotic systems and showed a high potential as an online trajectory planning method [RSS17; Tik+20; Sch+20; Krä+20; Mav+21; TB24].

This thesis focuses on developing an optimization-based task and motion planning (TAMP) approach allowing for a safe cooperation between multiple manipulators accomplishing a common goal. Prior to proposing the newly developed TAMP approach for a multi-manipulator system, the potential and limitations of different types of trajectory planners for a single manipulator system are analyzed. So far, no benchmark analysis exists between the existing state-of-the-art planners and MPC. To conduct a comparison between different types of planners, different types of static environments are considered and several performance metrics are selected to carry out a benchmark analysis based on experiments. To demonstrate the merits of MPC, dynamically changing environment involving sudden changes and moving targets are considered as well [Gaf+22c].

Considering a multi-manipulator system, coordination of a group of robots plays a crucial role. The advantages of using a group of robots come at the cost of increased complexity of robot coordination, where the potential for conflicts is high. To coordinate a group of robots in an optimal manner, an optimal scheduling approach is proposed by minimizing the maximum completion time needed to complete the tasks by all robots taking reachability constraints into account as well as reducing the potential for inter-robot collisions and deadlocks. Types of tasks considered in this work include pick-and-place tasks and disassembly.

Online replanning becomes even more complex in case of multiple manipulators working in a team, where each robot represents additional dynamical obstacles to each other that need to constantly be considered during operation. To split the problem into subproblems, an online motion control scheme based on distributed model predictive control (DMPC)

is proposed within this thesis [GYR21; Gaf+22a]. DMPC allows for each robot to plan its own trajectory, where communication between the systems is allowed. To realize a close operation between the robots, an efficient collision avoidance strategy based on ellipsoid-line segment (ELS) approach is introduced, where no intersections between approximated parts of the robot's body are allowed to occur. The proposed collision avoidance strategy ELS is integrated into each manipulator's trajectory planner DMPC, allowing for taking the predicted trajectory of each neighbor into account to avoid inter-robot collisions.

In general, MRS are prone to deadlocks provoking a standstill of the entire system. To this end, a special focus of this work also lies in proposing a reactive deadlock resolution method for a multi-manipulator system. This becomes especially important if the trajectories of the robots are not preplanned and each robot is allowed to react upon changes in its environment independently of its neighboring robots. In such a system, deadlocks between the robots can always occur and cannot be anticipated beforehand. Solutions for handling deadlocks in multi-manipulator systems that can reliably detect and resolve the occurring deadlocks are extremely limited. To this end, a reactive deadlock resolution scheme [Gaf+22a] is proposed that does not rely on preplanned trajectories of individual robots. A robotic team is clustered into deadlock-affected and deadlock-free groups of robots once a deadlock is detected. This allows to resolve locally occurred deadlocks without interrupting the entire robotic team.

This thesis is structured as follows. Chapter 2 provides an overview over different path and trajectory planning methods. The MPC method is introduced followed by presenting different types of collision avoidance strategies that might be integrated into the MPC framework. Further, a review of trajectory planning methods developed for manipulators is provided. The chapter is rounded off by elaborating on different aspects of multi-robot systems. An extensive benchmark analysis of the existing state-of-art planners and MPC is carried out for a single manipulator based on experiments in Chapter 3. Optimal task planning and predictive online motion control for a multi-manipulator system is introduced in Chapter 4. The main focus of this chapter is to introduce different layers of the newly developed TAMP approach handling multi-manipulator systems. Chapter 5 provides an extensive simulation-based study involving two benchmark analysis for a multi-robot setup. The first benchmark analysis compares the heuristic and optimization-based scheduling approaches. The second benchmark analysis deals with comparing the newly proposed DMPC-ELS method with several state-of-the-art trajectory planners. Chapter 6 deals with an experimental study of two manipulators working in cooperation and performing a series of pick-and-place tasks and disassembly. The thesis is concluded by an extensive discussion of the presented approaches and results followed by an outlook in Chapter 7.

2 State of the art

This chapter presents general concepts of path planning and trajectory planning in robotics. The main focus is directed towards model predictive control (MPC) as an optimization-based control method, which is the method of choice for realizing a collision-free operation of manipulators in this thesis. Therefore, the method as well as different collision avoidance strategies that might be incorporated into the MPC framework are introduced. Further, a thorough review on the application of MPC to single manipulator systems is given.

The research of this thesis is concentrated on multi-robot systems (MRS). Consequently, this chapter highlights critical aspects relevant for dealing with MRS. First, an extensive review on MRS involving manipulator systems is provided. Second, a particular emphasis is placed on deadlocks and techniques realizing a deadlock-free operation of MRS. Finally, approaches for multi-robot task allocation are introduced. The chapter is concluded by summarizing the main aspects relevant to realize an MRS.

2.1 Path planning

Path planning problem deals with finding a geometric path to traverse the robot from its initial to a final posture while avoiding collisions with its environment. Path planning problems are independent of time and are solved in the configuration space.

Some basic definitions need to be introduced in the following:

- $\mathcal{G}$ – geometry of a robot
- n – dimension of the robot's operation space
- $\mathcal{W} \subset \mathbb{R}^n$ – robot's workspace
- $\mathcal{O} \subset \mathcal{W}$ – obstacle region in $\mathcal{W}$
- $\mathcal{C}$ – configuration space
- $\mathbf{q} \in \mathcal{C}$ – configuration of a robot
- $\mathcal{C}_{\text{obs}} \subset \mathcal{C}$ – obstacle space in $\mathcal{C}$
- $\mathcal{C}_{\text{free}} \subset \mathcal{C}$ – free space in $\mathcal{C}$

- $\mathbf{q}_{\text{init}} \in \mathcal{C}_{\text{free}}$ – initial configuration of a robot
- $\mathbf{q}_{\text{goal}} \in \mathcal{C}_{\text{free}}$ – goal configuration of a robot

Consider a robot $\mathcal{G}$, which may consist of a kinematic chain, i.e., a series of rigid bodies connected by joints. The Euclidean space in which the robot operates is called workspace $\mathcal{W} = \mathbb{R}^n$, where $n = 2$ or $n = 3$ specifies its dimension. The configuration of the robot is represented by a vector of generalized coordinates $\mathbf{q} \in \mathcal{C}$ comprising translational and rotational degrees of freedom of the robot. For instance, a configuration of a six-axis manipulator is described by six generalized coordinates, i.e, $\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6]$. The configuration space $\mathcal{C}$, also shortened as *C-space*, describes the space with all possible configurations of a robot. The multi-dimensional space $\mathcal{C}$ results from Cartesian product of the spaces of the generalized coordinates.

Space occupied by the robot's body is described by the set $\mathcal{G}(\mathbf{q})$, which depends on a robot's configuration $\mathbf{q}$. The C-space $\mathcal{C}$ can be subdivided into obstacle space $\mathcal{C}_{\text{obs}}$ and free space $\mathcal{C}_{\text{free}}$, i.e.,

$$\mathcal{C} = \mathcal{C}_{\text{obs}} \cup \mathcal{C}_{\text{free}}. \tag{2.1}$$

While the obstacle space is defined as region occupied by a number of obstacles

$$\mathcal{C}_{\text{obs}} = \{\mathbf{q} \in \mathcal{C} \mid \mathcal{G}(\mathbf{q}) \cap \mathcal{O} \neq \emptyset\}, \tag{2.2}$$

the free space $\mathcal{C}_{\text{free}}$ describes the leftover space not occupied by any obstacles, i.e., where a robot can freely move

$$\mathcal{C}_{\text{free}} = \mathcal{C} \backslash \mathcal{C}_{\text{obs}}. \tag{2.3}$$

Fig. 2.1 illustrates a path planning problem in the configuration space $\mathcal{C}$. The task of a path planning algorithm is to compute a continuous path $\pi : [0, 1] \rightarrow \mathcal{C}_{\text{free}}$ in the free space from an initial $\pi(0) = \mathbf{q}_{\text{init}}$ to a goal configuration $\pi(1) = \mathbf{q}_{\text{goal}}$ or to output a failure if such a path does not exist. Different path planning algorithms differ from each other in terms of completeness. A path planning method is considered *complete*, if the discovery of a solution can be guaranteed in a finite amount of time and accurately report failure if no solution is found. Since the complexity of complete algorithms is not viable for practical use, more practical planners have been developed using the weaker concept of completeness [KF11]. Such planners can either guarantee *resolution completeness* or *probabilistic completeness*, but cannot report if a problem is not solvable. Resolution completeness depends on the resolution parameter to return a valid solution, if one exists. Probabilistic completeness can guarantee that with an increasingly dense graph, the probability to find a solution converges to one [LaV06].

Feasibility and optimality are the criteria to assess an outcome of a plan [LaV06]. A feasible plan guarantees that the goal state is reached not taking any optimality criterion

into account. An optimal plan is generated with respect to a predefined performance metric, such as reaching a goal in minimum time or in an energy-efficient manner, where the feasibility of the plan is assumed.

Further, path planning methods are differentiated between *online* and *offline* methods. *Offline* path planning methods compute a robot's path prior to execution. Such methods are suitable for static environments or repetitive tasks, where no dynamic changes occur along a robot's path. On the contrary, *online* path planners compute a robot's path during motion which allows for an interaction of the robot with its environment. Online path planning is usually required for highly dynamic and unstructured environments or for robot-human interaction. In these cases, it is assumed that a robot's path is allowed to change during a robot's motion [KW10].

In general, path planning algorithms can be divided into four categories: *roadmap techniques*, *cell decomposition methods*, *artificial potential field methods* and *sampling-based planners* [CG15; KL16]. They differ from each other in handling path planning problems with different degrees of complexity. The first two path planning approaches are efficient in solving a narrow class of problems by constructing one-dimensional roadmaps for two- or three-dimensional problems, but suffer from weak scalability for more general cases [KL16]. Moreover, the planners rely on an explicit representation of the obstacle space $\mathcal{C}_{\text{obs}}$ [Cho+05]. Therefore, sampling-based planners were developed that were successfully applied for high-dimensional configuration spaces and are suitable for a wide range of robotic applications [KF11]. Sampling-based planners belong to the well-established motion planners in robotics and are freely accessible to the robotics community, e.g., via the Open Motion Planning Library (OMPL) [ŞMK12], which is well integrated in the Robot Operating System (ROS) [Sta18] framework.

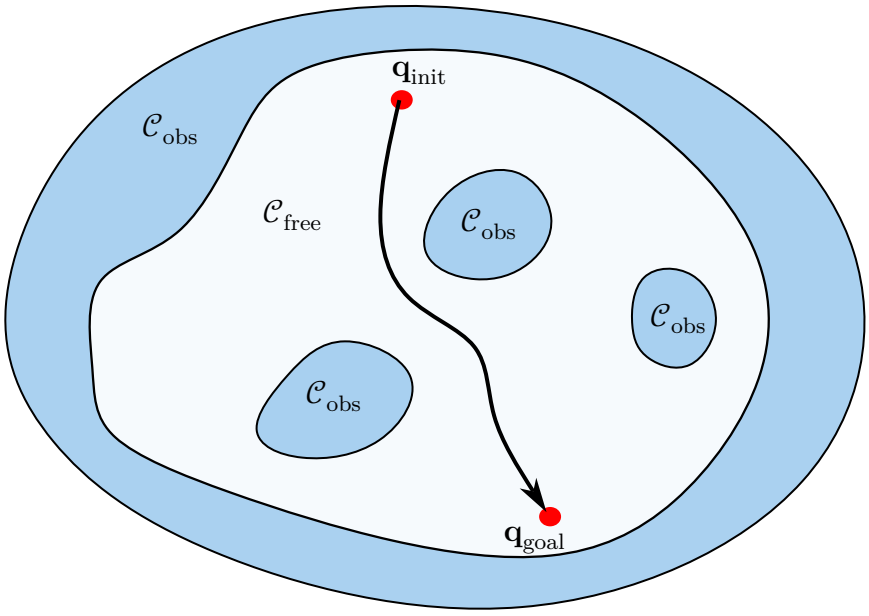


Figure 2.1: Path planning problem in configuration space.

2.1.1 Roadmap techniques

The roadmap techniques reduce the N-dimensional configuration space to a set of one-dimensional paths in $\mathcal{C}_{\text{free}}$ by generating a graph [Cho+05]. Each obstacle is modeled by a polygon, which requires an explicit representation of $\mathcal{C}_{\text{obs}}$. A graph is built by connecting all pairs of nodes of different polygons. More precisely, each vertex of one obstacle is connected to each vertex of another obstacle without violating the interior of an obstacle. Thus, the one-dimensional paths lie in the free space and represent a roadmap. A solution to the motion planning problem exists, if a continuous path can be found from the initial to the goal configuration. In case multiple solutions exist, the solution realizing the shortest path is chosen using the Dijikstra algorithm [Dij59]. There are various types of roadmap algorithms, which are primarily based on either visibility graphs [LW79] or Voronoi diagrams [Sug+09]. A solution obtained by a visibility graph leads to a path closely following the boundary of obstacles, while a path generated by a Voronoi diagram keeps maximum distance to the obstacles. Roadmap techniques belong to complete algorithms.

2.1.2 Cell decomposition methods

Cell decomposition methods subdivide the free configuration space $\mathcal{C}_{\text{free}}$ of the robot into cells. Two cells are considered adjacent if they share a common boundary. Based on the adjacency relationships between the cells, an adjacency graph is constructed, where a node represents a cell and edges connect pairs of adjacent cells. The cell decomposition method consists of two steps. In the first step, the planner determines the cells that contain the initial and goal configurations. In the next step, a continuous collision-free path is computed within the adjacency graph. This is realized by following adjacent cells from initial to goal configuration [Cho+05]. In contrast to other path planning methods, the cell decomposition method can be applied to achieve complete coverage, i.e., a robot visits each cell in its $\mathcal{C}_{\text{free}}$ [Cho+05].

Two main categories of decomposition techniques may be distinguished. The first category is called the *exact cell decomposition technique*, where the free space is exactly subdivided into non-overlapping trapezoidal or triangular cells. However, in some cases it is not possible to exactly decompose the free space. Therefore, an alternative approach, namely the *approximate cell decomposition*, can be applied. It is a recursive method that continues to subdivide a cell into four smaller cells until each cell lies completely in free space [CG15].

2.1.3 Artificial potential field methods

The artificial potential field (APF) approach was firstly introduced by Oussama Khatib [Kha85] in 1986 as a collision avoidance method for a manipulator, where the principle is introduced as follows:

"The manipulator moves in a field of forces. The position to be reached is an attractive pole for the end-effector and obstacles are repulsive surfaces for the manipulator parts."

The APF approach does not require an explicit representation of the configuration space, i.e., $\mathcal{C} = \mathcal{C}_{\text{obs}} \cup \mathcal{C}_{\text{free}}$. The search is carried out by an incremental exploration of the free space and simultaneously determining a collision-free path. Consequently, the method is applicable to a broader class of robots and high-dimensional configuration spaces, and is not limited to Euclidean spaces.

The method is based on attractive and repulsive forces, where the robot acts as a positively charged particle attracted to the negatively charged target. The obstacles act as positively charged particles, provoking repulsive forces so that the robot finds a path from the start to the goal configuration by avoiding them. Therefore, a potential function is defined as a differentiable real-valued function $U : \mathbb{R}^m \to \mathbb{R}$. The gradient of the potential function U can be interpreted as a force pointing in the direction of increasing potential and is given by $\nabla U(\mathbf{q}) = \left[\frac{\partial U}{\partial q_1}(\mathbf{q}), \dots, \frac{\partial U}{\partial q_m}(\mathbf{q})\right]^{\mathrm{T}}$. Gradient descent method is applied

$$\dot{\gamma}(t) = -\nabla U(\gamma(t)), \tag{2.4}$$

to compute a path $\gamma(t)$ along the negative gradient of the potential function.

The robot's motion ceases if the gradient of the potential function vanishes, i.e.,

$$\nabla U(\mathbf{q}^*) = \mathbf{0} \tag{2.5}$$

holds. However, the path may guide the robot to a local minimum that does not correspond to the desired goal configuration. Presence of local minima is the main drawback of the potential field method. Therefore, this class of path planning methods is not complete. The advantage of the APF method lies in its ability to plan paths online, due to its responsiveness to environmental changes detected by sensors [CG15].

To overcome the problem with local minima, several approaches were introduced. In 1989, Warren [War89] introduced an APF method as a global path planner to determine the entire path at once, which is less prone to local minima. This requires a priori knowledge of the environment and can only be applied for static environments. Harmonic potential field method was introduced, where potential (navigation) functions can be constructed such that a unique minimum at $\mathbf{q}_{\text{goal}}$ exists [KK92; RK92]. However, navigation functions can only be applied to a limited class of configuration spaces. An alternative to the former

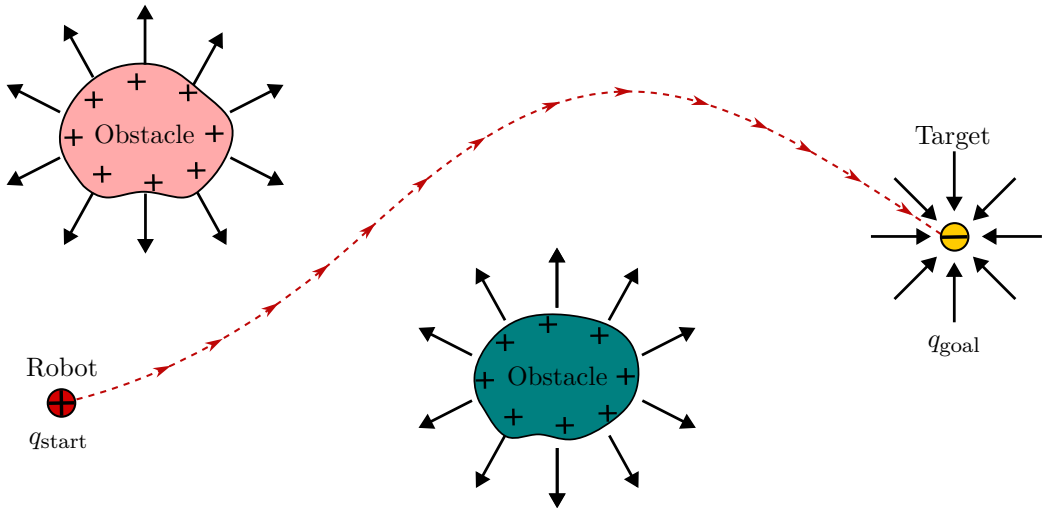


Figure 2.2: Robot navigating through potential field, where obstacles exert repulsive forces on the positively charged robot while the target attracts the robot resulting in a path from start configuration q_{start} to goal configuration q_{goal}.

approach is the Randomized Path Planner (RPP) [Bar+96], where a series of random walks is initiated to escape the local minimum once the robot gets trapped. Park et al. [Par+08] extended the APF approach by introducing dynamic potential fields, which allows to consider the velocity as well as the direction of the robot's end-effector.

2.1.4 Sampling-based planners

Traditional methods, such as graph search techniques, are not capable of handling high-dimensional configuration spaces, as they require an explicit representation of the configuration space $\mathcal{C}_{\text{obs}}$, which is computationally expensive. As a result, they are confined to low-dimensional spaces and simple environments with only a few obstacles. Sampling-based planners were developed to avoid reliance on an exact geometric representation of $\mathcal{C}_{\text{obs}}$, enabling them to manage high-dimensional spaces and obstacle-rich environments. Sampling-based planners are typically designed for static environments and are classified as global planners.

Sampling-based planners randomly select collision-free configurations (samples) of a robot, referred to as sampling process, to construct a roadmap or a tree containing those samples. For sampling, a collision-checking module is applied, which reports all geometric contacts (collisions) between objects. Different types of collision detection algorithms can be applied depending on the class of robots and the type of problem. Once the samples have been generated, a path planning problem is solved by connecting the nearest samples to obtain a collision-free path. The method can only guarantee probabilistic completeness, which is the main disadvantage of the method. Randomness can impact the quality and optimality of the solution. Sampling-based planners differ in their sampling methodologies and the

data structures they employ. Sampling-based planners can be categorized into *multi-query* and *single-query* planners [KL16].

Multi-query planners are employed for applications involving complex static environments, where multiple paths need to be planned. A roadmap, containing collision-free samples, is constructed only once to be reused for multiple queries [LaV06]. The most applied multi-query sampling-based planner is the Probabilistic RoadMaps method [Kav+96], abbreviated as PRM. PRM generates probabilistic roadmaps by sampling random configurations (vertices) of the robot and interconnecting adjacent vertices within a predefined distance by a local planner, e.g., connecting two nodes by a straight line. Once a roadmap exists, it can be used for new queries at any time in the same environment [LaV06]. However, the method does not guarantee to find an optimal path. Optimal Probabilistic RoadMaps (PRM*) [KF11] is an advanced version of the PRM method that guarantees asymptotically optimal paths.

The preprocessing phase to construct a roadmap is computationally exhaustive procedure, while path planning queries can be obtained very fast [SL02]. Solving a path planning problem in narrow passages is another drawback, where a path cannot easily be obtained [KL16]. Therefore a post-processing step is required to improve the quality of the paths [CG15]. A newly developed planner, the Hybrid Potential based Probabilistic Roadmap (HPPRM) [Rav+20] has been proposed, that is based on PRM and the artificial potential field method. The goal is to generate better planning queries with a smaller graph at lower computational costs and higher success rate. Another approach addressing the issue with narrow passages has been proposed for manipulators, which is based on a virtual force field to increase the density of samples in the area of narrow passages. The method proposes a better sampling technique and is applicable for manipulators with an arbitrary number of degrees of freedom [Che+21].

Single-query planners are applied for a single planning query to find a path from a start to a goal configuration. The method eliminates the need to create a global roadmap, i.e., preprocessing step is not required [KL00]. It explores only a part of the configuration space by randomly sampling it and incrementally building a path towards the goal configuration. The planners are based on constructing tree data structures rooted at the initial state. Given the initial and goal configurations, which are vertices of the search graph, the single-query planners attempt to determine a continuous path in the free configuration space by expanding a single or multiple trees to connect the two initial vertices [KL16]. The vertices are randomly selected configurations to bias the exploration towards unexplored regions of the configuration space [KL00]. Single-query planners cannot guarantee to find an optimal path attributed to randomly selecting samples. However, they are well-suited for applications that require rapid path planning in complex and dynamic environments.

The expansion of the tree data structure can be carried out in three different ways:

- Unidirectional: single tree, starting from initial configuration q_{init} [LK01],
- Bidirectional: two trees, one rooted in the initial configuration q_{init} and the other tree in the goal configuration q_{goal} [LK01],
- Multidirectional: multiple trees, rooted in multiple configurations [Str04].

Rapidly-exploring Random Tree (RRT) [LK01] or, a more general algorithm, the Rapidly-exploring Dense Tree (DRT) [LaV06], represent the most applied single-query algorithms. One of the most simplest and efficient variants of RRT is RRT-Connect which uses greedy heuristic to connect two trees rooted in initial q_{init} and goal configurations q_{goal} [KL00]. There exist a wide variety of tree-based algorithms, such as Manhattan-like RRT [CJS08], discrete RRT (dRRT) [SSH15] and subdimensional expansion [WC15], that are mainly applied to more complex problem settings like multi-robot motion planning. An asymptotically optimal extension of dRRT, named dRRT* [Sho+20] has been proposed and applied for four robotic arms, each with seven degrees of freedom solving a joint task.

2.2 Trajectory planning

Trajectory planning problem solves the task at what velocity and acceleration the robot's control system should execute the computed path subject to a selected optimality criterion, for instance, minimum-energy or minimum-time. To obtain velocities and accelerations, a trajectory should be at least twice-differentiable. Generating smooth trajectories is a prerequisite to limit jerks on robot's actuators and to guarantee a good tracking performance [CG15].

Similarly to path planning, trajectory planning can be performed either *online* or *offline*. An *online* trajectory planning method can recompute the trajectory during a robot's motion while the *offline* trajectory is planned beforehand to be executed subsequently. Online trajectory planners are capable of reacting upon sensor events, such as unforeseen changes in the environment, thus giving the robot the ability to operate within dynamic environments. Another reason employing online trajectory planners is to improve trajectory-following accuracy. Note, that the terms *real-time* and *online* are used as synonyms in the context of path/trajectory planning [KW10].

In general, considering robotic applications in a static environment, trajectory planning is preceded by path planning. For challenging applications, such as cooperation of robots or human-robot interaction, high flexibility of a robot is required, where path and trajectory planning may be solved at once. This is the case for point-to-point trajectories, where the specification of additional waypoints along a path is not required or possible as the

environment of the robot is continuously changing.

In general, a trajectory planning problem is solved by choosing an optimality criterion in terms of minimizing or maximizing certain metrics subject to certain constraints. Specifying an optimality criterion is crucial to assess the quality of the obtained solution. The most prevalent are minimizing execution time, traveled distance, energy or jerk [CG15]. To solve a trajectory planning problem, optimal control theory is applied. In most cases, the optimization problem cannot be solved analytically due to the complexity of the imposed kinematic and dynamic constraints of the considered robotic system. Therefore, two numerical approaches have been developed: *optimization-based method* and *grid-based search* [Cho+05].

2.2.1 Optimization-based methods

Consider a trajectory planning problem, which should be solved for a continuous-time dynamical system given in its general form

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \tag{2.6}$$

where $\mathbf{x}(t)$ describes the state vector of the system and $\mathbf{u}(t)$ represents the control input vector applied to the system. The basic idea of optimization-based methods is to formulate a trajectory planning problem as an optimal control problem (OCP). Considering a robotic arm system, the problem can be formulated either in the task space $\mathcal{W}$ or the configuration space $\mathcal{C}$ of the robot. In case the interaction between the robot and its environment are needed to be controlled, then the OCP is formulated as task space control [CFK16]. If the main goal is path-following control, then the problem is solved by formulating the OCP in the joint space resulting in a joint space control [CFK16]. Both kinds of control require either inverse or forward kinematics of the robot. A general formulation of an OCP in continuous time with states $\mathbf{x}(t)$ and control inputs $\mathbf{u}(t)$ is given as following

$$\begin{aligned}
\min_{\mathbf{u}(t)} \quad & J(\mathbf{x}(t), \mathbf{u}(t)) && \text{(2.7a)}\\
\text{s.t.} \quad & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad \forall t \in [t_0, t_\mathrm{f}] && \text{(2.7b)}\\
& \mathbf{x}(t_0) = \mathbf{x}_0, && \text{(2.7c)}\\
& \mathbf{x}(t_\mathrm{f}) = \mathbf{x}_\mathrm{f}, && \text{(2.7d)}\\
& \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t)) = \mathbf{0}, && \text{(2.7e)}\\
& \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \leq \mathbf{0}. && \text{(2.7f)}
\end{aligned}$$

The objective function $J(\mathbf{x}(t), \mathbf{u}(t))$, also referred to as the cost function, defines a quality criterion that the system should optimize. For instance, a time- or energy-optimal solution is often sought to satisfy the requirements (criterion) of fast and efficient production processes [Han+12]. In path-following control, the goal of trajectory planning is typically to

minimize tracking error, ensuring that the robot adheres closely to the reference trajectory. The trajectory planning problem (2.7a) includes a set of constraints, such as the robot dynamics formulated as an equality constraint in (2.7b) with a specified initial condition (2.7c), where computation of the trajectory should start. The terminal constraint (2.7d) enforces the system to reach the final state $\mathbf{x}(t_\mathrm{f})$ at the final time t_f. The final time may either be predefined or treated as an additional decision variable in the OCP. Additionally, system limitations, such as maximum torque and obstacle avoidance, should hold and can be incorporated into constraints (2.7e) and (2.7f) to ensure the safety of the robotic system.

Smoothness of the objective function and constraints with respect to the decision variable $\mathbf{u}(t)$, i.e., at least once or even twice-differentiable, is an important prerequisite to solve the OCP using numerical methods. Prior to solving the OCP, the infinite-dimensional problem (2.7) is discretized in time to obtain a finite-dimensional approximation. Thereafter, the discretized OCP is solved by applying numerical methods for nonlinear programming (NLP). In order to solve the continuous-time OCP (2.7) a direct or an indirect method can be used [Bet98]. An excellent overview of numerical techniques to solve trajectory optimization problems is given by Betts [Bet98]. For the convergence of the numerical algorithm, it is important to obtain gradients and Hessians of the objective function and constraints. By considering a linear robot dynamics, linear constraints and a quadratic cost function, the NLP reduces to a quadratic programming (QP) problem. QP problems are convex and thus possess a global optimum.

However, the real world is nonlinear and highly complex, and it is not always possible to formulate an appropriate QP. For certain robotic applications, nonlinear robot dynamics or non-convex collision avoidance constraints can be neither linearized nor neglected. Further, the presence of discontinuities is "the biggest obstacle encountered in the practical application of NLP methods" [Bet10]. Solving an NLP can be computationally expensive due to numerical evaluations of gradients and Hessians of the objective function and a potential large number of constraints. There exist several well established trajectory planners for manipulator systems, which are widely used for different robotic applications, such as Stochastic Trajectory Optimization for Motion Planning (STOMP) [Kal+11], Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [Rat+09] and Trajectory Optimization for Motion Planning (TrajOpt) [Sch+13]. STOMP and CHOMP realize collision avoidance by adding the Euclidean distance between the robot and the obstacles as a penalty term to the cost function. TrajOpt incorporates convex-convex collision checking as constraints to the OCP and solves it by means of sequential quadratic programming. In general, these methods are applied for static environments and a robot having a fixed goal.

2.2.2 Grid-based search methods

Grid-based search methods determine a path within a discrete grid environment. The environment is divided into a grid of cells, where each cell represents a state. Different types of search algorithms can be utilized to obtain a path from robot's initial to its target state. For instance, the breadth-first search (BFS) algorithm generates a search tree in a breadth-first fashion. Each node corresponds to a state, where the initial state forms the root of the search tree. Each level of the tree is obtained by applying discretized control inputs and computing states. In case the trajectory to a new node in the tree transits or comes close to an obstacle or violates specified requirements, for instance maximum speed or acceleration, the node is pruned from the search tree. The search is terminated when the goal state lies in a region with a certain tolerance specified by the user. The drawback of the grid-based search methods is the exponential growth in the dimension of the state space of the search tree. This limits the applicability of the approach for high-dimensional systems [Cho+05].

2.2.3 Alternative approaches

Artificial potential field (APF) method, introduced in Sec. 2.1.3, can be adapted to be applied for trajectory planning problems. The APF method can either be extended by a subsequent time parametrization, once a path has been obtained or by incorporating velocities and/or accelerations as dynamic constraints in the formulation of the APF problem [Cho+05].

Marcucci et al. [Mar+23] propose a newly developed approach, called Graph of Convex Sets (GCS), that combines trajectory optimization and convex optimization techniques for graph search. The corresponding optimization problem is a mixed-integer problem, where safe regions for the robot are defined to reach the final state and within these regions the optimal trajectory is solved. The algorithm was tested on experiments using a seven degrees of freedom (DoF) robotic arm manipulator.

2.3 Model predictive control

Model predictive control (MPC) is an optimization-based control method that predicts a system's future behavior based on a dynamical model of the underlying system [May+00; MH07]. MPC is an iterative approach, where the optimization problem is typically formulated for a finite prediction horizon and solved for each time step to obtain optimal control inputs in the future. MPC is also referred to as Receding Horizon Controller as the optimization problem is solved repeatedly over a shifted finite horizon [BBM17]. The

merits of the MPC framework are its predictive nature and the effectiveness in dealing with multivariable constrained control problems which makes the method appealing and extremely powerful for a wide range of industrial applications [Sch+21].

2.3.1 Mathematical formulation

In general, MPC as an OCP solves the path and trajectory planning problem at once by considering the kinematics and dynamics of a robot in the constraints of its optimization problem. In general, MPC problems are solved by applying direct methods which transform the continuous-time OCP (2.7) into an NLP by discretizing in time. Multiple shooting [Die+06], as a direct method, uses a full discretization, i.e., the states $\mathbf{x}^k := \mathbf{x}(t_k)$ as well as control inputs $\mathbf{u}^k := \mathbf{u}(t_k)$ are uniformly discretized at time steps $t_k = t_0 + k \cdot T_\mathrm{s}$ over a prediction horizon length N_p. Furthermore, with the following short-hand notation

$$\mathbf{x}^{0:N_\mathrm{p}} = \mathbf{x}^0, \mathbf{x}^1, \dots, \mathbf{x}^k, \dots, \mathbf{x}^{N_\mathrm{p}} \quad \text{and} \quad \mathbf{u}^{0:N_\mathrm{p}-1} = \mathbf{u}^0, \mathbf{u}^1, \dots, \mathbf{u}^k, \dots, \mathbf{u}^{N_\mathrm{p}-1}, \tag{2.8}$$

the general mathematical formulation for discrete-time MPC is given by

$$\begin{aligned}
\min_{\mathbf{u}^{0:N_\mathrm{p}-1}, \mathbf{x}^{0:N_\mathrm{p}}} \quad & J^\mathrm{f}(\mathbf{x}^{N_\mathrm{p}}) + \sum_{k=0}^{N_\mathrm{p}-1} J^\mathrm{c}(\mathbf{x}^k, \mathbf{u}^k) && \text{(2.9a)} \\
\text{s.t.} \quad & \mathbf{x}^{k+1} = f(\mathbf{x}^k, \mathbf{u}^k), \quad k = 0, \dots, N_\mathrm{p} - 1, && \text{(2.9b)} \\
& \mathbf{x}^0 = \mathbf{x}^\mathrm{s}, && \text{(2.9c)} \\
& \mathbf{x}^k \in \mathcal{X}, \quad k = 0, \dots, N_\mathrm{p}, && \text{(2.9d)} \\
& \mathbf{u}^k \in \mathcal{U}, \quad k = 0, \dots, N_\mathrm{p} - 1, && \text{(2.9e)}
\end{aligned}$$

consisting of a cost function subject to a set of constraints.

The objective function (2.9a) consists of running costs $\sum_{k=0}^{N_\mathrm{p}-1} J^\mathrm{c}(\mathbf{x}^k, \mathbf{u}^k)$ over the prediction horizon and terminal cost $J^\mathrm{f}(\mathbf{x}^{N_\mathrm{p}})$ at final time step of the prediction horizon. Discrete-time MPC subject to a terminal constraint as in equation (2.7d) may result in an infeasible problem due to the finite prediction horizon length. To address this issue, the constraint is relaxed and added as a terminal cost $J^\mathrm{f}(\mathbf{x}^{N_\mathrm{p}})$ to the objective function. The minimization problem (2.9) is subject to system's dynamics (2.9b), which can be linear or nonlinear, initial condition (2.9c) and equality and/or inequality constraints in states and control inputs (2.9d)-(2.9e) depending on the physical limitations or requirements of the considered system.

The basic idea of MPC is illustrated in Fig. 2.3. In the given example, the objective is to realize trajectory-following control ensuring that the system closely follows the given reference trajectory. At time step t_0, the open-loop OCP (2.9) is solved over a finite prediction horizon length N_p using the system's measured state $\mathbf{x}^0$ as initial condition.

The solution of the optimization problem are optimal sequence of states $\mathbf{x}^{*\,0:N_\mathrm{p}}$ and control inputs $\mathbf{u}^{*\,0:N_\mathrm{p}-1}$. To close the open-loop, the first optimal control input $\mathbf{u}^{*\,0}$ is applied to the system and the prediction horizon is shifted one time step further. Subsequently, the reaction of the system on the last optimal input is measured at the time instant $t_0 + T_\mathrm{s}$. Based on that information the open-loop optimal control problem (2.9) is solved again over the shifted prediction horizon. This procedure is repeated at each time step until the measured state of the plant is sufficiently close to the desired state.

2.3.2 Control schemes

Over the decades, MPC has evolved to handle large-scale systems prevalent in many engineering fields. Although the *Centralized MPC* solves a global optimal control system at once, the centralized scheme is often impractical and impossible to be applied for large-scale and networked systems [VRW07]. Reasons for this are versatile, such as spatial distribution of the system's components, considerable computational complexity in solving the centralized problem online, scalability issues resulting from a large number of decision variables and constraints to be solved as the system grows, difficulties in obtaining a dynamical model of the overall plant and lack of modularity to integrate new subsystems. Still, the centralized MPC serves as a benchmark control framework for assessing other controllers. It became evident that decomposing the large plant-wide problem into a set of simpler and smaller subproblems is more practical [RM09]. Thus, decentralized and distributed MPCs emerged to handle a set of subsystems instead of a large-scale system

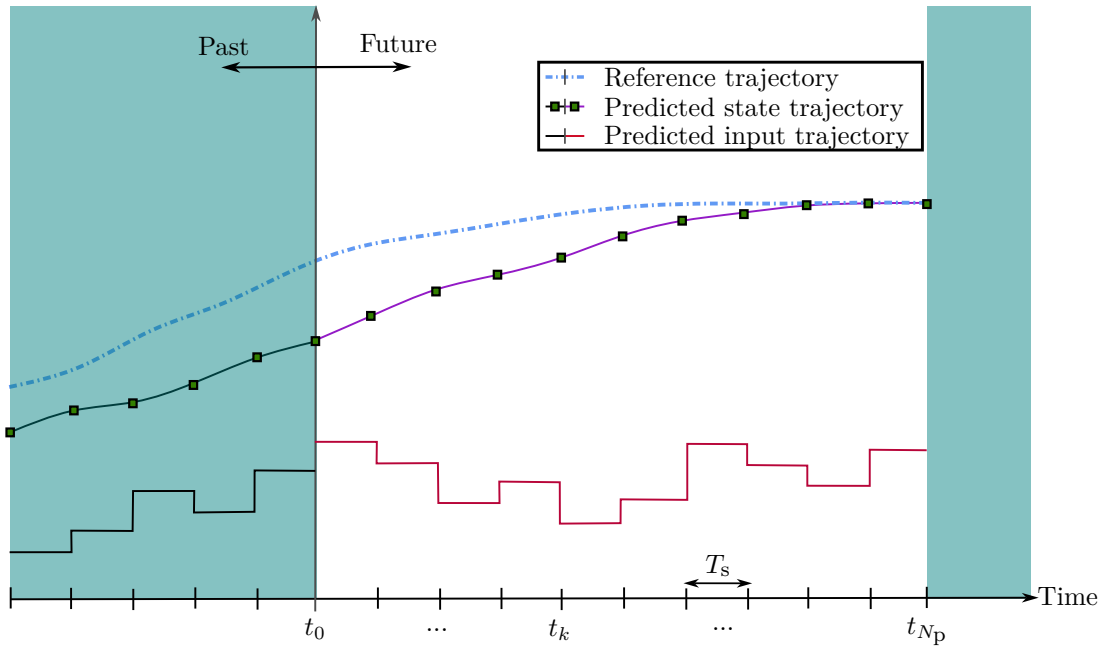


Figure 2.3: Model predictive control scheme with shifted prediction horizon illustrated as white window.

in a more computationally efficient and application-dependent manner.

Decentralized MPC may be applied for plants, where the interaction between the subsystems can be neglected, and thus, communication is not required. Nevertheless, using a decentralized control scheme typically results in a worse system-wide control performance compared to the centralized scheme [VRW07]. Therefore, a *Distributed MPC* is often chosen, where the inter-subsystem communication is allowed to share the predicted states and input trajectories to solve each subsystem's individual optimization problems. Both types of MPC schemes, decentralized and distributed, scale well with the size and complexity of the large-scale system and allow the design of a more reliable system where subsystems might fail during operation. In addition, both schemes have a lower computational burden than the centralized scheme. The primary limitation of applying MPC in the industry is the substantial modeling effort required, which can be exceedingly time-consuming [Hen98; Sch+21]. Another drawback of MPC is the considerable computational burden associated with solving the constrained optimization problem within the restricted time frame dictated by the system's sampling time [VDS11].

Solving a constrained optimization problem over a finite prediction horizon within a fixed sampling period requires efficient computer hardware and solvers. Initially, MPC was applied for processes with slow dynamics, where the sampling times are typically hours or days [QB03]. MPC was first applied in the petrochemical industry to handle multivariable constrained control problems [CR79]. Nowadays, with advances in computational power and robust numerical algorithms, the application of MPC spread not only to a broader range of engineering fields, such as building energy management systems (heating, ventilation and air conditioning (HVAC)) [Drg+20; THR22], manufacturing [Sch+21], but also agriculture [Din+18] and medicine [Haj+18]. In the last decade, systems with fast dynamics in the range of ms received significant attention. Areas where MPC was applied successfully include power electronics [Vaz+14], robotics [SZ21], or automotive systems [Sul+17]. The widespread use of MPC can be attributed to the availability of efficient OCP frameworks, such as ACADO [HFD11] or *CasADi* [And+19] and other similar tools, which systematically incorporate system dynamics and constraints within a receding horizon framework.

2.3.3 Collision avoidance

In general, developing a collision avoidance strategy for a robot operating in a dynamically changing environment is a challenging task that remains a subject of extensive research in robotics [ZLB21; Mav+21]. Protection of both the human body and the robotic structure is essential. Simply stopping the robot upon detecting a potential collision is not an efficient solution [Had+08], as it may obstruct the human or other robots from completing

their tasks [HDA17]. This section presents various approaches to collision avoidance that are suitable for integrating into an OCP.

In general, a collision avoidance strategy typically consists of two parts:

1. A geometric representation of collision-prone bodies, where these bodies are enclosed within convex shapes.
2. A collision-checking method that guarantees a safe distance between the collision-prone bodies.

The bounding volume (BV) method is considered as the state-of-the-art technique for the geometric representation of collision-prone bodies. This method is based on the idea to enclose objects within simple geometric shapes and checking for intersections between the BVs of these enclosed objects. Convex shapes such as spheres, cylinders, and other similar forms are typically used to represent full-dimensional objects. A new family of BVs was introduced by Larsen et al. [Lar+99] to achieve a tighter fit to the original shape of the objects. This method involves sweeping a certain radius over all points of a core primitive shape. The most applied swept sphere volumes are the point swept sphere (PSS) in the form of a sphere, the line swept sphere (LSS) representing a spherocylinder and the rectangle swept sphere (RSS) yielding a rounded box. For instance, LSSs are ideally suited to enclosing elongated objects, such as robot links.

The most commonly used collision-checking method is the GJK algorithm, often referred to as convex-convex collision-checking [GJK88]. This method is widely applied in both robotics [Sch+13; Mav+21] and computer graphics [JTT01; Ake+18]. The GJK algorithm computes the minimum distance between two convex sets by applying the Minkowski sum. Each object is represented by a polygon, and the goal is to determine the closest points of the two polygons and compute the Euclidean distance between them. The method operates on the principle that the distance between two convex sets and the origin is used instead of directly computing the distance between the convex sets, which are effectively the same. In the literature, the Euclidean distance metric is often used between isolated points distributed across a robot's body, resulting in a set of overlapping spheres, and obstacles enclosed by spheres [Rat+09; Kal+11; PPM12; Fla+12; Zub15; Nav+16; LX19; Zha+20; Tik+20].

Another approach that is frequently used, is the algorithm introduced by Lumelsky [Lum85] in 1985 to compute minimum distance between collision-prone bodies represented by line segments. Each line segment is formulated as a parametric equation

$$\mathbf{u} = \mathbf{p}_i^A + \alpha(\mathbf{p}_i^B - \mathbf{p}_i^A), \quad \alpha \in [0,1], \tag{2.10a}$$

$$\mathbf{v} = \mathbf{p}_k^A + \beta(\mathbf{p}_k^B - \mathbf{p}_k^A), \quad \beta \in [0,1]. \tag{2.10b}$$

The minimum distance between both line segments is obtained by solving an optimization

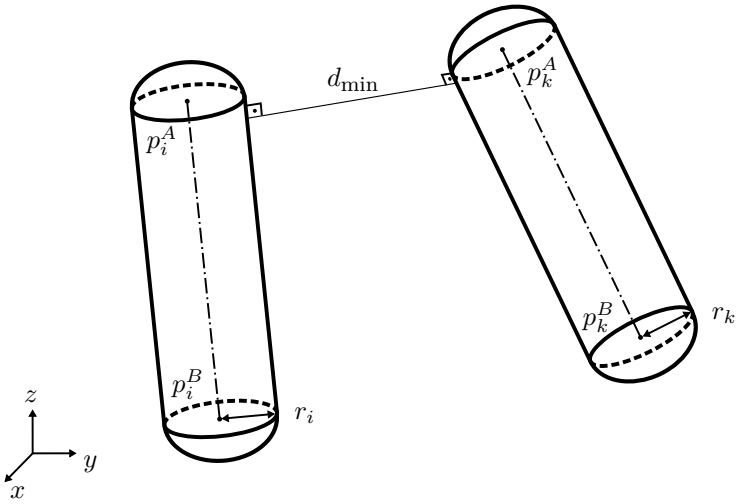


Figure 2.4: Two bodies i and k are encapsulated within LSS with radii r_i and r_k. A minimum distance $d_{\min}$ between two LSSs is computed by applying the algorithm from Lumelsky [Lum85].

problem, where the parameters α and β represent decision variables

$$d_{\min} = \min_{\alpha,\beta} ||\mathbf{u} - \mathbf{v}||_2 \tag{2.11a}$$

$$0 \leq \alpha \leq 1, \tag{2.11b}$$

$$0 \leq \beta \leq 1. \tag{2.11c}$$

The method can also be applied to compute distances between collision-prone bodies modeled as LSS [CCB90; Cas+09; Die+11; Log+18; Krä+20]. Fig. 2.4 illustrates two bodies represented by LSSs having the radii r_i and r_j, where the minimum distance $d_{\min}$ is computed using the algorithm from Lumelsky.

In 1986, Faverjon and Tournassoud [FT87] presented the velocity damping (VD) approach, which can be incorporated as a collision avoidance constraint within an OCP. The method is based on restricting relative velocities between two collision-prone bodies to slow down the motion as they approach each other. The method is described in detail with reference to Fig. 2.5. Bodies O_1 and O_2 represent two objects prone to collision, where $\mathbf{p}_1$ and $\mathbf{p}_2$ are the closest points of the respective bodies. A normal vector $\mathbf{n}$ is defined as

$$\mathbf{n} = \frac{\mathbf{p}_2 - \mathbf{p}_1}{||\mathbf{p}_2 - \mathbf{p}_1||_2}. \tag{2.12}$$

The minimal proximity of the two bodies should be greater than the safety distance d_s. To realize this, an inequality constraint is formulated

$$\mathbf{n}^\mathrm{T}(\dot{\mathbf{p}}_1 - \dot{\mathbf{p}}_2) = \mathbf{n}^\mathrm{T}(\mathbf{J}_{\mathrm{p}_1} - \mathbf{J}_{\mathrm{p}_2})\dot{\mathbf{q}} \geq -\frac{d - d_\mathrm{s}}{d_i - d_\mathrm{s}}, \quad \text{for} \;\; d < d_\mathrm{s}, \tag{2.13a}$$

where d_i describes the influence distance at which the constraint becomes active. The difference $d - d_\mathrm{s}$ calculates the distance along $\mathbf{n}$, while $\mathbf{J}_{\mathrm{p}_1}$ and $\mathbf{J}_{\mathrm{p}_2}$ are the Jacobian matrices for O_1 and O_2 determined respectively at points $\mathbf{p}_1$ and $\mathbf{p}_2$. The vector $\mathbf{q}$ denotes

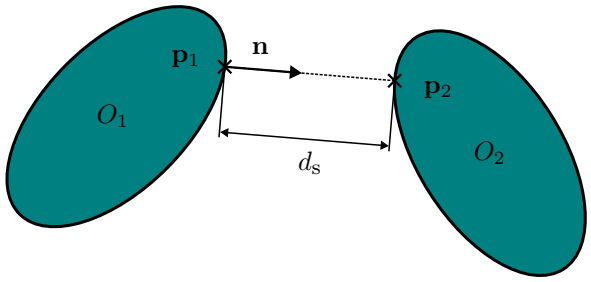


Figure 2.5: Two collision-prone bodies O_1 and O_2, where a certain distance threshold d_s should hold.

the joint velocity. The method gained a significant popularity in handling self-collision avoidance for humanoid robots [Kan+10; Die+11; Zho+16]. It was also applied for space manipulator avoiding obstacles in a static environment [WLW16] as well as the collision avoidance strategy between two robotic arms [BH09].

The main challenge of the collision avoidance methods mentioned previously lies in the distance function, which contains nested logical conditions with non-smooth derivatives. Additionally, describing a manipulator's motion requires mapping between Cartesian space and configuration space, necessitating the use of forward and inverse kinematics. When the OCP is formulated in configuration space, forward kinematics introduces nonlinear constraints. These non-smooth and nonlinear constraints make the OCP difficult to solve. To address this, interior-point methods from nonlinear optimization can be employed, converting the problem into an unconstrained one. An alternative approach is to use sequential quadratic programming (SQP), which linearizes the nonlinear constraints [FGW02].

2.3.4 Review for robotic arm systems

In the following, a review of MPC-based trajectory planners for single manipulator systems is presented, with a focus on studies that have demonstrated successful implementation in real-world applications.

In robotics, motion control strategies for manipulators are generally divided into path-following control and point-to-point motion. Tasks, such as milling, assembly or welding require path-following control. In the last decade, numerous works emerged that realize path-following control along a Cartesian path with MPC. Although the path is defined a priori, the velocities (and accelerations) need to be determined. The advantage of path-following MPC compared to offline methods are the feedback law and the freedom to trade off between productivity and accuracy [Dui+16]. An additional requirement for path-following problems is force control, where the robot must exert a desired force along the pre-computed path to perform tasks such as milling or sawing. A number of works

realized real-time capable implementations of the path-following approach on experiments utilizing a single robotic arm [LMG11; Fau+17; Mat+17; AGG17; GVG23]. Carron et al. [Car+19] introduced a novel approach by combining data-driven models with MPC to achieve high tracking accuracy for a non-industrial, compliant robotic arm with low stiffness and limited actuation power. An experimental validation on a six DoF robotic arm was performed and different types of controllers, such as PID controller and several MPC schemes, were investigated and compared.

Predefined paths are not always beneficial, as they introduce an additional layer of complexity to the motion planning task. Numerous works were recently published that employed MPC to generate point-to-point trajectories, e.g., for a ball-catching task [Ard+15; And+19], moving target positions [Krä+20] or pick-and-place tasks [TGB22; Mav+21]. Mavrommati et al. [Mav+21] demonstrated the potentials of MPC for a manipulator operating in a warehouse environment in presence of static and dynamic obstacles. The collision avoidance with the environment was realized using the GJK method.

To enhance the reachability of manipulators, mobile manipulation, which combines a mobile base with a robotic arm, has garnered increasing interest. Path-following control using nonlinear MPC has been applied in real-time simulations for both a single manipulator and a mobile manipulator, as demonstrated in [Zub15]. In this work, collision avoidance relied on the distance function between specific test points on the robot's body and the center points of obstacles. An online NMPC-based approach for optimal grasping with a mobile manipulator system was carried out in [Log+18]. Li et al. [LX19] addressed a mobile manipulation system that must maneuver while avoiding humans along its path. The authors proposed an MPC-based approach where humans are treated as dynamic obstacles. For collision avoidance, the robotic system was represented by a set of points uniformly distributed along both the manipulator's body and the mobile base. The minimum distance between the robotic system and obstacles was computed to ensure collision prevention.

MPC has gained substantial popularity in space manipulation, HRI, and humanoid robot locomotion. For space manipulation, Wang et al.[WLW16] formulated a trajectory planning problem using NMPC, where the space robot consists of a spacecraft and an n DoF manipulator. Rybus et al.[RSS17; Ryb18] investigated a free-floating manipulator mounted on a satellite, considering a fully nonlinear system model. The objective of the NMPC was to minimize the power consumption of the manipulator motors while following a precomputed reference trajectory without considering obstacles. The authors emphasized the high computational costs associated with NMPC, rendering it unsuitable for real-time applications with manipulators having more than four DoF. Collision avoidance with static obstacles was addressed using the VD approach. In the context of HRI, Krämer et al. [Krä+20] proposed an MPC approach that plans the robot's trajectory in the presence

of a human worker and a moving target pose. The human worker was treated as a dynamic obstacle in the robot's workspace, and BVs encapsulate both the robot and human bodies. To ensure safety, the Lumelsky algorithm computes the Euclidean distance between the robot links and the obstacles. A smooth motion of the robot in the presence of obstacles was achieved by adding separation costs to the objective function of the OCP and formulating the problem with a self-developed hypergraph [Rös+18]. The approach was validated through experiments. Due to the compliant structure of robotic arms and the high number of DoF that must be considered in the case of humanoid robots, some works have shown that MPC for motion planning significantly outperforms other approaches [RHK15; Bes+16]. The MPC framework enables the simultaneous generation of trajectories for the Zero Moment Point (ZMP) and Center of Mass (CoM) while satisfying both kinematic constraints and the ZMP balance condition. Numerous works have applied MPC as a trajectory planner, with successful experimental validation of the algorithm [Fen+16; Nav+16; Sci+20]. Moreover, in highly cluttered environments such as those encountered in home assistance or bin-picking, humanoid robots must find feasible paths by reaching into clutter. Killpack et al. [KKK16] successfully applied MPC to a seven DoF robotic arm of a humanoid robot operating in dense clutter. A dynamic MPC was presented, incorporating the robot's dynamics in contact with its environment into the MPC framework. The performance of the dynamic MPC was experimentally evaluated in multi-contact scenarios, demonstrating the efficacy of the proposed method.

Robotic manipulation requires visual sensing to manage bin-picking or pick-and-place tasks involving many objects and to handle dynamically changing environments with moving targets. A key challenge in this context is the occlusion problem, which arises when objects of interest become obscured from the robot's vision system. Occlusion can occur due to moving targets, obstacles blocking the line of sight, or self-occlusion, where the robot's own links obstruct its view of the target. These issues can be mitigated by employing an optimization-based controller that adjusts the vision system to avoid occluded viewpoints and modulates the robot's movements to reduce self-occlusions. MPC-based approaches addressing these challenges have been proposed and experimentally validated in several studies, including [Log+18; Bha+21; He+22].

2.4 Multi-robot systems

Over the past decades, multi-agent systems (MAS) have attracted considerable attention [Woo09; Cao+13; SY21]. MAS describes a group of interacting agents to accomplish a common goal by cooperating and coordinating their behavior [Woo09]. In the 1980s, the first works dealt with coordination and pathfinding problems for systems involving multiple robots [FH86; Tou86]. Such systems aimed to achieve common goals, such as

cooperative manipulation, formation and exploration, search and rescue, flocking, etc. An excellent overview of different MAS applications can be found in [DB17] and in the references therein.

This section focuses on multi-robot systems (MRS), representing a set of robots accomplishing tasks in the same environment [FIN04]. Multiple robots working together can carry out more complex tasks, work efficiently, increase the reliability of the overall system, and realize tasks that are infeasible for a single robot. However, employing MRS is challenging, where various problems, such as task allocation, coordination, communication, safety, and many other issues, must be solved [KHE15]. This is referred to as the task and motion planning (TAMP) problem, where tasks have to be assigned to individual robots to achieve a robotic team's desired (optimal) behavior, followed by solving the motion planning problem for each robot to complete the tasks. The TAMP problem can be viewed as discrete high-level task planning and continuous low-level motion planning, where the interplay between the two layers has to be managed as well [Gar+21].

Cortés and Egerstedt formulated four requirements for MRS that have to be satisfied [CE17]:

1. Each robot acts upon local information that is available through perception or communication between the robots (agents).
2. The control scheme must scale to multiple robots.
3. Safety needs to be ensured between the robots as well as their environment.
4. The global properties of the system must result from the local interaction rules.

2.4.1 Review

This section reviews works on MRS involving manipulator systems. Extensive research has been conducted in the field of MRS involving Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) [Mad+21], while literature dealing with a group of manipulators interacting with each other is sparse. Application of multi-arm robotic systems can be grouped into two categories: cooperative manipulation and assembly [Hay86].

Cooperative manipulation of a single object has been studied for several years, as heavy payloads and dexterity require multiple manipulators to cooperate. A wide range of tasks exist for cooperative manipulation, such as transporting objects in warehouses, assembly, or search and rescue. By jointly carrying an object, the robots are physically coupled, resulting in a closed kinematic chain. Thus, kinematic constraints on each robot's position and velocity must be considered, and the interaction forces need to be analyzed [EH16]. The task of each robot's controller is to evenly distribute the load among the robots by

applying a suitable force on the object in dependence on other robots' end-effector forces. To reach this goal, an impedance controller is usually applied [Hog84]. A suitable communication strategy must be chosen for the robots to control internal forces during their collaborative tasks. Although deploying a centralized communication scheme gives global information about the overall system, the scheme has several limitations, such as a lack of scalability, the assumption of sharing global information with the agents, and a high computational burden to solve the overall problem. Thus, either distributed [Alo+15; Mar18; DH20; HWL21; ZJW23] or decentralized [YSV21] communication typologies are proposed to handle cooperative manipulation. Dohmann et al. [DH20] developed an event-triggered distributed control scheme to reduce the rate of communication between the robots. To further increase the reachability of the manipulators, mobile platforms are used, where a manipulator is fixed on a mobile base and a common manipulation is performed [Alo+15; Mar18; Zha+21]. This allows for transporting a manipulated object from point A to point B, e.g., in a warehouse. An impressive work has been conducted by Yan et al. [YSV21] that deals with heterogeneous multi-robot teams involving fixed-based and mobile-based robotic arm systems, performing collaborative manipulation tasks. The authors developed a decentralized ability-aware adaptive controller under input constraints. Moreover, uncertain physical parameters of the manipulated object and environment are assumed, which is the case in real-world applications, guaranteeing stability and convergence.

Turning attention to the second category, assembly tasks in MRS can reduce working space, assembly time and increase overall efficiency. Similarly to a single robot system, a common goal, e.g., a minimum execution time or an energy-efficient solution, can be prescribed. Since robotic arm systems are stationary relative to their base, incorporating multiple manipulators enhances the system's reachability through collaboration among the robots. In contrast to a single robot, MRS can solve more complex assembly tasks, possibly substituting still required human workers. However, multiple issues arise when dealing with a fully autonomous MRS. These include task coordination, which involves assigning tasks to each robot individually, and planning collision-free paths for each robot to achieve a common goal.

Unlike cooperative manipulation, MRS performing assembly tasks are kinematically decoupled, but each robot still needs to consider the motion of surrounding robots. Indeed, an efficient collision avoidance strategy in multi-robot systems operating in a shared workspace is crucial, as collisions with the environment and between the robots must never occur. Due to the robots' complex kinematic chains, the full-dimensional geometry of each robot's links must be considered to guarantee a safe operation. Since the late 1980s, the problem of planning collision-free paths for MRS involving manipulators has been analyzed [FH86]. In the beginning, heuristic methods were proposed, where only a single robot was allowed to move to avoid collisions [OL89] or a time scheduling method based on a collision

map was presented [LG87]. For the first time, in 1990, Chang et al. [CCB90] introduced a constrained nonlinear OCP solved by a conjugate gradient method with barrier functions for two robotic arms sharing a common workspace. Since then, only a few works have emerged that allow for multiple manipulators to cooperate freely, i.e., path planners that can compute trajectories online.

Conservative collision avoidance approaches involve the computation of intersection regions between the robots' assumed precomputed paths. One approach is based on computing shadow spaces in a two-dimensional workspace and further discretized into grid cells [YCH19]. By combining the shadow spaces of the robots, a shadow space matrix was obtained, and collision-potential regions were determined. Collision-free coordination of tasks was performed by applying a genetic algorithm. Afaghani et al. [AAA23] presented a collision avoidance scheme based on a collision map computed a priori. Robots could only move in the shared workspace if no collisions had been detected. Once collisions were detected, time delays were added to the trajectory of the involved robots to avoid intersections. Multiple authors [OL89; AH02; MS13; Lee+14] have proposed similar approaches for the coordination of robots utilizing delays and identifying collision zones to avoid collisions between the robots, resulting in idle times during operation. The above-mentioned approaches rely on executing precomputed paths.

More promising concepts allowing for a greater flexibility in MRS are optimization-based methods. However, the number of works that consider collision avoidance between manipulators remains limited. Cascio et al. [Cas+09] applied BV using LSS to enclose the geometry of robots' links or objects and formulated a minimum distance function between two potentially colliding bodies. Additional decision variables were introduced for each collision-prone pair to overcome the problem of nested logical conditions in the distance function. The dimension of the optimization problem increases linearly with the number of obstacles. For a robot-robot cooperation, the authors formulated a minimum-time centralized motion planning, where each manipulator is considered a dynamic obstacle for the other robot. The algorithm was applied to two simple 2 DoF robotic manipulators. Similarly to the previous work, Bosscher et al. [BH09] applied the BV method for geometric representation of the robots and formulated a distance function as collision-checking method between the BVs. The proposed algorithm obtained optimal collision-free trajectories, where the optimization problem was solved in a centralized fashion. Experimental validation was performed on two six DoF manipulators sharing the same workspace. However, the authors did not report the control rate as well as the computation times of the algorithm. Tika et al. [TB21; TB24] developed a minimum-time collision-free online trajectory planning approach for two robotic manipulators. Collision avoidance was realized by approximating the robots' geometry by smooth Bézier curves, and the minimum distance between Bézier curves was incorporated into the constraints of the MPC problem.

The synchronicity of two robots sharing a common workspace is enforced to avoid unforeseen events, e.g., deadlocks. So far, all of the previously mentioned approaches applied a centralized scheme, which limits scalability.

Meanwhile, more complex assembly tasks can be performed by MRS, such as peg-in-hole tasks [Lee+22], screw assembly [SN11], furniture assembly [Kne+13; SZP18] and construction assembly [Ade+18; Dog+19; Har+23]. An impressive study by Suárez-Ruiz et al. [SZP18] showcased the capabilities of state-of-the-art robotic methods, validated through the assembly of an IKEA chair by two robots. The authors reported that it took slightly over 11 minutes to plan the robots' trajectories and almost 9 minutes to execute them. This example highlights that significant time was devoted to motion planning, which could have been minimized if collision-free paths were determined online during execution. It should also be noted that collision avoidance becomes more challenging as the construction grows, limiting the robots' operational range [Par+17]. A recent work by Hartmann et al. [Har+23] undertakes the first step to realize long-horizon assembly tasks for heterogeneous multi-robot setups. The authors proposed to break down the long-horizon assembly problem into a set of short-horizon subproblems containing a subset of robots and assigned tasks. The order of solving the subproblems was determined heuristically. At first, a set of setpoints was computed by applying an optimization-based approach. Further, path planning was conducted using a novel bi-directional space-time embedded RRT method that considers previously planned trajectories of the robots. The authors presented their approach for up to 10 robots assembling 15 to 113 objects on sophisticated simulation setups and experimental scenarios with two robots. However, the authors mentioned that the method is far too slow for online execution.

2.4.2 Deadlocks

In a multi-robot setup, where robots operate closely to each other, more than a task-based controller with an incorporated collision avoidance strategy is needed to guarantee that the robots reach their target states. Robots may block each other, causing a situation where none or only a subset of robots can proceed with their task without violating safety requirements, i.e., collisions between the robots or the surrounding environment are unavoidable. Thus, each robot stops moving and waits until the situation is resolved. This situation is referred to as a deadlock. Such states of the system can also be described as undesired local minima that a system should not converge to, as the robots would not reach their target states. Moreover, once a robot of an MRS is forced to stop temporarily, e.g., in case of passing-by humans, it should not affect the rest of the system to stop [ČGF16]. Such disturbances should be taken into account locally without causing a global delay. Therefore, coordination between the robots is necessary to ensure safety and increase performance [GLS21].

The ocorrence of deadlocks in MRS was already mentioned by Faverjon [FT87] while introducing the velocity damping approach in 1987. In 1989, O'Donnell and Lozano-Pérez [OL89] proposed a trajectory coordinator for manipulators that fulfills four tasks:

- Each manipulator's path should be planned independently.
- The planned trajectories should lead the robots to their target states.
- Execution of trajectories should be possible without time coordination between the robots.
- The robot's safety should not depend on the exact execution of the trajectories.

The authors introduced a solution based on scheduling, where two manipulators were coordinated asynchronously, i.e., one robot at a time was allowed to perform a task. At the same time, the other robot waited until its task space became collision-free and, thus, deadlock-free. This was realized by adding an idle time at the beginning of the precomputed trajectory of the waiting robot. The authors considered a structured and well-known environment without any unforeseen obstacles.

One approach for resolving deadlocks is to instruct individual robots to replan their trajectories until their motions are deadlock-free [JN01]. Another method for resolving deadlocks involves adding perturbation terms to the robots' controllers, enabling them to overcome standstill situations [RSS16; WAE17]. However, no formal guarantees exist that the robots would not be trapped again in a deadlock or violate safety requirements [GLS21]. Lately, control barrier functions gained increasing popularity as a safety-critical controller to coordinate robots in a multi-robot system. An excellent overview of this method is given in [Ame+19]. Grover et al. [GLS21] analyzed control barrier functions based quadratic programs (CBF-QPs) that were applied as local controllers in a distributed framework for a MRS involving mobile robots. CBF-QPs were formulated to ensure safety and a goal stabilization. Due to the distributed control scheme, CBF-QPs are prone to deadlocks. The authors proved that robots being trapped in a deadlock are on the verge of violating safety constraints. The authors proposed a deadlock resolution scheme in which the robots rotate around each other, forming a centroid until their positions are exchanged. Thereafter, the local PD controller of each robot is applied to ensure a safe distance between the robots. Although this scheme might be practical for mobile bases, it cannot be applied for manipulators with fixed bases. The work of Čáp et al. [ČGF16] showed that reactive collision-avoidance methods in MRS, e.g., such as ORCA [BGL11], often lead to deadlocks, where the robots execute the same reciprocal reactive algorithm except the robot that is disturbed. As a precise execution of trajectories by all robots cannot always be guaranteed, such systems are prone to deadlocks. To alleviate this issue and make MRS more reliable, Čáp et al. [ČGF16] suggested a control law that allows robots under disturbance to deviate from the planned trajectory, i.e., by temporarily stopping or

delaying while executing the computed trajectory without causing a global delay of the system.

Works dealing with deadlocks caused by multiple mobile robots exist to some extent [JN01; ČGF16; Tal+18; Zha+21; GLS21], while the number of works considering multiple manipulators is very sparse. Some research addresses the issue of deadlocks but do not offer effective resolution strategies [Dek+18]. The following works [AA14; ZA15; AAA23] proposed a method to handle deadlocks in a setup of multiple robotic manipulators where deviations from predefined paths are restricted. Collision avoidance was realized by adding time delays, resulting in a shifted trajectory of the robot that received the next target state. Collision checking was performed using a collision map, which is highly time-consuming. Time delays are computed during execution. A deadlock resolution approach was initiated in case the time-shifting method fails in unavoidable collisions. Their approach was based on incrementally sending manipulators to an escaping position until the deadlock was resolved. However, only one robot was allowed to proceed with its motion while all the other robots waited.

2.4.3 Multi-robot task allocation

Planning a coordinated sequence of motions is the objective of the multi-robot task allocation (MRTA) problem, where a set of tasks is to be optimally assigned to a set of robots while satisfying a set of constraints. Finding the optimal sequence of tasks to be assigned to a single robot is equivalent to the traveling salesman problem (TSP) [App+07], where a robot (salesman) visits the given number of goal states (cities) at minimum costs. The MRTA problem, also called scheduling, can be described as a multiple traveling salesman problem (mTSP) owing to the similarity between the two problems [Bek06]. Multiple robots (salesmen) travel to several goal states (cities) to achieve a common goal, e.g., minimize total distance or total time, that is to be formulated as the objective of the mTSP problem. Further, various MRTA problems can be considered, such as heterogeneous robots and different types of tasks. Each robot in a team might have its own restrictions, e.g., operating with a certain velocity or energy level [BHK13].

Most scheduling approaches for robotic manipulators focus on cyclic tasks, i.e., a robot visits all its task points only once and returns to its initial configuration afterward. Typically, the posed requirement is to execute all tasks in minimum time. Scheduling can be conducted offline prior to task execution, where kinematic constraints of the robots should be considered. Genetic algorithms (GA) belong to the class of metaheuristic optimization algorithms applicable to a small-scale scheduling problem. The framework allows for the implicit consideration of constraints that are encoded into the genes of the GA by employing a simulation-based optimization. The merit of the method is the capability to handle

non-continuous and procedural objective functions to find near-globally optimal solutions [ZA05]. Zacharia et al. [ZA05] suggested a GA to compute an optimal sequence of tasks for a single robot with a minimal total cycle time. In the proposed approach, several possible manipulator configurations were coded into the genes of the GA. Obstacle avoidance was not taken into account. Bonert et al. [BSB00] introduced a minimum-time approach employing a GA for collision-free motion planning of two robots performing an assembly task. Collision avoidance between the robots was realized by time (safety) delays incorporated in the proposed GA, where one robot's motion was delayed to avoid collisions with the other one. This collision avoidance approach is rather conservative and does not allow for the simultaneous operation of the robots in a shared workspace. Xidias et al. [XZA10] alleviated the problem by approximating the robots' bodies by B-spline curves and checking for intersections between them. The GA is augmented by adding a penalty function that allows for restricting to only collision-free tours of the robots. The work [YCH19] developed a GA to solve for collision-free task sequences assigned to two manipulators. The authors assumed linear transition between two states and the synchronous start of the robots. GAs are generally suitable for problems with well-known environments involving few resources and cyclic tasks. However, these methods suffer from high computational costs once the problem exceeds a certain complexity threshold. Thus, coupling constraints that result from cooperation between robots cannot be easily encoded into a GA [Coe02]. An alternative approach is proposed by Tika et al. [TGB22] for a single manipulator performing pick-and-place tasks to solve task allocation and trajectory planning in a hybrid model predictive control framework. The promising approach is demonstrated for a single manipulator performing six pick-and-place tasks, where trajectory planning and allocation of tasks are performed online.

Scheduling an assembly or construction plan is a highly complex MRTA problem as it involves many objects to be assigned to a set of robots. An assembly sequence is determined based on a final assembled structure. Most importantly, specific assembly steps must be carried out in a distinct order, resulting in constraints that need to be considered when solving the MRTA problem. As construction advances and the size of the assembly grows, expanding the space required within the robots' shared workspace, the objects introduce further constraints on their operations. Thus, the feasibility of a plan is the primary goal when solving a long-term scheduling problem. A constrained large-scale scheduling problem can be formulated as a constraint satisfaction problem (CSP) [Dec03]. The CSP framework allows for the formulation of high-level logical constraints [BSR10]. A decade ago, allocating tasks to the robots was not part of the scheduling problem. Instead, the ordering of the tasks was solved by a scheduling algorithm [CJS08; Kne+13]. Works by Lozano-Pérez et al. [LK14] and Dogar et al. [Dog+19] formulated a long horizon problem as sequential manipulation problems in the form of CSPs. Both works consider allocating tasks to the robots, the assembly sequence, robotic grasps, and the robots' trajectories, all

solved in the same framework. Naturally, solving problems that require long sequences of operations suffers from high computational complexity as scheduling and robot trajectories are computed a priori.

Another common approach is to separate task and motion planning by introducing a multi-layered approach, where task allocation and motion planning are handled in different layers [DB15; GLK18; Pan+21; Har+23; TB24]. Scheduling of tasks is considered successful if the motion planner can obtain a robot path for the scheduled task. Otherwise, a different plan has to be computed. In general, heuristics are applied to obtain a feasible plan for a team of robots. To further simplify a TAMP problem, almost all of the previously mentioned approaches compute the robots' trajectories offline to reduce the occurrence of any unforeseen events. So far, the existing methods do not consider real-world applications, where rescheduling is often required in case of unexpected issues that occur during task execution.

2.5 Summary

A comprehensive task and motion planning (TAMP) solution is essential to enable flexible and autonomous operation in single and multi-manipulator systems. The literature review on multi-manipulator systems reveals that current approaches often neglect online trajectory planning for manipulators or fail to address optimal task distribution. Current state-of-the-art planners, such as CHOMP, TrajOpt, PRM, and RRT, are still predominantly applied. However, these global planners generate the complete trajectory from an initial state to a desired target state in a single computation prior to execution, making them suitable primarily for offline trajectory planning. Replanning the trajectory is generally not possible in real-time. Recent studies [Krä+20; Mav+21; Tik+20; TGB22; TB24] have demonstrated the effectiveness of model predictive control (MPC) as an online trajectory planner for manipulators performing pick-and-place tasks in warehouse environments, fruit packing stations and interacting with human workers. MPC yields an intuitive formulation of the overall problem for online planning a feasible trajectory for a robot and anticipating the system's future behavior based on its dynamic model. However, the discussed control architectures lack modularity and fail to assess scalability, revealing limitations regarding the number of robots. The key challenges lie in achieving a collision-free operation for multiple robots working simultaneously and enabling them to respond to unforeseen events typical in real-world scenarios. The more a robot's environment gets sophisticated, the more constraints within the MPC formulation must be considered to guarantee safety. The bottleneck of MPC is its computational costs, which increase with the number of constraints. MPC's challenging factor is constructing an optimization problem that makes the trajectory planning problem solvable during runtime, i.e., *online* at

every control cycle. Online execution of the trajectory planning requires a valid trajectory at every time step [Ard+15]. However, one should be aware of situations where the optimization problem might become infeasible, sensors fail to provide the current state of the considered system, or the reference/target state during trajectory execution has changed [KW10].

Despite significant advancements in vision, planning, and control over the past two decades, robotic manipulation has not yet achieved full autonomy in unstructured environments within industrial applications. So far, task allocation problems have been predominantly solved using heuristic methods mainly due to two reasons. First, task coordination is carefully planned to avoid conflicts between robots during operation. Second, as the number of workpieces and associated tasks increases, the task allocation problem becomes increasingly complex to solve [Har+23]. The existing scheduling procedures can only be deployed successfully in well-known environments, with the robots executing tasks perfectly without any time delays. Once a robot in a team needs more time to accomplish a task or does not find a path through a cluttered environment, the pre-computed sequence of tasks cannot be guaranteed to be the optimal one as it influences the overall system and can even lead to a considerable poor performance for the whole team. Rescheduling of the plan is then required.

Additionally, the existing TAMP approaches usually do not account for the occurrence of deadlocks. Research on deadlock resolution for multi-manipulator systems is limited. Most existing methods [AA14; ZA15; AAA23] still rely on artificial time delays in preplanned trajectories to handle deadlocks. However, this results in a deteriorated performance of the overall system, where coordination of robots becomes an additional issue in the preplanning step.

3 Online motion control for a single manipulator

The limitations of current approaches, as discussed in Sec. 2.5, need to be addressed when designing a task and motion planning (TAMP) control architecture. To tackle the challenges associated with online trajectory planning, model predictive control (MPC), as detailed in Sec. 2.3, presents a promising concept. MPC optimizes the system's performance and anticipates the system's future behavior to respond to environmental changes. While recent works [Mav+21; TGB22] demonstrate MPCs' potential for online trajectory planning, a detailed comparison of MPC-based trajectory generation with state-of-the-art planners is still lacking.

This chapter introduces and demonstrates the benefits of MPC-based online motion control for planning a robot's trajectory. Unlike existing methods introduced in Sec. 2.4.3, the task allocation problem is solved by formulating an optimization-based scheduling model. The method allows for rescheduling in case modifications within the robot's workspace occur. The chapter also presents an experimental study focusing on a single manipulator system, evaluating the effectiveness of the MPC approach in simplified industrial scenarios, including a sorting task, a narrow passage problem, and a dynamically changing environment. The experimental results are analyzed and compared with selected state-of-the-art planners, offering a novel contribution to the existing literature. The contents of this chapter have been previously published in [Gaf+22c].

3.1 Assumptions

The primary distinction between the two fundamental types of robot motion control is that sensor-guided motion control relies on sensors to safely achieve the target state while trajectory-following control executes a predetermined trajectory through low-level joint control without using sensor feedback. Sensors are employed to perceive the robot's environment and detect environmental changes. Ideally, both control types should be integrated, seamlessly switching between them. However, commercially available robotic systems predominantly offer trajectory-following control, which focuses on minimizing the error between the actual and desired robot positions to adhere to a specified trajectory.

Consequently, these systems are inadequate for operating in dynamically changing environments, necessitating high-level control strategies [KW10].

Fig. 3.1 schematically illustrates a sensor-guided motion control structure introduced by Torsten Kröger [Krö10]. Sensor-guided robot motion relies on sensor signals that monitor the robotic system and its environment. For example, vision systems deliver signals $\mathbf{s}(t)$ within the robot's task space. Tasks are naturally defined in this task space. Conversely, a manipulator operates in joint space, where it provides its current configuration $\mathbf{q}(t)$, angular velocities $\dot{\mathbf{q}}(t)$, and accelerations $\ddot{\mathbf{q}}(t)$. To command the desired configuration and its derivatives $\ddot{\mathbf{q}}_\mathrm{d}(t)$, $\dot{\mathbf{q}}_\mathrm{d}(t)$ and $\mathbf{q}_\mathrm{d}(t)$ to the joints' actuators, inverse kinematics is employed to convert the desired setpoints $\ddot{\mathbf{p}}_\mathrm{d}(t)$, $\dot{\mathbf{p}}_\mathrm{d}(t)$ and $\mathbf{p}_\mathrm{d}(t)$ from task space to joint space. Inverse kinematics involves non-linear functions, which can lead to singularities and non-unique solutions [SHV06]. Consequently, a joint space controller is often used to avoid the complexities of inverse kinematics. This approach is preferred because constraints such as maximum angles, angular velocities, and robot dynamics can be more naturally expressed in joint space.

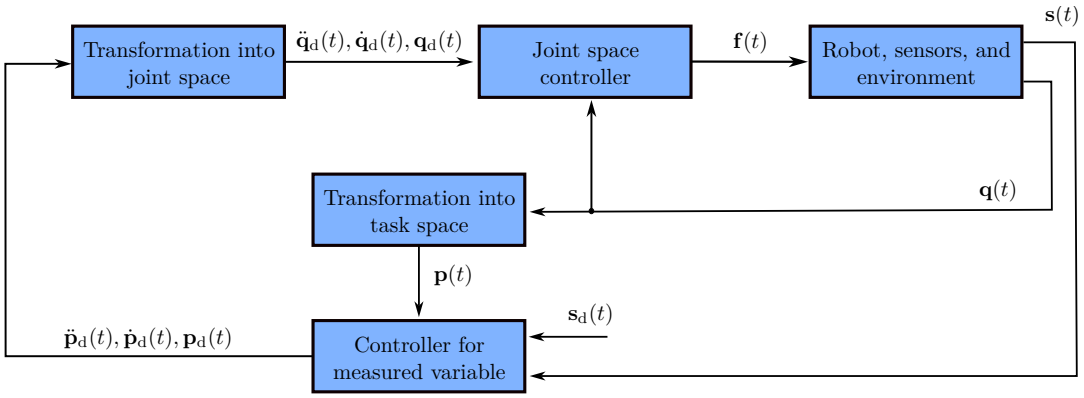


Figure 3.1: Sensor-guided robot motion control scheme introduced by Torsten Kröger [Krö10].

To verify whether the robot has reached its commanded setpoint, its pose is obtained in task space $\mathbf{p}(t)$ by applying forward kinematics. Forward kinematics establishes the relationship between the individual joints of the robot, represented as $\mathbf{q}(t)$ in configuration space, and the end-effector pose $\mathbf{p}(t)$ in task space [SHV06]. Additionally, forward kinematics can be utilized to determine the poses of individual robot links in task space, thereby enabling effective collision avoidance between these links and the robot's surrounding environment. Forward kinematics does not suffer from singularities and non-uniqueness of solutions but is associated with non-linearity.

Prior to realizing an optimal task and motion control approach for a single manipulator system, the following assumptions are met:

1. The start and goal configurations are given.
2. The start and goal configurations are collision-free.

3. The robot starts at zero velocity.
4. The goal state is allowed to change during the robot's motion.
5. The robot performs a point-to-point motion.
6. Trajectories are not specified a priori.
7. No force exertion between the robot and its workpiece/object is considered.

3.2 Experimental setup

An experimental study is conducted for a single manipulator performing a series of pick-and-place tasks in an environment preoccupied with obstacles. A special focus lies in investigating the performance of the MPC in static environments, in narrow passages and in a dynamically changing environment. The experimental setup, shown in Fig. 3.2, consists of the UR3 manipulator from *Universal Robots*, placed on top of a module (table) and equipped with a parallel Co-act EGP-C 40-N-N-URID[1] gripper from Schunk. The workspace of the robot is occupied by several products, two obstacles and two trays. Each tray contains three slots. The goal of the robot is to place products into the assigned slots while robot's workspace is exposed to changes. For the experimental setup, a single-lens camera ace acA1280-60gc from Basler is employed delivering 60 frames per second at 1.3 MP resolution.

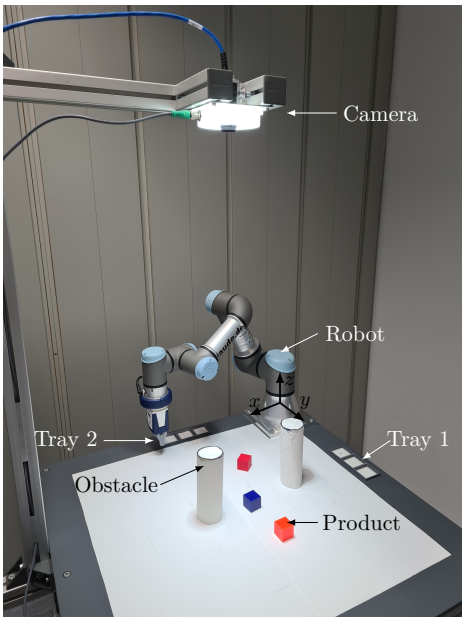


Figure 3.2: Experimental setup [Gaf+22c].

Three different product classes, which differ in terms of color (red, blue and orange), are considered. Each product class has a specific tray to which it can be assigned to: red products can only be placed in Tray 1 and blue products in Tray 2. The orange products can be assigned to both trays. The products considered in this setup are cubes of an edge length $l_{\text{cube}} = 30$ mm. The static obstacles in the workspace present the module and two cylinders that might be placed arbitrarily in robot's workspace. The cylinders have the height $h_{\text{cyl}} = 170$ mm and the diameter $d_{\text{cyl}} = 70$ mm. Note that the robot performs

[1] https://schunk.com/de/de/greiftechnik/parallelgreifer/co-act-egp-c/c/PGR_3995 (Last visited: 18.03.2024)

straight up and down movements to pick up or place the objects from predefined height $z_{\text{pick/place}} = 0.05$ m. For that, the setpoints are sent directly to robot's velocity tracking controllers.

The following control structure given in Fig. 3.3 is proposed to realize an optimal TAMP approach for a single manipulator system. Once modifications in the workspace have been conducted by an operator, e.g., a product has been moved to a new position, the *Camera* captures the current state in the form of RGB images of the robot's workspace and sends them to the *Object Detection* layer. The objects are detected and classified by the *Object Detection* algorithm. Subsequently, the current data $\underline{\mathbf{T}}_{\text{p}}$ of the products is sent to the *Scheduling* layer, and the data of the detected obstacles $\underline{\mathbf{T}}_{\text{obs}}$ to the *Trajectory Planner* layer. Based on the information about each product poses and the available slots, the scheduling algorithm computes an optimal sequence of tasks and sends a sequence of target setpoints $\underline{\mathbf{x}}^{W}$ to the robot. The trajectory can be either planned and executed online by sending the optimal control inputs $\mathbf{u}^{0*}$ at each control cycle or offline by sending the entire trajectory $\underline{\mathbf{u}}$, marked in a dashed line, to the robot's *Tracking Controller* layer. Both types of trajectory planning approaches will be investigated in Sec. 3.3.

The communication between different layers of the proposed online motion control architecture is established by using ROS Melodic [Sta18]. The scheduling problem is implemented in Julia [Bez+17] and solved using Gurobi [Gur23]. The MPC problem is formulated in CasADi [And+19], where the interior point solver IPOPT [WB06] is applied to solve the optimization problem. The optimal joint velocities computed at each sampling time by MPC are commanded via the velocity controller of Universal Robot ROS driver to the robot and current robot states are published to the controller. All of the developed algorithms run on a computer system unit equipped with an Intel Core i5-9600H at 2.9 GHz using 16 GB RAM under Ubuntu 18.04.

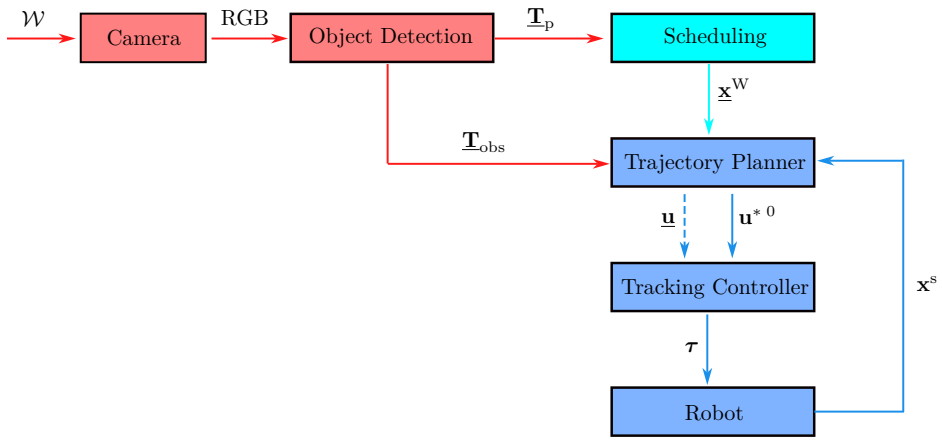


Figure 3.3: Control structure of an online task and motion planning approach for a single manipulator system.

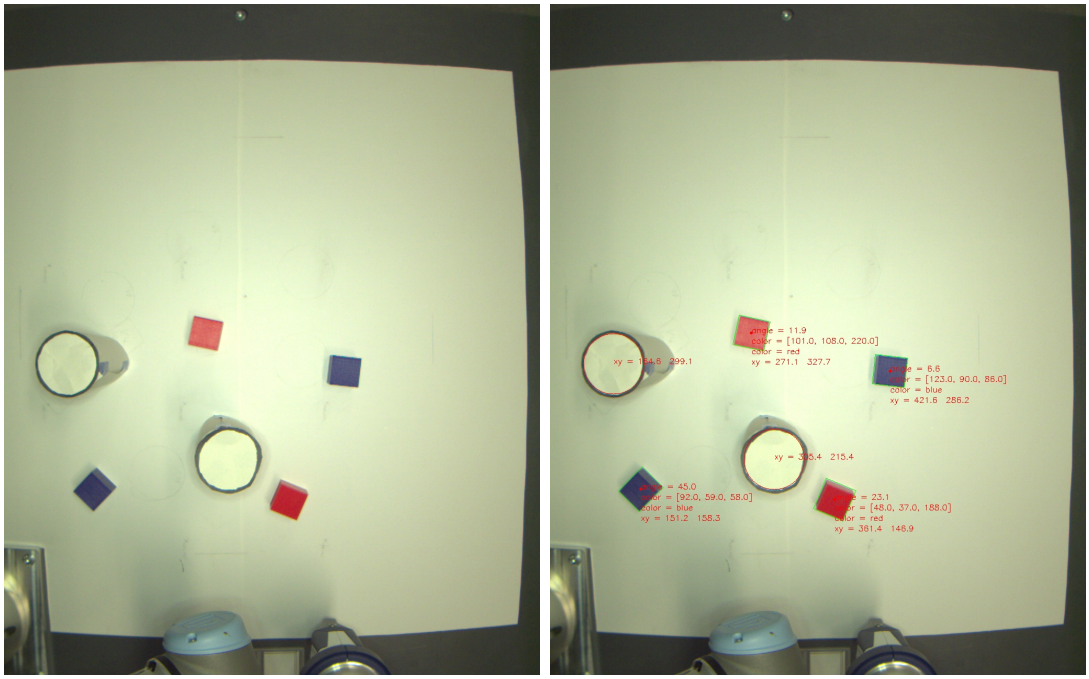
3.2.1 Object detection

A single-lens camera captures RGB images of the robot's workspace $\mathcal{W} \subset \mathbb{R}^3$ and sends them to the object detection algorithm. Note that the camera cannot provide depth information of the scene. The object detection algorithm detects different types of objects and classifies them into products or obstacles by their shape. For reasons of simplicity, two different shapes are considered, i.e., spherical objects are classified as obstacles and square objects as products. The algorithm determines the position, the orientation and the class of the detected products and stores the data in a tuple $\underline{\mathbf{T}}_{\mathrm{p}} \in \left(\mathbb{R}^2 \times [0, 2\pi[\times \Sigma\right)^{n_{\mathrm{p}}}$, respectively. The following classes of products are considered in the setup, defined as $\Sigma = \{\text{red, blue, orange}\}$. As the obstacles have a spherical shape, only the position is determined and stored in the vector $\underline{\mathbf{T}}_{\mathrm{obs}} \in \left(\mathbb{R}^2\right)^{n_{\mathrm{obs}}}$. The number of products and obstacles are denoted by n_{p} and n_{obs}, respectively. Fig. 3.4 shows the outputs of the single-lens camera and the object detection algorithm.

To correct the parallax error resulting from employing the single-lens camera, a linear transformation is applied

$$x_{\mathrm{e}} = \frac{H - h}{H}(x' - x_0) + x_0, \tag{3.1}$$

where x' stands for object's position captured by the camera and x_0 denotes the position of the camera along the x-axis. The camera is mounted at a certain height H. The height h of an object is assumed to be known.



(a) Capturing an RGB image from robot's workspace.

(b) Applying the object detection algorithm.

Figure 3.4: The outputs of the single-lens camera and the object detection algorithm.

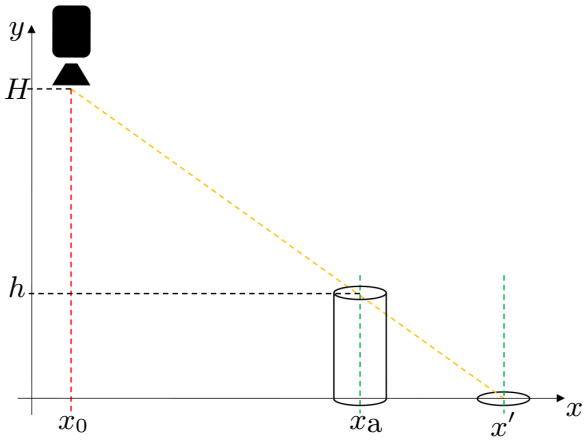


Figure 3.5: Parallax error correction [Gaf+22c].

3.2.2 Optimization-based scheduling model

The objective of the scheduling model is to assign objects to available slots in an optimal order, whereas the distance traveled by the robot's end-effector is minimized. The proposed scheduling model requires a set of objects $\mathcal{O}$ and a set of slots $\mathcal{S}$. Each object o can only be allocated to a subset of slots $\mathcal{S}_o \subseteq \mathcal{S}$ and only a subset of objects $\mathcal{O}_s \subseteq \mathcal{O}$ can be allocated to a slot s. Whether an object o is allocated to a slot s is expressed via the binary variable $Y_{os} \in \{0, 1\}$. Thus, the previous assumptions might be formulated by the following constraints: Each object o has to be allocated to exactly one slot

$$\sum_{\forall s \in \mathcal{S}_o} Y_{os} = 1, \ \forall o \in \mathcal{O} \tag{3.2}$$

and each slot can have at most one object allocated to it

$$\sum_{\forall o \in \mathcal{O}_s} Y_{os} \leq 1, \ \forall s \in \mathcal{S}. \tag{3.3}$$

It is assumed that the number of slots is greater than the number of objects ($|\mathcal{S}| \geq |\mathcal{O}|$), i.e. the constraint (3.3) is formulated as an inequality constraint.

In order to define a sequence, it is necessary to introduce the binary variables $F_o \in \{0, 1\}$ and $L_o \in \{0, 1\}$, where F_o denotes if object o is the first object in the sequence and L_o if object o is the last one. A sequence includes exactly one first and one last object. This is modeled by the following two linear constraints for the first

$$\sum_{\forall o \in \mathcal{O}} F_o = 1 \tag{3.4}$$

and the last object

$$\sum_{\forall o \in \mathcal{O}} L_o = 1. \tag{3.5}$$

For sequencing decisions, an immediate precedence-based binary variable $X_{oo'} \in \{0, 1\}$ is introduced, that specifies whether object o is grasped before object o'. Each object o has

exactly one successor, except for the last one,

$$\sum_{\forall o' \in \mathcal{O}} X_{oo'} = 1 - L_o, \ \forall o \in \mathcal{O} \tag{3.6}$$

and each object o' has exactly one predecessor, except for the first one,

$$\sum_{\forall o \in \mathcal{O}} X_{oo'} = 1 - F_{o'}, \ \forall o' \in \mathcal{O}. \tag{3.7}$$

The constraints (3.6) and (3.7) are not sufficient to eliminate subcycles in the sequence. Therefore, an additional continuous variable $C_o \geq 1$ is introduced, which represents the order of objects in the sequence. The variable C_o takes the value of 1, if object o is the first one in the sequence, i.e., if $F_o = 1$. This is expressed in the following constraint

$$C_o \leq 1 + |\mathcal{O}| \cdot (1 - F_o), \ \forall o \in \mathcal{O}. \tag{3.8}$$

If object o' is the successor of object o, then $X_{oo'} = 1$ holds and object o' fills the next position in the sequence, i.e., $C_{o'} = C_o + 1$. This is modeled as

$$C_{o'} \geq C_o + 1 - |\mathcal{O}| \cdot (1 - X_{oo'}), \tag{3.9a}$$

$$C_{o'} \leq C_o + 1 + |\mathcal{O}| \cdot (1 - X_{oo'}), \ \forall \ o, o' \in \mathcal{O}. \tag{3.9b}$$

If two objects are not in an immediate precedence relation, constraints (3.9) are trivially satisfied. The degrees of freedom of the scheduling problem are determined by the number of the decision variables specified for the optimization problem. All the decision variables introduced earlier are summarized in Table 3.1.

Table 3.1: Decision variables of the optimization-based scheduling model for a single robot.

Decision Variable	Type	Description
Y_{os}	Binary	Allocation of object o to slot s
F_o	Binary	First object
L_o	Binary	Last object
$X_{oo'}$	Binary	Immediate precedence variable takes value of one, if object o is picked directly before object o'
C_o	Continuous	Equal to an object's position within the sequence

Finally, the objective function penalizes the distance covered by the robot's end-effector. The scheduling model computes a distance-optimal solution that does not correspond to a time-optimal solution. For a time-optimal solution, information about the time required for a robot motion is needed. For that purpose, it is necessary to have knowledge about the robot dynamics. However, this would make the scheduling model intractable. Therefore, a linear motion of a robot is assumed. Besides that, the time needed to perform grasping is not considered. The objective function of the proposed approach consists of three parts. The first part represents the distance from the initial position of the end-effector to the first object. The second part measures the accumulated distance from the objects to be

picked to their allocated slots. The last part expresses the accumulated distance from the slots to the subsequent objects. The optimization problem for a single manipulator takes the following form

$$\min \quad \underbrace{\sum_{\forall o \in \mathcal{O}} F_o \cdot d_{or}}_{\text{Initial movement to the first object}} + \underbrace{\sum_{\forall o \in \mathcal{O}} \sum_{\forall s \in \mathcal{S}} Y_{os} \cdot d_{os}}_{\text{Movement from the picked object to its allocated slot}} + \underbrace{\sum_{\forall o \in \mathcal{O}} \sum_{\forall o' \in \mathcal{O} \setminus \{o\}} \sum_{\forall s \in \mathcal{S}} X_{oo'} \cdot Y_{os} \cdot d_{o's}}_{\text{Movement from a slot to the next object}}. \tag{3.10a}$$

$$\text{s.t.} \quad \text{Const. } (3.2) - (3.9). \tag{3.10b}$$

To evaluate the objective function, the distances between each object χ_o and the end-effector's initial position χ_e can be computed as following

$$d_{or} = \|\chi_o - \chi_e\|_2, \ \forall \ o \in \mathcal{O}. \tag{3.11}$$

Similarly, the distances between objects χ_o and slots χ_s can be determined as

$$d_{os} = \|\chi_o - \chi_s\|_2, \ \forall \ o \in \mathcal{O}, \ s \in \mathcal{S}. \tag{3.12}$$

Both distances d_{or} and d_{os} are pre-computed and cached.

3.2.3 Online trajectory generation using MPC

Consider a single manipulator assigned a series of pick-and-place tasks in an environment preoccupied with obstacles. Each task involves a point-to-point motion from an initial configuration $\mathbf{q}^{\mathrm{s}}$ to a goal configuration $\mathbf{q}^{\mathrm{f}}$. The robot's motion must be collision-free and smooth. These requirements can be addressed within the MPC framework as part of the planning stage. The following formulation of the MPC problem for a manipulator in the discrete-time form is proposed

$$\min_{\mathbf{u}^{0:N_{\mathrm{p}}-1}, \mathbf{x}^{0:N_{\mathrm{p}}}} \quad J^{\mathrm{f}}(\mathbf{x}^{N_{\mathrm{p}}}) + \sum_{k=0}^{N_{\mathrm{p}}-1} J^{\mathrm{c}}(\mathbf{x}^k, \mathbf{u}^k) \tag{3.13a}$$

$$\text{s.t.} \quad \mathbf{x}^{k+1} = \mathbf{A}^{\mathrm{d}}\mathbf{x}^k + \mathbf{B}^{\mathrm{d}}\mathbf{u}^k, \quad k = 0, \dots, N_{\mathrm{p}} - 1, \tag{3.13b}$$

$$\mathbf{x}^0 = \mathbf{x}^{\mathrm{s}}, \tag{3.13c}$$

$$\mathbf{x}^k \in \mathbb{X}, \quad k = 0, \dots, N_{\mathrm{p}}, \tag{3.13d}$$

$$\mathbf{u}^k \in \mathbb{U}, \quad k = 0, \dots, N_{\mathrm{p}} - 1, \tag{3.13e}$$

$$R(\mathbf{x}^k) \cap \mathcal{O} = \emptyset, \quad k = 0, \dots, N_{\mathrm{p}}, \tag{3.13f}$$

$$R(\mathbf{x}^k) \cap \mathcal{D}(k) = \emptyset, \quad k = 0, \dots, N_{\mathrm{p}}, \tag{3.13g}$$

comprising of a cost function and a set of constraints, which will be explained in the following.

Cost function

The main goal of the manipulator is to safely reach each given goal configuration. Note that the manipulator does not follow a given reference trajectory. To enforce the robot reaching the goal configuration, the deviation of the current state $\mathbf{x}^k$ to the desired state $\mathbf{x}^{\mathrm{f}} = [\mathbf{q}^{\mathrm{f}\,\mathrm{T}},\ \mathbf{0}^{\mathrm{T}}]^{\mathrm{T}}$ is penalized. Moreover, a smooth motion of the robot is desired to prevent unnecessary vibrations which can reduce lifespan of actuators. Note that the initial state $\mathbf{x}^0$ is set to be the measured state $\mathbf{x}^{\mathrm{s}}$ by the equality constraint (3.13c).

Consequently, the cost function (3.13a), consists of the stage cost $J^{\mathrm{c}}(\mathbf{x}^k, \mathbf{u}^k) : \mathbb{R}^{2N} \times \mathbb{R}^N \to \mathbb{R}$ and the terminal state cost $J^{\mathrm{f}}(\mathbf{x}^{N_{\mathrm{p}}}) : \mathbb{R}^{2N} \to \mathbb{R}$. The stage cost

$$J^{\mathrm{c}}(\mathbf{x}^k, \mathbf{u}^k) := (\mathbf{x}^k - \mathbf{x}^{\mathrm{f}})^{\mathrm{T}}\mathbf{Q}^x(\mathbf{x}^k - \mathbf{x}^{\mathrm{f}}) + \mathbf{u}^{k\,\mathrm{T}}\mathbf{R}^{\mathrm{u}}\mathbf{u}^k + \Delta\mathbf{u}^{k\,\mathrm{T}}\mathbf{R}^{\mathrm{d}}\Delta\mathbf{u}^k(T_{\mathrm{s}})^{-2} \tag{3.14}$$

penalizes the squared state error, control effort and control smoothness $\Delta\mathbf{u}^k = \mathbf{u}^k - \mathbf{u}^{k-1}$ with the positive definite weighting matrices $\mathbf{Q}^x \in \mathbb{R}^{2N\times 2N}$, $\mathbf{R}^{\mathrm{u}} \in \mathbb{R}^{N\times N}$ and $\mathbf{R}^{\mathrm{d}} \in \mathbb{R}^{N\times N}$, respectively. The terminal state cost punishes the terminal squared state error

$$J^{\mathrm{f}}(\mathbf{x}^{N_{\mathrm{p}}}) := (\mathbf{x}^{N_{\mathrm{p}}} - \mathbf{x}^{\mathrm{f}})^{\mathrm{T}}\mathbf{Q}^{\mathrm{f}}(\mathbf{x}^{N_{\mathrm{p}}} - \mathbf{x}^{\mathrm{f}}) \tag{3.15}$$

with the positive definite weighting matrix $\mathbf{Q}^{\mathrm{f}} \in \mathbb{R}^{2N\times 2N}$.

Mathematical model of the manipulator

MPC relies upon a mathematical model of the considered system to predict and optimize the future system behavior. In this case, a rigid body manipulator is considered with N joints. The dynamical model of a manipulator [SHV06] in the joint space can be described by Lagrange's equation in matrix form as

$$\mathbf{M}(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + \mathbf{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t))\dot{\mathbf{q}}(t) + \mathbf{g}(\mathbf{q}(t)) = \mathbf{Q}(t). \tag{3.16}$$

Here, the configuration $\mathbf{q}(t) = [q_1(t), \ldots, q_n(t)]^{\mathrm{T}} \in \mathbb{R}^N$ comprises the joint angles of the robot. The letter N specifies the number of degrees of freedom (DoF) of the dynamical system. The inertia matrix is represented by $\mathbf{M}(\mathbf{q}(t)) \in \mathbb{R}^{N\times N}$. The matrix $\mathbf{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t)) \in \mathbb{R}^{N\times N}$ maps the angular velocities $\dot{\mathbf{q}}(t)$ to Coriolis and centrifugal torques. The vectors $\mathbf{g}(\mathbf{q}(t)) \in \mathbb{R}^N$ and $\mathbf{Q}(t) \in \mathbb{R}^N$ represent the vectors of gravitational and generalized torques, respectively.

Inverse dynamics control is generally applied on nonlinear multivariable systems to follow a joint space trajectory. This control method allows for compensating the nonlinearity of

the dynamical system by applying a nonlinear state feedback law. More precisely, the task is to determine an auxiliary control input vector $\mathbf{u}(t) \in \mathbb{R}^N$ based on the manipulator's inverse dynamics to compensate the nonlinearity of the dynamical system. Applying a linearizing controller establishes a linear input/output relationship of the system with respect to the new input vector $\mathbf{u}(t)$. Consequently, the resulting system of the inner-loop control is linear and each joint's motion is independent of the other joints [CFK16; Sic+09].

In this work, it is assumed that the inner-loop control of the joint tracking controllers establishes a linear input/output relationship of the nonlinear system dynamics under control. In general, it is sufficient to model the manipulator's velocity tracking controllers as second-order system [Krä+20]. As a result, N decoupled, linear systems in state-space representation follow

$$\begin{bmatrix} \dot{\mathbf{q}}(t) \\ \ddot{\mathbf{q}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q}(t) \\ \dot{\mathbf{q}}(t) \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix} \mathbf{u}(t), \tag{3.17}$$

where the matrix $\mathbf{I} \in \mathbb{R}^{N \times N}$ specifies the identity matrix and $\mathbf{0} \in \mathbb{R}^{N \times N}$ denotes the zero matrix.

As the MPC formulation requires a discrete-time representation of the manipulator's dynamical system, the system (3.17) in terms of the state $\mathbf{x}(t) = [\mathbf{q}^{\mathrm{T}}(t),\ \dot{\mathbf{q}}^{\mathrm{T}}(t)]^{\mathrm{T}} \in \mathbb{R}^{2N}$ and control inputs $\mathbf{u}(t)$ are uniformly discretized at identical times t_k with $t_{k+1} - t_k = T_{\mathrm{s}}$. Subsequently, the following representation, formulated as equality constraint (3.13b) in the MPC problem, results

$$\mathbf{x}^{k+1} = \mathbf{A}^{\mathrm{d}}\mathbf{x}^k + \mathbf{B}^{\mathrm{d}}\mathbf{u}^k, \tag{3.18}$$

where $\mathbf{A}^{\mathrm{d}} \in \mathbb{R}^{2N \times 2N}$ denotes the discrete state matrix and $\mathbf{B}^{\mathrm{d}} \in \mathbb{R}^{2N \times N}$ the input matrix.

Kinematic constraints

Constraining instantaneous admissible motion of the robot is carried out by formulating kinematic constraints on states (3.13d) and inputs (3.13e). Consecutive robot links are not exposed to self-collisions, while all the other links might end up in collisions. To prevent self-collisions, lower and upper bounds for each joint angle can be defined. Further, to prevent unnecessary vibrations on actuators, the angular velocities are restricted as well, i.e., the robot state is constrained by specifying lower and upper bounds

$$\mathbb{X} := \{\mathbf{x}^k \in \mathbb{R}^N \mid \mathbf{x}_{\min} \leq \mathbf{x}^k \leq \mathbf{x}_{\max}\} \tag{3.19}$$

in the constraint (3.13d). Similarly, actuator torque limits are considered by constraining the torques applied to the robot's joints, i.e.,

$$\mathbb{U} := \{\mathbf{u}^k \in \mathbb{R}^N \mid \mathbf{u}_{\min} \leq \mathbf{u}^k \leq \mathbf{u}_{\max}\} \tag{3.20}$$

which is used for the constraint (3.13e) in the MPC problem (3.13).

Finally, to prevent collisions between the robot and its environment, collision-avoidance constraints (3.13f) are formulated. The collision avoidance strategy will be explained based on the Fig. 3.6a, which shows a simplified environment of the considered system from Fig. 3.2. The obstacles that the robot should avoid are the module (table) and the cylinder placed on top of it. As described in Sec. 2.3.3, the first step is to encapsulate the full body of the obstacles as well as of the robot by using geometric primitives. Fig. 3.6b illustrates the selected primitives to model the robot and its environment. To approximate the full body geometry of the robot, LSSs for each robot's link are used and modeled as a series of line segments described by the equation

$$\mathbf{s}_m(\mathbf{x}^k, \alpha_m) := \mathbf{b}_m(\mathbf{x}^k) + \alpha_m \mathbf{r}_m(\mathbf{x}^k), \quad \alpha_m \in [0, 1]. \tag{3.21}$$

This requires the forward kinematics of the manipulator to determine the base $\mathbf{b}_m(\mathbf{x}^k)$ and the direction of each robot's link $\mathbf{r}_m(\mathbf{x}^k)$. The parameter α_m restricts the length of the line segment to the length of the respective link. Note that the LSS forms a spherocylinder around each link of the robot.

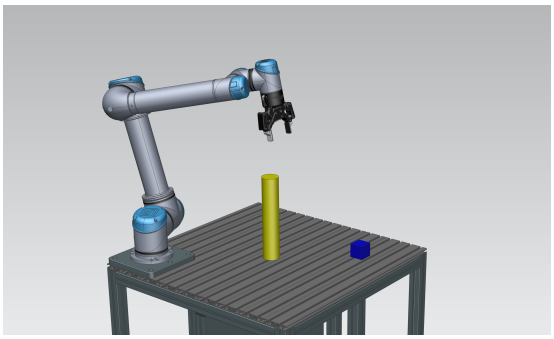
To prevent any collisions between the robot and the module (table), the intersection of the basis of each link $\mathbf{b}_m(\mathbf{x}^k)$ is restricted to the specified height along the z-axis, i.e.,

$$\mathbf{b}_m(\mathbf{x}^k) \geq \mathbf{z}_{\text{Table}} \tag{3.22}$$

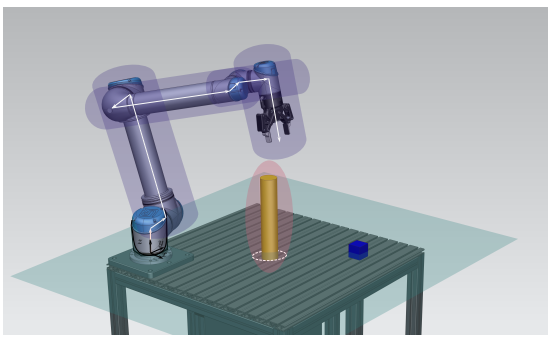
should hold. To encompass the geometry of the cylinder, an ellipsoid encapsulating the cylinder is chosen in the form

$$H(\mathbf{e}) = (\mathbf{e} - \mathbf{e}_0)^{\mathrm{T}} \mathbf{E} (\mathbf{e} - \mathbf{e}_0), \tag{3.23}$$

where $\mathbf{e}_0$ represents the center point of the ellipsoid, $\mathbf{e} \in \mathbb{R}^3$ describes a point on the



(a) Robot's environment.



(b) Representing the robot's environment using geometric primitives.

Figure 3.6: Utilizing geometric primitives to model the robot's environment for collision avoidance.

ellipsoid and the matrix $\mathbf{E}$ is given by the equation

$$\mathbf{E} = \text{diag}\left(\frac{1}{(h+s_\text{d})^2}, \frac{1}{(b+s_\text{d})^2}, \frac{1}{(w+s_\text{d})^2}\right) \in \mathbb{R}^{3\times 3} \tag{3.24}$$

containing the squared inverse principle semi-axes $h, b, w \in \mathbb{R}_{>0}$, standing for height, breadth and width of the cylinder, respectively. The parameter s_d represents a safety distance. To ensure no intersections between the robot and the cylinder, each link approximated by the line segment equation (3.21) should lie outside the ellipsoid occupied by the interior set of the cylinder, i.e.,

$$1 - H(\mathbf{s}_m(\mathbf{x}^k, \alpha_m)) \leq 0, \ \forall m \in \{1, \dots, N+1\}, \ \alpha_m \in [0,1]. \tag{3.25}$$

Following the same principle, collision avoidance of the robot with dynamic obstacles can be included into the constraints of the MPC problem (3.13). Note that the formulated collision avoidance constraints are non-convex resulting from the non-linearity of the forward kinematics.

3.2.4 Compensation of computation times

The number of active collision avoidance constraints change dynamically, as not all possible collisions might occur at a time step k. The number of active collision avoidance constraints considerably affects the computation times of the non-convex optimization problem (3.13). The computation times do not remain constant for each time step, instead T_c might even be greater than the sampling time T_s and therefore the optimal control input $\mathbf{u}^{*\,0}$ can not be obtained in time to be send to the robot's velocity tracking controllers. The larger the sampling time T_s, the smaller the influence of the computation time T_c on the DMPC. However, increasing the sampling time excessively can lead to tunneling, where collisions may be undetected.

The approach of Grüne and Pannek [GP17] proposed a solution to account for non-negligible computation times. Consider a single joint $q \in \mathbb{R}$, where the maximum computation times are greater than the sampling time T_s, i.e., $T_\text{c,max} > T_\text{s}$ holds. At time t_k, a new measurement q^k is received from the manipulator. Due to the fact that a new optimal control input might not be available at time t_k, the robot's controller is provided the optimal control input from the last optimization step for time t_k until $t_k + T_\text{c,max}$. Fig. 3.7 illustrates the concept for $T_\text{c,max} > T_\text{s}$, where the green line designates the optimal control inputs $[u^{*\,1} \ u^{*\,2}]$. Simultaneously, at time t_k, the state q^k is extrapolated into the future by integrating the manipulators dynamics $f(q(t), u(t), t)$

$$\hat{q}^k = q^k + \int_{t_k}^{t_k + T_\text{c,max}} f(q(t), u(t), t) dt. \tag{3.26}$$

With $\hat{q}^k$ as the initial condition, the optimization problem (3.13) is solved so that a new

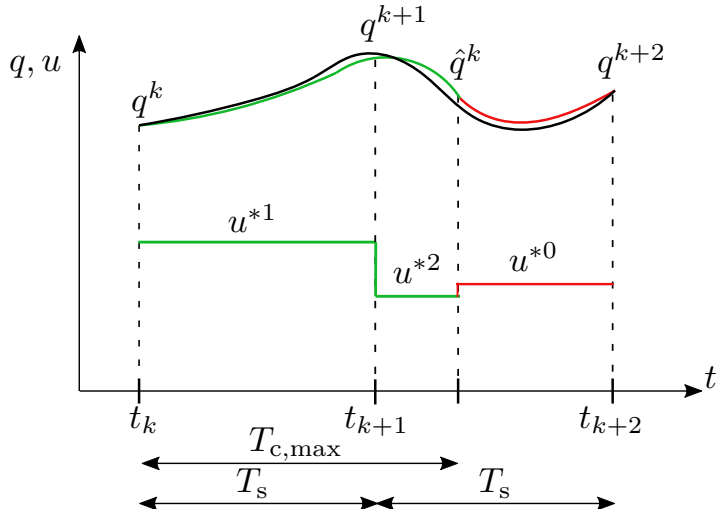


Figure 3.7: Compensation of computation times $T_{\mathrm{c,max}} \geq T_s$ for NMPC.

trajectory and control input sequence are available at the latest at time $t_k + T_{\mathrm{c,max}}$. At time $t_k + T_{\mathrm{c,max}}$ the first optimal control input $u^{*\,0}$, illustrated by the red line in Fig. 3.7, is sent to the robot. At time event t_{k+2} the MPC receives a new measurement q^{k+2} and the procedure is repeated. Thus, to compensate the maximum computation times, which can in some cases exceed the sampling time, an optimal control input sequence predicted into the future should be sent to the robot until the new optimal solution is available.

3.3 Experimental study

In the following section, an experimental study with the proposed control architecture of Fig. 3.3 is carried out for a single manipulator. Three different scenarios involving a simplified industrial production process subject to permanent modifications, narrow passage problem as well as a dynamically changing environment are carried out. To show the efficiency of the MPC approach, a comparative analysis is conducted with different state-of-the-art trajectory planners, which include single-query, multi-query sampling-based and optimization-based planner. The list of selected trajectory planners is provided in Table 3.2. Note, that the state-of-the-art planners plan the trajectory from an initial to a target state at once, prior to its execution and are considered as global planners. A detailed explanation is provided in Sec. 2.1.4 and Sec. 2.2. In contrast, MPC plans the robot's trajectory at each time step from its current to the target state and thus belongs to the type of local planners. The videos of the experiments are available at the following URL [Gaf+22b].

Table 3.2: List of selected trajectory planners and their characteristics.

Trajectory Planner	Category	Type	Description
PRM	Multi-query sampling-based	Global	Constructs a roadmap containing collision-free samples.
RRT-Connect (RRT-C)	Single-query sampling-based	Global	Tries to connect two trees by using a greedy heuristic.
BiTRRT	Single-query sampling-based	Global	Extension of RRT using stochastic optimization techniques.
CHOMP	Optimization-based	Global	Optimizes for a collision-free, smooth trajectory applying gradient descent.
MPC	Optimization-based	Local	Optimizes for a collision-free, smooth trajectory at each time step and predicts the robot's future behavior.

The exact parametrization of the MPC problem is provided in Table 3.3 for UR3 manipulator employed for the considered experimental setup.

Table 3.3: Selected parameters for the MPC problem (3.13) applied to UR3.

MPC Parameters	
Sampling time	$T_\mathrm{s} = 100$ ms
Prediction horizon length	$N_\mathrm{p} = 15$
Termination criterion (pose accuracy)	$\varepsilon_\mathrm{tol} = 4 \cdot 10^{-2}$ rad
Weighting matrices:	
- predicted states	$\mathbf{Q}_i^\mathrm{x} = \mathrm{diag}(1,1,1,1,1,1,1,1,1,0.1,0.1,0.1)^2$
- final state	$\mathbf{Q}_i^\mathrm{f} = 5\ \mathbf{Q}_i^\mathrm{x}$
- control inputs	$\mathbf{R}_i^\mathrm{u} = 0.1 \cdot \mathrm{diag}(1,1,1,1,1,1)\ \frac{\mathrm{s}^4}{\mathrm{rad}^2}$
- control input deviations	$\mathbf{R}_i^\mathrm{d} = \mathrm{diag}(1,1,1,0.1,0.1,0.1)\ \frac{\mathrm{s}^6}{\mathrm{rad}^2}$
Limits on joint velocities	$\dot{\mathbf{q}}_\mathrm{max/min} = \pm[\pi,\pi,\pi,2\pi,2\pi,2\pi]\ \frac{\mathrm{rad}}{\mathrm{s}}$
Limits on control inputs	$\mathbf{u}_\mathrm{max/min} = \pm[\pi,\pi,\pi,2\pi,2\pi,2\pi]\ \frac{\mathrm{rad}}{\mathrm{s}^2}$
Upper bound on joint angles	$\mathbf{q}_\mathrm{max} = [2\pi, 0, 0, \frac{1}{6}\pi, \pi, 2\pi]$ rad
Lower bound on joint angles	$\mathbf{q}_\mathrm{min} = [-2\pi, -\pi, -\frac{5}{6}\pi, -\frac{5}{6}\pi, 0, -2\pi]$ rad

3.3.1 Performance metrics

For the benchmark analysis, the following performance metrics are chosen to assess the quality of the generated trajectories: the path length, the execution time, the planning time, the trajectory smoothness and the safety distance. The distance covered by the end-effector is denoted as the path length L. The execution time T_e determines the time needed to execute a trajectory from the first to the last task of the robot. In case of global planners, the planning time T_p corresponds to the time to compute the complete trajectory before execution. In the context of MPC, the planning time refers to the average computation time required to solve a single MPC step. The trajectory smoothness $s = \int_0^{T_\mathrm{e}} ||\ddot{\mathbf{q}}(t)||_2 \mathrm{d}t$ is determined as the Euclidean norm of the integral of joint accelerations. Last, the safety distance of the robot to its environment is considered, to assess if safety requirements can be guaranteed with the proposed collision avoidance approach. The safety distance S_d is defined as the Euclidean distance between the robot's end-effector and the obstacles.

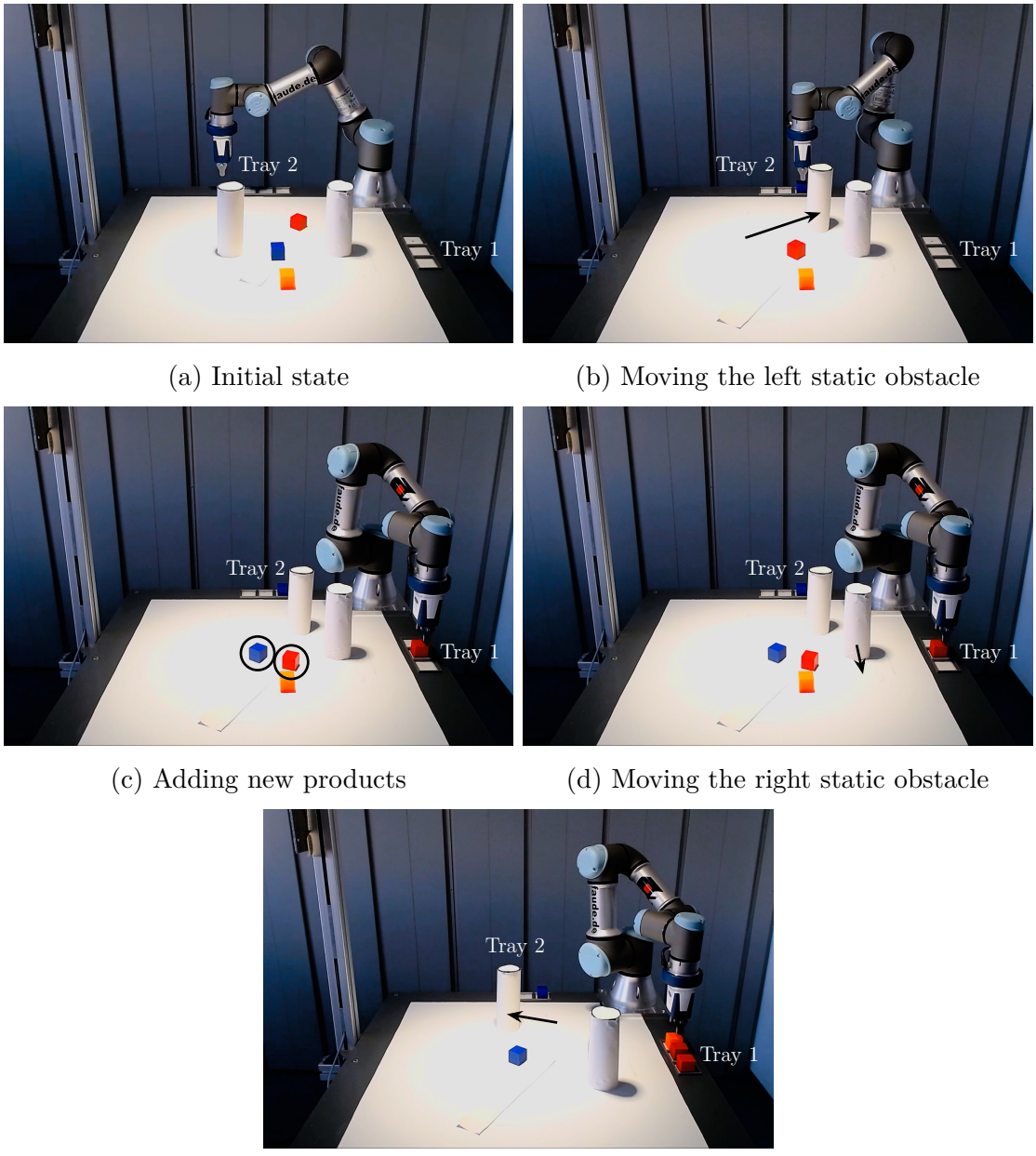
3.3.2 Sorting task

The following scenario demonstrates a basic industrial production process, where products and obstacles can be freely placed in robot's workspace or added by an operator. The following experiment investigates different types of modifications that can occur in a production process, such as adding new products, placing products or obstacles at new positions. The state-of-the-art planners require a static environment, i.e., no changes in robot's workspace are allowed to occur during the execution of robot's trajectory. Thus, to keep the environment static, any modifications in the workspace are conducted prior to the execution of the TAMP approach. Every change requires a rescheduling, where the scheduling plan is updated and new sequence of setpoints is sent to the robot. The scheduling algorithm, presented in Sec. 3.2.2, computes an optimal task sequence for a single robot and assigns the products to slots of the two trays.

The spatial arrangement of products and obstacles in the workspace of a simplified production process are depicted in Fig. 3.8, where all of the modifications in robot's workspace are highlighted. The initial state of robot's workspace is given in Fig. 3.8a that includes three products in red, blue and orange. Further, two obstacles are placed close to the products. After the first product in blue is sorted into Tray 2, the objects' setup is altered by moving the left obstacle and the product in red. Fig. 3.8b shows the previously mentioned modifications. Further, Fig. 3.8c shows the red product moved into Tray 1 and new red and blue products added to the workspace. Once the second product in red is placed into its assigned slot, the right static obstacle is placed between robot's current state and its new target state (orange product), presented in Fig. 3.8d. The final scene in Fig. 3.8e shows one last product in blue and the left obstacle, which is placed between the product and the tray.

The performance metrics, introduced in Sec. 3.3.1, are evaluated to assess the quality of the generated trajectories and are presented in Fig. 3.9. The results show that MPC and BiTRRT yield the shortest execution times and shortest pathlength compared to the other planners, see Fig. 3.9a and Fig. 3.9b. However, a short execution time does not necessarily correspond to a short pathlength. For instance, while CHOMP exhibits a shorter execution time than PRM, it produces the longest pathlength among the planners considered. Long pathlength can be attributed to far-reaching motions around the obstacles. The far-reaching motion planned by CHOMP is illustrated in Fig. 3.10b at one selected time step around a static obstacle and can be compared to the robot's current state computed using MPC. A closer examination of the figure shows a significantly shorter distance between the robot and the obstacle in the MPC case than CHOMP. Concerning trajectory smoothness, presented in Fig. 3.9c, MPC computed the smoothest trajectories, almost 50 % better than the considered state-of-the-art planner. The former might be attributed to the fact that state-of-the-art planners plan jerky movements close to obstacles. A direct comparison

between the planning time and the computation time of the state-of-the-art planners with MPC can not be performed since the entire trajectory is planned at once, from the initial to the target state. In contrast, MPC plans iteratively, once every time step. Fig. 3.9d reveals significant differences in the planning times among the state-of-the-art planners. In the case of PRM and CHOMP, it takes on average between $2 \cdot 10^3$ ms and over $5 \cdot 10^3$ ms to plan a path from the initial to the target state. In contrast, the planning times for RRTC and BiTRRT are significantly shorter and comparable to the computation times of MPC. The average computation times of MPC amount to 52.2 ms and thus lie well below the sampling time of $T_\mathrm{s} = 100$ ms.



(a) Initial state

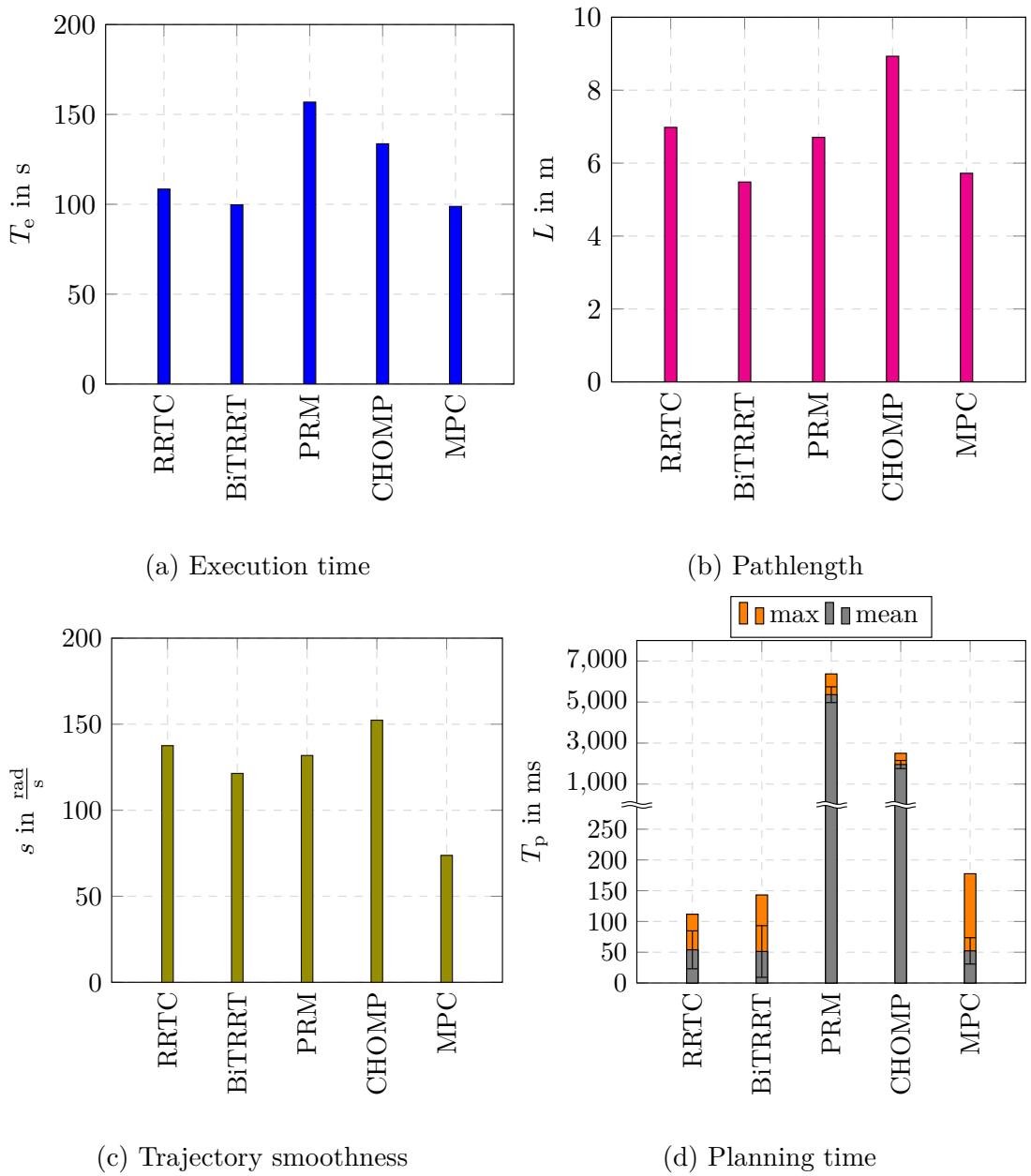
(b) Moving the left static obstacle

(c) Adding new products

(d) Moving the right static obstacle

(e) Moving the left static obstacle

Figure 3.8: Modifications to the experimental setup conducted during the sorting task [Gaf+22c]. The video of the experiment is available at the following URL [Gaf+22b].



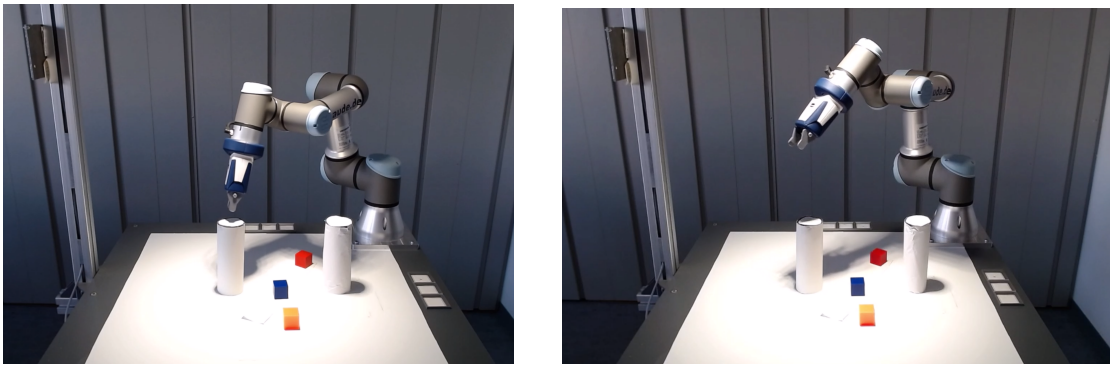
(a) Execution time

(b) Pathlength

(c) Trajectory smoothness

(d) Planning time

Figure 3.9: Evaluation of performance metrics for the sorting task [Gaf+22c].



(a) Robot's state around a static obstacle computed by MPC

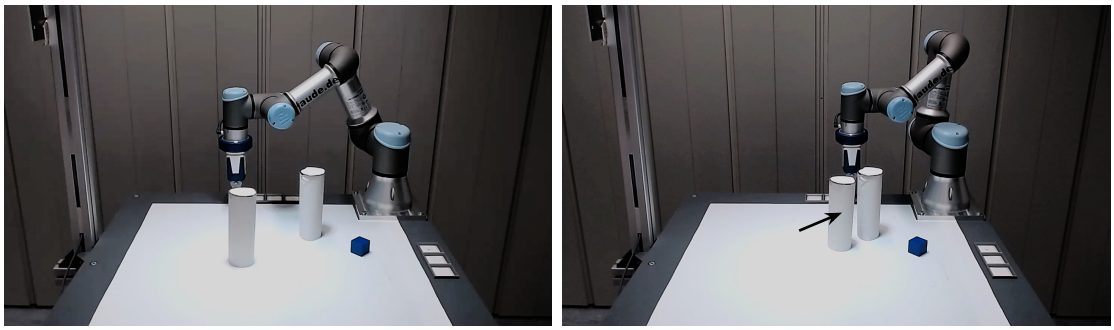
(b) Robot's state around a static obstacle computed by CHOMP

Figure 3.10: Selected time step illustrating the maneuvering around a static obstacle using MPC and CHOMP.

3.3.3 Narrow passage problem

In the following experimental setup, path planning through a narrow passage within a static environment is studied. By this scenario, it is investigated how well the state-of-the-art-planners and MPC plan trajectories in narrow passages and how potential evasive maneuvers influence the path length and execution time. To this end, two cylindrical obstacles forming a narrow passage are considered. The experiment consists of two investigations involving an open and a closed passage, depicted in Fig. 3.11. The first investigation, presented in Fig. 3.11a, allows the robot to pass through the narrow passage to grasp the blue product, which should be placed into the assigned slot of Tray 2. In the next investigation, illustrated in Fig. 3.11b, the distance between the obstacles is reduced, which does not allow the robot to pass through the passage. The product to be grasped by the robot is placed again behind the obstacles, which should also be placed into another slot of the same tray. The shortest path to the target in that case is passing over or around the obstacles.

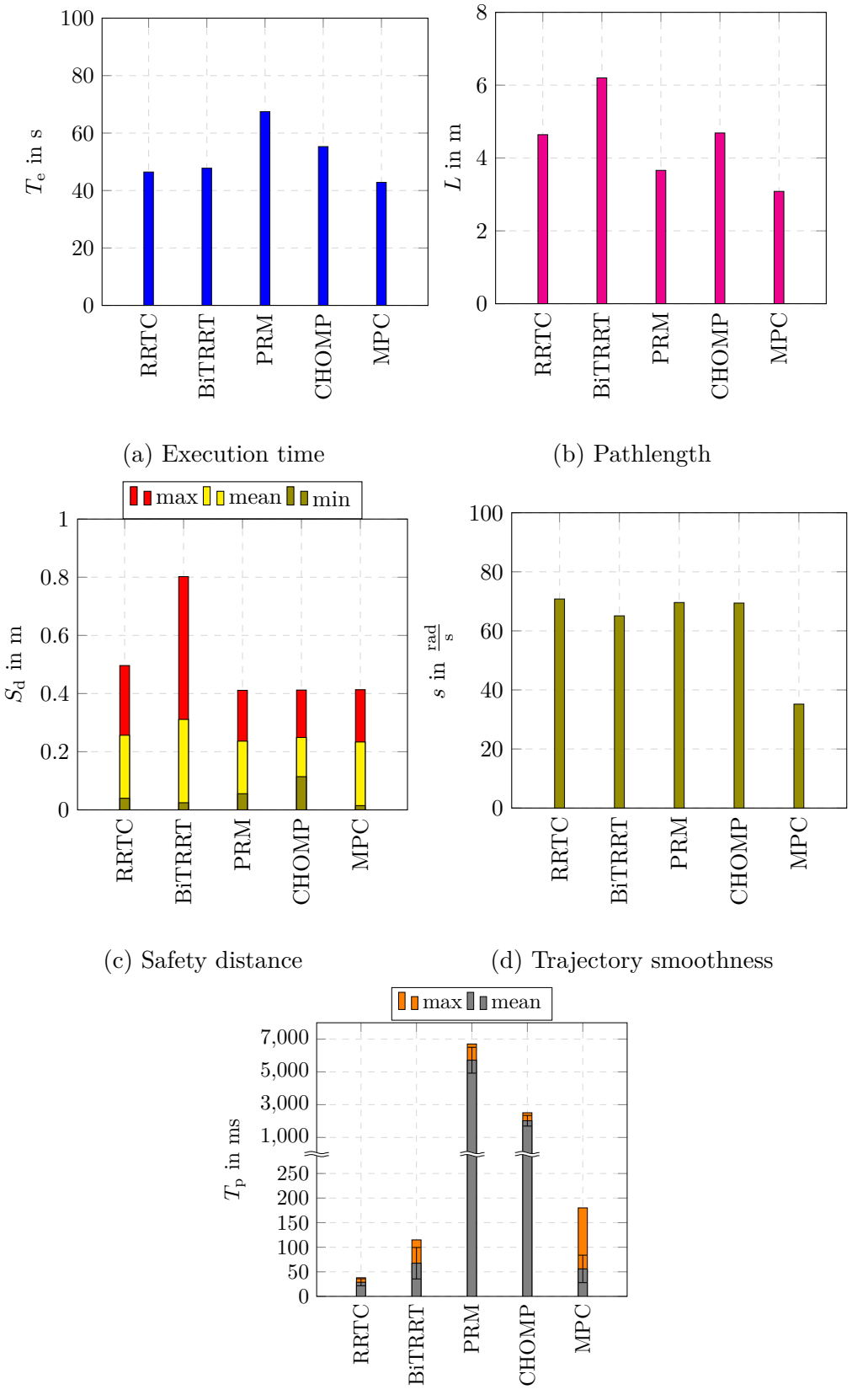
Assessing the experiment, the collision avoidance approach of each trajectory planner plays a crucial role in the robot's behavior in passing through the narrow passage. The performance metrics evaluated for the experiment are summarized in Fig. 3.12 including the evaluation of the safety distance S_d of the robot to the obstacles. The trajectories planned by PRM resulted in the longest execution times. This is attributed to the longest planning times in the case of PRM. Trajectories planned by other planners require 15 s to 20 s less in their execution. Although PRM needs more time to plan the trajectories, the results yield the shortest pathlength among the state-of-the-art planners. Notably, BiTRRT is the only global planner that computed a path through the narrow passage toward the target state, whereas all the other global planners chose a path above the open passage. However, Fig. 3.12b shows the longest pathlength in the case of BiTRRT, because all the following trajectories resulted in far-reaching and even jerky motions. Additionally,



(a) Narrow passage allows the robot to pass through.

(b) Closing the gap between the static obstacles.

Figure 3.11: Experimental setup presenting the narrow passage problem [Gaf+22c]. The video of the experiment is available at the following URL [Gaf+22b].



(a) Execution time

(b) Pathlength

(c) Safety distance

(d) Trajectory smoothness

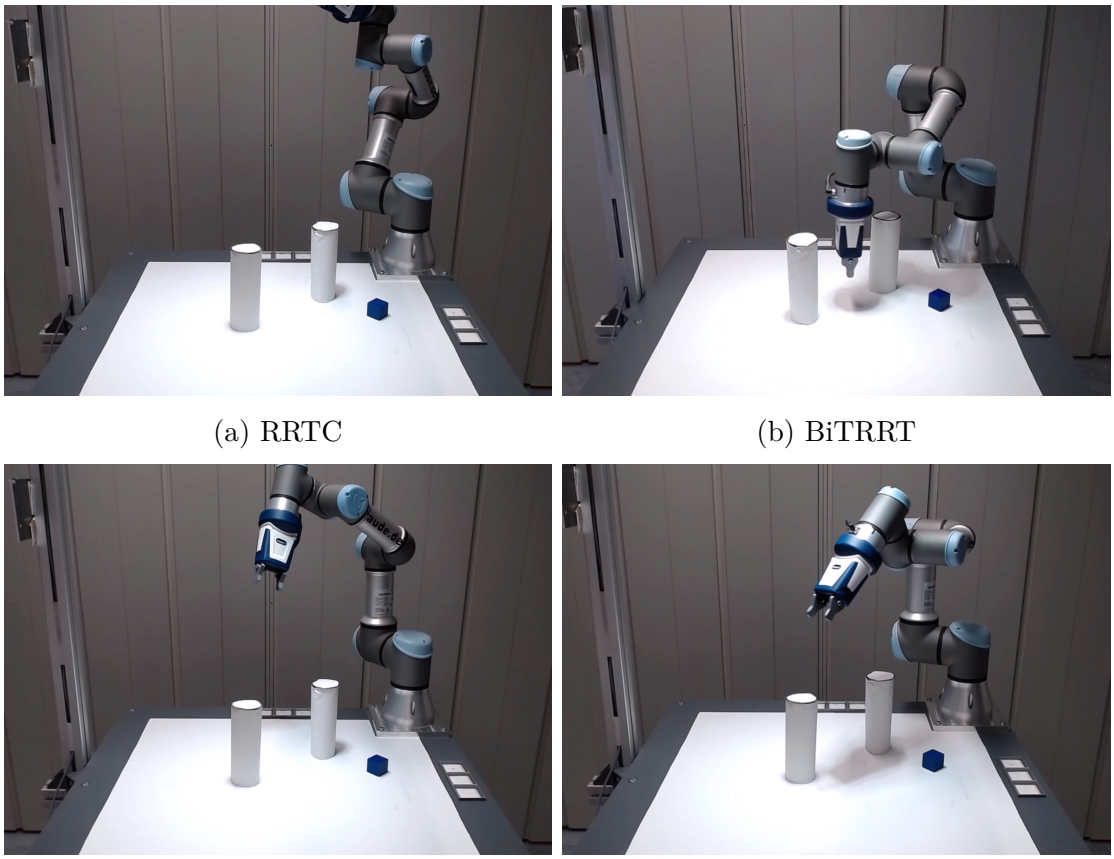
(e) Planning time

Figure 3.12: Evaluation of performance metrics for the narrow passage problem [Gaf+22c].

the robot overran the target state and had to return, resulting in the robot executing the additional path. A comparison of the robot's behavior passing the narrow passage for the state-of-the-art planners can be viewed in Fig. 3.13. An extreme, far-reaching motion was performed by the robot with RRTC while passing the open passage.

In contrast, MPC managed to compute the shortest path to the target through the narrow passage, which can be observed for the time instant $t = 2$ s in Fig. 3.14a. The follow-up motions are executed in close proximity to the obstacles, which can be observed in small values for safety distance. The safety distance S_d represents the values to the closest cylindrical obstacle, where the minimum, maximum and mean values have been evaluated. The far-reaching motions of BiTRRT can also be observed in the high maximum values of the safety distance. The smoothness of the executed trajectories yield no considerable difference between the global planners. MPC show significantly smooth execution of the trajectories.

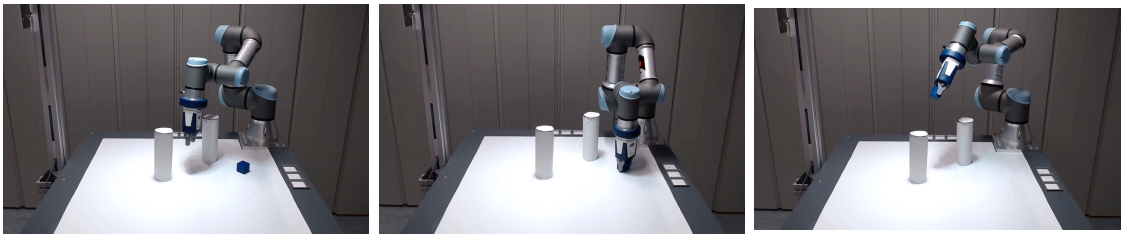
Distribution of the planning times is similar to the results from the previous experimental setup. PRM followed by CHOMP need more than $2 \cdot 10^3$ ms to plan a trajectory from the initial to the target state. On the contrary, RRTC and BiTRRT need comparatively



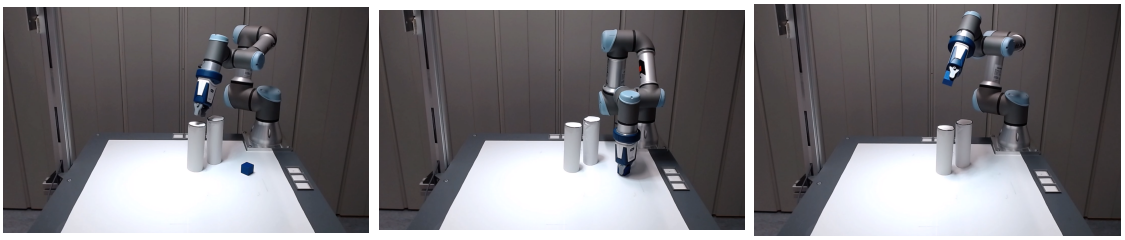
(a) RRTC (b) BiTRRT

(c) PRM (d) CHOMP

Figure 3.13: Snapshots from robot's state passing through the narrow passage planned by RRTC, BiTRRT, PRM and CHOMP.



(a) $t = 2$ s (b) $t = 6$ s (c) $t = 9$ s



(d) $t = 20$ s (e) $t = 23$ s (f) $t = 27$ s

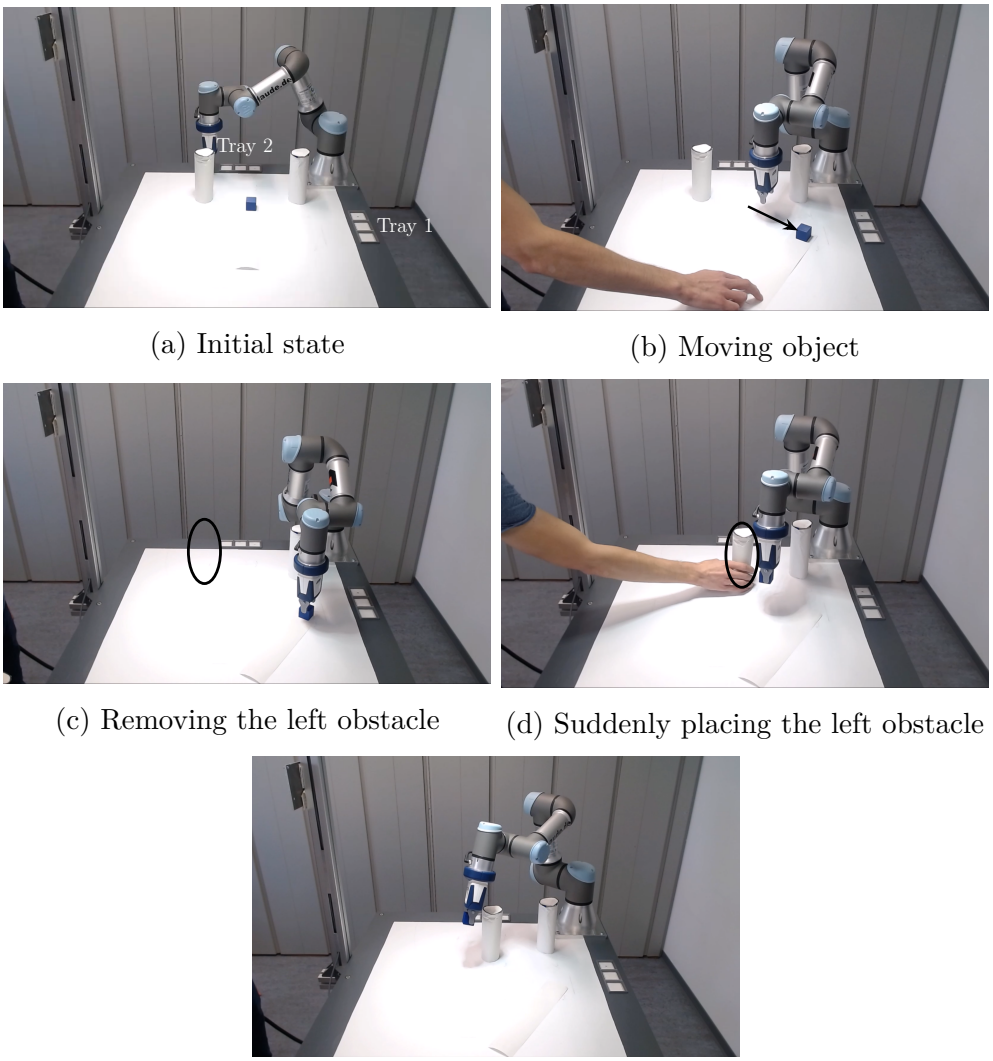
Figure 3.14: Selected time frames showing the robot executing trajectories planned by MPC.

short planning times between 40 ms and 70 ms. Considering the computation times of MPC, the mean value amounts to 56 ms and thus lies well below the sampling time of $T_\mathrm{s} = 100$ ms.

3.3.4 Dynamically changing environment

In the final experiment, the MPC approach is employed for a dynamically changing environment. The purpose of the following experiment is to investigate whether MPC is able to track a moving target and how fast it can react to a dynamically changing environment by placing abruptly an obstacle during task execution. The experimental setup given in Fig. 3.16 consists of two cylindrical obstacles and a blue product. At the beginning of the setup, illustrated in Fig. 3.15a, the product is placed in the middle of the two obstacles. Once the robot starts moving towards the target, the object is pulled out of the passage and moved continuously by the operator to the other end of the robot's workspace, preserving a constant speed. This can be observed in Fig. 3.15b. The object's position serves as a reference trajectory, which is transformed in the configuration space of the robot using inverse kinematics. Once the object has stopped moving, the robot is able to grasp it, see Fig. 3.15c. By a closer examination of Fig. 3.15c, the left obstacle is removed from the workspace of the robot. The slot assigned to the object is located in Tray 2 on the other side of the workspace. Thus, once the obstacle is removed, the robot is able to take the direct path to the assigned slot in Tray 2. However, during robot's motion towards the slot, the removed obstacle is abruptly placed by the operator in front of the robot, which can be seen in Fig. 3.15d. The final setup in Fig. 3.15e demonstrates, that the robot has to plan a path by taking two obstacles into account.

At the beginning, MPC leads the robot through the passage, formed by two cylinders. As the target starts moving, the robot follows it until it stops moving. This can be seen in the joint angles given in Fig. 3.16 until 12 s, where a continuous progression towards the object's pose can be observed. At time $t = 15$ s the joint angles q_1, q_3, q_4 and q_6 show a slight kink as at this time the robot deviates from its planned path not to collide with the obstacle that is placed abruptly in front of it. Especially, a significant deviation can be observed for joint angles q_2 and q_5. MPC needs 42.5 ms on average to solve the underlying optimization problem at each time step, which is far below the fixed sampling time $T_\mathrm{s} = 100$ ms. That means, the proposed approach with MPC is real-time capable for the considered dynamic environment.



(a) Initial state

(b) Moving object

(c) Removing the left obstacle

(d) Suddenly placing the left obstacle

(e) Two static obstacles

Figure 3.15: Steps demonstrating the conducted modifications during the experiment [Gaf+22c]. The video of the experiment is available at the following URL [Gaf+22b].

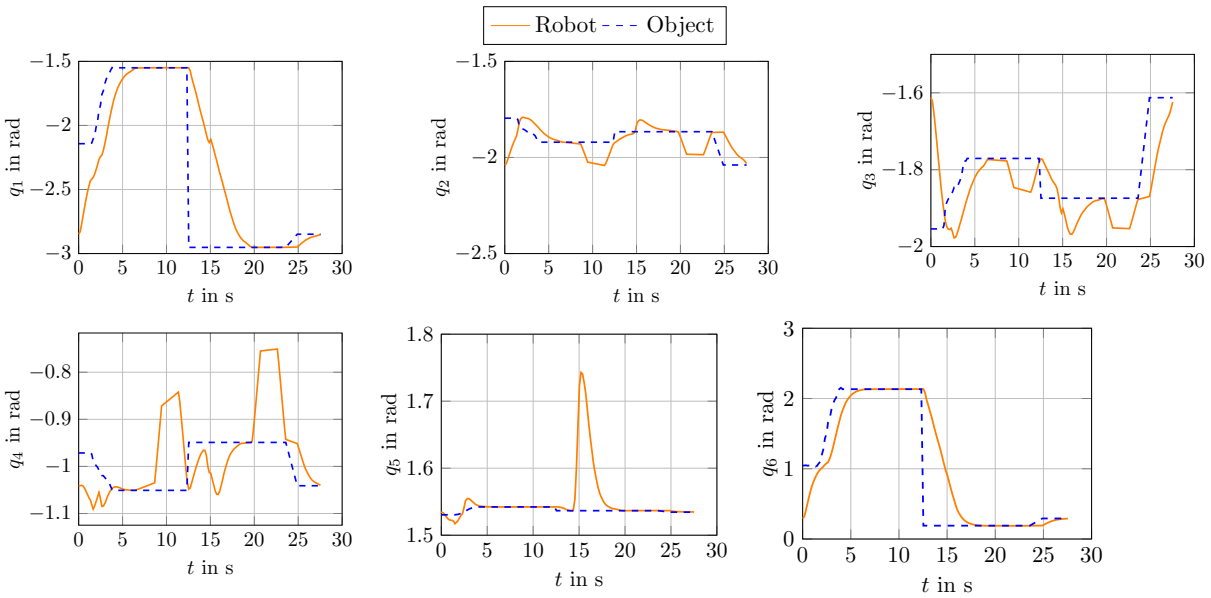


Figure 3.16: The joint angles of the robot [Gaf+22c].

3.4 Summary

The objective of this chapter was to demonstrate the advantages and significant potential of MPC for manipulators operating in real-world applications subject to environmental changes. To this end, various experiments were conducted and analyzed, comparing the performance of MPC with several state-of-the-art planners. The benchmark analysis revealed that MPC consistently outperformed global state-of-the-art planners across all selected performance metrics, even in static environments. The global planners tended to choose far-reaching motions near obstacles, resulting in non-intuitive and potentially hazardous maneuvers. Thus, in more complex environments with additional obstacles, it is essential to fine-tune the parameters of global planners to improve their performance. Similarly, minor environmental changes might significantly affect the performance of optimization-based global planners, which was the case with CHOMP. Consequently, considerable engineering effort is required to pre-tune the parameters of these planners for all considered setups. In contrast, MPC can be universally applied to complex static and dynamically changing environments. MPC ensures that the robot remains safely clear of obstacles without executing excessive maneuvers while maintaining a smooth motion. The following chapters will address the online motion control for multiple robots solving a joint task.

4 Online motion control for a multi-manipulator system

Given the promising results of using MPC for a single manipulator system, presented in Chapter 3, the next step involves enabling the simultaneous operation of multiple manipulators to accomplish a joint task. Fig. 4.1 depicts an illustrative example of a customized production process, where multiple robots work together simultaneously alongside human workers. Multiple manipulators can perform more complex tasks and enhance overall efficiency and reliability compared to a single robot. However, realizing a multi-manipulator system poses significant challenges, particularly in effective task coordination among the robots and the online planning of collision-free trajectories, as each robot represents a dynamic obstacle to the others. Recently, some promising concepts have been developed for handling HRI and multi-manipulator systems by applying MPC [Krä+20; TB24]. Centralized model predictive control (CMPC) has been proposed for two manipulators [Tik+20; TB24], offering the advantage of generating deadlock- and collision-free trajectories by solving the overall problem centrally, thereby eliminating conflicts between robots. However, centralized approaches limit system scalability, increase computational costs, hinder real-time operation, and impose a strong coupling between interacting systems. A distributed approach to trajectory planning should be explored to decompose the system into subsystems, thereby distributing computational efforts among the robotic systems. Moreover, literature on deadlock resolution in multi-manipulator systems is scarce. Ex-

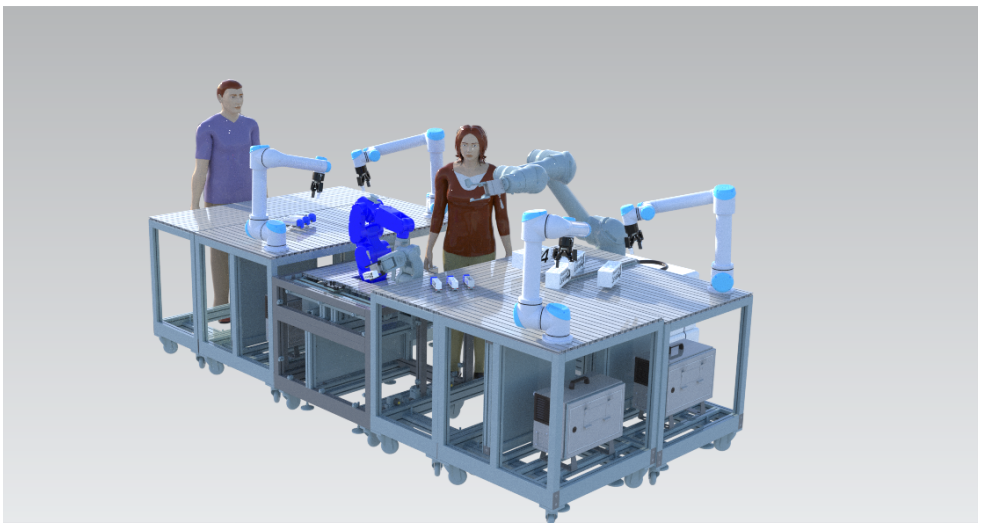


Figure 4.1: Multi-manipulator system involving human-robot interaction.

isting methods typically manage deadlocks by incorporating artificial time delays within preplanned trajectories [OL89; AA14; ZA15; AAA23], which results in a reduced overall system performance and complicates the robot coordination during preplanning.

This chapter introduces the theoretical background of the novel TAMP approach for a multi-manipulator system. The types of tasks considered here involve pick-and-place tasks, which can be found in a wide variety of industrial processes, such as bin-picking, material transfer, assembly, or disassembly. Pick-and-place tasks consist of a series of point-to-point motions, as the robot is required only to reach its target state without adhering to a predefined path. The proposed optimization-based TAMP approach is hierarchically structured and consists of several layers. The task allocation problem is formulated as a mixed-integer programming problem and solved by optimally distributing the subtasks among the robots to accomplish the overall goal in minimum time. Additionally, conflicting goals of the robots are excluded a priori to alleviate the problem of deadlocks. The motion planning problem is solved online to enable the reactive behavior of each robot in response to environmental changes during motion. The proposed method is based on distributed model predictive control (DMPC), where each robot plans its own trajectory by taking the trajectory of its neighbors into account. Considering the predicted motions of neighboring manipulators ensures that no inter-robot collisions occur and smooth interaction between multiple robots is possible. A reactive deadlock resolution approach is proposed to address deadlocks that may occur during a robot's motion, involving only the deadlock-affected robots in the resolution procedure. The contents of this chapter have been published in [GYR21; Gaf+22a]. The proposed TAMP approach is validated through extensive simulation-based and experimental studies presented in Chapter 5 and Chapter 6, respectively.

4.1 Requirements and assumptions

Fig. 4.2 shows an example of a simplified industrial environment where two manipulators cooperate to perform tasks typical for a customized production. A human worker might intervene in that process whenever assistance is required. The two manipulators should carry out their tasks simultaneously and autonomously, receiving only a common task: sorting. Overlapping workspaces allow the robots to cooperate, i.e., jointly sort the given objects placed in the individual robot's workspaces into corresponding stations. The complexity of this setup can be easily extended by adding even more robots, each having its subtask.

To realize an MRS, the following questions should be addressed, which were partly mentioned by LaValle in 2006 [LaV06]:

1. How should tasks within a group of robots be distributed?
2. How can collisions between the robots be avoided?
3. Should the coordination of the robots be centralized or decentralized?
4. Are robots allowed to communicate?
5. How can each robot replan its trajectory in case of environmental changes?
6. How can deadlocks reliably be detected and resolved?

Several requirements must be established before realizing an MRS to address the above-mentioned questions. First, an object detection system should be employed to monitor workpieces and detect any modifications within the workspace, such as the replacement or addition/removal of objects during runtime. This capability is crucial for rescheduling tasks in response to changes. Optimal sequencing and allocation decisions can significantly enhance the system's overall performance. However, reachability constraints between each robot and object must be considered, especially when repositioning objects. Limitations in reachability may necessitate handovers between robots, requiring intelligent coordination. Distributed schemes demand communication between robot controllers to ensure collision-free trajectories during operation. Furthermore, an online trajectory planning strategy should be implemented to respond effectively to environmental changes. However, online trajectory planning for individual robots can lead to conflicting goals within the team, potentially resulting in deadlocks at any point during operation. Thus, deadlocks must be reliably detected and resolved. Particular attention should be given to distinguishing between deadlock-affected and deadlock-free groups of robots. Interrupting the entire robotic team to resolve a deadlock would waste resources and degrade the overall system performance.

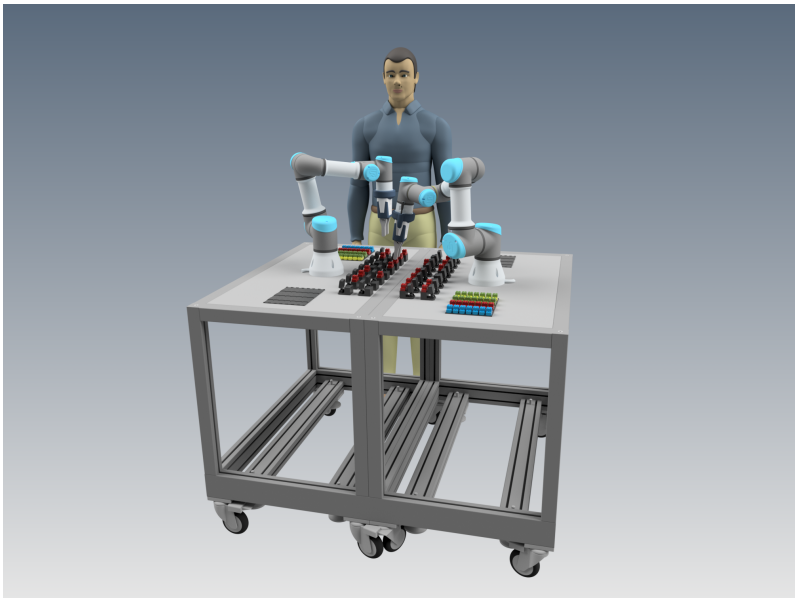


Figure 4.2: Two-manipulator setup involving human-robot interaction.

Prior to realizing a TAMP approach for a multi-manipulator system, the following assumptions are met:

1. The start and goal configurations are given to each robot.
2. The start and goal configurations of the robots are collision-free.
3. Each robot starts at zero velocity.
4. Goal state is allowed to change during the robot's motion.
5. The robot performs a point-to-point motion.
6. Trajectory of each robot is not specified a priori.
7. Communication between the robots is allowed.
8. A team of robots fulfills a joint task.
9. Each robot can operate in its entire workspace unless physical constraints, such as walls, tables, etc., restrict it.
10. No force exertion between the robot and its workpiece/object is considered.

4.2 Task and motion planning architecture

The following section introduces a novel control architecture based on MPC that enables the autonomous and simultaneous operation of a multi-manipulator system. The control structure of the proposed TAMP, illustrated in Fig. 4.3 and comprising several layers, will be explained in detail.

- **Object Detection System** First, an *Object Detection System*, such as a camera, monitors the common workspace of the robots, denoted as $\mathcal{W} = \bigcup_{i=1}^{M} \mathcal{W}_i$. Next, the object detection algorithm classifies the detected objects into two categories: parts that need to be manipulated by the robots and obstacles that must be avoided. The tuple $\underline{\mathbf{T}}_{\text{obj}} \in \left(\mathbb{R}^3 \times SO(3) \times \mathbb{R}^3 \times \Sigma\right)^{n_{\text{obj}}}$ summarizes the properties of the detected objects, including their positions, orientations, dimensions (length, width, and height), and classes. Here, n_{obj} represents the total number of objects.
- **Task Planner** The *Task Planner* layer represents the first layer of the proposed TAMP approach. This layer is application-dependent, with different types of objects assigned to predefined groups based on their properties, such as color, geometry, etc. The tasks may include sorting, assembly, or disassembly and serve as input to the *Task Planner* layer. For instance, in the case of a disassembly task, objects of the same type should be disassembled and sorted into the same tray. Therefore, all objects of this type are assigned to the same group.

- **Scheduling** Once different groups of objects are defined in the *Task Planner* layer, a *Scheduling* layer computes a sequence of subtasks for each robot. Optimal sequencing and allocation decisions can positively influence the system's overall performance. The input to the layer is a set of objects $\mathcal{O}$ occupying the workspace of one or multiple robots, depending on the composition, and a set of slots $\mathcal{S}$ to which the objects must be allocated. The *Scheduling* layer is responsible for distributing the objects between the robots and allocating them to the slots in an optimal or heuristic manner. A formulation for each type of scheduling problem is presented in Sec. 4.3. The output of the *Scheduling* layer is a sequence of goal poses $\underline{\mathbf{x}}_i^{\mathrm{W}} = [\mathbf{x}_1^{\mathrm{W}}, \dots, \mathbf{x}_Q^{\mathrm{W}}]$ in the workspace (Cartesian space) of robot i, where Q represents the number of setpoints. An optimal sequence of tasks should ensure that the joint goal assigned to the robots is achieved, e.g., all tasks are accomplished by all robots as quickly as possible. Furthermore, the task allocation problem may help alleviate the occurrence

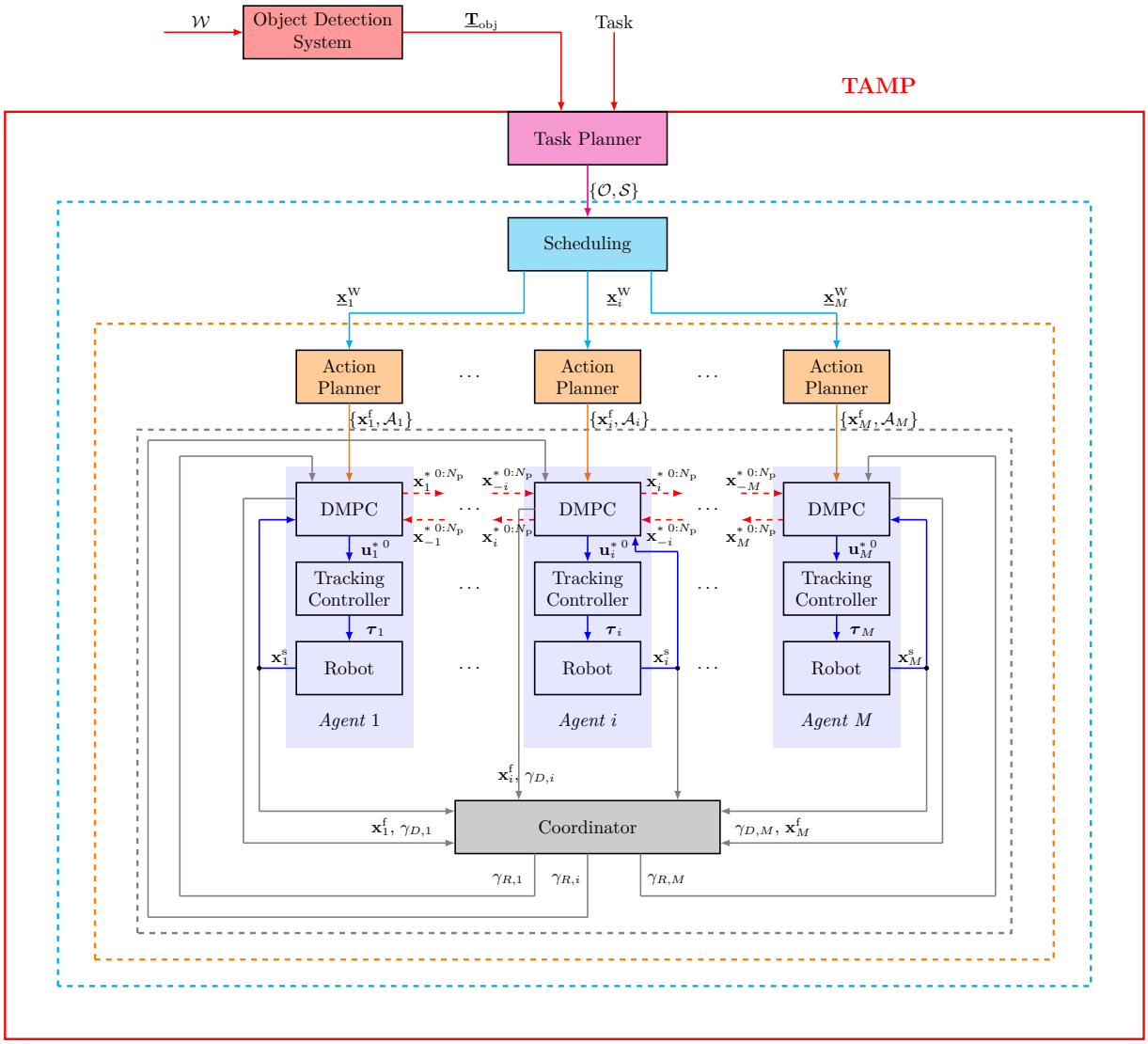


Figure 4.3: Control structure of an online task and motion planning for a multi-manipulator system.

of deadlocks by considering their reproducible occurrence a priori. For instance, objects lying too close to each other should not be assigned to the robots at the same time to avoid conflicts during simultaneous grasping.

- **Action Planner** In the next step, the *Action Planner* layer transforms the sequence of goal poses $\underline{\mathbf{x}}_i^{\mathrm{W}}$ from the Cartesian space into the configuration space $\underline{\mathbf{x}}_i^{\mathrm{f}} = [\mathbf{x}_1^{\mathrm{f}}, \dots, \mathbf{x}_Q^{\mathrm{f}}]$ of robot i using inverse kinematics. Additionally, this layer generates a sequence of actions $\mathcal{A}_i = [\alpha_1, \dots, \alpha_T]$ for each goal configuration $\mathbf{x}_i^{\mathrm{f}}$ of robot i. For example, a pick-and-place task, which is a component of the overall sorting task, involves moving to the object, grasping it, and then placing it into the designated slot of a tray. Consequently, each setpoint must be associated with a corresponding sequence of actions. Table 4.1 summarizes the various actions required for executing a pick-and-place task.

Table 4.1: Types of actions for a pick-and-place task.

Actions	Description
α_1	Move to the setpoint with an offset in height
α_2	Move down to the setpoint
α_3	Move up from the setpoint
α_4	Open gripper
α_5	Close gripper

Subsequently, two sequences of actions for a pick-and-place task are defined as an ordered set

$$\mathcal{A}_{\text{pick}} = \{(\alpha_1, \alpha_2, \alpha_5, \alpha_3)\}, \quad \mathcal{A}_{\text{place}} = \{(\alpha_1, \alpha_2, \alpha_4, \alpha_3)\}.$$

- **DMPC** The generated sequence of goal poses and the corresponding actions are subsequently forwarded to the robots' trajectory planners individually, denoted as the *DMPC* layer. Each trajectory planner utilizes DMPC to optimize the trajectory of its corresponding robot over a prediction horizon. Each robot cooperates with its neighbors by considering their respective motion. To achieve this, the robots exchange their current states and predicted trajectories, denoted as $\mathbf{x}_i^{*\,0:N_{\mathrm{p}}}$, where $i = 1, \dots, M$. Based on the robots' current states and the shared optimal trajectories of their neighboring robots, each *DMPC* layer computes the optimal control inputs.

- **Tracking Controller** The optimal control input for the first time step, $\mathbf{u}_i^{*\,0}$, is sent to the robots' underlying *Tracking Controller*. Each *Tracking Controller* layer computes the joint actuator torques, τ_i, which are applied to the robot's joints. Subsequently, each robot executes its motion and sends the current state, $\mathbf{x}_i^{\mathrm{s}}$, to its respective *DMPC* layer to calculate the next optimal control input for the *Tracking Controller* layer. This process involves computing a trajectory from the current state to the goal state in an iterative manner. In other words, each *DMPC* layer replans

its trajectory at each time step, allowing it to respond to changes. Further details about the *DMPC* layer are provided in Sec. 4.4.

- **Coordinator** Deadlocks lead to a blocking behavior of the robots, necessitating the intervention of a supervisory instance to resolve the deadlocks. This task is performed by the *Coordinator* layer. Each *DMPC* layer of the corresponding robot detects whether it is currently in a deadlock and sends its deadlock status, $\gamma_{\mathrm{D},i}$, to the *Coordinator*. Upon receiving the deadlock status from each robot, the *Coordinator* clusters each robot into either a deadlock-affected or a non-deadlock-affected group. To resolve a deadlock, the *Coordinator* sends an activation or deactivation status, $\gamma_{\mathrm{R},i}$, to each robot. The activation status allows a robot to resume its motion, while the deactivation status sends a robot away from its current goal to a neutral pose. It is important to note that the *Coordinator* has no knowledge of the system dynamics of a robot and therefore relies exclusively on the information exchanged by the robots. Further details about the *Coordinator* layer are provided in Sec. 4.5. The occurrence of deadlocks can be mitigated by incorporating certain constraints into the optimization problem of the scheduling approach in the *Scheduling* layer, as explained in more detail in Sec. 4.3.4.

Dashed markings in Fig. 4.3 represent different types of execution loops of the proposed control architecture. The inner marking in gray encompasses the *DMPC*, *Tracking Controller* and *Robot* layers, collectively denoted as an *Agent*, as well as the communication with the *Coordinator* layer. The inner control loop of each *Agent* is executed at every time step t_k to solve the DMPC problem. Bidirectional communication with the *Coordinator* layer for detecting and resolving deadlocks also occurs at every time step t_k. The subsequent control loop, marked in orange, is action-based. Once a sequence of actions, α_i, for a specific setpoint (goal state) $\mathbf{x}_i^{\mathrm{f}}$, is executed by a robot, the *Action Planner* layer sends the next setpoint along with the corresponding sequence of actions to an *Agent*. Finally, the outer control loop includes the *Scheduling* layer, which is executed only if a new object has been added into the workspace or the position of existing objects has changed. In such cases, the *Task Planner* updates the groups of objects by adding or removing items, necessitating rescheduling to successfully complete the task.

4.3 Multi-robot scheduling

This section introduces a heuristic and optimization-based model for distributing tasks within a group of robots. The type of tasks considered here is a series of pick-and-place tasks required for material transfer. Later, a heuristic scheduling model is utilized to show the benefits of the optimization-based scheduling model [GYR21]. Compared to the heuristic scheduling model, the optimization-based framework allows the incorporation of

several constraints to solve a complex multi-robot task allocation (MRTA) problem. The proposed optimal approach [GYR21] is based on a mixed-integer bilinear programming (MIBLP) problem. The objective function to be minimized is the maximum completion time needed by all robots to complete the tasks. Reproducible deadlocks, prevalent in pick-and-place tasks, are treated by introducing additional constraints into the optimization problem. Further, the approach is extended to treat assembly or disassembly tasks, where a specific order of objects needs to be enforced.

4.3.1 Definitions

In the following, sets and subsets for the multi-robot scheduling model are introduced. Let M represent the number of manipulators and let $\mathcal{M} = \{1, \dots, M\}$ denote the index set of manipulators. The objective of the multi-robot scheduling is to allocate each object $o \in \mathcal{O}$ to manipulator $i \in \mathcal{M}$ and slot $s \in \mathcal{S}$. Each tray $\tau \in \mathcal{T}$ consists of a subset of slots $\mathcal{S}_\tau \subseteq \mathcal{S}$. The subset $\mathcal{O}_\tau \subseteq \mathcal{O}$ contains all objects that must be placed into tray τ, whereby the subset $\mathcal{S}_o \subseteq \mathcal{S}$ describes all slots the associating object o might be allocated to. Correspondingly, the subset of objects that can be allocated to slot s is denoted by $\mathcal{O}_s \subseteq \mathcal{O}$. All manipulators that can reach slot s are designated with the subset $\mathcal{M}_s \subseteq \mathcal{M}$. The reachability of slot s by manipulator i depends on the robot's range ρ_i, the Cartesian coordinates of the manipulator's base χ_i^b and the Cartesian coordinates of slot χ_s. Thus, the subset $\mathcal{M}_s \subseteq \mathcal{M}$ is defined as following

$$\mathcal{M}_s = \left\{ i \in \mathcal{M} \mid \|\chi_i^b - \chi_s\|_2 \leq \rho_i \right\}. \tag{4.1}$$

Reversed, the subset $\mathcal{S}_i \subset \mathcal{S}$ denotes all slots reachable by manipulator i and is defined as

$$\mathcal{S}_i = \left\{ s \in \mathcal{S} \mid i \in \mathcal{M}_s \right\}. \tag{4.2}$$

Manipulators that can reach an object o comprise a subset $\mathcal{M}_o \subseteq \mathcal{M}$. Therefore, the distance between the robots' bases and object o should be considered. Further, it must be guaranteed that a slot is reachable by a robot to which the object can be allocated,

$$\mathcal{M}_o = \left\{ i \in \mathcal{M} \mid \|\chi_i^b - \chi_o\|_2 \leq \rho_i \ \wedge \ \mathcal{S}_o \cap \mathcal{S}_i \neq \emptyset \right\}. \tag{4.3}$$

All objects that can be reached by manipulator i are contained in the subset $\mathcal{O}_i \subseteq \mathcal{O}$, which is defined as

$$\mathcal{O}_i = \left\{ o \in \mathcal{O} \mid i \in \mathcal{M}_o \right\}. \tag{4.4}$$

The distance between all objects and the initial location of each manipulator's end-effector,

$$d_{oi}^0 = \|\chi_o - \chi_i^e\|_2, \ \forall \ o \in \mathcal{O}, \ i \in \mathcal{M}, \tag{4.5}$$

and between all objects and slots d_{os}, defined in (3.12), are computed beforehand.

4.3.2 Heuristic scheduling model

The proposed heuristic scheduling model assigns the objects to the robots based on their distance to the end-effectors' initial positions, which is computed using equation (4.5). In case object $o \in \mathcal{O}$ is only accessible by a single robot, this object is assigned to that robot. In case multiple robots are able to pick up an object, then the object is assigned to the robot containing the least allocated objects so far to guarantee a balanced utilization of available resources. If the available robots have the same number of assigned objects, the current object will be assigned to the nearest robot. Concerning the allocation of objects to slots, a first-come-first-served heuristics is used. The heuristic-based scheduling model does not consider minimal distances for grasping and placing of objects.

Algorithm 1
Heuristc Scheduling

Initialize: $\text{AssignedObjects}_i \leftarrow \{\},\ \forall i \in \mathcal{M}$
for $o \in \mathcal{O}$ **do**
  if $|\mathcal{M}_o| = 1$ **then**
    $i \leftarrow \mathcal{M}_o$
    $\text{AssignedObjects}_i \leftarrow \text{AssignedObjects}_i \cup \{o\}$
  else if $|\text{AssignedObjects}_j| = |\text{AssignedObjects}_{j'}|,\ \forall j, j' \in \mathcal{M}_o$ **then**
    $i \leftarrow \arg\min_{i \in \mathcal{M}_o} d^0_{oi}$
    $\text{AssignedObjects}_i \leftarrow \text{AssignedObjects}_i \cup \{o\}$
  else
    $i \leftarrow \arg\min_{j \in \mathcal{M}_o} |\text{AssignedObjects}_j|$
    $\text{AssignedObjects}_i \leftarrow \text{AssignedObjects}_i \cup \{o\}$
  end if
end for

4.3.3 Optimization-based scheduling model

In the following, the scheduling model for a single robot, introduced in Sec. 3.2.2, is extended to multiple robots. The goal of the scheduling model is to optimally allocate a set of objects to multiple manipulators and slots. Therefore, the precedence relations between objects is enforced for each robot individually. Besides that, not all objects and all slots might be reachable by all robots, i.e., not all objects can be allocated to all slots and should be optimally distributed among available manipulators. Therefore, a number of distinct trays reachable by a number of manipulators and comprising a required number of slots is considered. For instance, three types of objects should be sorted into three trays by two manipulators. The first requirement is that each tray should contain at least the number of slots equal to the number of objects. The second requirement would be that

only objects reachable by manipulators can be sorted into corresponding trays, that in turn should be reachable by the serving manipulator.

The MIBLP problem comprises several linear equality and inequality constraints. In order to ensure that all objects are served by manipulators, a decision variable W_{oi} is defined. Each object has to be assigned to exactly one manipulator, which is expressed by the following equality constraints

$$\sum_{\forall i \in \mathcal{M}_o} W_{oi} = 1, \ \forall \ o \in \mathcal{O}. \tag{4.6}$$

Not all objects and slots are reachable by all manipulators. Setting the corresponding decision variable to zero

$$W_{oi} = 0, \ \forall \ o \in \mathcal{O}, \ i \in \mathcal{M} \backslash \mathcal{M}_o \tag{4.7}$$

prevents the allocation of manipulators to non-accessible objects.

The same constraints (3.2), (3.3), (3.4) and (3.5) as for a single robot should hold. The constraints (3.4) and (3.5) must be only extended by the index set of manipulators and have the following form

$$\sum_{\forall o \in \mathcal{O}_i} F_{oi} = 1, \ \forall \ i \in \mathcal{M}, \tag{4.8}$$

$$\sum_{\forall o \in \mathcal{O}_i} L_{oi} = 1, \ \forall \ i \in \mathcal{M}. \tag{4.9}$$

To preserve a sequence for each manipulator with the first and the last object allocated to manipulator i, additional constraints are formulated

$$F_{oi} \leq W_{oi}, \ \forall \ o \in \mathcal{O}, \ i \in \mathcal{M}_o, \tag{4.10a}$$

$$L_{oi} \leq W_{oi}, \ \forall \ o \in \mathcal{O}, \ i \in \mathcal{M}_o. \tag{4.10b}$$

Similar to the constraints (3.6) and (3.7) for a single manipulator, a sequence of objects must be defined for each manipulator. Therefore, each assigned object has exactly one successor, except for the last one,

$$\sum_{\forall o' \in \mathcal{O}_i} X_{oo'i} = W_{oi} - L_{oi}, \ \forall \ o \in \mathcal{O}, \ i \in \mathcal{M}_o. \tag{4.11}$$

Furthermore, each assigned object has exactly one predecessor, except for the first one, within a manipulator's sequence

$$\sum_{\forall o \in \mathcal{O}_i} X_{oo'i} = W_{o'i} - F_{o'i}, \ \forall \ o' \in \mathcal{O}, \ i \in \mathcal{M}_{o'}. \tag{4.12}$$

To eliminate the case that an object is its own successor or predecessor, the following equality constraints have to hold

$$X_{ooi} = 0, \ \forall \ o \in \mathcal{O}, \ i \in \mathcal{M}_o. \tag{4.13}$$

In addition, two objects can only have a precedence relation, if they are assigned to the same manipulator,

$$2 \cdot (X_{oo'i} + X_{o'oi}) \leq W_{oi} + W_{o'i}, \ \forall o, o' \in \mathcal{O}, \ o \neq o', \ i \in \mathcal{M}_o \cap \mathcal{M}_{o'}. \quad (4.14)$$

Similar constraints as for a single robot (3.8), (3.9) are formulated to avoid subcycles for each manipulator. The continuous variable $C_o \geq 1$ is equal to an object's position within a manipulator's sequence. To guarantee that an object is the first one within a manipulator's sequence, the following inequality constraint should hold

$$C_o \leq 1 + |\mathcal{O}| \cdot (1 - F_{oi}), \ \forall \ o \in \mathcal{O}, \ i \in \mathcal{M}_o. \quad (4.15)$$

The inequalities (4.15) are trivially satisfied, if $F_{oi} = 0$ holds, otherwise C_o takes the value of 1. The order of objects is determined by the following inequality constraints

$$\begin{aligned} C_{o'} &\geq C_o + 1 - |\mathcal{O}| \cdot (3 - X_{oo'i} - W_{oi} - W_{o'i}) \\ C_{o'} &\leq C_o + 1 + |\mathcal{O}| \cdot (3 - X_{oo'i} - W_{oi} - W_{o'i}), \\ &\forall \ o, o' \in \mathcal{O}, \ o \neq o', \ i \in \mathcal{M}_o \cap \mathcal{M}_{o'}. \end{aligned} \quad (4.16)$$

In case the binary variables $X_{oo'i}$, W_{oi} and $W_{o'i}$ are equal to one, the equality $C_{o'} = C_o + 1$ is imposed. Otherwise the inequality constraints are trivially satisfied. Although the variable C_o can only take integer values, due to the constraints (4.15) and (4.16), it is defined as a continuous variable to keep the computational burden of the algorithm manageable. Furthermore, constraints (4.16) ensure that no subcycles occur in the set of feasible solutions. The following Table 4.2 summarizes the relevant decision variables.

Table 4.2: Decision variables of the optimization-based scheduling model for multiple robots.

Decision Variable	Type	Description
W_{oi}	Binary	Specifies, if object o is grasped by manipulator i
Y_{os}	Binary	Allocation of object o to slot s
$X_{oo'i}$	Binary	Immediate precedence variable takes value of one, if object o is picked directly before object o' by manipulator i
F_{oi}	Binary	First object picked by manipulator i
L_{oi}	Binary	Last object picked by manipulator i
$C_o \geq 1$	Continuous	Equal to an object's position within the sequence
Z_{op}	Binary	Equal to one, if object o is in position p of its manipulator's sequence (i.e. if $C_o = p$)

A similar objective function (3.10a) as for a single manipulator is formulated. In case of multiple manipulators, it is important to perform all tasks in the fastest possible way. Due to the fact that the exact duration of the tasks is not known beforehand, linear motion with mean velocities v_i of the manipulators' end-effectors is assumed. The task completion

time of each manipulator can be formulated in the following form

$$\Phi_i = \underbrace{\sum_{\forall o \in \mathcal{O}_i} F_{oi} \cdot d^0_{oi}/v_i}_{\text{Initial movement to the first object}} + \underbrace{\sum_{\forall o \in \mathcal{O}_i} \sum_{\forall s \in \mathcal{S}_i} W_{oi} \cdot Y_{os} \cdot d_{os}/v_i}_{\text{Movement from the picked object to its allocated slot}} + \underbrace{\sum_{\forall o \in \mathcal{O}_i} \sum_{\forall o' \in \mathcal{O}_i \setminus \{o\}} \sum_{\forall s \in \mathcal{S}_i} X_{oo'i} \cdot Y_{os} \cdot d_{o's}/v_i}_{\text{Movement from a slot to the next object}} \tag{4.17}$$

The optimization problem of the scheduling model for multiple manipulators is formulated in the form, where the maximum completion time, i.e., the makespan, is minimized

$$\min \quad \Phi_{\max} \tag{4.18a}$$

$$\text{s.t.} \quad \Phi_{\max} \geq \Phi_i, \ \forall i \in \mathcal{M} \tag{4.18b}$$

$$\text{Const. } (3.2), (3.3), (4.6) - (4.16). \tag{4.18c}$$

Note that constraints cannot be implemented in the form (4.17), since this would result in quadratic equality constraints, which are nonconvex. Instead, the epigraph formulation (4.18) is used to model the makespan without explicitly defining the variables Φ_i. The scheduling model results in a mixed-integer quadratically constrained programming (MIQCP) problem.

4.3.4 Proximity and concurrency constraints

In general, the occurrence of deadlocks cannot be anticipated, especially in cases the trajectories are planned and executed online. However, by analyzing the given setup, it is possible to reduce reproducible types of deadlocks. Two types of deadlocks can potentially occur in a pick-and-place task. The first type involves deadlock during grasping, i.e., when the objects lie too close to each other. The second type of deadlock can occur during placing, in case objects are assigned to the same tray.

The occurrence of deadlocks can already be reduced in *Scheduling* layer by incorporating additional constraints in the scheduling model, formulated in (4.18). To exclude simultaneous picking of adjacent object and simultaneous placing to adjacent slots, the sequence in which objects are allocated to robots should be taken into account. Therefore, an additional set $\mathcal{P}$ and a binary variable Z_{op} are introduced. The set $\mathcal{P}$ represents the position of an object within a manipulator's sequence

$$\mathcal{P} = \{1, \dots, \max_{i \in \mathcal{M}} |\mathcal{O}_i|\}. \tag{4.19}$$

Further, the binary variable Z_{op} designates object o in position p of manipulator's se-

quence. It takes a value of one, if $C_o = p$ and otherwise zero. For instance, if object $o = 2$ is the fourth object grasped by a manipulator i, $C_2 = 4$ and $Z_{2,4} = 1$ holds. Hence, each object can only occupy exactly one position in a sequence

$$\sum_{\forall p \in \mathcal{P}} Z_{op} = 1, \ \forall \ o \in \mathcal{O}. \tag{4.20}$$

Additionally, the variable corresponding to the position C_o can be set equal to one

$$\sum_{\forall p \in \mathcal{P}} Z_{op} \cdot p = C_o, \ \forall \ o \in \mathcal{O}. \tag{4.21}$$

The first type of deadlock can be circumvented by allowing two manipulators to only grasp objects that have a minimum distance $d^{\min}$ between each other in the same position of their sequence,

$$d_{oo'} + (2 - Z_{op} - Z_{o'p}) \cdot d^{\min} \geq d^{\min}, \ \forall \ o, o' \in \mathcal{O}, \ p \in \mathcal{P}, \tag{4.22}$$

where the distance between two objects o and o' is defined as

$$d_{oo'} = \|\chi_o - \chi_{o'}\|_2, \ \forall o, o' \in \mathcal{O}. \tag{4.23}$$

The second deadlock situation is avoided by restricting the allocation of objects with the same position in their manipulator's sequence to slots of the same tray,

$$\sum_{\forall s \in \mathcal{S}_\tau} (Y_{os} + Y_{o's}) \leq 3 - Z_{op} - Z_{o'p}, \ \forall \tau \in \mathcal{T}, \ o, o' \in \mathcal{O}_\tau, \ o \neq o', \ p \in \mathcal{P}. \tag{4.24}$$

More precisely, two objects, where each of them is ordered in the same position of each robot's sequences are not allowed to be placed into the same tray. Otherwise, the robots would serve the same tray simultaneously. For the considered scheduling model, it is assumed that the manipulators perform a linear motion and reach their goals at the same time. As the assumption is quite conservative, it is not sufficient in a case of large number of objects to compute an optimal scheduling plan only once. Therefore, the scheduling plan should iteratively be updated.

The proposed scheduling model already excludes conflicting goals of the robots a priori. However, they still can happen in case deadlocks occur during robot's motion and the operation process of one robot is considerably delayed while the other robot performed a set of tasks. In that case, the coordinator is still capable to handle the deadlocks.

4.3.5 Constraints enforcing specific order of objects

For assembly or disassembly tasks, certain objects (components of a product), should be assembled or disassembled in a specific order. For instance, in a disassembly task, certain objects may not be accessible instantly without extracting other objects first. If several

robots are manipulating the same product, complexity increases further. For this reason, additional constraints are required to enforce a specific order of objects belonging to a product, which is to be assembled or disassembled. Let $\mathcal{R} \subseteq \mathcal{O}$ denote an ordered set of objects, which need to be picked or placed in the given order. Then, this order might be enforced by the following inequality constraints

$$C_r + 1 \leq C_{r'}, \ \forall \ r, r' \in \mathcal{R}, \ r \leq r', \tag{4.25}$$

where object r is a predecessor of object r'.

4.4 Online trajectory generation using DMPC

Consider a system of M independent manipulators solving a joint task and sharing the same workspace. Hereby, each manipulator $i = 1, \dots, M$ receives its own subtask, where the main objective is to safely plan a trajectory from its current to its goal state taking each manipulator's environment into account. The considered environment of each robot involves static obstacles as well as dynamic obstacles, which result from the interaction between multiple robots. The proposed trajectory planning method computes for each manipulator an optimal trajectory, taking the motion of its neighbors into account. For collision avoidance, the ellipsoid-line segment (ELS) approach is employed in the optimization problem of each robot's trajectory planner to ensure a safe robot-robot interaction, which is referred to in the following as DMPC-ELS method and has been published in [Gaf+22a]. The discrete form of the proposed DMPC-ELS problem for each involved manipulator i is formulated in the joint space as

$$\min_{\mathbf{u}_i^{0:N_\mathrm{p}-1}, \mathbf{x}_i^{0:N_\mathrm{p}}} \quad J_i^\mathrm{f}(\mathbf{x}_i^{N_\mathrm{p}}) + \sum_{k=0}^{N_\mathrm{p}-1} J_i^\mathrm{c}(\mathbf{x}_i^k, \mathbf{u}_i^k) \tag{4.26a}$$

$$\text{s.t.} \quad \mathbf{x}_i^{k+1} = \mathbf{A}_i^\mathrm{d}\mathbf{x}_i^k + \mathbf{B}_i^\mathrm{d}\mathbf{u}_i^k, \quad k = 0, \dots, N_\mathrm{p} - 1, \tag{4.26b}$$

$$\mathbf{x}_i^0 = \mathbf{x}_i^\mathrm{s}, \tag{4.26c}$$

$$\mathbf{x}_i^k \in \mathbb{X}_i, \quad k = 0, \dots, N_\mathrm{p}, \tag{4.26d}$$

$$\mathbf{u}_i^k \in \mathbb{U}_i, \quad k = 0, \dots, N_\mathrm{p} - 1, \tag{4.26e}$$

$$R(\mathbf{x}_i^k) \cap \mathcal{O} = \emptyset, \quad k = 0, \dots, N_\mathrm{p}, \tag{4.26f}$$

$$R(\mathbf{x}_i^k) \cap \mathcal{R}(\underline{\mathbf{x}}_{-i}^{*\,k}) = \emptyset, \quad k = 0, \dots, N_\mathrm{p}. \tag{4.26g}$$

Cost function

The cost function for each manipulator is formulated identically to the trajectory planning problem of a single manipulator, introduced in Sec. 3.2.3. The constraint (4.26c) sets the measured joint state $\mathbf{x}_i^\mathrm{s}$ of manipulator i as the initial condition of the state vector $\mathbf{x}_i^k$

at time $t_k = 0$. The cost function of the DMPC-ELS problem consists of the stage cost $J_i^{\mathrm{c}}(\mathbf{x}_i^k, \mathbf{u}_i^k)$ and the terminal state cost $J_i^{\mathrm{f}}(\mathbf{x}_i^{N_\mathrm{p}})$. The stage cost $J_i^{\mathrm{c}} : \mathbb{R}^{2N} \times \mathbb{R}^N \to \mathbb{R}$,

$$J_i^{\mathrm{c}}(\mathbf{x}_i^k, \mathbf{u}_i^k) := (\mathbf{x}_i^k - \mathbf{x}_i^{\mathrm{f}})^{\mathrm{T}} \mathbf{Q}_i^x (\mathbf{x}_i^k - \mathbf{x}_i^{\mathrm{f}}) + \mathbf{u}_i^{k\,\mathrm{T}} \mathbf{R}_i^{\mathrm{u}} \mathbf{u}_i^k + \Delta \mathbf{u}_i^{k\,\mathrm{T}} \mathbf{R}_i^{\mathrm{d}} \Delta \mathbf{u}_i^k (T_{\mathrm{s}})^{-2} \tag{4.27}$$

punishes the squared state error, i.e., the deviation of the state $\mathbf{x}_i^k$ from the desired state $\mathbf{x}_i^{\mathrm{f}} = [\mathbf{q}_i^{\mathrm{f}\,\mathrm{T}},\ \mathbf{0}^{\mathrm{T}}]^{\mathrm{T}}$, the magnitude of the control input $\mathbf{u}^k$ and the control smoothness, i.e., the magnitude of $\Delta \mathbf{u}_i^k = \mathbf{u}_i^k - \mathbf{u}_i^{k-1}$, with the positive definite weighting matrices $\mathbf{Q}_i^x \in \mathbb{R}^{2N \times 2N}$, $\mathbf{R}_i^{\mathrm{u}} \in \mathbb{R}^{N \times N}$ and $\mathbf{R}_i^{\mathrm{d}} \in \mathbb{R}^{N \times N}$, respectively. The terminal state cost $J_i^{\mathrm{f}} : \mathbb{R}^{2N} \to \mathbb{R}$

$$J_i^{\mathrm{f}}(\mathbf{x}_i^{N_\mathrm{p}}) := (\mathbf{x}_i^{N_\mathrm{p}} - \mathbf{x}_i^{\mathrm{f}})^{\mathrm{T}} \mathbf{Q}_i^{\mathrm{f}} (\mathbf{x}_i^{N_\mathrm{p}} - \mathbf{x}_i^{\mathrm{f}}) \tag{4.28}$$

punishes the terminal squared state error with the positive definite weighting matrix $\mathbf{Q}_i^{\mathrm{f}} \in \mathbb{R}^{2N \times 2N}$.

Mathematical model of a manipulator i

The DMPC-ELS relies upon the system dynamics of the corresponding manipulator, for which the optimization problem (4.26) is solved. The discrete-time linear system from (3.18) for a rigid body manipulator with N joints is incorporated as the first constraint (4.26b) into the DMPC-ELS problem.

Kinematic constraints

Self-collision constraints as well as limitation of joint velocities and accelerations are realized similarly to a single manipulator system, introduced in Sec. 3.2.3, i.e.,

$$\mathbb{X}_i := \{\mathbf{x}_i^k \in \mathbb{R}^{2N} \mid \mathbf{x}_{i,\min} \leq \mathbf{x}_i^k \leq \mathbf{x}_{i,\max}\}, \tag{4.29a}$$

$$\mathbb{U}_i := \{\mathbf{u}_i^k \in \mathbb{R}^{N} \mid \mathbf{u}_{i,\min} \leq \mathbf{u}_i^k \leq \mathbf{u}_{i,\max}\}. \tag{4.29b}$$

Static collision avoidance is formulated in the constraint (4.26f) to prevent collisions with the static environment of manipulator i. It is important to note, that the position of an object is usually described in the task space, expressed in Cartesian coordinates. Therefore, it is necessary to transform joint positions, described in configuration space, of manipulator i into Cartesian coordinates by the nonlinear forward kinematics of the robot. Thus, $R(\mathbf{x}_i^k)$ is introduced, which represents the set of interior points in Cartesian space of manipulator i with state $\mathbf{x}_i^k$. Further, the set $\mathcal{O}$ is defined that contains the interior Cartesian points of all static obstacles. By enforcing that the intersection of $R(\mathbf{x}_i^k)$ and $\mathcal{O}$ for state $\mathbf{x}_i^k$ is empty, it is ensured that no collisions between the robot and its static environment occur.

Dynamic collision avoidance prevents inter-robot collisions formulated in the constraints (4.26g). The insight on the future trajectory of each robot is utilized for collision avoidance

between the robots. To evaluate the constraints (4.26g) for manipulator i, knowledge about the optimal trajectories of all neighboring robots is required. For this, the trajectory vector, which comprises the optimal trajectory of all manipulators excluding manipulator i is introduced

$$\underline{\mathbf{x}}_{-i}^{*\,0:N_\mathrm{p}} = [\mathbf{x}_1^{*\,0:N_\mathrm{p}}, \dots, \mathbf{x}_{i-1}^{*\,0:N_\mathrm{p}}, \mathbf{x}_{i+1}^{*\,0:N_\mathrm{p}}, \dots, \mathbf{x}_M^{*\,0:N_\mathrm{p}}]. \tag{4.30}$$

By defining the set

$$\mathcal{R}(\underline{\mathbf{x}}_{-i}^{k}) := \bigcup_{j=1, j\neq i}^{M} R(\mathbf{x}_j^k), \tag{4.31}$$

the constraints (4.26g) guarantee collision avoidance of robot i with all other manipulators in its neighborhood. How the constraints (4.26g) are efficiently implemented is the topic of the subsequent section 4.4.1 and will be explained in more detail. Note that the constraint (4.26g) establishes a coupling in states and leads to M coupled DMPC-ELS problems.

Communication

Communication between the robots is essential to prevent inter-robot collisions. Each manipulator solves its own trajectory planning problem based on shared information from its neighbors to obtain an optimal trajectory to its goal state. Bidirectional communication is performed once at each time step, where each robot shares its current and predicted states with its neighbors. The information with a neighbor is shared in case the workspaces of the considered manipulators overlap.

The optimization problem (4.26) is solved for each robot i in parallel. Thus, it is necessary to obtain the optimal trajectories of each robot's neighbors, collected in $\underline{\mathbf{x}}_{-i}^{*\,0:N_\mathrm{p}}$ a priori. To obtain an approximation of $\underline{\mathbf{x}}_{-i}^{*\,0:N_\mathrm{p}}$, the extrapolation approach of Christofides et al. [Chr+13] is used. Suppose

$$\hat{\underline{\mathbf{x}}}^{*\,0:N_\mathrm{p}} = [\hat{\mathbf{x}}_1^{*\,0:N_\mathrm{p}}, \dots, \hat{\mathbf{x}}_M^{*\,0:N_\mathrm{p}}] \tag{4.32}$$

denotes the optimal trajectories of all manipulators of the last converged DMPC step. The required state vector $\underline{\mathbf{x}}^{*\,0:N_\mathrm{p}}$ (and thus also $\underline{\mathbf{x}}_{-i}^{*\,0:N_\mathrm{p}}$) is obtained by shifting $\hat{\underline{\mathbf{x}}}^{*\,0:N_\mathrm{p}}$ by one time step and extrapolating the last state using the discrete system dynamics

$$\mathbf{x}_i^{*\,0:N_\mathrm{p}} \mathrel{\widehat{=}} \hat{\mathbf{x}}_i^{*\,1:N_\mathrm{p}+1} = [\hat{\mathbf{x}}_i^{*\,1:N_\mathrm{p}}, \quad \mathbf{A}_i^\mathrm{d}\hat{\mathbf{x}}_i^{*N_\mathrm{p}} + \mathbf{B}_i^\mathrm{d}\mathbf{u}_i^{*N_\mathrm{p}-1}]. \tag{4.33}$$

It is assumed that the two last optimal inputs in the sequence are equal, i.e.,

$$\mathbf{u}_i^{*N_\mathrm{p}-1} = \hat{\mathbf{u}}_i^{*N_\mathrm{p}-1} = \hat{\mathbf{u}}_i^{*N_\mathrm{p}}. \tag{4.34}$$

Summing up, each model predictive controller of manipulator i receives its current state $\mathbf{x}_i^k$ and the extrapolated predicted state sequence of its neighbors $\underline{\mathbf{x}}_{-i}^{*\,0:N_\mathrm{p}}$ to avoid inter-robot collisions.

4.4.1 Inter-robot collision avoidance

In the following section, the dynamic collision avoidance approach based on ellipsoid-line segment (ELS) bounding volumes is introduced. The proposed approach offers an efficient implementation of the constraints (4.26g) in the DMPC-ELS problem (4.26). Special care is taken to account for the whole geometry of a robot to guarantee a collision-free trajectory rather than formulating a distance function between arbitrarily chosen points on two links. Additionally, the goal is to derive an efficient and smooth formulation to overcome the problem with nested logical conditions, which is a prevalent problem in existing collision avoidance strategies, see Sec. 2.2 for more details.

The collision avoidance strategy, introduced in Sec. 3.2.3, was employed for a single manipulator operating in a static environment. Taking a multi-manipulator system into account, each robot represents a dynamical obstacle to the other one. Consider two manipulators, illustrated exemplary in Fig. 4.4, performing pick-and-place tasks in close proximity to each other, where collisions between the robots are imminent. The links of one manipulator are encapsulated by line swept spheres (LSSs), resulting in a spherocylindrical shape, and the links of the neighbored robots by ellipsoids. To guarantee that constraints (4.26g) are fulfilled, it is necessary to ensure that no intersections between LSSs and ellipsoids occur at each time step $k = 0, \dots, N_\mathrm{p}$ for all involved manipulators. Enforcing the intersections of sets to be empty has the advantage that the whole geometry of manipulators is considered and thus a collision avoidance between all links of two or more robots is guaranteed. Therefore, it is important to choose proper dimensions of the ellipsoids with a suitable safety margin.

To explain the derivation of the proposed approach in more detail, collision avoidance

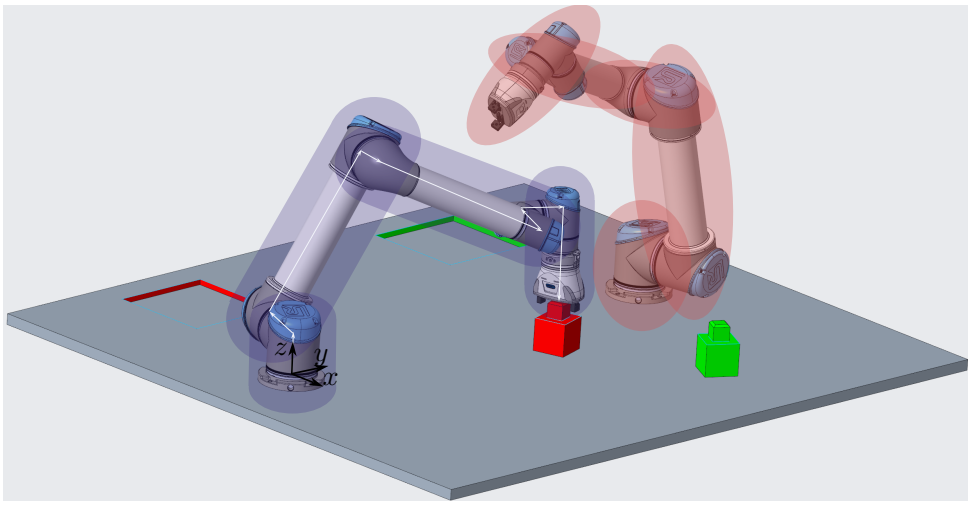


Figure 4.4: Illustrative representation of the robots' geometry, where left robot is modeled by a series of line segments resulting in LSSs (spherocylinders) and the right robot by several ellipsoids.

between two manipulators i and j is considered. Starting from the perspective of manipulator i, for which the optimization problem (4.26) is solved, each link $m \in \mu - 1$ of manipulator i is modeled by LSSs. To this end, a line segment $\mathbf{s}_m$ is defined as

$$\mathbf{s}_m(\mathbf{x}_i^k, \alpha_m) := \mathbf{b}_m(\mathbf{x}_i^k) + \alpha_m \mathbf{r}_m(\mathbf{x}_i^k), \quad \alpha_m \in [0, 1], \tag{4.35}$$

where the vector $\mathbf{b}_m(\mathbf{x}_i^k) \in \mathbb{R}^3$ denotes the position vector to the basis of the considered link and the vector $\mathbf{r}_m(\mathbf{x}_i^k) \in \mathbb{R}^3$ describes the direction from $\mathbf{b}_m(\mathbf{x}_i^k)$ to $\mathbf{b}_{m+1}(\mathbf{x}_i^k)$ of the subsequent link. The parameter α_m restricts the line segment $\mathbf{s}_m$ to the length of the considered link.

Each link $n \in \eta - 1$ of the manipulator j is modeled by an ellipsoid

$$\{\mathbf{e} \in \mathbb{R}^3 \mid H_n(\mathbf{e}, \mathbf{x}_j^k) = 1\} \tag{4.36}$$

for a given state $\mathbf{x}_j^k \in \mathbb{R}^{2N}$. The ellipsoid is parameterized by the quadratic function

$$H_n(\mathbf{e}, \mathbf{x}_j^k) := (\mathbf{e} - \mathbf{e}_{0,n}(\mathbf{x}_j^k))^{\mathrm{T}} \mathbf{R}_n(\mathbf{x}_j^k) \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}}(\mathbf{x}_j^k)(\mathbf{e} - \mathbf{e}_{0,n}(\mathbf{x}_j^k)), \tag{4.37}$$

where $\mathbf{e} \in \mathbb{R}^3$ denotes a point on the ellipsoid. The centre point $\mathbf{e}_{0,n}(\mathbf{x}_j^k)$ is computed from the position vectors of the considered link $\mathbf{b}_n(\mathbf{x}_j^k)$ and the subsequent link $\mathbf{b}_{n+1}(\mathbf{x}_j^k)$ as

$$\mathbf{e}_{0,n}(\mathbf{x}_j^k) = \frac{1}{2}(\mathbf{b}_{n+1}(\mathbf{x}_j^k) + \mathbf{b}_n(\mathbf{x}_j^k)). \tag{4.38}$$

Further, the rotation matrix $\mathbf{R}_n(\mathbf{x}_j^k) \in \mathrm{SO}(3)$ describes the rotation of link n relative to the inertial frame. Finally, the diagonal matrix

$$\mathbf{E}_n = \mathrm{diag}\left(\frac{1}{l_1^2}, \frac{1}{l_2^2}, \frac{1}{l_3^2}\right) \in \mathbb{R}^{3\times 3} \tag{4.39}$$

comprises the squared inverse principal semi-axes $l_1, l_2, l_3 \in \mathbb{R}_{>0}$ of the ellipsoid.

The principal semi-axes of each ellipsoid should be chosen sufficiently large, i.e., at least twice as large as the width of a robot link, and should cover the two joints connecting the link. This is illustrated in Fig. 4.5.

To ensure that the line segment m of manipulator i does not intersect with the ellipsoid n of manipulator j, the following condition has to hold

$$1 - H_n(\mathbf{s}_m(\mathbf{x}_i^k, \alpha_m), \mathbf{x}_j^k) \leq 0, \quad \{i, j\} \in M,\ n \in \nu,\ m \in \mu. \tag{4.40}$$

The former condition introduces an additional decision variable $\alpha_m \in [0, 1]$ into the optimization problem (4.26), which significantly increases computational costs depending on the number of collision-prone links to be considered. To overcome this problem, the condition (4.40) is reformulated into a constrained optimization problem to determine α_m^* by

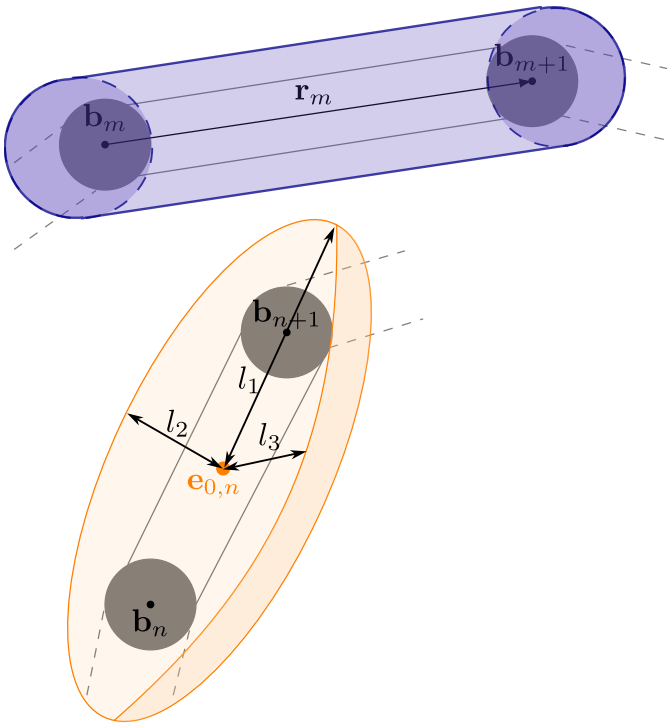


Figure 4.5: Illustrative example of a collision-prone pair of robotic links.

solving

$$\min_{\alpha_m} \quad H_n(\mathbf{b}_m(\mathbf{x}_i^k) + \alpha_m \mathbf{r}_m(\mathbf{x}_i^k), \mathbf{x}_j^k) \tag{4.41}$$

$$s.t. \quad 0 \leq \alpha_m \leq 1. \tag{4.41a}$$

For sake of readability, the explicit reference to $\mathbf{x}_i^k$ and $\mathbf{x}_j^k$ is dropped in the following. To solve the optimization problem (4.41), the unconstrained optimization problem is solved first, i.e.,

$$\frac{\mathrm{d}}{\mathrm{d}\hat{\alpha}_m}(\mathbf{b}_m + \hat{\alpha}_m \mathbf{r}_m - \mathbf{e}_{0,n})^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} (\mathbf{b}_m + \hat{\alpha}_m \mathbf{r}_m - \mathbf{e}_{0,n}) \stackrel{!}{=} 0, \tag{4.42}$$

$$\mathbf{r}_m^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} (\mathbf{b}_m - \mathbf{e}_{0,n}) + \hat{\alpha}_m \mathbf{r}_m^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m = 0, \tag{4.43}$$

where the optimal solution

$$\hat{\alpha}_m^* = -\frac{(\mathbf{b}_m - \mathbf{e}_{0,n})^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m}{\mathbf{r}_m^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m} \tag{4.44}$$

holds. However, the solution of $\hat{\alpha}_m^*$ is unbounded and valid in the range $\hat{\alpha}_m^* \in (-\infty, \infty)$. The solution (4.44) is guaranteed to exist since $\mathbf{E}_n$ is positive definite, i.e., $\mathbf{r}_m^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m > 0$ holds. To bound the solution of $\hat{\alpha}_m^*$ onto the unit interval, a projection operator $P : (-\infty, \infty) \to [0, 1]$ is applied to determine the solution of the constrained optimization problem (4.41) in a closed form

$$\alpha_m^* = P(\hat{\alpha}_m^*) = P\left(-\frac{(\mathbf{b}_m - \mathbf{e}_{0,n})^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m}{\mathbf{r}_m^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m} \right). \tag{4.45}$$

To transform P into a continuously differentiable function, an approximation in the general form

$$\hat{P}(x) = x\ \Phi(x) - (x-1)\ \Phi(x-1) \tag{4.46}$$

is chosen, where a smooth approximation of the Heaviside function is applied

$$\Phi(x) = \frac{1}{1+\exp(-cx)}, \tag{4.47}$$

with $c \in \mathbb{R}_{>0}$ as a scaling parameter. Note that for $c \to \infty$ the function $\hat{P}$ converges towards P. By choosing $c = 20$, the maximum absolute error of $\hat{P}(x)$ amounts to $1.13{\cdot}10^{-2}$. Both functions $\hat{P}$ and P are given in Fig. 4.6.

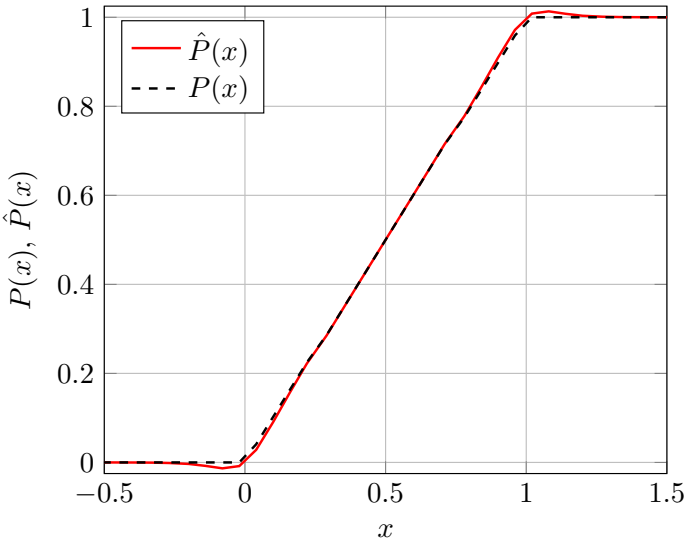


Figure 4.6: A comparison between the projection operator $P(\alpha)$ and the approximated function $\hat{P}(\alpha)$ with parameter $c = 20$ [Gaf+22a].

4.4.2 Comparison of centralized and distributed optimization

In the following, the solution quality of the DMPC solution is investigated by comparing it to the centralized one. Therefore, a simple trajectory is chosen, where two manipulators of type UR3 are commanded to move to opposite sides. To successfully reach the goal state, the robots must avoid each other. Note that an ideal dynamical model of decoupled integrators, introduced in (3.17), has been used for UR3 model. The trajectory is computed in the distributed and centralized scheme for prediction horizon lengths $N_\mathrm{p} = \{10, 15, 20, 25, 30\}$. The results of three joint angles of one robot for prediction horizon lengths $N_\mathrm{p} = \{10, 20, 30\}$ are given in Fig. 4.7. As can be seen from the results, the deviation of the distributed solution from the centralized solution becomes smaller with increasing prediction horizon length, where the solution of the distributed scheme closely follows the solution of the centralized one. Fig. 4.8a illustrates the evolution of the mean squared error of the joint angles of centralized and distributed MPC over the prediction

horizon length. The greatest decrease of the error can be observed from $N_\mathrm{p} = 15$ to $N_\mathrm{p} = 20$. From the prediction horizon length $N_\mathrm{p} = 20$ upwards, the mean squared error of the joint angles remains almost constant and slowly converges towards zero. Fig. 4.8b shows how the mean computation times T_c of CMPC and DMPC scale with increasing prediction horizon length. By determining the ratio of computation times $rt = \frac{\mathrm{mean}(T_{\mathrm{c,CMPC}})}{\mathrm{mean}(T_{\mathrm{c,DMPC}})}$ between the two schemes, the ratio rt increases significantly from 1.5 for prediction horizon length of $N_\mathrm{p} = 10$ to 1.9 for prediction horizon length of $N_\mathrm{p} = 30$. Solving the DMPC problem for a setup of two manipulators in parallel reduces the computation times by factor of almost two.

The results demonstrate that the distributed solution converges towards the centralized one for an increasing prediction horizon length and takes significantly less computation time. In addition, a prediction horizon length of $N_\mathrm{p} = 20$ is sufficient to obtain results with DMPC that are close to the centralized one. The evaluation of the results show that solving the problem of trajectory planning in real time for a multi-robot system in the centralized scheme is intractable. Due to limited computational resources in real time applications, it is necessary to solve the problem of trajectory planning for a multi-robot system in a distributed scheme.

4.5 Reactive deadlock resolution approach

Deadlocks can always occur within a team of manipulators sharing a common workspace and operating each autonomously, i.e., each robot's motion controller plans and executes its tasks online. Such kind of systems have been rarely studied as an intelligent coordination between the robots is imminent to keep the robotic system collision- and deadlock-free. The proposed deadlock resolution approach introduced in the following allows for an autonomous operation of each manipulator sharing a common workspace, where deadlocks can be detected and resolved during task execution. Each robot is able to detect if it is currently stuck in a deadlock and a supervisory instance is capable of resolving any occurring deadlock within a team of manipulators.

The proposed reactive deadlock approach builds upon the work of Jäger et al. [JN01]. Deadlocks may occur in a setup of multiple manipulators, where one or more manipulators block each others motion, so that their respective goal remains unreachable until the deadlocks are resolved. Cases in which deadlock between multiple manipulators can occur are:

- Densely overlapping workspaces: due to manipulator's complex kinematic chain many trajectory planners fail to find a path for every manipulator.
- Regions of manipulators' initial or goal states lie too close to each other. Due to

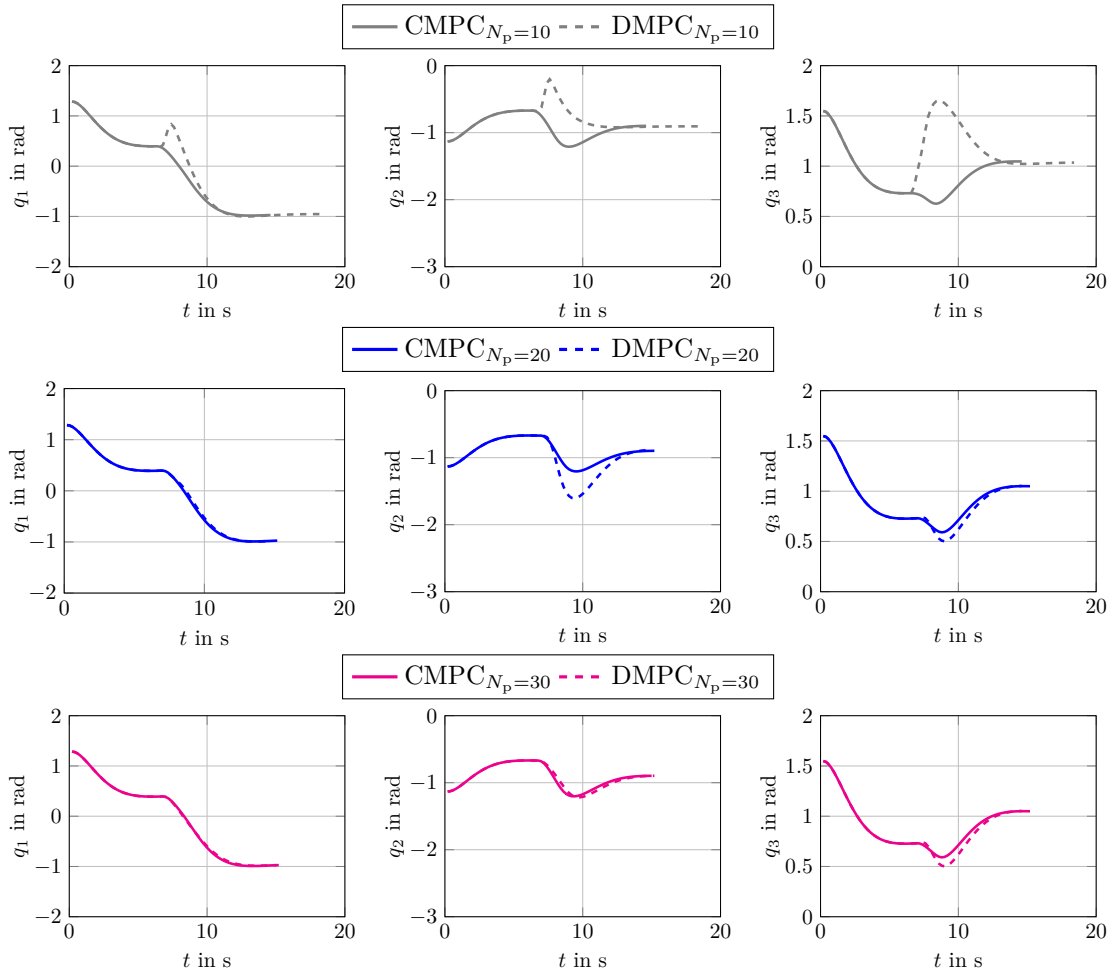
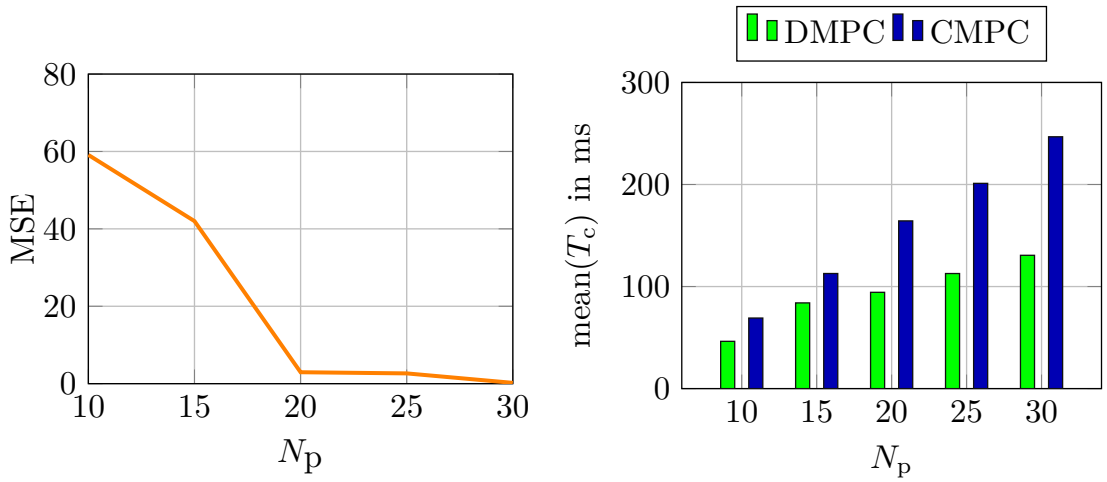


Figure 4.7: Comparing solutions from distributed and centralized MPC for three joint angles with different prediction horizon lengths.



(a) Mean squared error of joint angles over prediction horizon length.

(b) Dependency of mean computation times over prediction horizon length.

Figure 4.8: Mean squared error of joint angles and mean computation times over prediction horizon length.

certain safety margins for collision avoidance, it might become impossible to place an object in the working region of another manipulator. In that case a sequential processing of tasks is necessary.

In case a deadlock occurs, the solution of the optimization problem (4.26) is stuck in a local minimum. The exchange of information between the DMPCs of manipulators about their current and predicted states is not sufficient to resolve an occurred deadlock. Therefore, a coordinator in the role of a supervisory instance is proposed, that takes over the following tasks:

- Communication with the robots about their deadlock status.
- Clustering of robots in deadlock-affected and non-deadlock-affected groups of robots.
- Selecting robots, which are given a higher priority to finish their tasks and disabling all other robots.
- Deciding whether the disabled robots can proceed with their previous tasks.

The coordinator initiates its supervisory role, if one or more robots report a deadlock. For this, a local deadlock detection algorithm is proposed, where each manipulator i monitors if its currently in a deadlock and shares this information with the coordinator. Manipulator i reports to be in a deadlock if both of the following conditions hold:

1. The L_2 norm of the joint velocities must be sufficiently small.
2. The deviation of the robot's measured state $\mathbf{x}_i^{\mathrm{s}}$ and the desired state $\mathbf{x}_i^{\mathrm{f}}$ must be sufficiently large.

The two conditions can be formulated logically as

$$\left\|\dot{\mathbf{q}}_i^{*\,0}\right\|_2 \leq \varepsilon_v \quad \wedge \quad \left\|\mathbf{x}_i^{\mathrm{s}} - \mathbf{x}_i^{\mathrm{f}}\right\|_2 \geq \delta_x \quad \Rightarrow \quad \gamma_{D,i} = 1. \tag{4.48}$$

The parameter $\gamma_{D,i} \in \{0, 1\}$ denotes the deadlock parameter that is communicated from each robot to the coordinator. Once, a single robot or multiple robots report a deadlock, the coordinator initiates its clustering and resolving procedure. The clustering procedure, summarized as pseudo-code in Algorithm 2, aims at distributing robots into deadlock-affected and non-deadlock-affected groups by their mutual distances. To this end, the coordinator receives current and goal states of each robot to compute the distances between them. In the first step, all robots R_i, $i = 1, \ldots, M$ belong to M disjoint clusters $\mathcal{C}_i$, i.e., $\mathcal{C}_i = \{R_i\}$. In case of a deadlock for a manipulator R_i, i.e., $\gamma_{D,i} = 1$, the coordinator computes the distances between the robot R_i and all its neighbors. Thereafter, all robots which are sufficiently close to manipulator i are moved to the same cluster $\mathcal{C}_i$. This clustering procedure ensures that only deadlock-affected robots are assigned to the same cluster while the cluster of non-deadlock-affected robots remain unchanged. Depending

on the number of robots, there may exist multiple clusters of deadlock-affected robots.

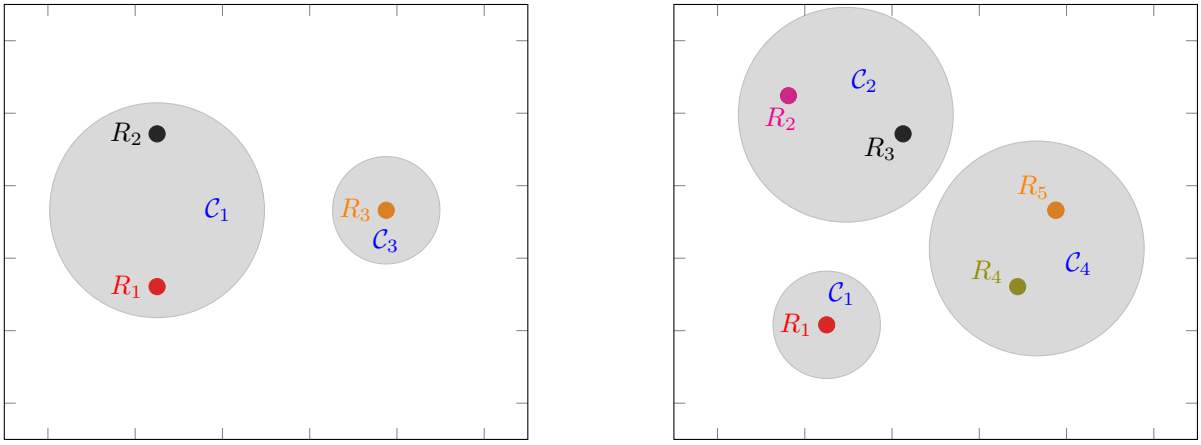
Fig. 4.9 illustrates two examples of the clustering procedure. In the first example, presented in Fig. 4.9a, the robot R_1 reports to the coordinator that it is currently in a deadlock. As robot R_2 is the closest to robot R_1, the coordinator clusters them to the same group $\mathcal{C}_1$. Thus, the cluster $\mathcal{C}_2$ remains empty. As robot R_3 is not involved in the deadlock, it remains in its own cluster $\mathcal{C}_3$. Another example, depicted in Fig. 4.9b, presents two deadlocks occurred independently of each other. One deadlock is detected between robots R_2 and R_3, while the other one is present between robots R_4 and R_5. As robot R_1 is not involved in any of the deadlocks, the coordinator determines three clusters. The first cluster $\mathcal{C}_1$ of robot R_1 remains unchanged. The second cluster comprises the two robots R_2 and R_3. As robot R_3 has been moved to cluster $\mathcal{C}_2$, the cluster $\mathcal{C}_3$ is empty. The same holds true for the last two clusters $\mathcal{C}_4$ and $\mathcal{C}_5$. While the cluster $\mathcal{C}_4$ consists of the robots R_4 and R_5 that are currently in a deadlock, the cluster $\mathcal{C}_5$ remains empty.

To resolve a deadlock, the following steps are required, summarized as pseudo-code in Algorithm 3. The coordinator sends the resolving parameter $\gamma_{\mathrm{R},i} = 0$ to all robots in a cluster $\mathcal{C}_j$, except for the robot $i_{\min}$ with the smallest distance to its target pose, denoted as residual $\mathrm{res}(i_{\min})$, that receives $\gamma_{\mathrm{R},i_{\min}} = 1$. A resolving parameter of $\gamma_{\mathrm{R},i} = 0$ stands for a new neutral goal state $\mathbf{x}_i^{\mathrm{D}}$. The neutral pose is chosen in the manner, that a robot minimizes its occupancy in the workspace by shrinking the links and thus leaving more space to the robot that is allowed to proceed to its goal. Once the robot finishes its task, i.e., reaches its goal state with the allowed tolerance $\varepsilon_{\mathrm{tol}}$, the clusters are reset by sending all involved robots to their respective cluster $\mathcal{C}_j = \{R_i\}$. Further, each robot's resolving parameter is set to $\gamma_{\mathrm{R},i} = 1$, assigning the robots their former goals to complete their previous tasks.

Algorithm 2

Clustering

```
for i = 1 to M do
  if γ_{D,i} = 1 then
    Check which robots are sufficiently close
    for j = 1 to M do
      if i ≠ j & dist(R_i, R_j) ≤ d_min then
        Add robot R_j to the cluster C_i
      else
        Robot R_j remains in its own cluster
      end if
    end for
  end if
end for
```



(a) Three robots in two clusters.

(b) Five robots in three clusters.

Figure 4.9: Two examples of clustering procedure.

Algorithm 3

Resolving deadlocks

for $j = 1$ **to** M **do**
  $i_{\min} = \arg\min_{i \in \mathcal{C}_j} \text{res}(i)$
  Send $\gamma_{R,i} = 1$ to robot $i = i_{\min}$
  Send $\gamma_{R,i} = 0$ to all robots $i \in \mathcal{C}_j \backslash i_{\min}$
  if $\text{res}(i_{\min}) < \varepsilon_{\text{tol}}$ **then**
    Send $\gamma_{R,i} = 1$ to all robots $i \in \mathcal{C}_j$
    Redistribute all robots to their original clusters
  end if
end for

4.6 Summary

This chapter introduces a novel optimization-based TAMP approach for multi-manipulator systems that handles both task and robot coordination. The hierarchical structure of the proposed control scheme enables the effective deployment of a group of manipulators, providing both modularity and scalability. The novelty of the scheduling approach lies in its formulation as a mixed-integer quadratically constrained programming problem, which allows for the easy incorporation of various constraints. This enables the prevention of certain deadlocks a priori and optimal task distribution among the robots. Furthermore, a specific order of objects can be enforced for assembly and disassembly tasks. The overall objective of task coordination is to minimize the maximum completion time required for all robots to complete the task. To demonstrate the benefits of the optimization-based scheduling model, a heuristic model will be employed for benchmarking through simulations and experiments.

The trajectory planning problem is addressed using model predictive control (MPC), allowing for online trajectory planning for each robot with a distributed MPC (DMPC) scheme. The novel collision avoidance approach, termed ELS, allows a close operation of manipulators in the common workspace, where the whole geometry of collision-prone bodies are considered. Communication between robots is managed through the exchange of predicted trajectories between the DMPC-based motion controllers, which is allowed only between neighbored robots. The collision avoidance strategy, integrated into each robot's DMPC as DMPC-ELS, enables trajectory replanning to prevent collisions with neighboring robots. The system's reliability is further enhanced by a coordinator that reacts to and resolves deadlocks involving only deadlock-affected robots. An efficient clustering procedure ensures that robots not involved in a deadlock can operate without interruption.

5 Simulation-based study

In this chapter, a simulation-based study comprising a team of manipulators is conducted. The joint goal of the robots is to sort different types of objects into corresponding trays. First, a setup of two robots is considered, where the task assignments are carried out with the heuristic or the optimization-based scheduling approach, introduced in Chapter 4.3. The study comprises 20 samples, where in each sample six objects are randomly distributed in the shared workspace. The execution times and success rates are evaluated to draw conclusions about the performance of both scheduling models. Further, the influence of prediction horizon length on the performance of the DMPC-ELS is analyzed.

A benchmark analysis of the DMPC-ELS approach and several state-of-the-art sampling- and optimization-based planners is conducted in the next step. For this purpose, five performance metrics are introduced to evaluate the quality of the planned trajectories: path length, execution time, planning time, smoothness, and success rate. Finally, the scalability of the DMPC-ELS approach of up to four robots is demonstrated. For that, different constellations of robots are considered to perform pick-and-place tasks in a static environment. The contents of this chapter have been published in [GYR21; Gaf+22a].

5.1 General settings

The proposed TAMP approach, introduced in Sec. 4.2, is applied for a two-manipulator system, illustrated in Fig. 5.1. The simulation setup comprises two modules, two collaborative robotic manipulators UR3 from *Universal Robots*, six objects and two trays. A module is comprised of a table, on top of which a single manipulator is placed. The modular approach allows for flexibility by coupling multiple modules to each other, making cooperation between robots possible. The common workspace of the robots is occupied by two trays and six objects. The objective of each manipulator is to pick up several objects in the shared workspace and to place them into the assigned trays. Each tray contains three slots the objects can be placed into and both trays can be served by both robots. For each sample, the six objects are randomly placed in the common workspace by applying random sequential adsorption (RSA) [Eva93]. Additional reachability constraints ensure that the objects are placed accessible to both robots.

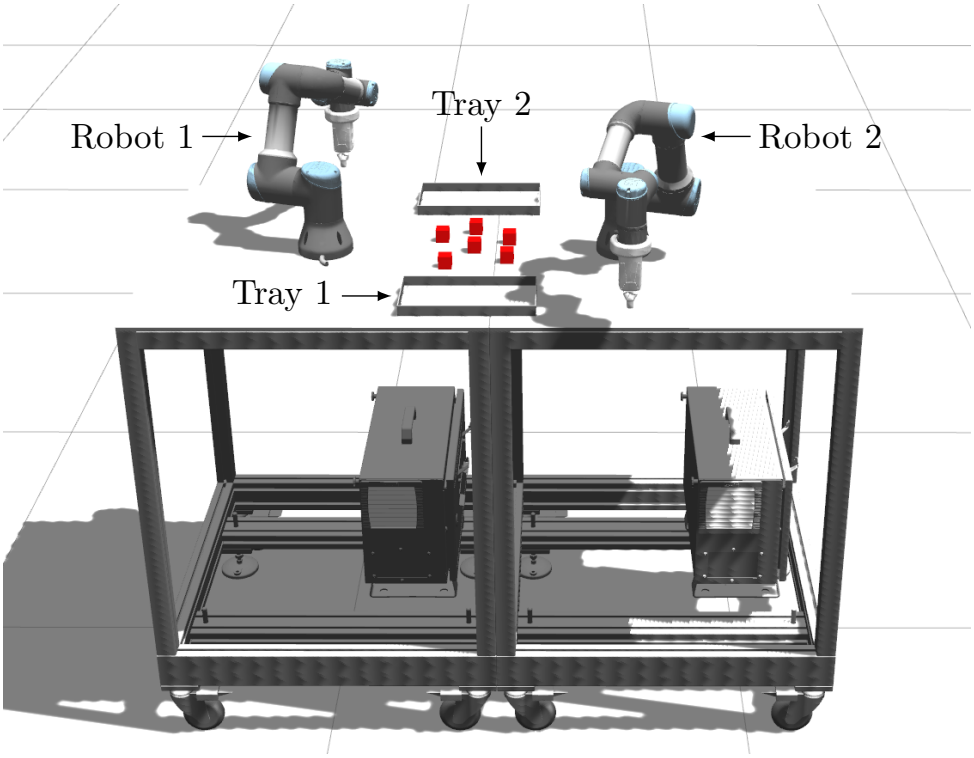


Figure 5.1: Simulation setup with 2 modules of UR3 robots.

The sequence of setpoints assigned to each robot is determined by utilizing the heuristic or the optimization-based scheduling model from Sec. 4.3. Prior to sending a sequence of setpoints to the DMPC-ELS, inverse kinematics is applied to transform setpoints from Cartesian space into configuration space of each manipulator. As a result, the action planner transforms a set of setpoints into a set of target states, which are sent subsequently to each DMPC-ELS of the robots. Once the robot successfully reached its previously assigned target state, the action planner sends the next target state to the robot. In case deadlocks occur during a task execution, the coordinator resolves them. Further, the action $\mathcal{A}_{\text{p\&p}}$ for pick-and-place task is assigned to each target state of the robot, introduced in Sec. 4.2. As grasping of objects is not the main focus of this work, the picking or placing procedure is only performed from above. Straight up and down movements from a small distance to corresponding objects are carried out by sending the setpoints directly to velocity tracking controllers of the robot.

The gripper attached to each robot is from *Schunk* with the model name Co-act EGP-C 40-N-N-URID[1]. *Gazebo* provides an interface to control the robots using *ROS*, where the communication to the *Universal Robots ROS* driver is established by the *ROS* Action Protocol. Each robot is controlled using a velocity controller hardware interface. Using the *ROS* interface as a means of communication between subsystems allows for an easy substitution of the *Gazebo* simulation model with a real testbed, see Chapter 6, without adapting the control algorithm in any aspect.

[1]https://schunk.com/de/de/greiftechnik/parallelgreifer/co-act-egp-c/c/PGR_3995 (Last visited: 18.03.2024)

The simulation study is conducted in the robotic simulation environment *Gazebo* [KH04]. The properties of objects are specified a priori while simulating the environment, omitting the Object Detection layer. The multi-robot scheduling problem, formulated as MIQCP problem (4.18), is solved using Gurobi [Gur23]. The control algorithms are implemented in MATLAB using *CasADi* [And+19] for setting up the NLP (4.26) for the DMPC-ELS. *CasADi* is chosen due to its capability of automatic differentiation, where the first- and second-order derivatives of the cost function and constraints are computed and provided to the interior point solver IPOPT [WB06]. Further, *MA27* [Lib18] is used to solve the underlying linear system. In addition, *CasADi* is instructed to pre-compile the optimization problems (4.26) for each robot using just-in-time compilation. Both model predictive controllers run in parallel on a machine equipped with an Intel i7-11800H CPU with 8 cores and 32 GB RAM under Ubuntu 20.04. The coordinator is employed on the same machine and communicates with the MPCs via the UDP protocol. The UDP protocols are also used for exchange of information between robots. The parameters for DMPC-ELS and the deadlock algorithm are listed in Table 5.1.

Table 5.1: Parameters for DMPC-ELS and deadlock algorithm.

DMPC-ELS Parameters	
Sampling time	$T_\mathrm{s} = 200$ ms
Termination criterion (pose accuracy)	$\varepsilon_\mathrm{tol} = 4 \cdot 10^{-2}$ rad
Weighting matrices:	
- predicted states	$\mathbf{Q}_i^\mathrm{x} = \mathrm{diag}(1,1,1,0.2,0.2,1,1,1,1,0.1,0.1,0.1)^2$
- final state	$\mathbf{Q}_i^\mathrm{f} = 5\ \mathbf{Q}_i^\mathrm{x}$
- control inputs	$\mathbf{R}_i^\mathrm{u} = \mathbf{I}^{6\times6}\ \frac{\mathrm{s}^4}{\mathrm{rad}^2}$
- control input deviations	$\mathbf{R}_i^\mathrm{d} = \mathbf{I}^{6\times6}\ \frac{\mathrm{s}^6}{\mathrm{rad}^2}$
Limits on joint velocities	$\dot{\mathbf{q}}_\mathrm{max/min} = \pm[\pi,\pi,\pi,2\pi,2\pi,2\pi]\ \frac{\mathrm{rad}}{\mathrm{s}}$
Limits on control inputs	$\mathbf{u}_\mathrm{max/min} = \pm[\pi,\pi,\pi,2\pi,2\pi,2\pi]\ \frac{\mathrm{rad}}{\mathrm{s}^2}$
Upper bound on joint angles	$\mathbf{q}_\mathrm{max} = [2\pi, 0, 0, \frac{1}{6}\pi, \pi, 2\pi]$ rad
Lower bound on joint angles	$\mathbf{q}_\mathrm{min} = [-2\pi, -\pi, -\frac{5}{6}\pi, -\frac{5}{6}\pi, 0, -2\pi]$ rad
Deadlock Parameters	
Velocity tolerance	$\varepsilon_v = 1.5 \cdot 10^{-3}\ \frac{\mathrm{rad}}{\mathrm{s}}$
State tolerance	$\delta_x = 1.2 \cdot 10^{-2}$ rad
Clustering parameter	$d_\mathrm{min} = 0.2$ m
Setup Parameters	
Table height	$z_\mathrm{T} = 1.107$ m
Table offset	$\mathbf{z}_\mathrm{min} = [0, 0, 0.04]$ m
Grasping offset	$z_\mathrm{offset} = 0.06$ m

5.2 Environmental modeling

The simulation environment, illustrated in Fig. 5.1, is designed to evaluate the performance of the DMPC-ELS approach, with each robot acting as a dynamically changing obstacle to the other. In the considered setup, the table represents a static obstacle. To prevent collisions between the robot and the table, a plane is defined along the table. For each robot, the intersection of the basis of each link $\mathbf{b}_m$ with the plane is constrained by

$$\mathbf{b}_m \geq \mathbf{z} + \mathbf{z}_{\min}, \tag{5.1}$$

where $\mathbf{z} = [0\ 0\ z_{\mathrm{T}}]^{\mathrm{T}}$ represents the table's height and $\mathbf{z}_{\min}$ is an offset. This constraint enforces the condition (4.26f) in the DMPC-ELS problem (4.26). The offset $\mathbf{z}_{\min}$ accounts for the radius of the two wrist joints (Wrist 1 and Wrist 2) and the end-effector, i.e., the last wrist including the gripper. For safety reasons, 20% greater dimensions than necessary are chosen for $\mathbf{z}_{\min}$. The setup parameters are specified in Table 5.1. The kinematic chain and the corresponding joint nomenclature of the utilized robot from *Universal Robots* is given in Fig. 5.2. The kinematics and DH-parameters of UR3 are provided on the official website of *Universal Robots*[3].

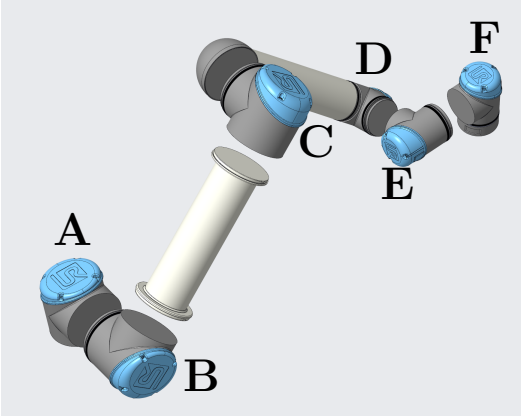


Figure 5.2: The joint nomenclature of a collaborative manipulator from *Universal Robots* is as follows A:Base, B:Shoulder, C:Elbow, D:Wrist 1, E:Wrist 2, F:Wrist 3

To prevent inter-robot collisions, the ELS method from Sec. 4.4.1 is used. Determining the necessary number of collision avoidance constraints is a matter of pre-processing. The geometric representation of collision-prone bodies for the two robots is illustrated in Fig. 4.4. The geometry of each robot, for which the DMPC problem is solved, is modeled by $N_{\mathrm{L}} = 8$ line segments. The body of the respective neighbored robot is enclosed within $N_{\mathrm{E}} = 5$ ellipsoids, encapsulating the basis, subsequent three links (Shoulder, Elbow, Wrist 2) and the end-effector including the gripper, referred to as Wrist 3. This results in $N_{\mathrm{dyn}} = N_{\mathrm{L}} \cdot N_{\mathrm{E}} = 40$ collision constraints per time step of the prediction horizon. By using sufficiently large ellipsoids, the three short line segments connecting Basis and Shoulder,

[3] https://www.universal-robots.com/articles/ur/application-installation/dh-parameters-for-calculations-of-kinematics-and-dynamics/ (Last visited: 18.03.2024)

Table 5.2: Collision-prone pair of links for the two-robot setup.

Robot i (Line Segments)	**Robot** j (Ellipsoids)	$\mathbf{N}_{\text{dyn}} \cdot \mathbf{N}_{\text{p}}$
Wrist 2, Wrist 3	Shoulder	$2 \cdot N_{\text{p}}$
Elbow, Wrist 2, Wrist 3	Elbow	$3 \cdot N_{\text{p}}$
Elbow, Wrist 2, Wrist 3	Wrist 2	$4 \cdot N_{\text{p}}$
Shoulder, Elbow, Wrist 2, Wrist 3	Wrist 3	$5 \cdot N_{\text{p}}$

Shoulder and Elbow and Wrist 1 and Wrist 2 can be omitted, reducing the number of collision constraints to $N_{\text{dyn}} = 25$ for a single time step.

The number of collision constraints can be further reduced by considering the specific setup. Due to the relative distances between the robots, not all potential collision pairs need to be considered, as some pairs may never intersect geometrically. For instance, in the given setup, one robot cannot reach the base of the other, and the shoulders of both robots cannot collide. Consequently, the number of constraints to be considered reduces to $N_{\text{dyn}} = 14$ for each time step. For a prediction horizon of $N_{\text{p}} = 20$, this totals 280 constraints, in addition to other constraints in the DMPC problem (4.26). Table 5.2 provides a summary of the collision-prone link pairs after preprocessing. In the considered setup, two DMPC problems of the form (4.26) need to be solved.

Finally, to prevent self collisions, upper and lower bounds on joint angles are imposed to restrict the angle motion of each joint. The chosen parameters are listed in Table 5.1.

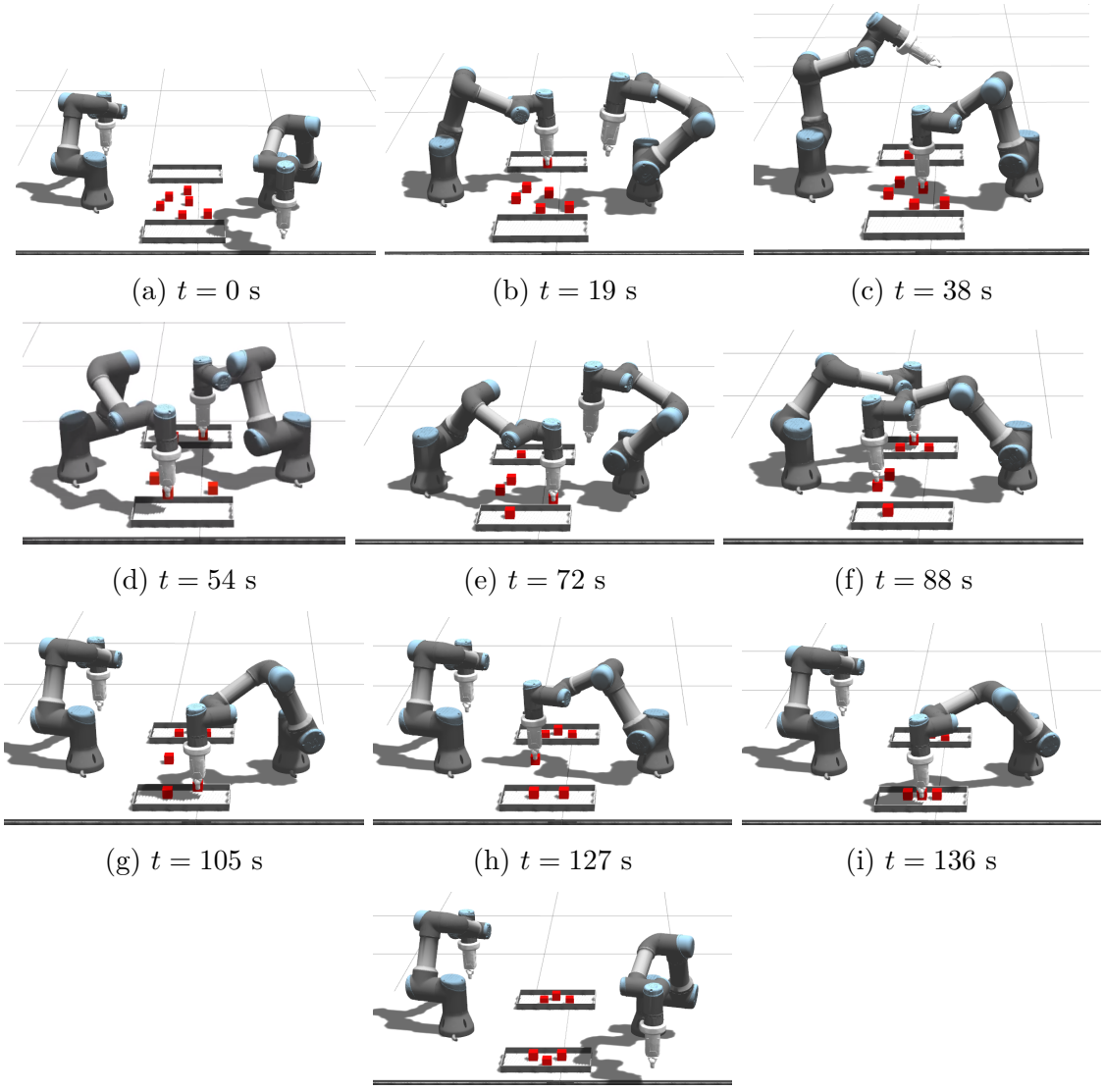
5.3 Benchmark analysis of scheduling approaches

In the following, a benchmark analysis of the heuristic and the optimization-based scheduling approaches is conducted, see Sec. 4.3.3 and Sec. 4.3.2. To this end, 20 samples of six consecutive pick-and-place tasks with different object positions in the shared workspace are investigated. Further, three different prediction horizon lengths $N_{\text{p}} \in \{10, 15, 20\}$ are considered to investigate the influence of the prediction horizon length on the performance of the DMPC-ELS method.

Choosing a proper prediction horizon length can reduce computational burden, but it might have a crucial effect on the performance of MPC. On the one hand, too short prediction horizon length can result in a jerky motion of the robot in close proximity to an obstacle, as MPC might be unable to anticipate an obstacle in time and hence, have difficulties to plan a smooth trajectory circumventing it. On the other hand, long horizon length can have a consequence of high computational load, that is impractical for real-time motion planning [Zhu+24].

5.3.1 Heuristic task assignment

The performance of the proposed control approach is demonstrated exemplary on the basis of one selected sample, for which a prediction horizon length of $N_\mathrm{p} = 20$ is chosen. The tasks are equally distributed among both robots by employing the heuristic scheduling approach, see Sec. 4.3.2. Considering six objects in total, each robot's task is to place three of the randomly distributed objects into slots of the assigned trays. The position of each robot's basis and the objects as well as the distances between the assigned objects are provided in Table 5.3. Fig. 5.3 illustrates the robots' trajectories for ten distinct time steps extracted from the simulation environment *Gazebo*. The robots in their initial position are shown in Fig. 5.3a. The heuristic scheduling approach does not consider minimal



(a) $t = 0$ s (b) $t = 19$ s (c) $t = 38$ s

(d) $t = 54$ s (e) $t = 72$ s (f) $t = 88$ s

(g) $t = 105$ s (h) $t = 127$ s (i) $t = 136$ s

(j) $t = 150$ s

Figure 5.3: Selected time frames from *Gazebo* simulation during the sorting task using the heuristic task assignment. The video of the simulation is available at the following URL [Gaf+22a].

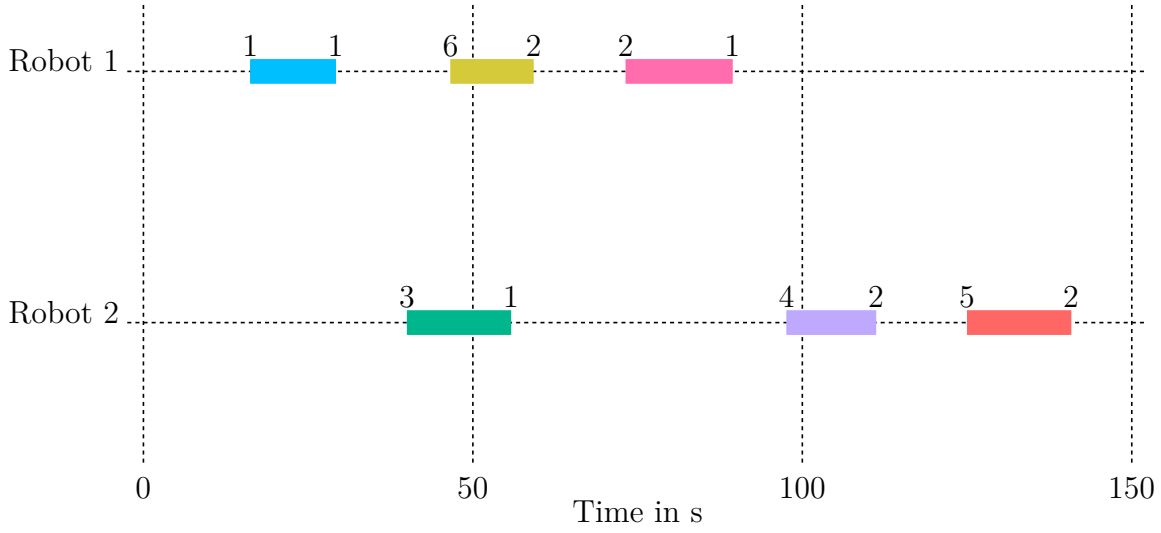


Figure 5.4: Gantt chart of heuristically computed schedule.

distances between adjacent objects. Thus, two objects might be assigned to both robots which cannot be grasped simultaneously resulting in a deadlock. A sequential grasping is then required to resolve the deadlock. This is the case for the first assigned objects to the robots, where the distance between the assigned objects is below the minimum deadlock-free distance, i.e., $d_{13} < 0.12$ m, leading to a deadlock between the robots. The deadlock is resolved, where Robot 2 moves to its neutral pose allowing Robot 1 to grasp its object, illustrated in Fig. 5.3b. This results in a delay of Robot 2 which grasps its first object while Robot 1 already finished its first task. The second deadlock occurs for the last task of Robot 1 and the second task of Robot 2, which is resolved at time $t = 72$ s as shown in Fig. 5.3e. The objects lie on opposite sides of each robot, so that simultaneous grasping is physically impossible. Once the Robot 1 grasps its last object and moves away, Robot 2 can move thereafter to its objects, depicted in Fig. 5.3f. Comparing Fig. 5.3g–5.3i, this leads to an additional delay of Robot 2 resulting in the last two pick-and-place tasks of Robot 2 to be performed by this robot alone.

The Gantt chart in Fig. 5.4 shows the distribution of the six objects to the trays between the two robots and their execution of the tasks over time. The number on the top-left of each bar indicates the ID $\in \{1, 2, 3, 4, 5, 6\}$ of the object which has to be grasped by the respective robot, whereas the number on the top-right of each bar specifies the assigned tray. The length of each bar represents time-to-completion for a specific pick-and-place task. The total duration to accomplish all pick-and-place tasks by both robots is denoted as execution time T_e. The Gantt chart illustrates the effect of both deadlocks on the execution time of Robot 2 resulting in an execution time of 140 s. A closer look at the Gantt chart indicates that the first two tasks were assigned to the same tray. In other words, both robots have to serve the same tray which leads in any case to a deadlock,

as the objects cannot be placed into the same tray simultaneously. The chosen scenario demonstrates that the heuristic scheduling approach often leads to deadlocks in realistic pick-and-place tasks.

Table 5.3: Robot and object positions and distances for the selected sample.

Robot's basis in m	
$R_{1,0} = [0.0,\ 0.0,\ 1.107]$	$R_{2,0} = [0.7,\ 0.0,\ 1.107]$
Object positions in m	**Distance between the objects in m**
$\mathbf{p}_1 = [0.356,\ 0.0949,\ 1.107]$	$d_{13} = 0.1056,\ d_{14} = 0.1826,\ d_{15} = 0.0947$
$\mathbf{p}_2 = [0.437,\ -0.125,\ 1.107]$	$d_{23} = 0.1366,\ d_{24} = 0.1974,\ d_{25} = 0.222$
$\mathbf{p}_3 = [0.364,\ -0.01,\ 1.107]$	$d_{31} = 0.1056,\ d_{32} = 0.1366,\ d_{36} = 0.1332$
$\mathbf{p}_4 = [0.252,\ -0.055,\ 1.107]$	$d_{41} = 0.1826,\ d_{42} = 0.1974,\ d_{46} = 0.1229$
$\mathbf{p}_5 = [0.283,\ 0.035,\ 1.107]$	$d_{51} = 0.0947,\ d_{52} = 0.2220,\ d_{56} = 0.1852$
$\mathbf{p}_6 = [0.34,\ -0.141,\ 1.107]$	$d_{63} = 0.1332,\ d_{64} = 0.1229,\ d_{65} = 0.1852$

5.3.2 Optimization-based task assignment

In the following, the tasks are distributed among the robots using the optimization-based scheduling approach, introduced in Sec. 4.3.3. To conduct a comparison between the two scheduling methods, the identical sample of six pick-and-place tasks is considered as in the previous Sec. 5.3.1 with a prediction horizon length of $N_\mathrm{p} = 20$. The sequence of pick-and-place tasks assigned to both robots can be viewed in the Gantt chart presented in Fig. 5.5. In addition, the Gantt chart provides information about the execution of each pick-and-place task.

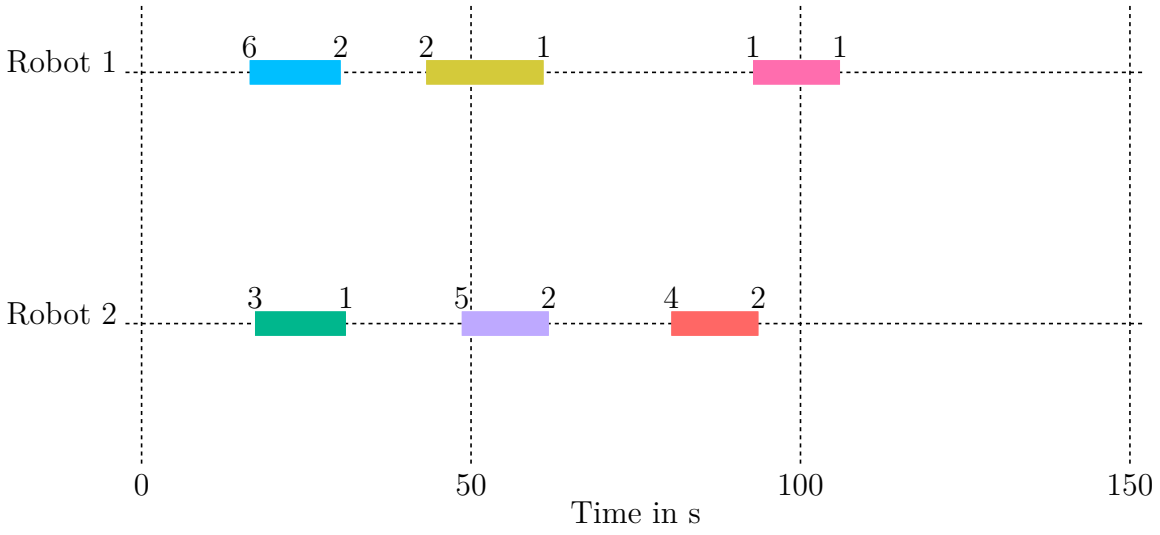
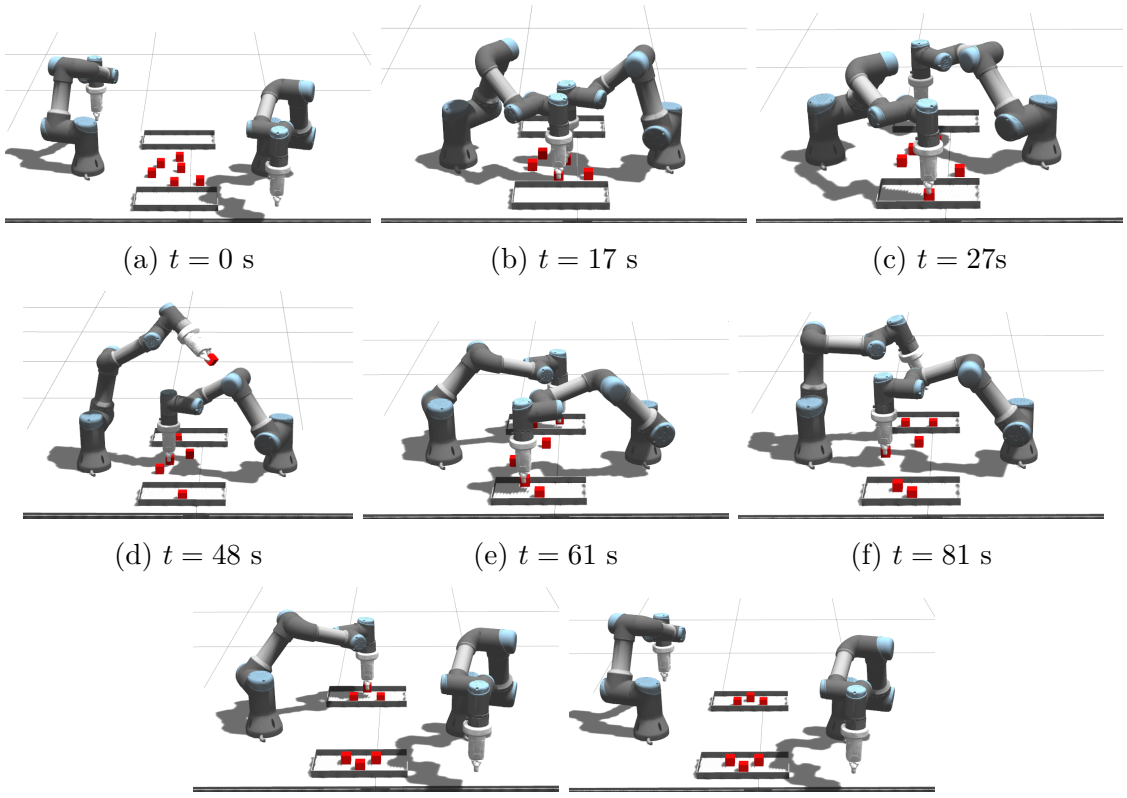


Figure 5.5: Gantt chart of the optimization-based scheduling approach.

The Gantt chart of the optimization-based scheduling approach shows that the scheduler managed to assign both robots to different trays at the same time. Fig. 5.6 illustrates several time frames of the respective *Gazebo* simulation. The alternating serving of the trays can be seen throughout the selected time frames, where robots serve different trays at the same time. In addition, simultaneous grasping is possible for the first two assigned objects, as the optimization-based scheduling approach considers a minimum grasping distance. However, as only a minimum distance between the objects and not between the robots' links is considered, it may occur that a simultaneous grasping is not always feasible. Fig. 5.6f illustrates this case for the last assigned objects. However, as Robot 2 reaches its object earlier than Robot 1, no deadlock occurs between the robots. Once Robot 2 has grasped its object, Robot 1 can proceed with its last task and places the last object into the assigned tray some time later, see Fig. 5.6g. This introduced delay between the robots can be determined from the Gantt chart and amounts to 12 s. Nevertheless, the delay is still considerably lower than resolving an occurred deadlock. Thus, as demonstrated, it is not always possible to prevent all deadlocks a priori by optimal task scheduling alone.



(a) $t = 0$ s (b) $t = 17$ s (c) $t = 27$s

(d) $t = 48$ s (e) $t = 61$ s (f) $t = 81$ s

(g) $t = 104$ s (h) $t = 115$ s

Figure 5.6: Selected time frames from *Gazebo* simulation during the sorting task using the optimization-based task assignment.

5.3.3 Discussion

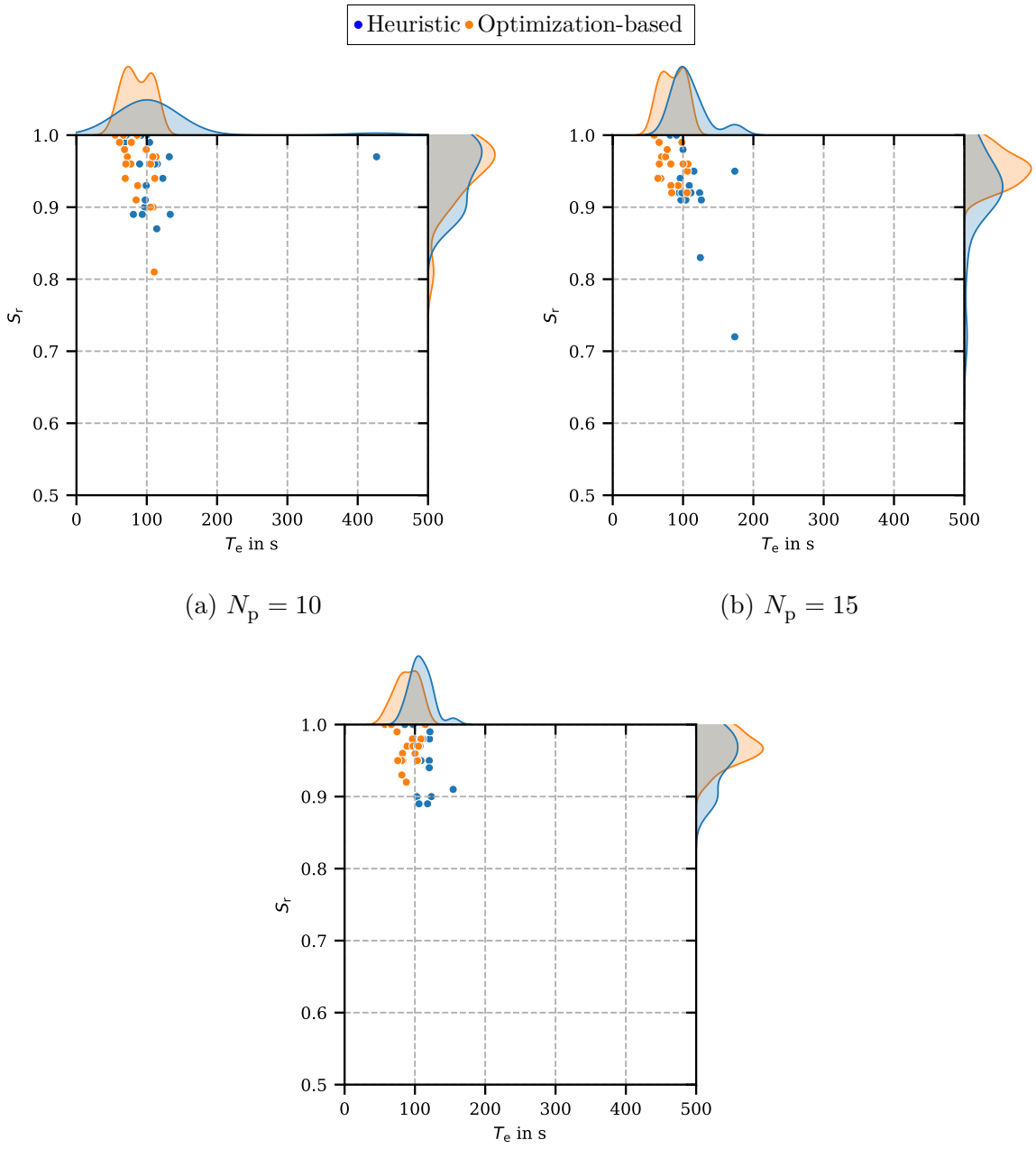
In the following section, the results of all 20 sampled pick-and-place scenarios performed with the proposed TAMP approach and two different scheduling methods are compared and discussed. In addition, the influence of the prediction horizon length on the DMPC-ELS approach is investigated. Therefore, success rate S_r over execution time T_e has been evaluated for different prediction horizon lengths $N_\mathrm{p} \in \{10, 15, 20\}$. The results are presented in Fig. 5.7. The success rate S_r represents the number of time steps without any deadlocks divided by the total number of time steps. Additionally, Table 5.4 provides means and standard deviations of execution times and success rates for both scheduling approaches.

Table 5.4: Mean and standard deviations of success rates and execution times for 20 samples.

	Heuristic		**Optimization-based**	
$\mathbf{N_p}$	$\mathbf{T_e} \pm$ **std**$(\mathbf{T_e})$	$\mathbf{S_r} \pm$ **std**$(\mathbf{S_r})$	$\mathbf{T_e} \pm$ **std**$(\mathbf{T_e})$	$\mathbf{S_r} \pm$ **std**$(\mathbf{S_r})$
10	122.36 ± 21.61	0.947 ± 0.0438	86.98 ± 19.24	0.953 ± 0.0462
15	113.55 ± 24.35	0.923 ± 0.0610	85.43 ± 16.51	0.955 ± 0.0235
20	108.04 ± 14.41	0.952 ± 0.0373	87.37 ± 15.04	0.968 ± 0.0230

The results for the heuristic scheduling approach show that the mean execution time and its standard deviation considerably drop with longer prediction horizon lengths. Heuristic task assignments can often lead to unfortunate intersections between the robots that can result in potential collisions between multiple links that have to be avoided or even cause multiple deadlocks. With a longer prediction horizon each robot receives predicted trajectories of its neighbors over longer period of time that makes possible to react upon potential collisions faster and compute better trajectories resulting in lower execution times. In comparison, no significant influence of the prediction horizon length on the mean execution time can be observed in case of the optimization-based scheduling approach. Nevertheless, the mean execution time is significantly lower compared to the heuristic scheduling approach. Concerning the success rate for using the heuristic scheduling approach, the means and standard deviations remain in the same range, almost independent of the prediction horizon length. The same holds true for the optimization-based scheduling approach. Furthermore, the scattering of success rates, i.e., the standard deviations, decreases with longer prediction horizon length for both scheduling methods. However, the success rate for the optimization-based scheduling approach is higher than for heuristic scheduling approach. This can be attributed to the fact that on average more deadlocks occur and thus more outliers are present in case of heuristic task assignment. With the optimization-based scheduling approach deadlocks can be prevented by assigning tasks in an optimal manner. In other words, deadlocks can be considered a priori by assigning objects above a deadlock-free distance to be grasped and by the constraint that each tray

may only be served by one robot at a time. Overall, employing the optimization-based scheduling approach in the proposed TAMP leads to considerably shorter execution times and higher success rates. This investigation demonstrates that the occurrence of deadlocks can considerably be reduced by imposing additional constraints in the scheduling layer and assigning tasks to the robots in an optimal manner.



(a) $N_p = 10$ (b) $N_p = 15$

(c) $N_p = 20$

Figure 5.7: Distribution of success rate over execution time of the 20 considered samples with the heuristic and the optimization-based scheduling methods for different prediction horizon lengths.

5.4 Benchmark analysis of trajectory planners

In the following, the performance of the DMPC-ELS approach is compared with several state-of-the-art trajectory planners. The same setup involving two robots, introduced in Sec. 5.3, is used. As in the benchmark analysis in Sec. 3.3 for a single manipulator system, three sampling-based planners – namely PRM, PRM*, and RRT-Connect – and the optimization-based planner CHOMP are selected. The methodologies of the trajectory planners are provided in Table 3.2. The main difference in this comparison is the substitution of BiTRRT with PRM*[KF11]. This substitution was made to include an optimal planner among the sampling-based planners, allowing its performance to be assessed relative to CHOMP and DMPC-ELS. PRM* is an asymptotically optimal version of PRM. More details about each planner can be found in Sec. 2.1.4 and Sec. 2.2.

For the benchmark analysis, the selected state-of-the-art trajectory planners are employed to compute the trajectory from initial to target pose of both robots simultaneously. After the planning phase, the trajectories of both robots are executed concurrently. In total, 20 samples of consecutive pick-and-place tasks are considered, with each sample involving six randomly distributed objects, as shown in Fig. 5.1. For each sample, the tasks are allocated between the robots using both the heuristic and the optimization-based scheduling method. To ensure comparability between the planners, the same deadlock resolution procedure outlined in the proposed TAMP, introduced in Sec. 4.2, is implemented across all planners.

5.4.1 Performance metrics

The same performance metrics used in the benchmark analysis in Sec. 3.3.1 for a single robot are also selected to evaluate the quality of the generated trajectories for multiple robots: path length, execution time, planning time, and trajectory smoothness. It is important to note that the execution time T_e refers to the time required to complete all tasks by both robots. The safety distance to the environment has been replaced with the success rate S_r, which represents the percentage of successfully planned tasks simultaneously completed by all robots. For the DMPC-ELS, the success rate indicates the number of time steps during which both robots performed pick-and-place tasks simultaneously, divided by the total number of time steps. For global planners, the success rate reflects the number of planning cycles in which a solution was found for both robots, divided by the total number of planning cycles. During the remaining time steps or planning cycles, when robots are resolving a deadlock, only one robot executes its task while the rest of deadlock-affected robots wait in their neutral pose.

5.4.2 Heuristic task assignment

The results for the 20 samples of six consecutive pick-and-place tasks obtained with the heuristic scheduling approach are provided in Table 5.5 for both robots. In the table, the planner RRT-Connect is abbreviated as RRTC. Also, Fig. 5.8 summarizes the four considered performance metrics for Robot 1. The results for Robot 2 are similar to Robot 1 and are therefore omitted. As the planning time is part of the execution time, the two

Table 5.5: Results for 20 pick-and-place task samples using a two-robot setup and the heuristic scheduling approach [Gaf+22a].

Planner	$\mathbf{N_p}$	**mean($\mathbf{T_p}$)± std($\mathbf{T_p}$)** in ms		**$\mathbf{T_e}$± std($\mathbf{T_e}$)** in s	**L± std(L)** in m		**s± std(s)** in $\frac{rad}{s}$		**$\mathbf{S_r}$**
		Robot 1	**Robot 2**		**Robot 1**	**Robot 2**	**Robot 1**	**Robot 2**	
DMPC	10	32.88 ± 14.34	31.84 ± 18.24	122.36 ± 21.61	5.02 ± 1.35	5.80 ± 1.58	27.40 ± 7.24	28.33 ± 6.57	**95%**
	15	57.47 ± 24.41	59.89 ± 26.53	113.55 ± 24.35	5.03 ± 1.25	5.41 ± 1.47	25.62 ± 6.67	26.20 ± 8.27	**95%**
	20	75.99 ± 34.76	78.66 ± 34.21	108.04 ± 14.41	5.01 ± 1.01	5.29 ± 0.85	25.67 ± 6.18	24.09 ± 2.84	**95%**
CHOMP	-	4643 ± 1129.90		266.66 ± 56.30	4.93 ± 1.04	6.71 ± 1.97	48.80 ± 26.48	49.78 ± 19.64	**43%**
PRM	-	299.50 ± 357.34		70.47 ± 13.10	7.30 ± 2.46	8.10 ± 2.90	45.04 ± 11.10	48.74 ± 9.58	**75%**
PRM*	-	10066 ± 29.81		308.59 ± 56.61	5.73 ± 1.47	5.79 ± 1.55	38.72 ± 7.92	43.98 ± 6.95	**76%**
RRTC	-	53 ± 15.93		65.60 ± 12.08	6.19 ± 1.98	7.43 ± 2.76	42.27 ± 8.85	47.00 ± 9.57	**72%**

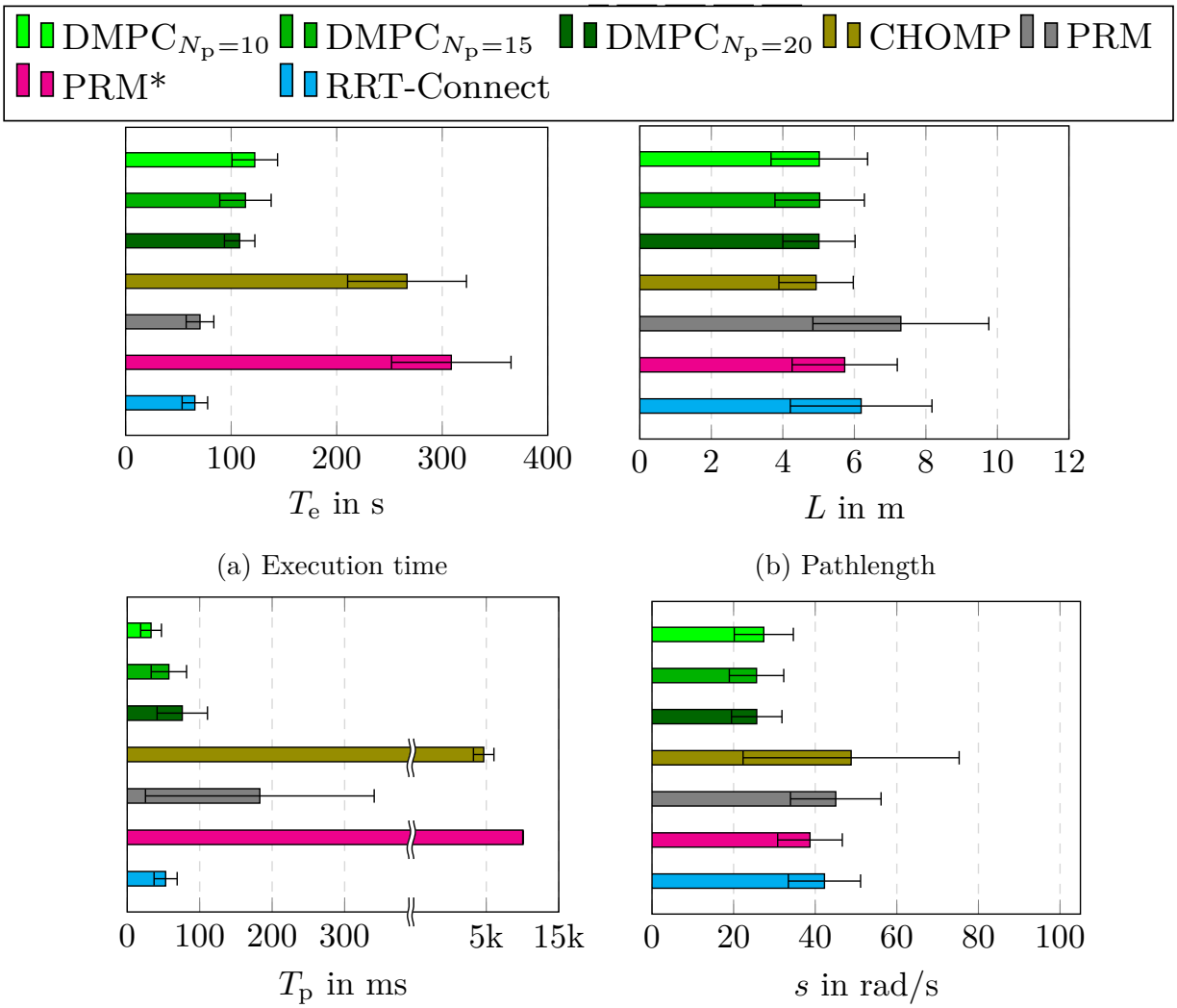
performance metrics cannot be considered independently of each other. The longest execution time can be observed for PRM*, see Fig. 5.8a. Sampling-based planners guarantee probabilistic completeness, i.e., a solution is found within a finite amount of time if one exists. In case of PRM* this comes at the cost of predefining a maximum planning time that stops improving the found trajectory. A maximum planning time of 10 s is chosen for PRM*. As can be seen in Fig. 5.8c, PRM* needs on average about 10^4 ms to find a path to the target, i.e., the planner needs the maximum planning time of 10 s that is specified. As the longest planning time arises for PRM*, it also leads to the longest execution time. The second longest execution time is observed for CHOMP. CHOMP needs on average 4643 ms to plan a trajectory for both robots, the second longest of all planners, which explains the results. The execution times of the other planners are considerably shorter. This can be mainly attributed to the significantly shorter planning times of PRM and RRT-Connect. This is especially true for RRT-Connect, that needs on average only 50 ms to plan a trajectory for both robots. In case of DMPC-ELS, a deacreasing tendency of execution times with increasing length of prediction horizon can be observed. However, the computation time increases with longer prediction horizon as larger optimization problems have to be solved.

A short execution time does not necessarily correlate with a short pathlength. The planners PRM and RRT-Connect yield the shortest execution times, however resulting also in the longest pathlength compared to the other planners. The planners compute a feasible solution rapidly which might not necessarily be the best one. The results of DMPC-ELS show no significant improvement of pathlength with increasing prediction horizon length. Also CHOMP manages to compute the shortest pathlength similar to the DMPC-ELS

approach.

Concerning the smoothness of the generated trajectories, the DMPC-ELS approach computes the smoothest trajectories compared to all state-of-the-art planners. This is due to the fact, that the investigated global planners provide jerky movements. Especially CHOMP shows the highest standard deviations of trajectory smoothness.

Finally, the success rate of the planners is investigated. CHOMP yields the lowest success rate among all the planners which amounts to 43 %. Although, CHOMP manages to find a trajectory for both robots in half of the cases, several collisions between the robots could be observed. In 57 % of cases CHOMP fails completely to find a solution for both robots simultaneously. The success rates of the considered sampling-based planners are comparable and lie between 72 % and 76 %. The DMPC-ELS planner manages to find a solution on average in 95 % of time steps, independent of the prediction horizon length.



(a) Execution time

(b) Pathlength

(c) Planning time

(d) Smoothness

Figure 5.8: Evaluation of performance metrics for the sorting task for 20 samples of pick-and-place tasks using the heuristic scheduling approach [Gaf+22a].

5.4.3 Optimization-based task assignment

In the next step, the identical 20 sampled pick-and-place tasks, the same planners and the optimization-based scheduling approach are considered. The performance metrics for both robots are reported in Table 5.6. In addition, the results for Robot 1 are illustrated in Fig. 5.9.

Table 5.6: Results for 20 pick-and-place task samples using a two-robot setup and the optimization-based scheduling approach.

Planner	$\mathbf{N_p}$	**mean(T_p)± std(T_p)** in ms		**T_e± std(T_e)** in s	**L± std(L)** in m		**s± std(s)** in $\frac{rad}{s}$		$\mathbf{S_r}$
DMPC	10	34.03 ± 18.86	34.27 ± 17.73	86.98 ± 19.24	4.77 ± 1.06	4.94 ± 1.13	13.76 ± 3.88	12.49 ± 3.75	**95** %
	15	57.89 ± 28.14	62.56 ± 59.76	85.43 ± 16.51	4.86 ± 1.21	4.85 ± 1.12	14.10 ± 5.00	13.07 ± 4.50	**96** %
	20	81.31 ± 44.52	83.86 ± 51.49	87.37 ± 15.04	4.96 ± 0.86	4.87 ± 1.11	14.84 ± 5.00	12.97 ± 5.37	**97** %
CHOMP	-	9376.50 ± 3902.96		427.70 ± 155.88	3.16 ± 1.62	3.16 ± 2.05	26.61 ± 15.82	23.63 ± 13.98	**42** %
PRM	-	199.50 ± 275.50		60.73 ± 12.90	5.86 ± 1.94	6.40 ± 2.39	38.11 ± 6.84	41.00 ± 10.51	**76** %
PRM*	-	10073.00 ± 36.14		297.17 ± 37.31	5.44 ± 1.27	5.28 ± 1.82	37.46 ± 5.99	40.03 ± 7.07	**80** %
RRTC	-	66.50 ± 40.56		58.37 ± 8.50	5.57 ± 1.37	6.03 ± 2.21	37.34 ± 6.50	40.71 ± 6.90	**76** %

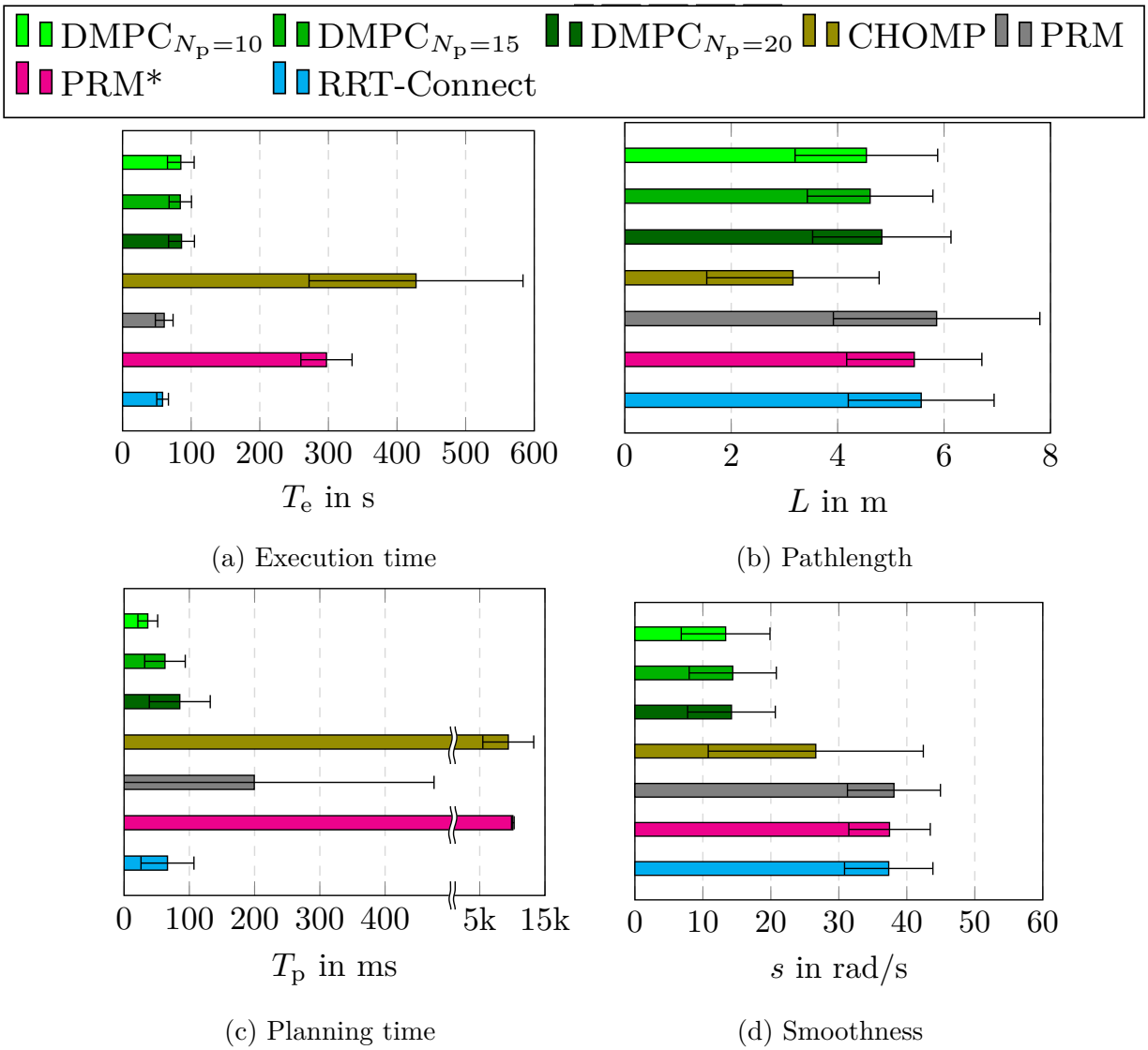
In contrast to the previous section, CHOMP leads to the longest execution time. This can be attributed to the fact that CHOMP has the second longest planning time, only slightly less than the planning time observed for PRM*. Incidentally, even though PRM* has the longest planning times, the planned trajectories need 120 s less on average to plan and execute a pick-and-place task for both robots. All other planners have significantly shorter execution and planning times. Particularly noticeable are the high standard deviations of planning times in case of PRM. These high fluctuations in planning times can be attributed to the lack of asymptotic optimality of the algorithm [KF11]. For the DMPC-ELS approach, it can be observed that an increase of the prediction horizon length has no significant influence on the length of the execution time. As expected, only computation times increase with increasing prediction horizon length as larger optimization problems have to be solved.

Concerning the pathlength of the planners, PRM, PRM* and RRT-Connect compute on average paths of similar length to a target for both robots. In contrast, CHOMP delivers the best results compared to all the other planners. The DMPC-ELS approach shows a slight increase in pathlength with increasing prediction horizon length.

The smoothest trajectories are computed by the DMPC-ELS approach with results that are almost independent of the prediction horizon length. PRM, PRM* and RRT-Connect compute trajectories of similar smoothness, while CHOMP delivers slightly better results than the other global planners, which still fall short of the DMPC-ELS approach by a huge margin.

Considering the success rate, CHOMP yields the worst results, i.e., 42 %, of all the investigated planners. In over half of the cases CHOMP fails to find a solution for both robots.

PRM* delivers the best result among the sampling-based planners with 80 % success rate. PRM and RRT-Connect have the same success rate, that is only 4 % lower than of PRM*. The DMPC-ELS approach yields the best results compared to all planners, that amounts to 97 % for a prediction horizon length of $N_\mathrm{p} = 20$.



(a) Execution time

(b) Pathlength

(c) Planning time

(d) Smoothness

Figure 5.9: Evaluation of performance metrics for the sorting task for 20 samples of pick-and-place tasks using the optimization-based scheduling approach.

5.4.4 Scalability analysis

In a multi-robot setup, especially if more than two robots are solving a joint task, reliable and efficient algorithms for trajectory planning are necessary. Therefore, it is investigated how well the state-of-the-art planners and the DMPC-ELS approach scale to more than two robots. The scalability is assessed by the success rate of the planners as it indicates how many time steps a planner was able to find a path to a target for all robots without executing a sequential pick-and-place procedure of one robot at a time. In conjunction to

the already investigated two robot setup, setups with three and four robot modules are investigated as well.

Fig. 5.10 visualizes a setup with three modules, which are connected in series. The manipulator in the middle has to cooperate with two robots in different workspaces, while the outer robots have only one neighbor. A setup with nine randomly placed objects in two different workspaces is considered. Each robot is assigned three objects which need to be placed in one of the four trays. Each tray can be served by two robots. The objects in the common workspace are accessible by both robots sharing the common workspace. Fig. 5.11 shows several selected time steps from one sample in a *Gazebo* simulation. Since the focus lies especially on the proposed DMPC-ELS approach, the selected time frames show trajectory planning executed with the DMPC-ELS approach. At the beginning, the interaction between two robots occurs between the robot on the right and the robot in the middle, see Fig. 5.11a. Fig. 5.11b illustrates the first assigned task of placing the grasped objects into the specified trays. The robot in the middle switches to the workspace on its left side, shown in Fig. 5.11c. Due to the fact, that the objects are in close proximity a deadlock occurs. Therefore, the robot on the left has to wait in its neutral pose, while the robot in the middle finishes its assignment. Several time steps later, the robot in the middle overcomes the collisions with its neighbor by choosing a trajectory over it to place its object into the assigned tray, see Fig. 5.11d. Next, the robot in the middle returns to the workspace shared with the robot on the right side. A closer look at Fig. 5.11e shows that at time step $t = 62$ s another collision avoidance between two robots occurs. Fig. 5.11f illustrates that after 80 s all the tasks are accomplished by all robots. For all samples the robots were able to reliably detect deadlocks and avoid collisions between each other.

Next, a setup with four modules and thus four manipulators is considered, see Fig. 5.12. The four manipulators are placed in the manner to form one common workspace. Thus,

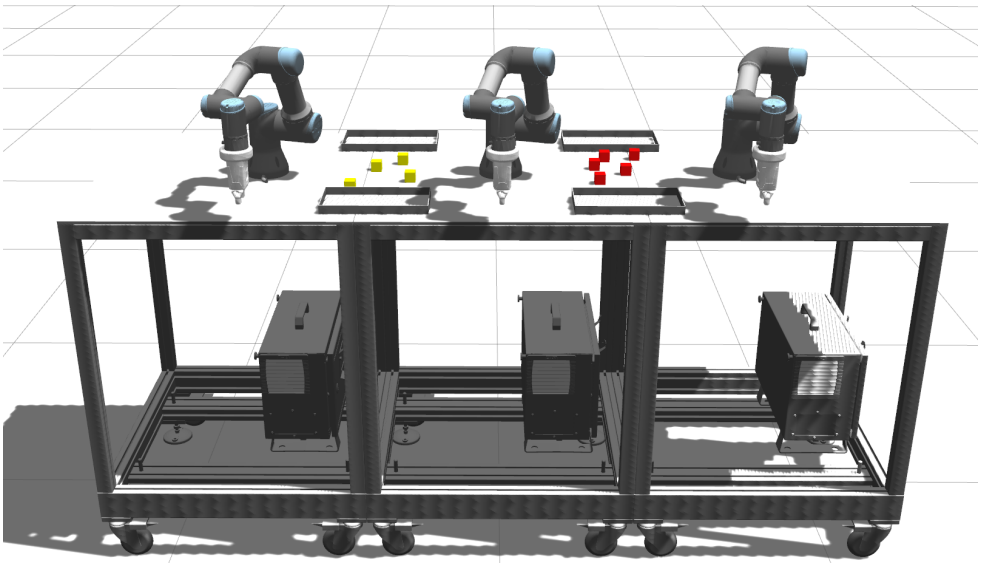
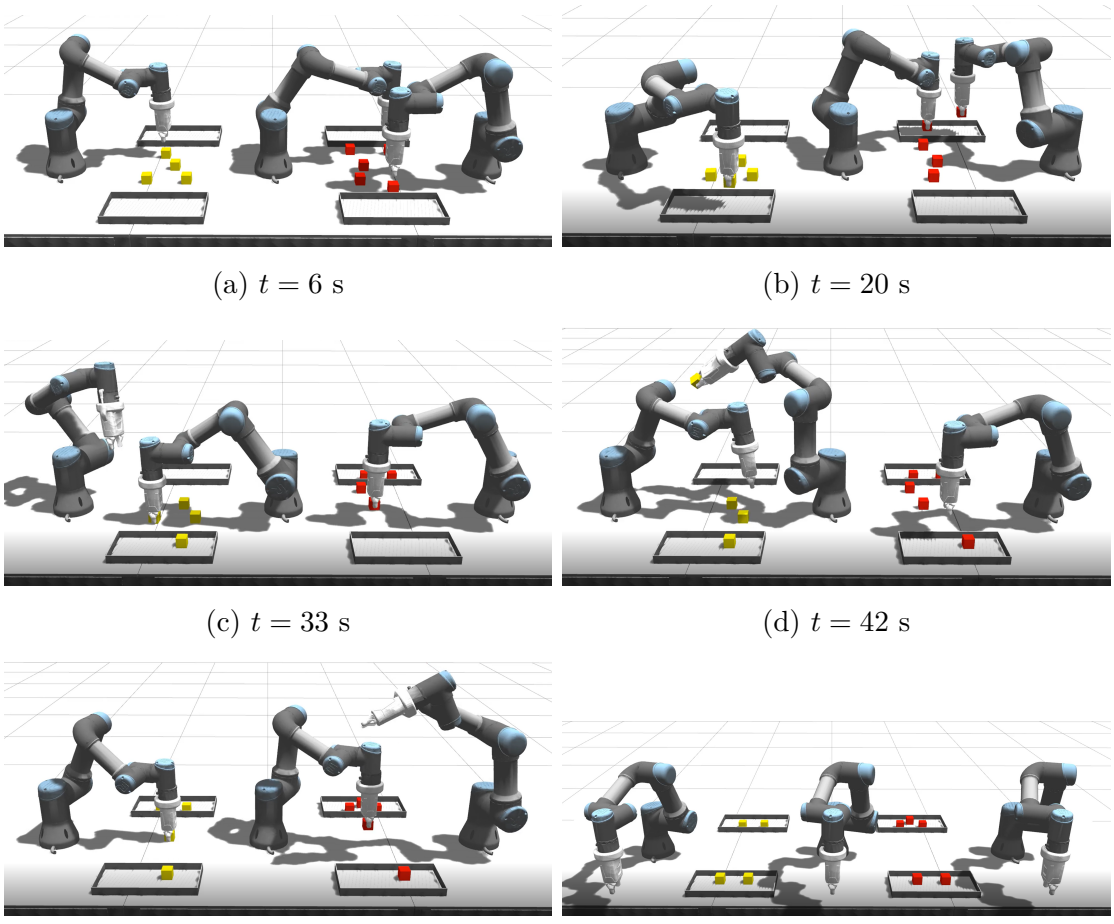


Figure 5.10: Simulation setup with 3 modules of UR3 robots [Gaf+22a]

multi-robot cooperation takes place for each robot with three neighbors. The robots are placed close to each other so that collisions between three robots are imminent. In this setup, twelve randomly placed objects and four trays are considered. Each tray can only be served by two robots. The goal of the robots is to sort all objects into the assigned trays. Two selected time frames in Fig. 5.12 show multi-robot cooperation performed with the DMPC-ELS approach. This constellation requires special care, as the number of collision avoidance constraints increases with the number of robots and a cluster of deadlock-affected robots should be reliably identified and differentiated from robots that are not involved in a deadlock. The robots successfully accomplished the joint task by avoiding collisions with each other and successfully resolving deadlocks.

Fig. 5.14a shows the success rates for the considered planners. The success rate of the DMPC-ELS approach is almost independent of the number of robots, i.e., in 95 % of time steps no deadlock is detected or resolved. The high success rate demonstrates that the proposed TAMP is highly efficient and reliable for different constellations of robots. Considering the global planners, the success rate decreases for an increasing number of robots, in general, except for CHOMP, where the planner delivers better results for the



(a) $t = 6$ s (b) $t = 20$ s

(c) $t = 33$ s (d) $t = 42$ s

(e) $t = 62$ s (f) $t = 80$ s

Figure 5.11: Selected time frames from the *Gazebo* simulation for three robots. The video of the simulation is available at the following URL [Gaf+22a].

linear setup of three robots than for two robots. The reason might lie in the specific setup of three modules, i.e., two different workspaces and a middle robot which alternates between both. That means, one of the outer robots operates for several pick-and-place tasks completely alone leading to a high success rate, as one of the three robots remains always deadlock-free. However, the performance of CHOMP drastically deteriorates for the setup of 4 robots, where the success rate merely amounts to 30 %. The results of the other planners, namely PRM, PRM* and RRT-Connect lie almost in the same range and decrease almost linearly with the number of robots. PRM* delivers slightly better results than PRM and RRT-Connect for a setup of two and three robots. Nevertheless, RRT-Connect leads to better results for a setup of four robots.

Further, the computation times of DMPC-ELS approach for the two setups of three and four robots are analyzed to draw conclusions about its applicability to multi-robot setups.

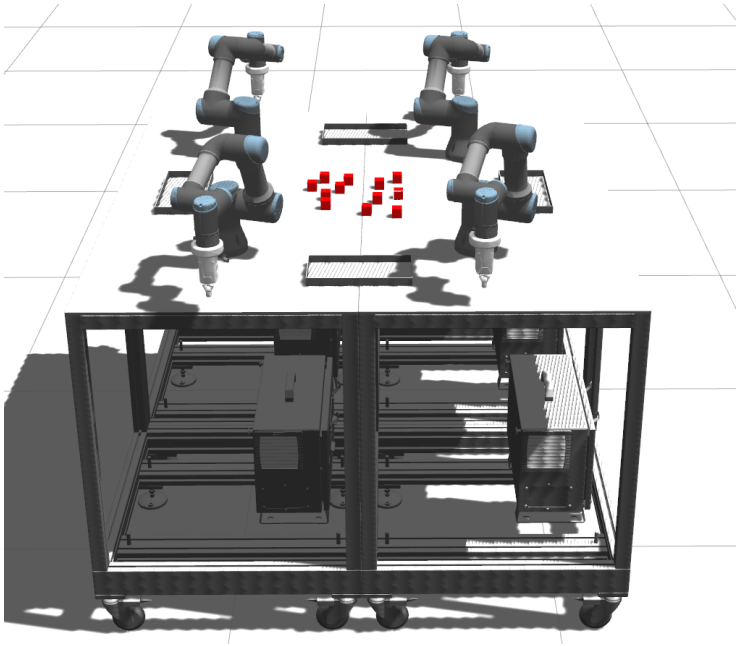
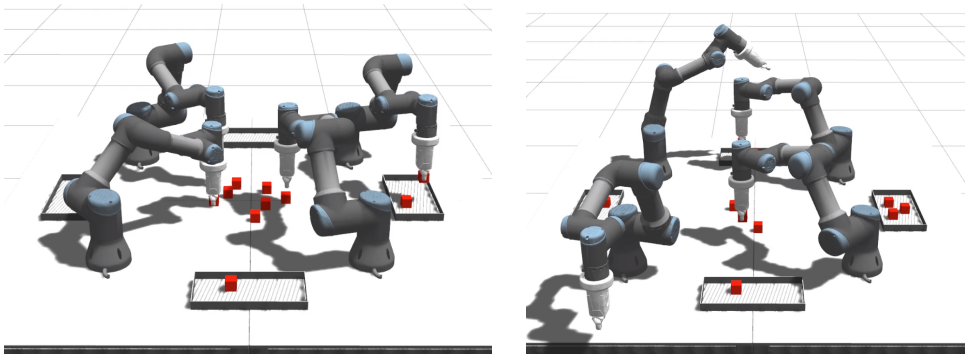
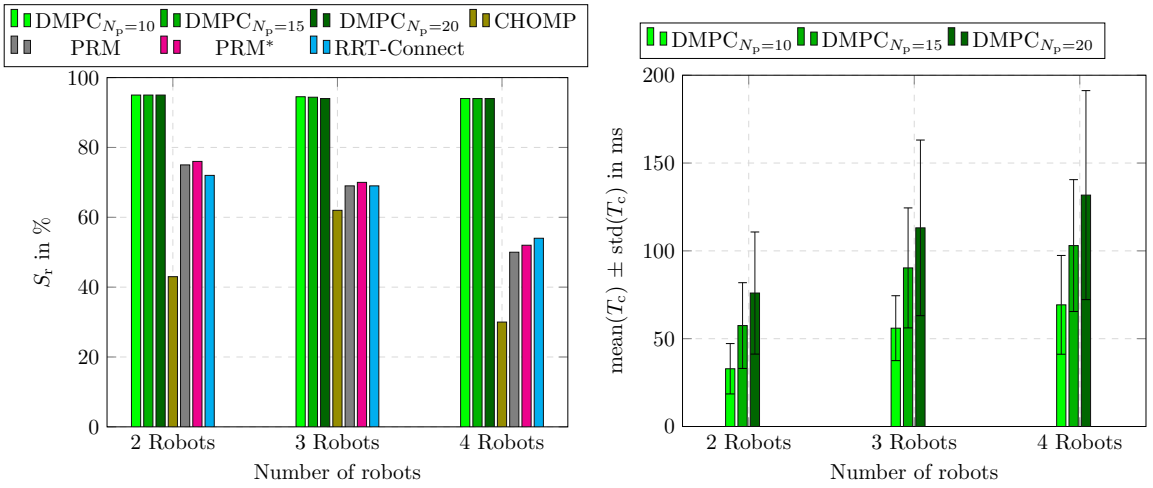


Figure 5.12: Simulation setup with four modules of UR3 robots [Gaf+22a]



(a) $t = 27$ s (b) $t = 46$ s

Figure 5.13: Selected time frames from the *Gazebo* simulation for four robots. The video of the simulation is available at the following URL [Gaf+22a].



(a) Success rate of the planners.

(b) Computation times of the DMPC-ELS.

Figure 5.14: Success rate of the planners and computation times of the DMPC-ELS approach [Gaf+22a].

Fig. 5.14b shows the mean computation times depending on the number of robots. In addition, the influence of the prediction horizon lengths on computation times are investigated. As expected, the mean computation times increase superlinearly with an increasing number of robots and prediction horizon length. However, the mean computation times and standard deviations do not exceed the sampling time of $T_s = 200$ ms for any number of robots and prediction horizon lengths. Thus, the approach shows a potential to compute optimal trajectories for more than 4 robots for shorter prediction horizon lengths.

5.4.5 Discussion

In summary, no significant impact of the scheduling approach, whether heuristic or optimization-based, was observed on the performance of the global planners. However, the performance of the global planners, assessed by the performance metrics, varied notably in certain cases.

PRM* demonstrated the highest success rate among the global planners, likely due to its asymptotic optimality, a characteristic absent in PRM and RRT-Connect. The dense search graph constructed by PRM* also resulted in the smoothest trajectories. However, PRM* consistently utilized the maximum predefined planning time, leading to higher execution times. While RRT-Connect and PRM performed well across the selected metrics, PRM slightly outperformed RRT-Connect with the heuristic scheduling approach, though both had similar success rates with the optimization-based approach. In contrast, CHOMP performed poorly across nearly all metrics. Regardless of the scheduling approach, CHOMP failed to generate a path for both robots simultaneously in over half of the cases, necessitating a sequential pick-and-place process that significantly increased execution time. More severely, several collisions were encountered between the robots for

trajectories planned by CHOMP. This might be attributed to the use of soft constraints for collision avoidance, which does not guarantee collision-free paths.

Scalability analysis of the global planners revealed difficulties in finding feasible paths as the number of robots increased. CHOMP exhibited the poorest scalability, with a success rate of only 30 % for a four-robot setup. The sampling-based planners showed an almost linear decline in success rate as the number of robots increased, achieving successful path planning in about 50 % of cases for four robots. These findings indicate that global planners are not well-suited for multi-robot setups, even in static environments as considered in this chapter.

The DMPC-ELS approach outperforms the investigated global planners across all selected performance metrics. The high success rate of over 95 % regardless of the scheduling approach, highlights its efficiency. Although computation times remain well below the sampling time of $T_\mathrm{s} = 200$ ms, they pose a bottleneck if additional constraints are introduced. A suitable choice of the prediction horizon length may alleviate the computational burden. Since the performance metrics show minimal improvement beyond $N_\mathrm{p} = 15$, this horizon length is optimal to choose. In scenarios involving robot-robot cooperation, each robot can independently replan its trajectory at each time step, ensuring that a change in one robot's target does not affect the other. This demonstrates that DMPC-ELS offers high flexibility for various multi-robot setups in static environments.

5.5 Summary

The simulation-based study presented in this chapter demonstrated the performance of the proposed TAMP approach applied to a team of manipulators operating within a common workspace. The study included several benchmark analyses to compare the proposed method with established techniques, particularly in the areas of task scheduling and trajectory planning. The simulations involved two manipulators, two trays, and six objects. The robots' joint goal was to sort randomly distributed objects in the shared workspace into the corresponding trays. To achieve this, 20 different sorting task samples were analyzed. By optimally assigning tasks between the robots, the execution times were significantly reduced. The optimization-based scheduling approach accounts for certain deadlock scenarios a priori. Although it cannot completely eliminate deadlocks, there is a notable improvement by optimally assigning tasks to the robots.

Additionally, the quality of the planned trajectories was assessed using five performance metrics: path length, execution time, planning time, smoothness, and success rate. The results indicated that the DMPC-ELS approach outperformed state-of-the-art planners in nearly all metrics. While state-of-the-art planners perform well in static environments,

they are effective only for single-robot setups. For example, CHOMP exhibited significant drawbacks in computing trajectories for multiple robots, with only 30% of tasks being executed simultaneously by four robots. Even with successfully planned trajectories, collisions between robots occurred. This leads to the conclusion that state-of-the-art planners are not suitable for multi-robot systems (MRS). Finally, the scalability analysis demonstrated that DMPC-ELS could plan collision-free trajectories for up to four robots with acceptable mean computation times. For shorter prediction horizons, setups with even more robots may be feasible.

6 Experimental study

This chapter is devoted to the experimental investigation of the proposed TAMP approach on a multi-manipulator system with two collaborative manipulators. The robots' joint goal is to solve a given task with minimal makespan through cooperation. The considered product is a truck composed of three parts: a cab, a chassis, and a load. The load can be either a trailer or a tanker. An object detection algorithm is trained to identify and classify objects in the robots' workspace. Various experiments are conducted to demonstrate collision avoidance, deadlock resolution, and the performance of the robots in disassembling or sorting different types of objects. The first experiment shows how effectively imminent potential collisions between the robots are avoided by employing the DMPC-ELS method. Additionally, three reproducible types of deadlocks between the two robots are demonstrated. Finally, two different experimental setups of robot-robot cooperation are introduced in detail. In the first setup, the robots sort different parts of a truck into the assigned trays. In the second, the two robots cooperatively disassemble three trucks into single parts, where the order of disassembly steps is crucial. Identical performance metrics are applied for comparing the proposed trajectory planner and state-of-the-art planners, as in the simulation-based study in Chapter 5.4.1. The task assignments for both experimental setups are executed using the heuristic and optimization-based scheduling approach. The videos of the experiments are available at the following URL [GWR24].

6.1 Experimental setup

The following experimental setup, depicted in Fig. 6.1, consists of two collaborative UR5e manipulators from *Universal Robots* with a payload of 5 kg each. With $N = 6$ joints, composed of Base, Shoulder, Elbow, Wrist 1, Wrist 2 and Wrist 3, each robot has a maximum reach of 850 mm. Each manipulator is mounted on top of a portable table, which is referred to as a module, having a dimension 80 cm $\times$ 80 cm $\times$ 107 cm. The distance between the robots' bases is 1.1 m in the positive y-direction. Both robots are equipped with a *Robotiq* 2F-85[1] gripper, which is attached to the end-effector (Wrist 3) with a maximum stroke of 85 mm. Grasping is performed identically to the previous

[1] https://robotiq.com/de/produkte/adaptiver-2-finger-robotergreifer-2f85-140 (Last visited: 18.03.2024)

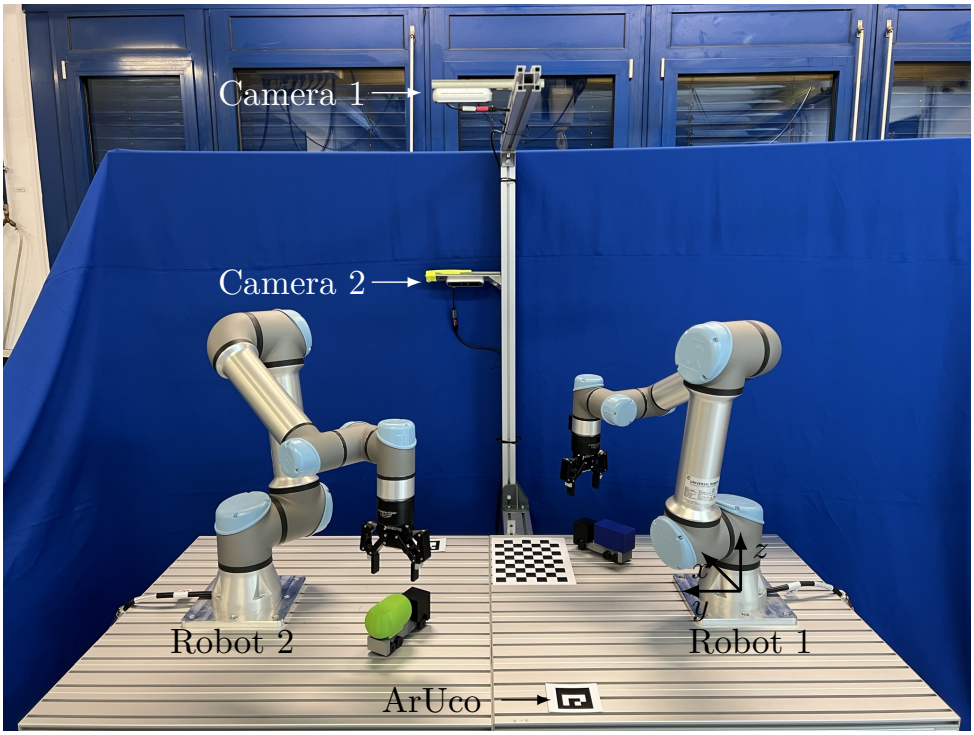


Figure 6.1: Experimental setup with two UR5e collaborative robots and two cameras.

experiments involving a single UR3 manipulator and is thus limited to straight up and down movements towards the object or away from it. The height from which an object is picked or placed for the considered experimental setup is denoted as grasping offset $g_{\text{offset}} = 0.13$ m.

For reasons of reliability, two Intel RealSense D455[2] depth cameras are employed. The upper camera (Camera 1), shown in Fig. 6.1, captures the top view of the workspace, whereas the side camera (Camera 2) captures the side view of the workspace. In case when objects become invisible to one of the cameras, i.e., an occlusion occurs, the other camera can still detect the objects in the common workspace. An occlusion might occur if robot's body blocks the visibility of a camera and thus the objects cannot be detected. A checkerboard pattern is used to calibrate the cameras and transform the objects' positions to the coordinate frame of a robot. In the considered experimental setup, the reference frame is set in the base of Robot 1.

Each ArUco marker [Gar+14] serves as a virtual tray for the objects. Uniqueness of each marker is guaranteed, where each marker has its own ID and to which a specific class of objects is assigned. Moreover, each ArUco marker can be freely placed in the workspaces of the robots, which allows to test different setup constellations. The multi-robot scheduling problem, formulated as MIQCP problem (4.18), is solved using Gurobi [Gur23]. The communication between the object detection unit, the robot control unit and the TAMP unit is established via ROS, running each on a different computer system unit.

[2]https://www.intelrealsense.com/depth-camera-d455/ (Last visited: 18.03.2024)

Each computer unit publishes and subscribes the necessary data within the same ROS network. The depth cameras are connected to the object detection unit, while the robots with the grippers are coupled to the robot control unit. ROS Master which itself runs on the robot control unit, receives optimal control inputs from DMPC-ELS and sends current robot's states back to it. The coordinator continuously receives data from both robots, however the deadlock resolution scheme is initiated once a deadlock has been detected. The hardware and software used for the experimental testbed are given in Table 6.1. The parameters chosen for DMPC-ELS and deadlock algorithm are provided in Table 6.2.

Table 6.1: Hardware and software used for the experimental testbed.

	Object Detection Unit	**Robot Control Unit**	**TAMP Unit**
Processor	Intel Core i5-8500@3.00GHz	Intel Core i5-9600K@3.7GHz	Intel Core i7-11800H@2.30GHz
RAM	16 GB	16 GB	16 GB
dGPU	Nvidia Quadro P1000	-	Nvidia T1200 Laptop GPU
OS	Ubuntu 18.04	Ubuntu 18.04	Ubuntu 20.04
ROS	Melodic Morenia	Melodic Morenia (ROS Master)	Noetic Ninjemys
Testbed Components	2×Intel RealSense D455	2×UR5e 2×Robotiq 2F-85 gripper	-

Table 6.2: Parameters for DMPC-ELS and deadlock algorithm for UR5e.

DMPC-ELS Parameters	
Sampling time	$T_\mathrm{s} = 200$ ms
Prediction horizon length	$N_\mathrm{p} = 15$
Termination criterion (pose accuracy)	$\varepsilon_\mathrm{tol} = 4 \cdot 10^{-2}$ rad
Weighting matrices:	
- predicted states	$\mathbf{Q}_i^\mathrm{x} = \mathrm{diag}(1,1,1,0.2,0.2,1,2,0.1,0.1,0.01,0.01,0.01)^3$
- final state	$\mathbf{Q}_i^\mathrm{f} = 3\ \mathbf{Q}_i^\mathrm{x}$
- control inputs	$\mathbf{R}_i^\mathrm{u} = 0.1 \cdot \mathrm{diag}(1,1,1,0.1,0.1,0.1)\ \frac{\mathrm{s}^4}{\mathrm{rad}^2}$
- control input deviations	$\mathbf{R}_i^\mathrm{d} = 5 \cdot \mathrm{diag}(1,1,1,0.1,0.1,0.1)\ \frac{\mathrm{s}^6}{\mathrm{rad}^2}$
Limits on joint velocities	$\dot{\mathbf{q}}_\mathrm{max/min} = \pm[\pi/2, \pi, \pi, \pi, \pi, \pi]\ \frac{\mathrm{rad}}{\mathrm{s}}$
Limits on control inputs	$\mathbf{u}_\mathrm{max/min} = \pm[\pi/2, \pi, \pi, \pi, \pi, \pi]\ \frac{\mathrm{rad}}{\mathrm{s}^2}$
Upper bound on joint angles	$\mathbf{q}_\mathrm{max} = [2\pi, 0, 2.4, \pi, \pi, 2\pi]$ rad
Lower bound on joint angles	$\mathbf{q}_\mathrm{min} = [-2\pi, -\pi, -\frac{2}{3}\pi, -\pi, -\pi, -2\pi]$ rad
Deadlock Parameters	
Velocity tolerance	$\varepsilon_v = 1.5 \cdot 10^{-3}\ \frac{\mathrm{rad}}{\mathrm{s}}$
State tolerance	$\delta_x = 1.2 \cdot 10^{-2}$ rad
Clustering parameter	$d_\mathrm{min} = 0.23$ m

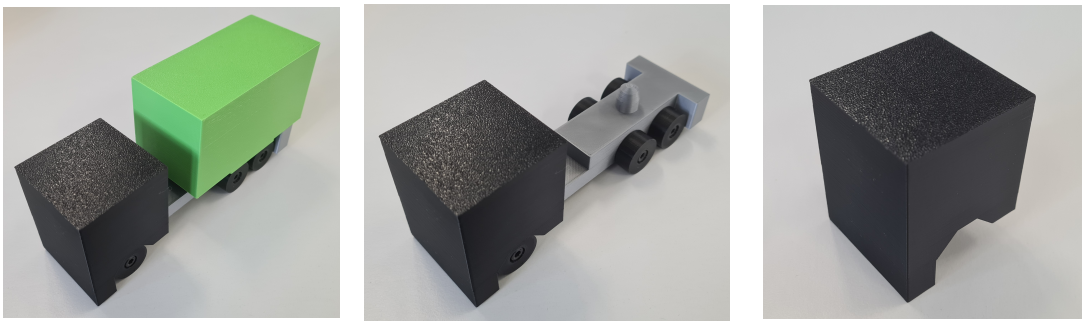
6.2 Object detection

In the following, the object detection algorithm is introduced which involves identification, localization and classification of objects. The main objectives are to localize and classify objects into different classes, and identify the objects' positions, orientations and dimensions. An object class consists of the object's geometry and its color. Prior to per-

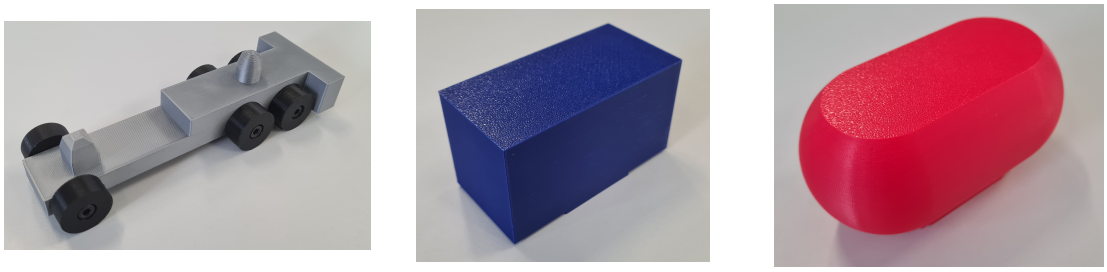
forming a pick-and-place task, each robot's task planner requires the object's position and its orientation relative to the world coordinate system (WCS).

An OpenCV [Bra00] checkerboard pattern, which is located between both robots, is used for camera calibration and extrinsic parameter estimation. First, each camera is calibrated individually. Calibration involves finding the focal length, optical center, and distortion coefficients. Second, the location and orientation of the checkerboard pattern with respect to the WCS needs to be determined. This is done by tracing specific points, i.e., the corners of the squares, with the end-effector of Robot 1 and logging its pose. Using the recorded positions of the checkerboard corners in the WCS, the rotation and translation vectors that transform points from the checkerboard's coordinate system to the WCS is determined using the Kabsch-Umeyama algorithm [Kab76; Ume91]. Third, the extrinsic parameters, i.e., rotation and translation vectors, which relate each camera's coordinate system to the WCS are determined. To this end, images of the checkerboard pattern are captured from both cameras[4]. All of these steps combined give translation vectors and rotation matrices to translate the captured objects from the local coordinate systems of the cameras into the WCS.

Fig. 6.2 shows a truck (Fig. 6.2a) and its components (Fig. 6.2c-6.2f), which are used as object classes for the following experiments. The truck consists of a cab, a chassis and a trailer or a tanker. A truck can also be partially assembled, as illustrated in Fig. 6.2b.



(a) Truck (b) Truck without trailer (c) Cab



(d) Chassis (e) Trailer (f) Tanker

Figure 6.2: Object classes for object detection algorithm.

Fig. 6.3 presents the architecture used for the object detection algorithm. Each camera

[4]https://github.com/IntelRealSense/librealsense/blob/master/wrappers/python/examples/box_dimensioner_multicam/box_dimensioner_multicam_demo.py

delivers RGB images and a point cloud. For classification of objects, the YOLOv5 [Joc20] real-time object detection algorithm is used. For model identification, a joint model of both cameras is trained, for which 756 images are captured by the upper camera and 758 images by the side camera, resulting in 1514 RGB images in total. The aim was to capture different constellations of trucks and single components in the common workspace. In the next step, the objects in the images are labeled by hand, i.e., suitable bounding boxes and object classes are assigned. The training data for YOLOv5 is split into a training (70 % of samples), validation (20 % of samples) and test set (10 % of samples). The number of training epochs amounted to 1000.

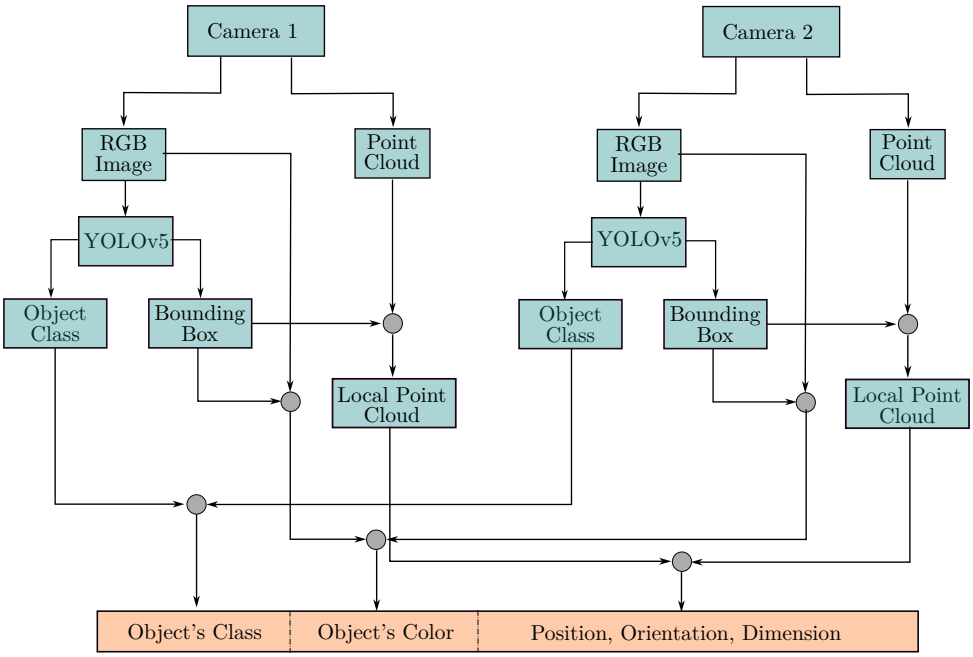


Figure 6.3: Architecture of the object detection algorithm.

After YOLOv5 is identified, the model is used for the output of both cameras. Fig. 6.4 demonstrates the output from YOLOv5 with respect to Camera 1 and Camera 2. For both cameras, YOLOv5 identifies the four objects placed in the common workspace and returns bounding boxes and classes. The location of the bounding boxes are already transformed into the WCS. In the next step, the color of each object is identified from the RGB images and the bounding boxes by using the KNN [MPP09] algorithm. Following color classes are distinguished for the experimental setup, which are red, blue, green, yellow, gray and black. The depth cameras are able to provide point clouds to obtain the dimensions of identified objects. The bounding boxes allow to restrict the region to the area occupied by an object in order to obtain a local point cloud. By merging the local point clouds from both cameras, the position, orientation and dimensions of the identified objects are obtained. The dimensions of an object include length, width and height.

The output of the object detection algorithm can be summarized as the vector of tuples $\underline{\mathbf{T}}_{\text{obj}} \in (\mathbb{R}^3 \times SO(3) \times \mathbb{R}^3 \times \Sigma)^{n_{\text{obj}}}$, where each tuple is comprised of the position vector

(a) Top view (Camera 1) (b) Side view (Camera 2)

Figure 6.4: Localization and classification of object classes with the upper (Camera 1) and side (Camera 2) cameras.

$\mathbf{p} \in \mathbb{R}^3$, the orientation vector $\mathbf{o} \in SO(3)$, the dimensions $\mathbf{d} \in \mathbb{R}^3$ and the object class $\sigma \in \Sigma$. The object class of an assembled or partially assembled truck is

$$\Sigma = \Sigma^{\text{Cab}} \times \Sigma^{\text{Chassis}} \times (\Sigma^{\text{Trailer}} \cup \Sigma^{\text{Tanker}}).$$

The object classes of truck's components, which are tanker, trailer, chassis and cab, are defined as follows

$$\Sigma^{\text{Tanker}} = \{\epsilon\} \cup \{\text{Tanker}\} \times \{\text{red, blue, green, yellow}\} = \{\epsilon, \text{rTA}, \text{bTA}, \text{gTA}, \text{yTA}\},$$

$$\Sigma^{\text{Trailer}} = \{\epsilon\} \cup \{\text{Trailer}\} \times \{\text{red, blue, green, yellow}\} = \{\epsilon, \text{rTL}, \text{bTL}, \text{gTL}, \text{yTL}\},$$

$$\Sigma^{\text{Chassis}} = \{\epsilon\} \cup \{\text{Chassis}\} \times \{\text{gray}\} = \{\epsilon, \text{gCH}\},$$

$$\Sigma^{\text{Cab}} = \{\epsilon\} \cup \{\text{Cab}\} \times \{\text{black}\} = \{\epsilon, \text{bCB}\}.$$

The letter ϵ indicates that the respective part of the truck was not recognized, i.e. is not present.

6.3 Collision avoidance

In the following experimental setup, the proposed collision avoidance approach of DMPC-ELS is demonstrated in case of imminent collisions between two robots. Fig. 6.5 presents the robots' states at selected time frames from the experiment. A blue trailer and a green container are used in the setup. The blue trailer is assigned to Robot 1, while the green tank has to be grasped by Robot 2. Both objects have to be placed on top of ArUco markers lying on opposite sides of the common workspace to provoke imminent collisions between the robots. The resulting joint angles for both robots are depicted in Fig. 6.5.

The states of the robots before and after picking up both objects are shown in Fig. 6.5b

and Fig. 6.5c, respectively. Figure 6.5 illustrates the grasping procedure, showing the progression of the joint angles at time $t = 8$ s during the straight down and up movements of both robots, affecting the joint angles q_2, q_3 and q_4. Grasping and placing of objects are performed from a specified height after the reference pose has been reached. To place the grasped objects into the assigned slots, each robot moves to the opposite side of the workspace. Fig. 6.5d illustrates that Robot 2 chose a path above Robot 1 to avoid inter-robot collisions. Besides that, Robot 1 avoided collisions with the module by tilting the gripper, which was performed by the joint angle q_5. The tilting of the joint can be observed in Fig. 6.5d. Due to the risk of collisions, both robots could not follow a direct path to their target poses. This can be observed for the joint angles q_2, q_3, q_4 and q_5, which show deviations in their trajectories from time step $t = 10$ s. To evaluate the closest proximity between the robots, the Euclidean distances between each link of one robot to each link of the other, including the grippers, are evaluated. The minimum distance between the links is shown in Fig. 6.6a. As depicted in the graph, the minimum distance between the robots decreases until time $t = 8$ s, as both robots move from their initial poses towards common workspace. The minimum distance remains constant during the grasping procedure and decreases further as the robots approach each other to place their objects on opposite sides of the shared workspace. Subsequently, the relative distance increases as the robots place their objects on top of the ArUco markers and move to their initial poses. Fig. 6.5e and Fig. 6.5f depict the placement of the objects and the robots returning to their starting positions.

The computation times for the experiment are given in Fig. 6.6b. Table 6.3 summarizes the mean, maximum and standard deviation of the computation times for the experiment. Mean and maximum computation times are significantly below the sampling time, with a small standard deviation for both robots. It should be noted that the pick-and-place procedure is executed by directly sending the setpoints to the underlying robots' velocity controllers, resulting in virtually zero computation times for MPC. The time before grasping the objects, that is, up to time $t = 7$ s, the mean computation times are approximately 57 ms for both robots, during which the robots' paths do not intersect. During the placement phase, the robots have to cross their common workspace to place their objects. As a result, the computation times increase to 120 ms right after the robots grasped their objects. The computation times rise as each robot receives the predicted trajectory of the other robot for the subsequent 3 s with most collision avoidance constraints active

Table 6.3: Mean, maximum and standard deviation of computation times of Robot 1 and Robot 2.

Computation times	**Robot 1**	**Robot 2**
Mean	72.15 ms	69.4 ms
Maximum	126.82 ms	127.37 ms
Standard deviation	17 ms	16.6 ms

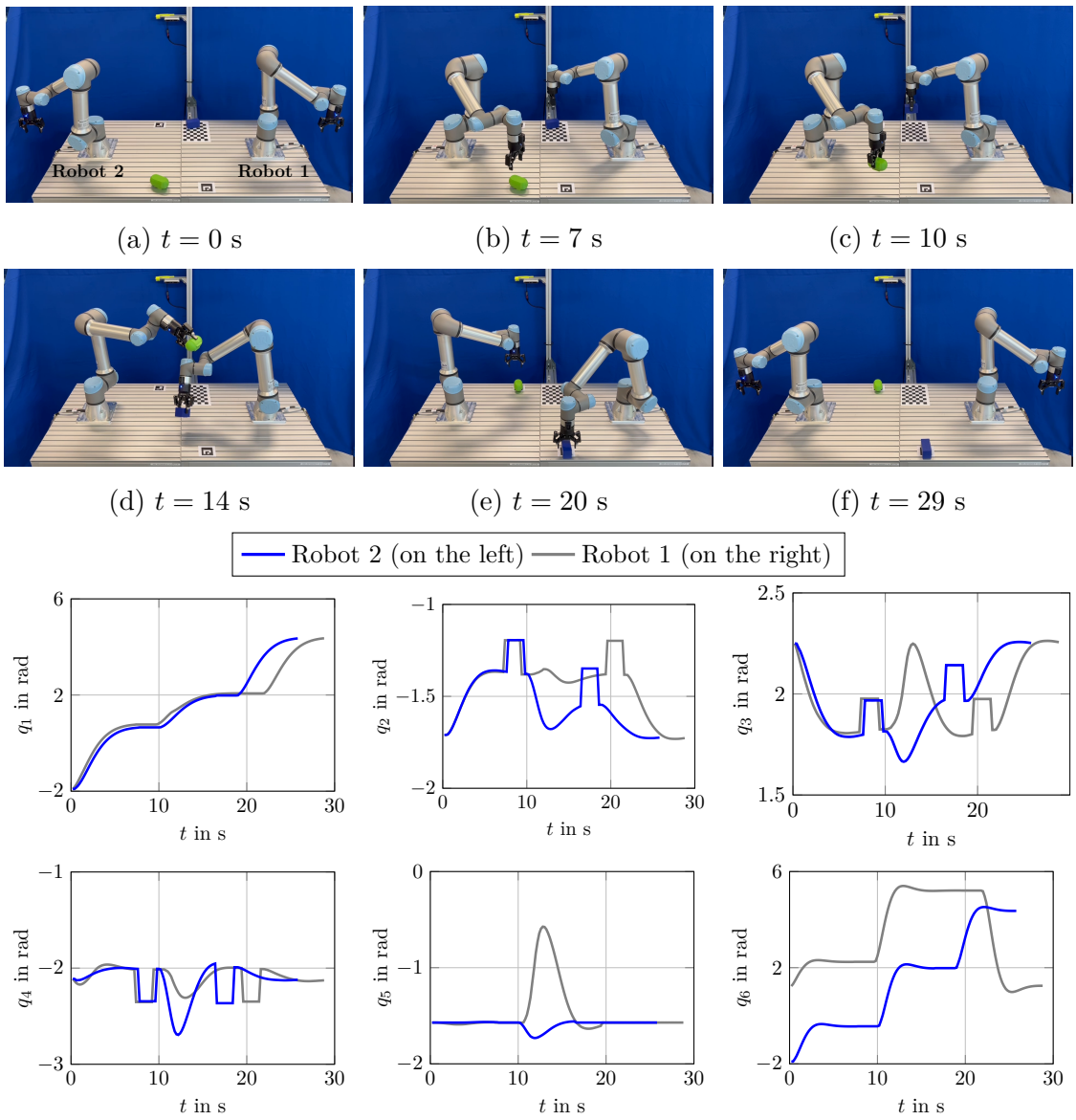
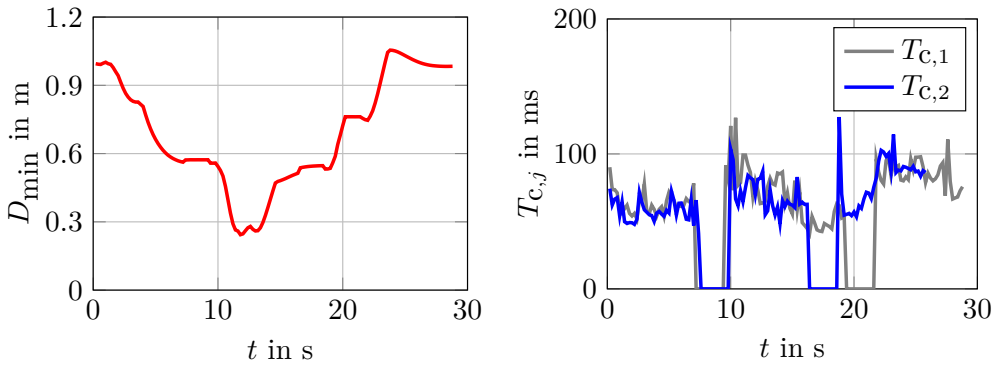


Figure 6.5: Selected time frames and joint angles of Robot 1 and Robot 2. The video of the experiment is available at the following URL [GWR24].



(a) Minimum distance (b) Computation times

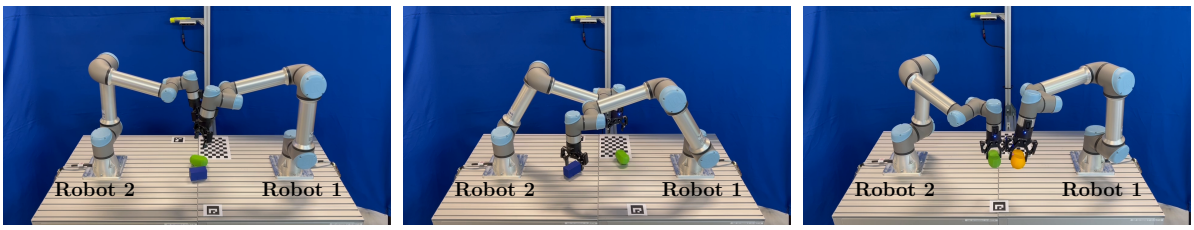
Figure 6.6: The minimum distance between the robots and computation time profiles for both robots.

during this period to prevent inter-robot collisions. Subsequently, the computation times decrease until the objects are placed. As Robot 2 places its object near the camera frame, which must be avoided, the computation times increase again at time $t = 20$ s again. This increase is not observed for Robot 1. Once the robots placed their objects, they return to their initial poses.

6.4 Deadlock

The following experimental investigation examines three distinct scenarios that result in deadlocks in a two-robot setup, each illustrated in Fig. 6.7. These scenarios investigate reproducible types of deadlocks that inhibit the simultaneous grasping or placement of objects by the robots. Generally, a deadlock may arise from the strict adherence to safety boundaries designed to prevent collisions between the robots.

In scenario 1, depicted in Fig. 6.7a, a deadlock is provoked by the small distance between two objects that cannot be simultaneously grasped by the robots. Such a scenario is often found in bin-picking applications that reliably leads to a deadlock and is therefore ideal for evaluating the deadlock resolution algorithm, introduced in Sec. 4.5, using a realistic example. Fig. 6.7b shows scenario 2 investigating the case when one robot blocks the other robot from grasping an object. In this scenario, the objects are placed symmetrically in the common workspace, where the farthest objects are assigned to each robot. As a result, the grasping is physically impossible. The last scenario in Fig. 6.7c demonstrates a deadlock in case the objects are to be placed in close proximity, i.e., they are assigned to the same tray. In such a case, a simultaneous placing of objects is impossible.



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 6.7: Experimental demonstration of three types of deadlocks. The videos of the experiments are available at the following URL [GWR24].

6.4.1 Scenario 1: Objects in close proximity

The first scenario investigates the limitation of simultaneous grasping. The setup consists of two objects: a blue trailer and a green container placed in the middle of the common workspace and two ArUco markers, one for each object, see Fig. 6.8a. The distance

between the objects is chosen so that the robots cannot pick the objects at the same time. The joint angles of the two robots are given in Fig. 6.8.

At the beginning, the robots move to their assigned objects, shown in Fig. 6.8b. However, due to the small distance between the objects, the collision avoidance algorithm does not allow a simultaneous grasping of the objects resulting in a deadlock. The deadlock can be noticed in the standstill of both robots by observing the joint angles q_1, q_4 and q_6, which

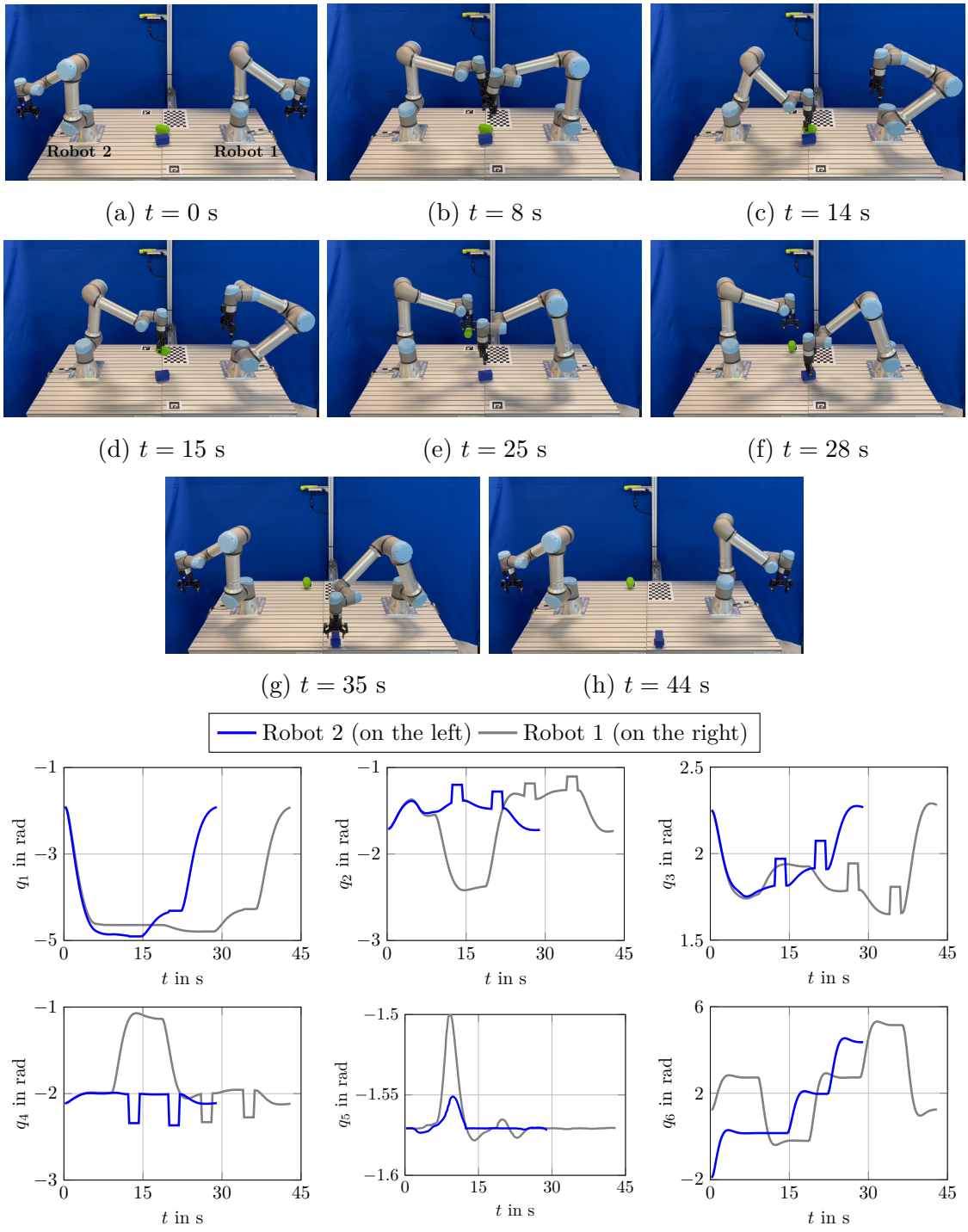


Figure 6.8: Selected time frames and joint angles of Robot 1 and Robot 2 for Scenario 1. The video of the experiment is available at the following URL [GWR24].

remain constant during the time period $t = 8$ s and $t = 12$ s. As the distance from Robot 2 to its target is smaller, the coordinator allows to proceed with its motion while Robot 1 is sent to its neutral pose, compare Fig. 6.8b–6.8d. As a result, the deflection in the joint angles can be observed for Robot 1 from time $t = 12$ s, while Robot 2 continuously approaches its reference pose followed by grasping its assigned object. Robot 1 has to wait almost 10 s to proceed with its initially assigned task. Once Robot 2 has grasped its object, Robot 1 is allowed to move to its target pose, see Fig. 6.8e. To resolve the deadlock, a sequential execution of both tasks was required resulting in idle times for Robot 1, while Robot 2 completed its task earlier. This can be observed for all joint angles in Fig. 6.8.

6.4.2 Scenario 2: Objects lying on opposite sides of each robot

The following scenario investigates the case, when an object is blocked by another robot and grasping is not possible. This type of a deadlock can always occur when the robots operate in the common workspace. Due to manipulator's geometry occupying a part of the workspace, not all objects might be accessible to the other robots. The setup of scenario 2 also consists of the same types of objects and two ArUco markers. The objects are placed symmetrically in the common workspace and the farthest objects are assigned to the robots, respectively. Robot 1 is assigned to pick up the blue trailer, while Robot 2 has to grasp the green tanker. Selected time frames and joint angles of the two robots from the scenario are shown in Fig. 6.9.

Fig. 6.9b and 6.9c show that a simultaneous grasping of objects is not possible, even if the objects are at a sufficient distance to be gripped. Due to the fact that the farthest objects are assigned to the robots, collisions are imminent within the lower segments of the robots. As a consequence, the collision avoidance approach prohibits approaching the robots too close, resulting in a deadlock. The deadlock is resolved by sending Robot 1 to its neutral pose and thus allowing Robot 2 to grasp its assigned object, see Fig. 6.9d–6.9e. The order of executing the tasks result from comparison of robots' residuals to their targets. As Robot 2 is closer to its target, Robot 1 is sent to its neutral pose to subsequently proceed with its previously assigned task. The deadlock, i.e. standstill of the robots' motion, can especially be observed in preserving the same joint angles during the time period $t = 8$ s and $t = 12$ s. The replanning of the motion towards the neutral pose can be seen for the most of the joint angles of Robot 1, where significant deflections from time $t = 12$ s can be observed. Robot 1 can proceed with its previously assigned task after Robot 2 has grasped its object and received a new task to place the object into the assigned slot. This can be seen in the time frames from the experiment in Fig. 6.9e–6.9g. Due to the idle times caused by resolving the deadlock, Robot 1 required 15 s longer to finish its tasks compared to Robot 2.

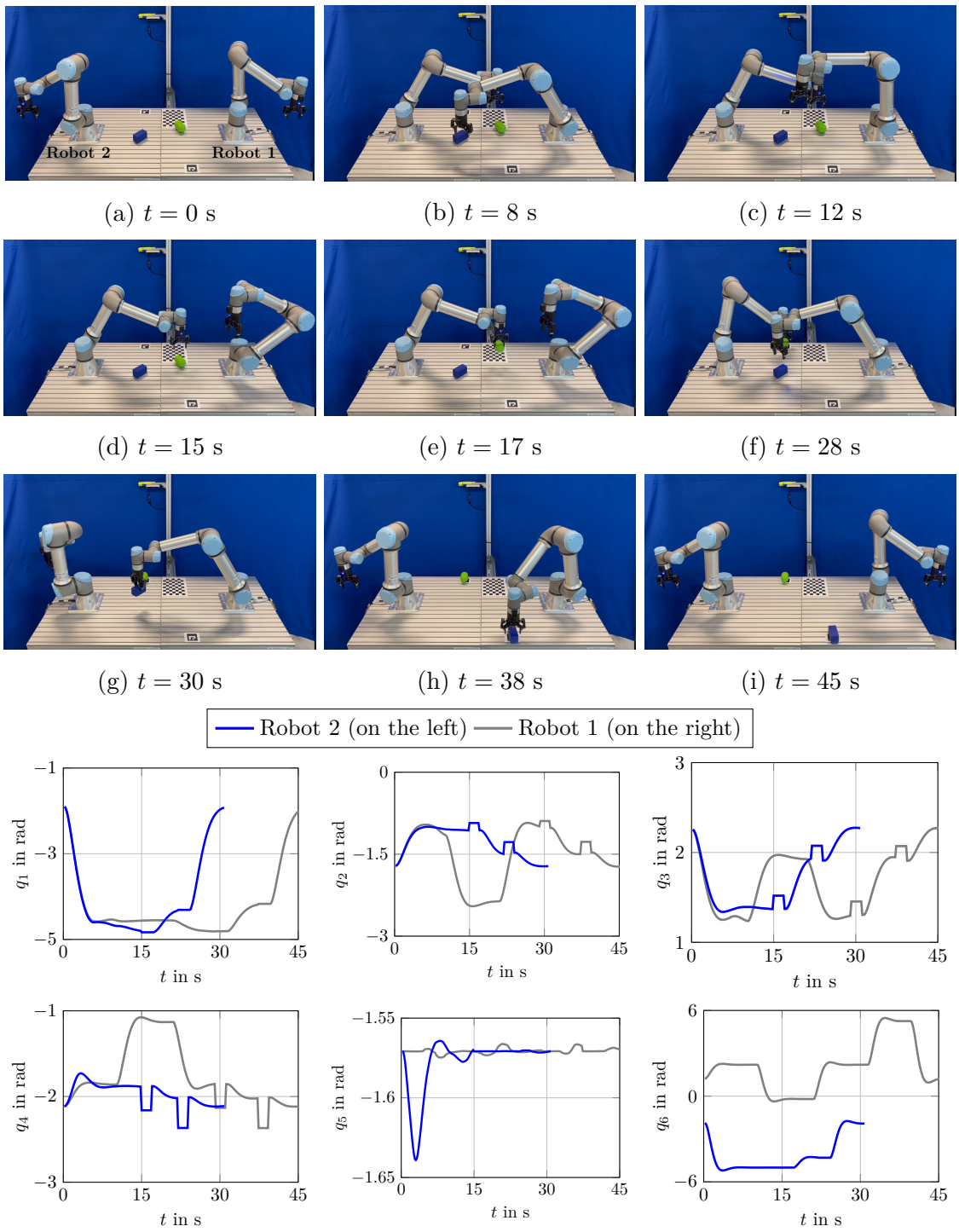


Figure 6.9: Selected time frames and joint angles of Robot 1 and Robot 2 for Scenario 2. The video of the experiment is available at the following URL [GWR24].

6.4.3 Scenario 3: Assignment of objects to the same tray

The last scenario investigates the case, when a simultaneous placing of objects is not possible. This situation is provoked by assigning two objects to the same tray. Note, that in that case the tray comprises two virtual slots having a small distance to each other. As a result, placing both objects into the tray at the same time is impossible. The yellow tanker is assigned to Robot 1 and the green tanker to Robot 2. Selected time frames and

the joint angles for both robots from the experiment are shown in Fig. 6.10.

At time $t = 8$ s, a simultaneous grasping of objects is possible, which can be viewed from the experiment in Fig. 6.10b–6.10c. However, the active collision avoidance between the grippers does not allow a simultaneous placing of the objects, which can be evidently seen in Fig. 6.10d–6.10e. The robots try to place their objects for about 2 s until the deadlock is detected and reported to the coordinator. To resolve the deadlock, the coordinator

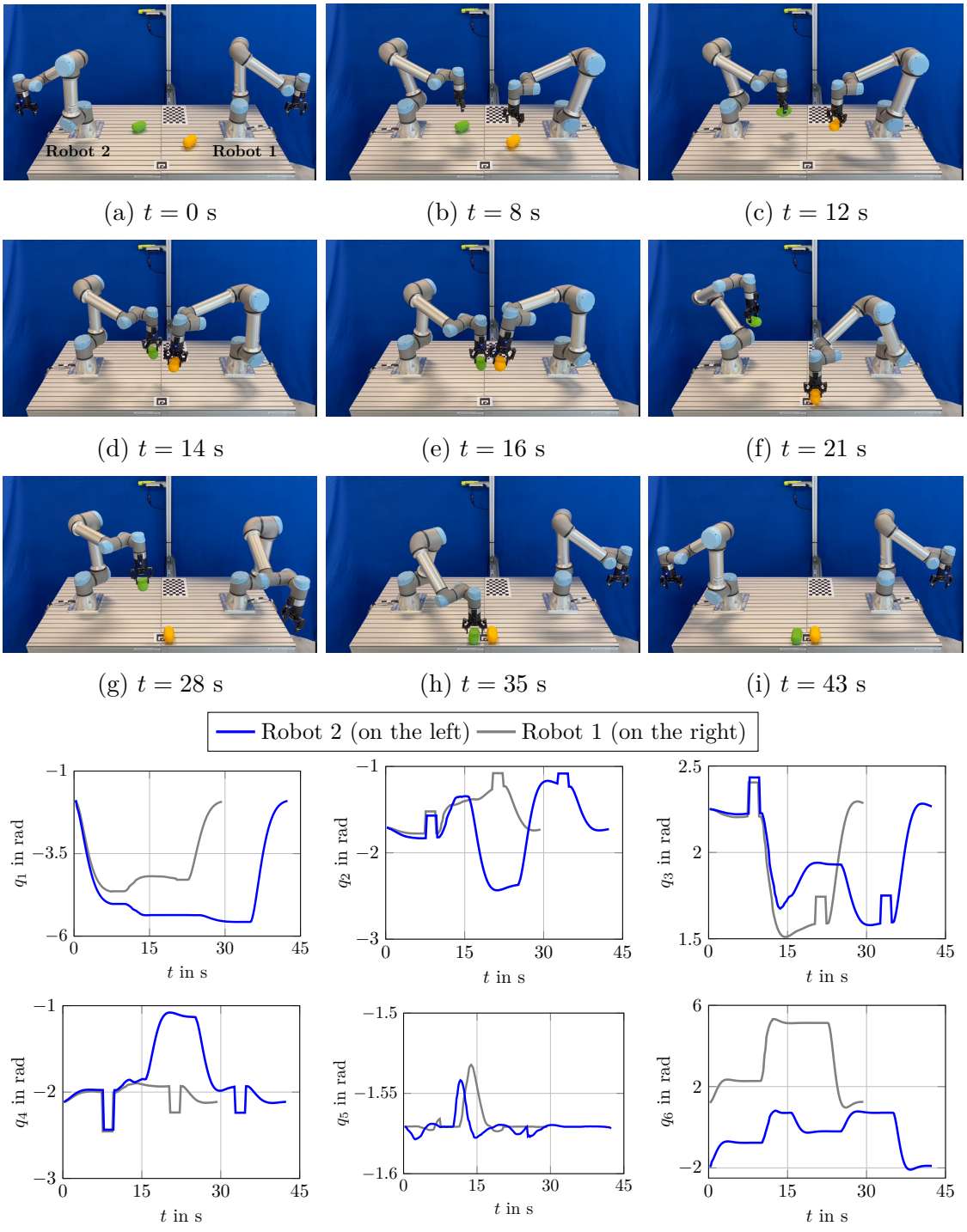


Figure 6.10: Selected time frames and joint angles of Robot 1 and Robot 2 for Scenario 3. The video of the experiment is available at the following URL [GWR24].

sends Robot 2 to its neutral pose while Robot 1 can place its object into the assigned slot, compare Fig. 6.10f. After the deadlock has been detected at time $t = 16$ s, Robot 2 moves to a new setpoint, that can be observed by the deflection of the joint angles in Fig. 6.10. After finishing the task by Robot 1, Robot 2 leaves its neutral pose to place its object as well, while Robot 1 returns to its initial position. This sequence can be seen for selected time frames in Fig. 6.10g-6.10h. Similar to the previous scenarios, Robot 2 needs more time to finish its tasks compared to Robot 1.

6.5 Sorting task

Fig. 6.11 illustrates the experimental setup of a sorting task. The joint task of the manipulators is to sort six objects into three trays in minimum time. The trays are marked by ArUco markers, where at most four objects in the area of each ArUco marker can be placed. The components of two trucks are arbitrarily distributed in the common workspace, which are two chassis, two trailers in yellow and blue and two tankers in green and red. Which class of an object can be sorted into which tray is indicated in Fig. 6.11. All objects are reachable by both robots. In contrast, only the tray for trailers can be served by both robots, while the other two are only reachable by one of the robots, i.e., the tray for chassis by Robot 2 and the tray for tankers by Robot 1. Similar to the simulation-based study in Chapter 5, a comparison between the heuristic and optimization-based scheduling approaches are carried out.

Initially, three classes of objects, i.e., chassis, trailer and tanker, are detected and classified including their properties by the object detection algorithm. A set of object classes yields

$$\Sigma = \{\mathrm{gCH}, \mathrm{gCH}, \mathrm{gTA}, \mathrm{rTA}, \mathrm{bTL}, \mathrm{yTL}\}.$$

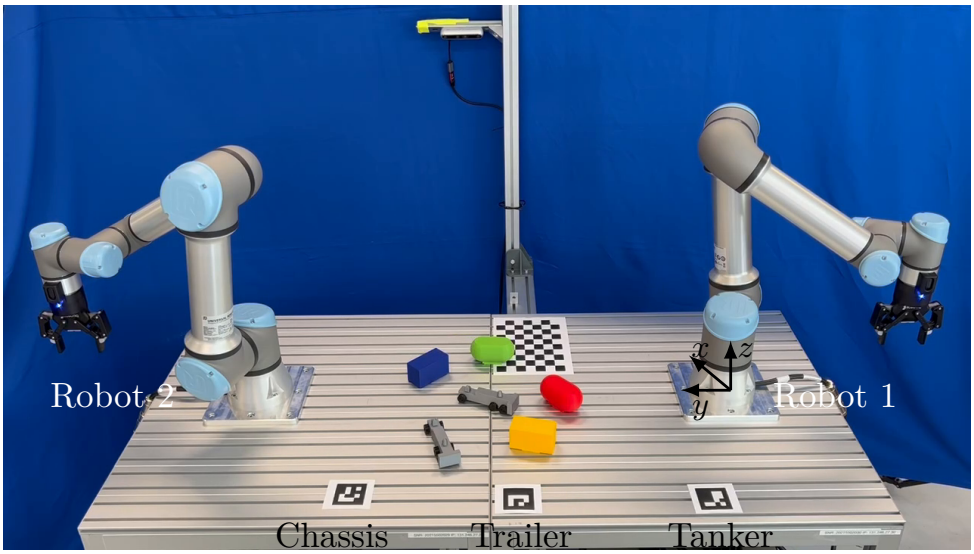


Figure 6.11: Experimental setup for sorting components from two trucks into three trays.

The number of objects of the same class determines the number of slots within a corresponding tray. For instance, if two tankers have to be sorted into a tray, then the task planer divides the area of the ArUco marker into two slots. Consequently, for the considered setup each tray contains two slots, to which one of the objects of the same class can be sorted. The assignment of a sequence of objects

$$\mathcal{O} = \{o_1^{\mathrm{gCH}}, o_2^{\mathrm{gCH}}, o_3^{\mathrm{gTA}}, o_4^{\mathrm{rTA}}, o_5^{\mathrm{bTL}}, o_6^{\mathrm{yTL}}\}.$$

into slots represents the degrees of freedom of the scheduling model. A pool of slots yields

$$\mathcal{S} = \{s_1^{\mathrm{CH}}, s_2^{\mathrm{CH}}, s_1^{\mathrm{TL}}, s_2^{\mathrm{TL}}, s_1^{\mathrm{TA}}, s_2^{\mathrm{TA}}\}.$$

The scheduling algorithm distributes the tasks in a heuristic or optimization-based manner, resulting in individual sequences of tasks for each robot. The action planner of each robot transforms a set of setpoints into a set of target states by applying inverse kinematics of the corresponding manipulator. The action planner sends a target state in case the robot successfully reached its previous target state. Moreover, each robot requires an action related to the given target state. For instance, an object that has to be picked is related to the action set $\mathcal{A}_{\mathrm{pick}}$ and a slot has to be served by applying the action set $\mathcal{A}_{\mathrm{place}}$. More details are provided in Sec. 4.2. In case deadlocks occur during a task execution between the robots, the coordinator aims to resolve them.

6.5.1 Heuristic task assignment

In the following, the tasks are distributed between the two robots employing a heuristic scheduling approach from Sec. 4.3.2. The executed sequences of tasks for each robot are given in the Gantt chart in Fig. 6.12. The Gantt chart also shows how much time each robot needed to accomplish a single task as well as all task assignments. The color of a bar and the designation above a bar yield the class name of an object. The experiment is illustrated by selected time frames from the experiment given in Fig. 6.13. Additionally, the joint angles of both robots are depicted as well.

The first assigned tasks are accomplished by both robots successfully, i.e., a simultaneous grasping of the objects and placing them into slots is possible. The corresponding time frames can be viewed in Fig. 6.13b–6.13c. In the next step, presented in Fig. 6.13d, Robot 2 has already grasped the gray chassis, while the collision avoidance constraints do not allow Robot 1 to simultaneously grasp the yellow trailer resulting in a deadlock. This type of a deadlock was investigated in Sec. 6.4 for scenario 1 caused by objects lying in close proximity to each other. After grasping, Robot 2 received a new target state, i.e., placing the chassis into the assigned slot. As a deadlock is detected shortly afterwards, the distance between Robot 2 and its new target state is now greater than of Robot 1 and the yellow trailer. Consequently, the coordinator sends Robot 2 to its neutral pose while Robot 1 is

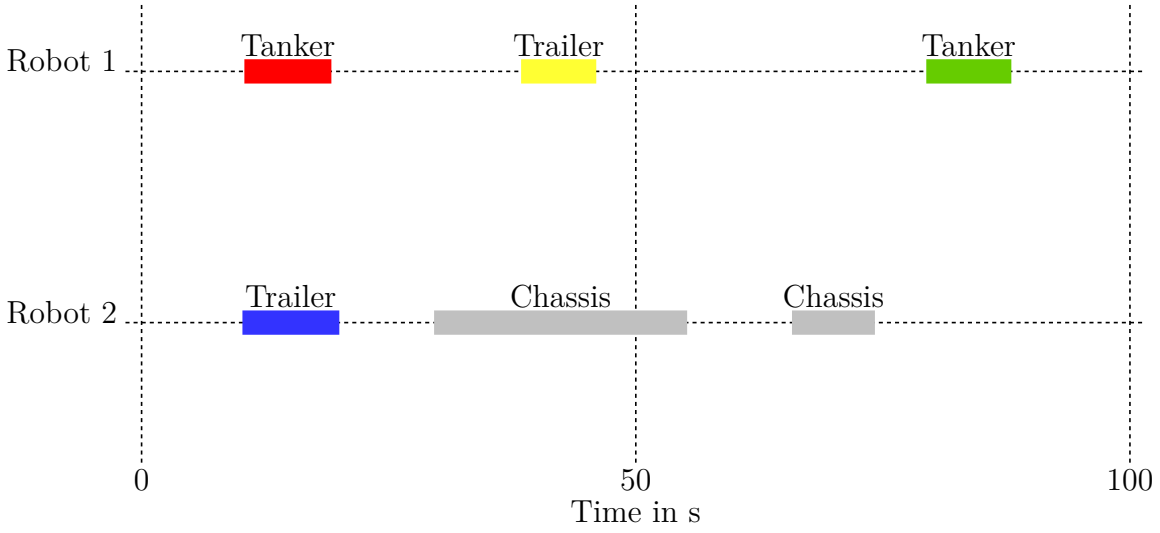


Figure 6.12: Gantt chart for the sorting task using the heuristic task assignment.

allowed to grasp its object, see Fig. 6.13e. The Gantt chart in Fig. 6.12 illustrates, that Robot 2 needs considerably longer for finishing its second pick-and-place task. This can be attributed entirely to the detection and resolution of the deadlock. Further, placing of the gray chassis and yellow trailer causes another deadlock. This type of a deadlock was presented in Sec. 6.4 for scenario 3 in case objects are assigned to the same tray or the assigned slots lie in close proximity to each other. In both cases, a simultaneous placing of objects into slots is not possible. This results in a sequential placing of objects into the assigned slots, compare Fig. 6.13f-6.13h. For the robots' final tasks, the grasping of the last objects causes another deadlock. The heuristic scheduler assigns chassis to Robot 2 and green tanker to Robot 1, which lie in close proximity to each other. Consequently, only a sequential grasping of the objects by the robots is possible. Robot 2 is allowed to proceed with its last task, while Robot 1 waits for Robot 2, which can be observed in Fig. 6.13i-6.13k. Due to several idle times caused by multiple deadlocks, the robots need almost 100 s to accomplish the sorting task. Multiple deadlocks can be evidently seen for the joint angles of both robots by short standstills accompanied by multiple deflections, i.e., every time a robot is sent to its neutral pose a deflection of a joint angle occurs. This is the case for the joint angles q_2, q_3, q_4 and q_5. In addition, all joint angles present smooth trajectories although multiple collisions had to be prevented between the two robots.

6.5.2 Optimization-based task assignment

In the following investigation, the tasks of the same experimental setup in Fig. 6.11 are optimally distributed between the two robots. The optimal sequence of tasks determined by the optimization-based scheduling approach from Chapter 4.3.3 can be viewed in the

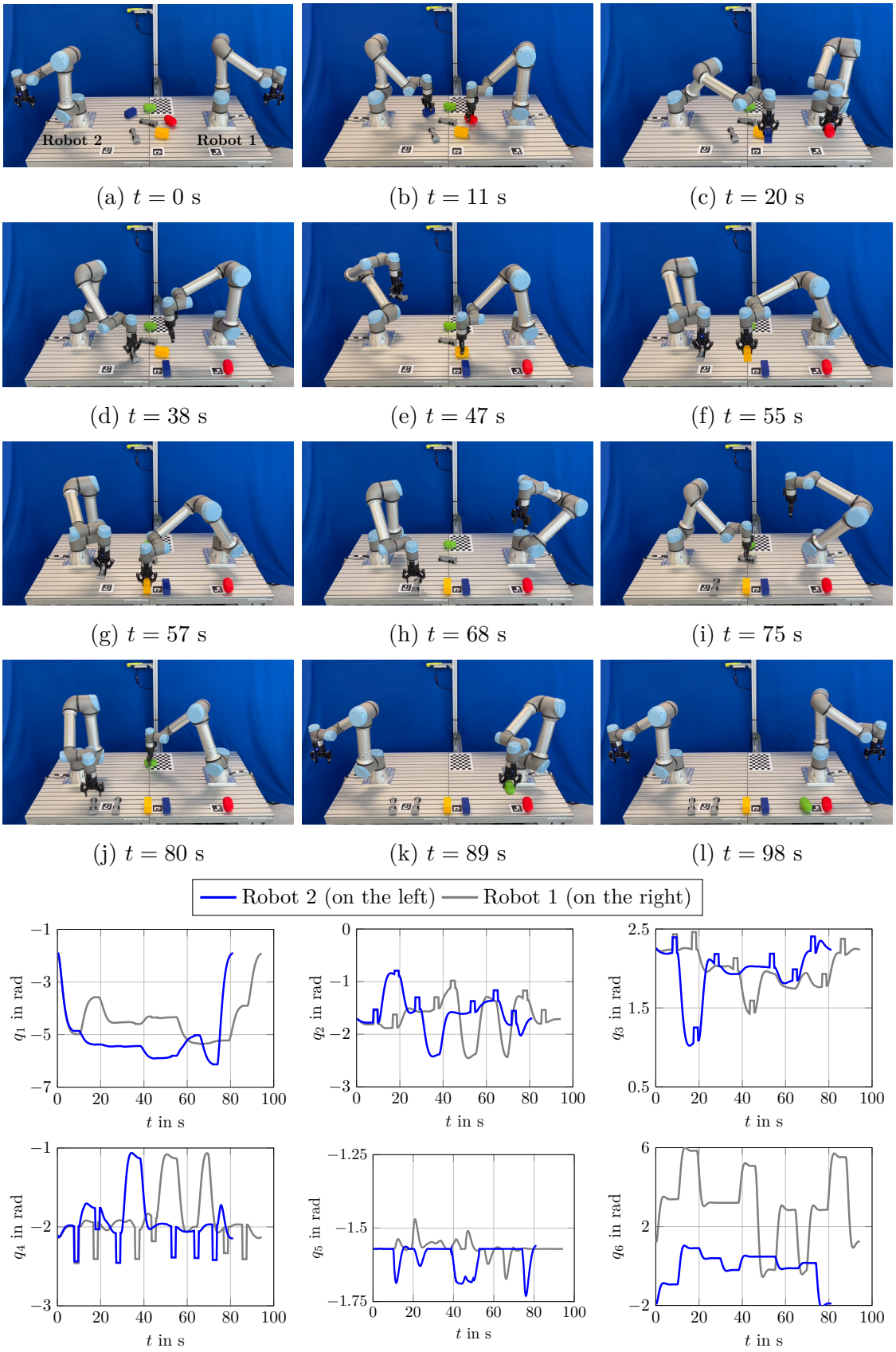


Figure 6.13: Selected time frames and joint angles of Robot 1 and Robot 2 during the sorting task using the heuristic task assignment. The video of the experiment is available at the following URL [GWR24].

Gantt chart in Fig. 6.14. The difference in the sequence of objects can be observed for Robot 1. The green tanker is assigned first and the red tanker as the last object. In case of heuristic scheduling approach it is vice versa. Some selected time frames from the experiment and the joint angles are given in Fig. 6.15.

As the first assigned objects lie in close proximity to each other, a simultaneous grasping of the objects is not possible resulting in a deadlock, which can be observed in Fig. 6.15b. The deadlock is resolved by sending Robot 1 to its neutral pose, as it has the longest distance to its target compared to Robot 2. Once Robot 2 has grasped its object, Robot 1 resumes its previous goal and grasps its object 8 s later, which can be seen in Fig. 6.15c–6.15d. The deflections of the joint angles q_2, q_3 and q_4 can be observed for Robot 1 only once at the beginning, when a deadlock occurs. The deadlock can be evidently observed in the Gantt chart, where Robot 1 shows a long idle time at the beginning. Nevertheless, all the following tasks can be carried out without delays, which can be seen in the follow-up time frames in Fig. 6.15e-6.15j. This leads to a considerable speed-up of the overall process compared to the results obtained by the heuristic scheduling approach. All task assignments are carried out in 77 s, which is 22% faster compared to the heuristically computed schedule. In general, the optimization-based scheduling approach tries to assign the objects that do not lie close to each other allowing a simultaneous grasping of objects. In addition, the algorithm takes care of not assigning the objects into the same tray preventing the case of simultaneous placing. However, as can be seen from the experiment, all objects are closely distributed in the common workspace and a deadlock cannot be entirely prevented due to the objects' constellation. Nevertheless, the task assignments with optimization-based scheduling approach shows considerable improvements upon the heuristic approach, where in total only one deadlock occurred compared to three deadlocks in case of heuristic scheduling.

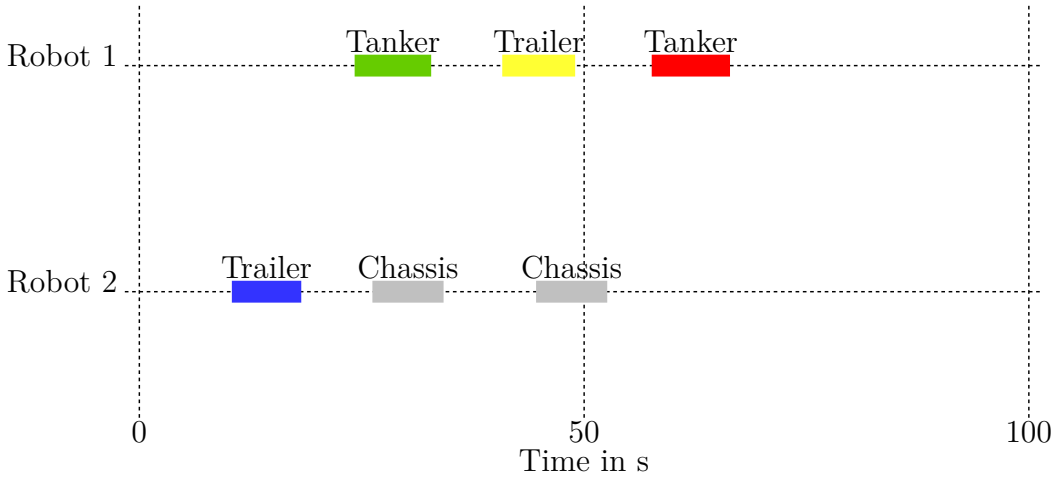
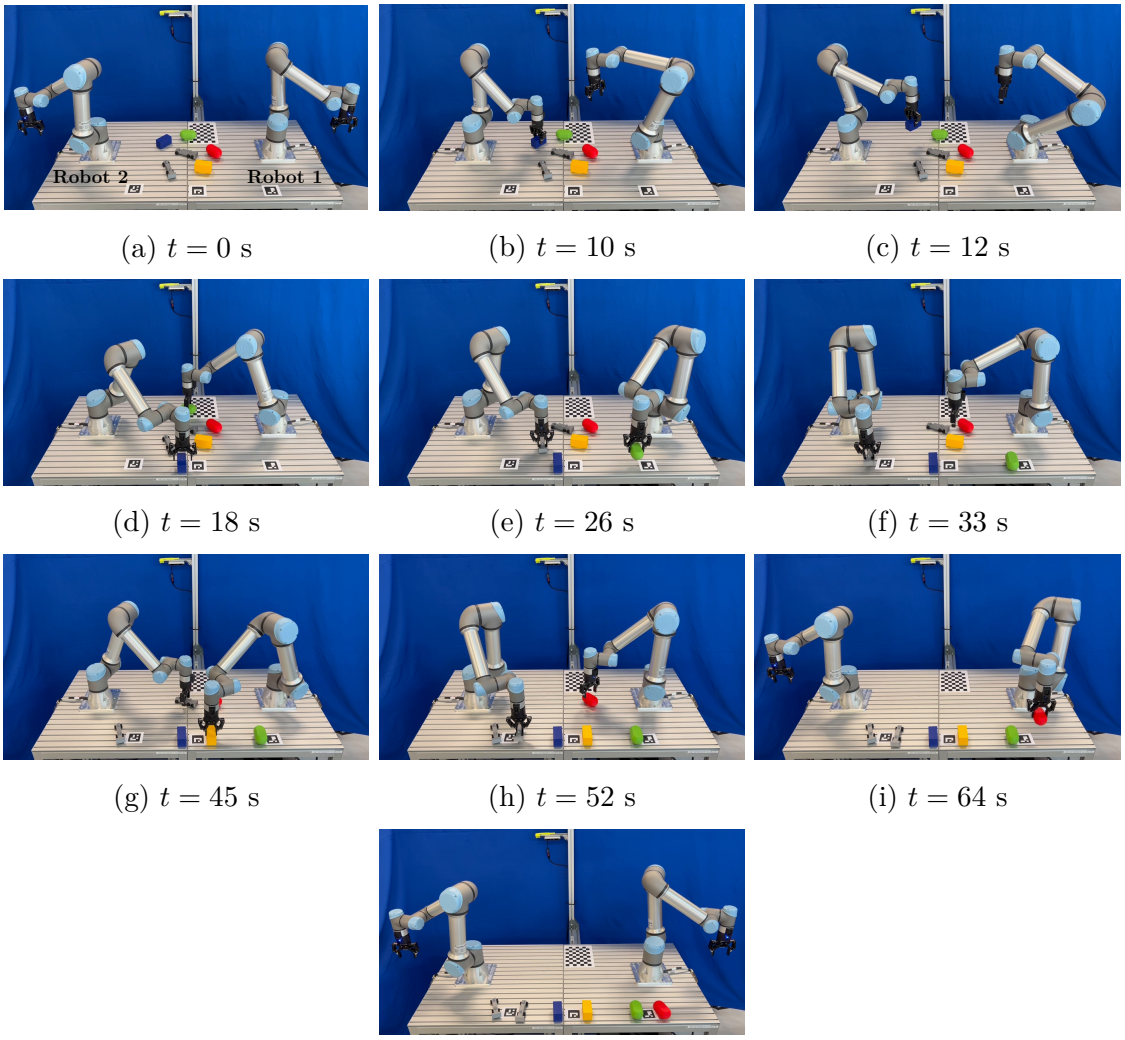


Figure 6.14: Gantt chart for the sorting task using the optimization-based task assignment.



(a) $t = 0$ s (b) $t = 10$ s (c) $t = 12$ s

(d) $t = 18$ s (e) $t = 26$ s (f) $t = 33$ s

(g) $t = 45$ s (h) $t = 52$ s (i) $t = 64$ s

(j) $t = 77$ s

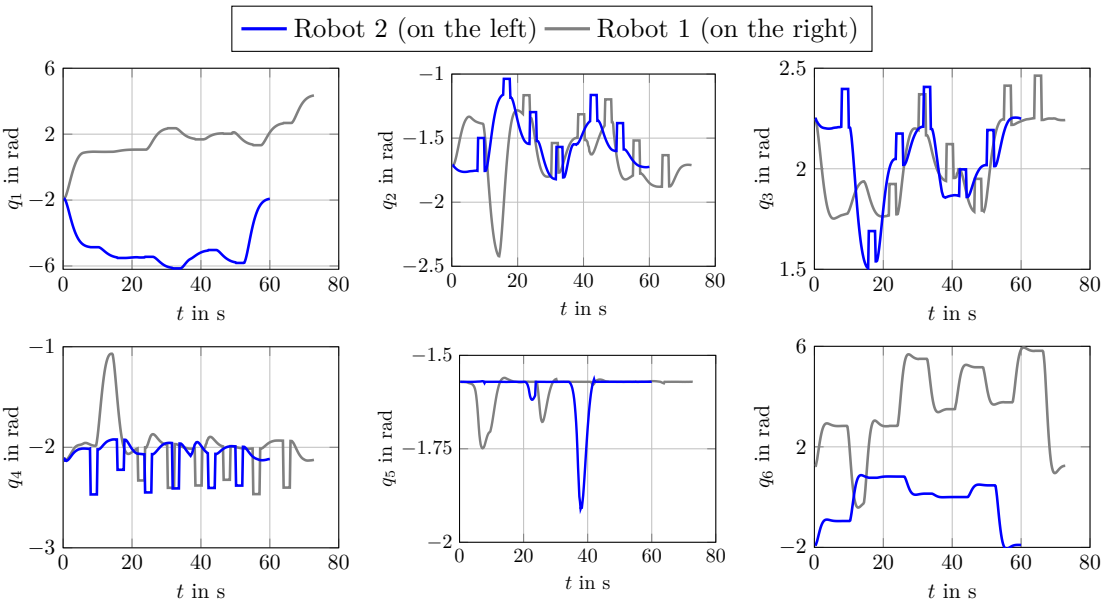


Figure 6.15: Selected time frames and joint angles of Robot 1 and Robot 2 during the sorting task using the optimization-based task assignment. The video of the experiment is available at the following URL [GWR24].

6.6 Disassembly task

In the final experiment, presented in Fig. 6.16, a disassembly of three trucks carried out by two manipulators is considered. Each truck consists of three parts: a chassis, a cab and a load. A truck can be loaded with either a tanker or a trailer. The common goal of both robots is to disassemble three trucks into individual components and place them into the assigned trays in minimum time. Each ArUco marker serves as a tray, where in total four ArUco markers are placed on top of the two modules. One ArUco marker is placed inside a box, where all cabs are to be sorted in. The trays for the chassis and the cabs are accessible by both robots, while the tray for the tankers are only reachable by Robot 2 and the tray for the trailers by Robot 1. Similar to the previous experiment, a comparison between the heuristic and optimization-based scheduling approaches is carried out. The main difference to the previous investigations is that additional constraints enforcing the order of disassembly are incorporated in both scheduling approaches. This is an important requirement to be able to reach for a chassis, which is only possible if a cab and a load are disassembled first. More precisely, any order in which a chassis is the last component of a truck assigned to one of the robots is feasible. More details about the order constrains for the proposed scheduling models are given in Chapter 4.3.5.

For the experimental setup, illustrated in Fig. 6.16, the object detection algorithm classifies nine objects in the common workspace, which are

$$\Sigma = \Sigma^{\text{Truck}} \cup \Sigma^{\text{Cab}} \cup \Sigma^{\text{Tanker}} \cup \Sigma^{\text{Trailer}} \tag{6.1a}$$

$$= \{\text{bTR}, \text{yTR}, \text{gTR}\} \cup \{\text{bCB}, \text{bCB}, \text{bCB}\} \cup \{\text{bTA}, \text{gTA}\} \cup \{\text{yTL}\}. \tag{6.1b}$$

Evidently, the chassis are not identified. Instead, three trucks, $\text{bTR}, \text{yTR}, \text{gTR}$, with a color of the corresponding load appear in the set of classified objects, i.e., a truck with a

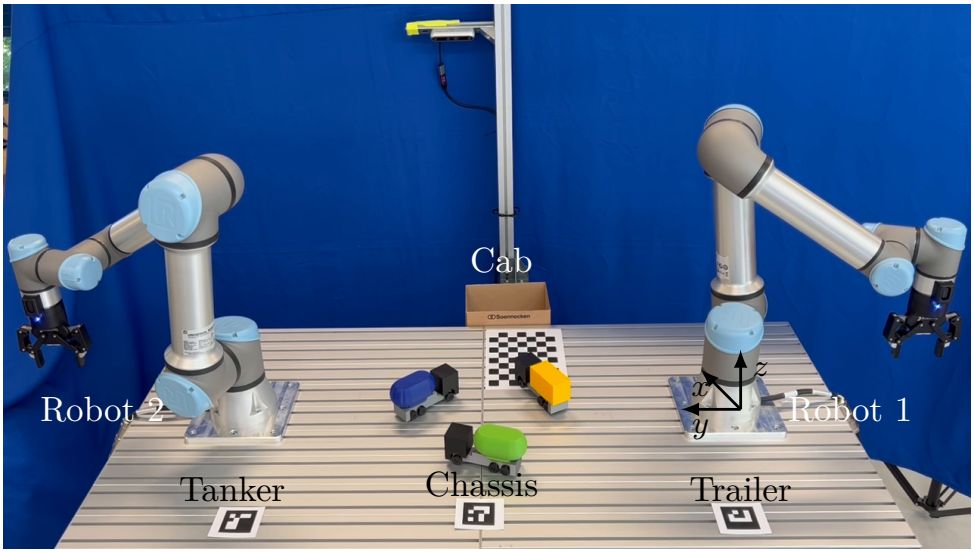


Figure 6.16: Experimental setup for disassembling three trucks into four trays.

yellow trailer is declared as an object class 'Yellow Truck' and is designated yTR. Due to the fact that a cab and a load conceal the chassis, it cannot be identified by the cameras as an additional truck's component. This is necessary, as all truck's components have to be sorted into the corresponding trays. To overcome this problem, the object detection algorithm has been trained to identify a truck as its own object class containing all three components: a cab, a load and a chassis. The position and orientation of each truck are used later for its corresponding chassis.

In the next step, the set of classified objects Σ are enumerated yielding a sequence of objects for the scheduling layer

$$\mathcal{O} = \{o_1^{\mathrm{bTR}}, o_2^{\mathrm{yTR}}, o_3^{\mathrm{gTR}}, o_4^{\mathrm{bCB}}, o_5^{\mathrm{bCB}}, o_6^{\mathrm{bCB}}, o_7^{\mathrm{bTA}}, o_8^{\mathrm{gTA}}, o_9^{\mathrm{yTL}}\}.$$

However, the object detection algorithm cannot classify which components belong to which truck. Thus, prior to sending the sequence of objects to the scheduling algorithm, the task planner clusters the individual components of the trucks by their relative distance. Each of these clusters forms one of the three trucks, consisting of a chassis, a cab and a load, which are to be dissembled by the two robots. Each cluster receives a unique ID yielding

$$\mathcal{O} = \mathcal{O}_1^{\mathrm{Truck}} \cup \mathcal{O}_2^{\mathrm{Truck}} \cup \mathcal{O}_3^{\mathrm{Truck}} \tag{6.2a}$$

$$= \{o_7^{\mathrm{bTA}}, o_4^{\mathrm{bCB}}, o_1^{\mathrm{gCH}}\} \cup \{o_9^{\mathrm{yTL}}, o_5^{\mathrm{bCB}}, o_2^{\mathrm{gCH}}\} \cup \{o_8^{\mathrm{gTA}}, o_6^{\mathrm{bCB}}, o_3^{\mathrm{gCH}}\}. \tag{6.2b}$$

Consequently, the following pool of slots are determined by the task planner

$$\mathcal{S} = \{s_1^{\mathrm{CH}}, s_2^{\mathrm{CH}}, s_3^{\mathrm{CH}}, s_1^{\mathrm{CB}}, s_2^{\mathrm{CB}}, s_3^{\mathrm{CB}}, s_1^{\mathrm{TA}}, s_2^{\mathrm{TA}}, s_1^{\mathrm{TL}}\},$$

which are sent to the Scheduling layer. All disassembly tasks are performed as pick-and-place tasks. Consequently, the action planner assigns the action set $\mathcal{A}_{\mathrm{pick}}$ to grasp an object and the action set $\mathcal{A}_{\mathrm{place}}$ to place an object into its assigned slot.

6.6.1 Heuristic task assignment

The first experiment is carried out by assigning the tasks to the robots with the heuristic scheduling approach from Sec. 4.3.2. The Gantt chart in Fig. 6.17 shows the executed sequence of tasks for each robot. To make the disassembly process more comprehensible, the color of each bar represents the components that belong to one of the three trucks. The length of the bar shows the time needed to accomplish a disassembly of one component, which consists of grasping an object and placing it to its assigned slot. The disassembly process is visualized by selected time frames from the experimental setup, given in Fig. 6.18. The joint angles of both robots are provided as well.

As can be seen from the Gantt chart, the heuristic scheduling algorithm assigns five parts to Robot 1 and four parts to Robot 2. More importantly, the Gantt chart shows that the

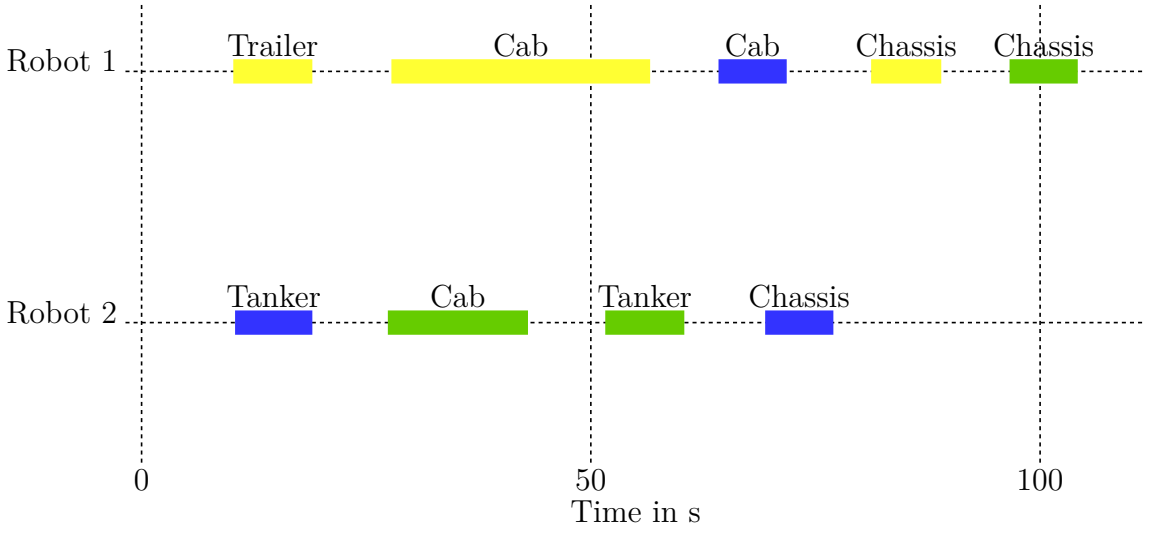
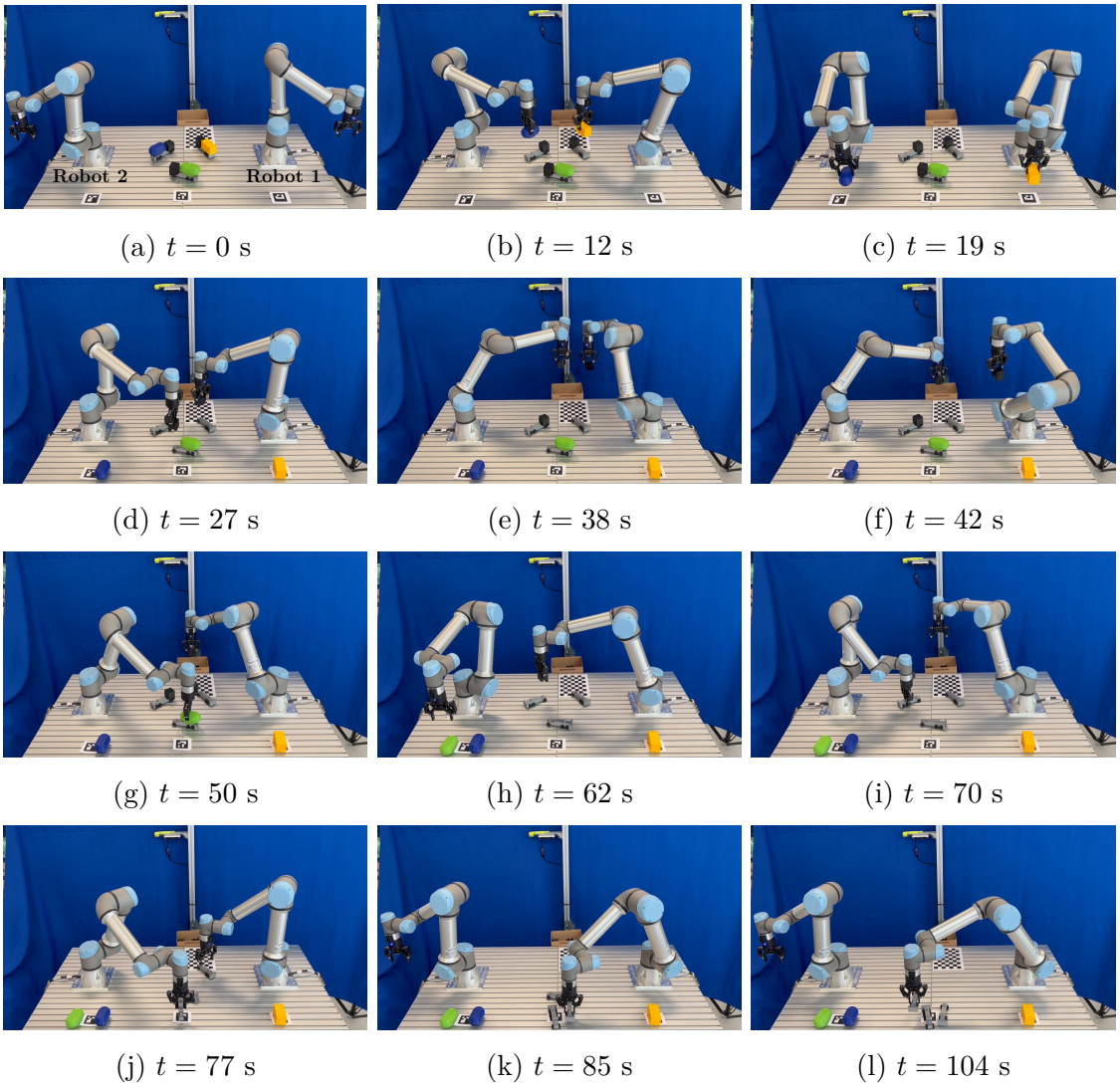


Figure 6.17: Gantt chart for the disassembly of three trucks using the heuristic task assignment.

chassis are assigned to both robots as the last parts of the three trucks to be disassembled, whereas the order of loads and cabs are arbitrary. Thus, the requirement of specific order successfully holds. The first task assignment, i.e., picking the yellow trailer by Robot 1 and the blue tanker by Robot 2, allows for a simultaneous grasping and placing of the objects, shown in Fig. 6.18b–6.18c. In the next disassembly step, the robots successfully remove two cabs, as can be seen in Fig. 6.18d. However, the placing of the cabs into the box leads to a deadlock between the robots. This type of a deadlock has been investigated in scenario 3 from Sec. 6.4, where simultaneous placing into the same tray is not possible. Fig. 6.18e shows the robots trying to simultaneously place their objects. The coordinator resolves the deadlock by sending Robot 1 to its neutral pose that allows Robot 2 to place the cab into the box, see Fig. 6.18f. A deflection in the joint angle q_2 at time $t = 40$ s can be seen, when Robot 1 moves to its neutral pose. The Gantt chart also shows a considerable amount of time for Robot 1 to accomplish its task assignment, which takes 28.8 s, while Robot 2 requires only 15.6 s. Consequently, Robot 1 continues with a considerable time delay to proceed with its tasks. All successive tasks are carried out by both robots without any deadlocks. While Robot 2 accomplishes its tasks after 77 s of operation, Robot 1 needs 104.2 s, i.e., 27 s more, to accomplish all of its tasks. The times for the robots to reach their initial positions are not considered here.



(a) $t = 0$ s (b) $t = 12$ s (c) $t = 19$ s

(d) $t = 27$ s (e) $t = 38$ s (f) $t = 42$ s

(g) $t = 50$ s (h) $t = 62$ s (i) $t = 70$ s

(j) $t = 77$ s (k) $t = 85$ s (l) $t = 104$ s

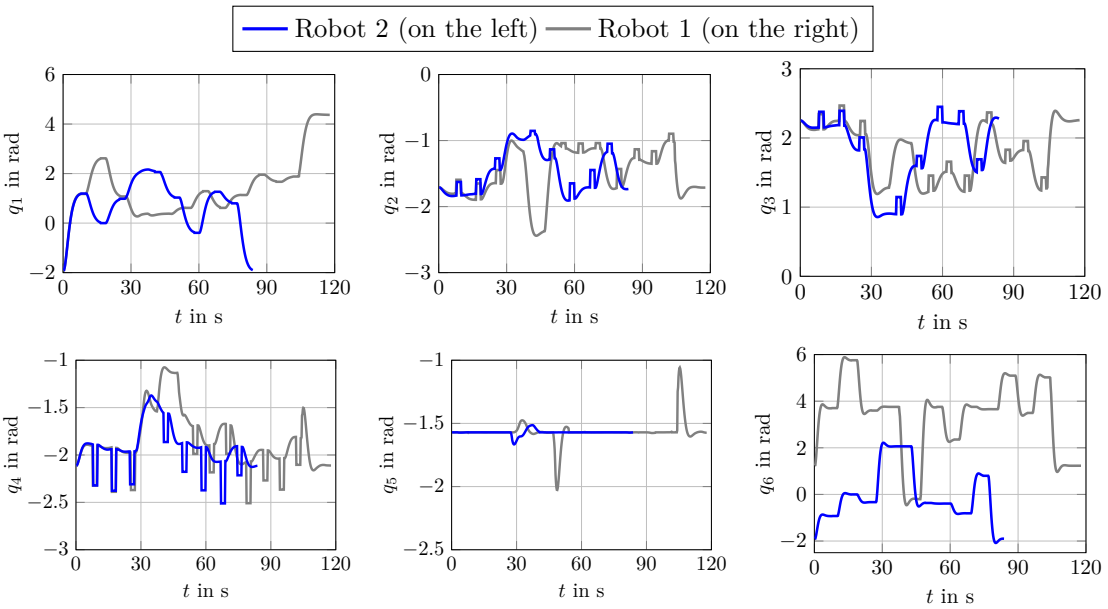


Figure 6.18: Selected time frames and joint angles of Robot 1 and Robot 2 during the disassembly task using the heuristic task assignment. The video of the experiment is available at the following URL [GWR24].

6.6.2 Optimization-based task assignment

In the following investigation, the tasks between the robots are assigned by utilizing the optimization-based scheduling method, introduced in Chapter 4.3.3. The Gantt chart in Fig. 6.19 shows the task assignments for both robots and the time needed to execute them. Fig. 6.20 presents selected time frames showing the robots' current states from the experiment and the corresponding joint angles of both robots.

From the Gantt chart it can be observed that the overall operation was carried out without any deadlocks, as each disassembly task was carried out in similar amount of time. Further, no idle times can be observed in execution of the tasks. Thus, simultaneous grasping and placing was possible for all disassembly tasks. Constraints enforcing a specific order, where chassis should be the last component of a truck to be disassembled by one of the robots have also been successfully fulfilled. Robots avoiding successfully collisions between each other can be observed at time $t = 50$ s, in Fig. 6.20g, where Robot 2 has to choose a path above Robot 1 to reach the box with the cabs, see Fig. 6.20h. No time delays can be observed in the Gantt chart during this time, although Robot 2 had to choose a longer path to reach its target state. Moreover, a smooth operation of both robots can be observed in the progression of the joint angles. Similar to the results obtained by the heuristic scheduling, Robot 1 receives five parts for disassembly and Robot 2 four disassembly components. Consequently, Robot 1 needs more time to finish its tasks. In total, the optimal task assignment leads to 16% shorter makespan compared to heuristically computed plan.

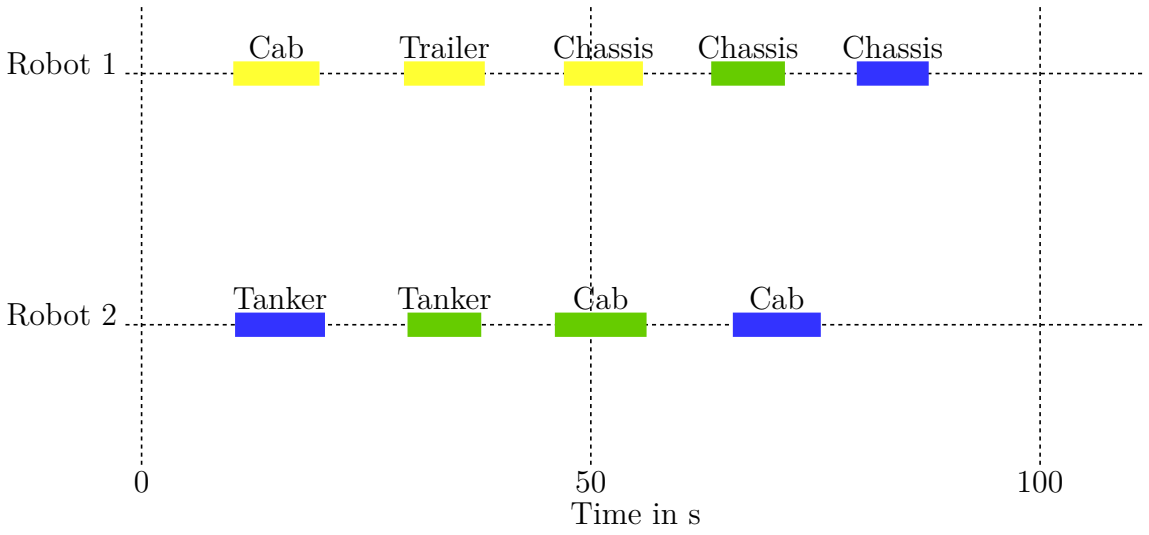


Figure 6.19: Gantt chart for the disassembly of three trucks using the optimization-based task assignment.

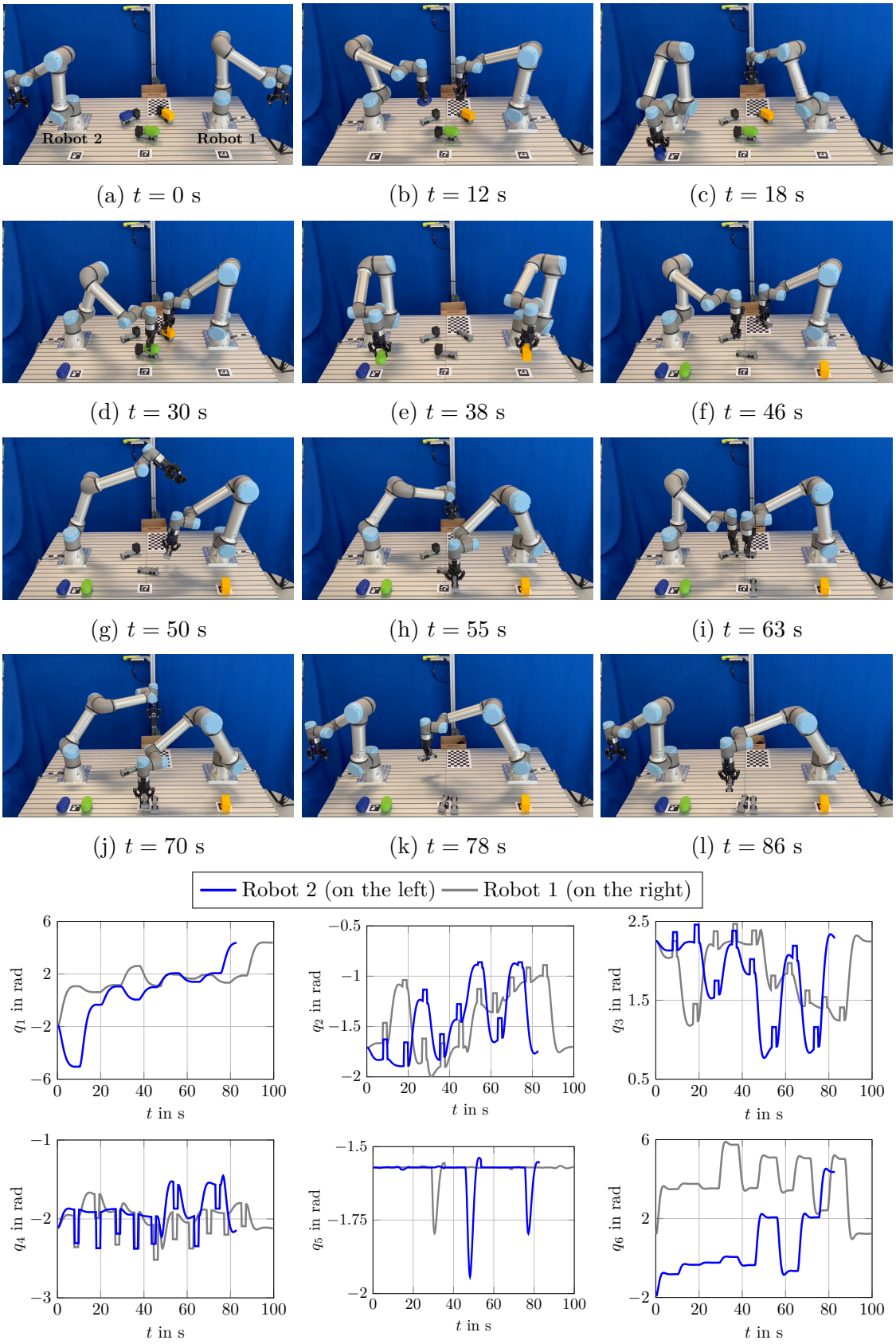


Figure 6.20: Selected time frames and joint angles of Robot 1 and Robot 2 during the disassembly task using the optimization-based task assignment. The video of the experiment is available at the following URL [GWR24].

6.7 Discussion

The experimental study demonstrates that deadlocks and collisions between multiple robots sharing a common workspace are imminent. A simple experiment in Sec. 6.3 served as an example demonstrating that a safe and reliable collision avoidance strategy is required to avoid inter-robot collisions as well as collisions with a robot's environment. The experimental study investigating several types of deadlocks in Sec. 6.4 showed that simultaneous grasping and placing of objects can reproducible cause a deadlock. By optimally assigning tasks between the robots, where the scheduling model does not allow assigning objects in close proximity to each other and placing objects into the same tray, can considerably reduce the occurrence of deadlocks. This was shown in the experiments carried out for sorting the components of two trucks in Sec. 6.5 and disassembly task of three tasks in Sec. 6.6. In both experiments, almost no deadlocks occurred by assigning tasks to the robots in an optimal manner. However, in case of heuristically computed schedule, the robots were not able to simultaneously place or grasp their objects resulting in a long makespan. The assumption made for the optimization-based scheduling model, that all manipulators traverse with the same mean velocity, implies synchronicity between the robots. In other words, time needed to accomplish a task is assumed to be identical. However, in cases where multiple collisions between the robots during a task execution should be avoided or more time is required to place an object, might result in time delays. Consequently, although the scheduling model takes care of certain cases of deadlocks, there is still no guarantee that the operation will be deadlock-free. Finally, it cannot be guaranteed that the constraints preventing from assigning objects in close proximity to each other and placing objects into the same tray always hold. This case was demonstrated in the sorting task experiment in Sec. 6.5, where the optimization-based scheduling approach assigned two objects lying close too each other to the robots. This was attributed to the dense positioning of the objects in the common workspace.

The third type of a deadlock, presented in scenario 2 in Sec. 6.4, demonstrates a non-reproducible type of a deadlock that cannot be considered a priori in the optimization-based scheduling model. This type of a deadlock occurs due to the intersection of robots' paths. In that case, the coordinator is required to resolve the occurred deadlock. The merit of the proposed TAMP approach is that the coordinator can reliably resolve any kind of deadlocks resulting however in a sequential execution of tasks causing idle times. In other words, one robot finishes its assigned tasks earlier than the other one. The approach however does not require a priori knowledge of robots motions coordinating them, thus allowing for a flexible operation of each manipulator without the intervention from the engineering side. By comparing the makespans of the two experiments carried out with different scheduling approaches yield, that robots could finish 22% faster the sorting task and 16% shorter the disassembly task in case of optimally assigned tasks to the robots.

7 Conclusion and Outlook

Achieving flexible and autonomous operation of a team of manipulators demands effective task coordination and online planning of collision-free trajectories, considering that each robot acts as a dynamic obstacle to the others. Existing task and motion planning (TAMP) methods typically depend on preplanned trajectories, which are sufficient for single-manipulator systems performing repetitive tasks in static, well-defined environments. However, real-world applications are often cluttered and subject to constant changes, requiring manipulators to execute complex tasks in dynamic and unpredictable settings. In the last decade, model predictive control (MPC) has gained widespread popularity in robotics research due to its predictive nature, formulation as an optimal control problem, and online execution. However, no benchmark analysis currently demonstrates the advantages of MPC over existing planners for manipulator systems.

In this thesis, an experimental benchmark analysis was conducted for a single manipulator system. An UR3 manipulator, a single-lens camera, two trays, two obstacles, and several objects were utilized for the experimental setup. The first experiment involved permanent changes within a production process, such as adding new products, placing products or obstacles at new positions, and observing how well the planners adapted. The second experiment investigated a narrow passage problem to evaluate how well the collision avoidance approach incorporated in each trajectory planner performed near obstacles. The results of both experiments showed that the state-of-the-art planners tended to select far-reaching motions in the presence of obstacles or execute evasive maneuvers, leading to long path lengths and execution times. MPC exceeded the performance of state-of-the-art planners across all evaluated metrics. The trajectories planned by MPC were notably smooth, effectively navigating through narrow passages, maintaining safe distances from obstacles, opting for shorter paths, and requiring less execution time. Additionally, the behavior of MPC towards sudden environmental changes has been investigated. MPC could replan the robot's trajectory at each time instant to follow the moving target and plan a collision-free and smooth trajectory around an obstacle suddenly placed in the robot's workspace. The results motivated the development of a suitable formulation of MPC for multi-manipulator systems.

This thesis proposed an optimal and reactive TAMP scheme for multi-manipulator systems sharing a common workspace and performing a series of pick-and-place tasks. Novel

algorithms developed and proposed in this thesis include

1. an optimal task allocation algorithm,
2. an online trajectory planning algorithm and
3. a reactive deadlock detection and resolution procedure.

An optimal task allocation algorithm for MRS assigns tasks to a group of robots, accounting for constraints such as reachability, maximum acceleration, velocity, and avoiding concurrent actions in close proximity to reduce the occurrence of deadlocks. The DMPC-ELS method, a distributed model predictive control approach, plans trajectories online from any initial to target state, with iterative replanning at each time step to address environmental changes. Predicted trajectories are shared among robots to anticipate and avoid collisions using a novel collision avoidance method, termed ELS, which checks for intersections of geometric primitives (ellipsoids and spherocylinders) rather than computing distances. Additionally, a reactive deadlock detection and resolution procedure allows the closest robot to the target to proceed while others wait, effectively resolving deadlocks online without preprocessing.

The proposed optimal and reactive TAMP scheme is thoroughly investigated through both simulations and experiments. The tasks considered in this work include sorting workpieces into designated compartments and disassembly. First, it is investigated if the trajectory planning problem should be solved in a centralized manner (CMPC), i.e., considering the overall system comprising all manipulators, or in a distributed manner (DMPC), i.e., treating each manipulator as a subsystem. To this end, a simple scenario involving imminent collisions between two manipulators was considered. A comparison of DMPC and CMPC computation times for various prediction horizon lengths revealed that real-time centralized trajectory planning for multi-robot systems is infeasible due to computational constraints. The distributed approach achieves near-centralized performance with a prediction horizon length $N_\mathrm{p} = 20$ while requiring significantly less computation time, allowing for parallelization.

The simulation-based study focused on two benchmark analyses: one for scheduling models and another for various trajectory planning methods, using 20 sampled pick-and-place scenarios. The study involved a two-manipulator system that shared a common workspace and sorted six objects into two distinct compartments, with objects randomly placed in each scenario. In the first benchmark analysis, task allocation was addressed using both the optimization-based and the heuristic scheduling approach. The results indicated that optimal task allocation significantly reduced the occurrence of deadlocks compared to the heuristically computed plans. Longer prediction horizons in DMPC-ELS improved the mean execution times of heuristically computed schedules, but increased computation times due to the added constraints. However, the prediction horizon length had no

notable effect on the execution time for optimal task assignment. Thus, using a suitably short prediction horizon with optimal task assignment can help manage computation times and decrease deadlock occurrences. A further benchmark analysis involving several selected state-of-the-art planners was carried out to assess the quality of the trajectories obtained by the DMPC-ELS approach. Similar to the single-robot case, state-of-the-art planners exhibited far-reaching motions and lower success rates in finding collision-free paths for multiple robots. Notably, CHOMP achieved collision-free trajectories in only 43 % of the cases. In contrast, DMPC-ELS achieved success rates of up to 97 % and generated the smoothest trajectories. The lower performance of state-of-the-art planners suggests that they are not well-suited for simultaneous trajectory planning used for multiple manipulators in static environments. Additional analysis has been carried out to answer how well the proposed DMPC-ELS approach scales with an increasing number of robots. Different robot setups with two, three and four robots and a different number of objects were investigated performing pick-and-place tasks. The scalability analysis showed that DMPC-ELS could plan collision-free trajectories for up to four robots with the mean computation times not exceeding the sampling time. The main bottleneck of the proposed DMPC-ELS approach lies in computation times that rise with the system's complexity.

For the multi-robot experimental study, two UR5e manipulators and two depth cameras capturing different views of the robots' workspaces have been utilized. An object detection algorithm has been trained to detect and identify different classes of trucks consisting of three parts: a cab, a chassis, and two types of load. Three commonly occurring classes of deadlocks were investigated, where simultaneous grasping and placing of objects were impossible and resulted in a deadlock. Here, the focus lay not on optimally assigning the tasks to the robots but on how each deadlock was resolved by applying the proposed reactive deadlock approach. The first experiment of the robot-robot cooperation investigated a sorting task, where two robots had to place components of a truck lying in the shared workspace into corresponding trays. By applying the heuristically computed schedule, three deadlocks occurred between the two robots invoked by trying to grasp and place the assigned objects simultaneously. This was attributed to the fact that objects were lying too close to each other, and a pair of objects had been assigned to the same tray, making simultaneous grasping and placing of objects impossible. The occurrence of deadlocks considerably dropped, namely to one deadlock, by optimally assigning the tasks. One deadlock occurred when grasping objects placed close to each other, highlighting that the optimization-based scheduling approach does not always ensure a deadlock-free schedule. Still, the execution times of the optimal schedule were reduced by 22% compared to the heuristically computed plan. Consequently, a reactive deadlock detection and resolution approach is imminent to handle deadlocks between the robots reliably. The second experiment involved the cooperative disassembly of three trucks, where each truck's component had to be removed and placed into its assigned tray. The disassembly was successfully

accomplished, where nine parts of three trucks were disassembled and sorted into the assigned trays. Similar to the results obtained from the sorting task, optimal task assignment led to shorter execution times, namely by 16% shorter, with no occurrence of deadlocks, which was not the case for the heuristically computed schedule.

This chapter is rounded off by giving a few directions toward future research in multi-robot cooperation. First of all, homogeneous robotic arm systems were exclusively considered. Thus, the proposed approach could extend to heterogeneous multi-robot systems, such as stationary manipulators cooperating with aerial robotic or mobile manipulators. This could further increase the overall reachability of the system, and more complex tasks could be carried out and automated. The proposed collision avoidance approach has to be extended and automatized to allow cooperation between different types of manipulators and also human-robot interaction (HRI). Depending on the kinematics of a robotic system or a human body representing a number of dynamic obstacles, a suitable choice of geometric primitives should be chosen. Moreover, uncertainties in the motion of robots and humans should also be considered.

The selection of an appropriate number of ellipsoids and line-swept spheres to encapsulate the entire body of each robot depends on the robot's kinematics and geometry. This preprocessing stage has been realized for homogeneous robots and is particularly important when dealing with heterogeneous robots, as it involves choosing a sufficient number of geometric primitives to ensure safe robot-robot interactions. Additionally, preprocessing should eliminate collision avoidance constraints from the optimization problem that could never become active. This applies in cases where the links of the robots cannot come into contact with each other, meaning collisions between those links are impossible under any circumstances.

Further, hand-over tasks between homogeneous and heterogeneous multi-robot systems should be considered. This can be done using a fixed hand-over station accessible to both robots. That means one robot places the object at a fixed place, from which the other robot can pick up the object at some point later. This would be rather a simplified hand-over task. Considering a hand-over of an object between the robots, i.e., by simultaneously allowing physical contact of both robots on an object, is highly complex. On the one hand, grasping an object plays a crucial role, and the coordination between the robots, as well as how and where the hand-over of an object should take place, has to be taken into account. On the other hand, if the robotic team consists of multiple robots, where hand-over tasks can be carried out spontaneously, special care should be taken regarding collision avoidance.

Further investigations are necessary for tracking problems in MRS, particularly when the target can change its position at any time. Additionally, the occurrence of deadlocks and the performance of the proposed reactive deadlock resolution procedure should be

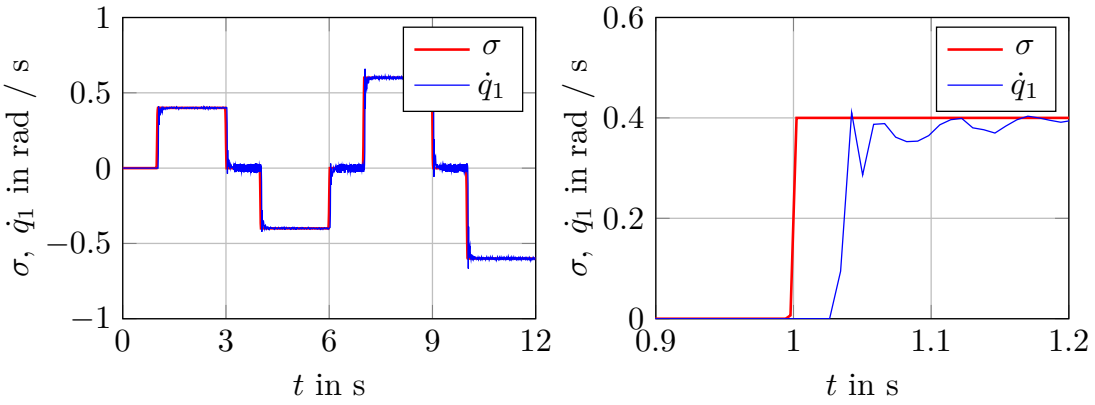
examined across different types of applications.

Moreover, the issue of sensor malfunction should be addressed. For example, sensors may provide inaccurate information due to process delays, noise, or other factors, which could compromise system safety. To ensure safety under such conditions, the proposed collision avoidance approach should be extended to account for sensing uncertainties [RSS16].

Appendix

A Model identification of an UR3

To investigate the dynamics of the velocity tracking controllers of the UR3 manipulator, test signal with step functions over a duration of 12 s are applied to each joint independently with only a single joint active at a time. The step response of each joint is recorded with a rate of 125 Hz. Fig1a shows the angular velocity step response $\dot{q}_1$ of the first joint and the applied step functions σ. As can be seen from the enlarged view of $\dot{q}_1$ in Fig1b, the system reacts with a time delay of around 60 ms with only a minor overshoot. The reference is closely tracked after a short transition time.



(a) Test signal with step response (b) Enlarged view

Figure 1: Excitation and angular velocity step response for joint 1 of the UR3.

Krämer et al. demonstrated that is suffices to approximate the joint dynamics of an UR10 robotic manipulator by a zero-order dynamical system including dead time [Krä+20]. This approach comes with the great advantage that only a single dead time per joint need to be considered in the (D)MPC and no additional degrees of freedom are introduced, as would be the case for an assumed first- or second-order dynamics. Thus, the transfer function of each joint i is modeled by

$$G_i(s) = e^{-s \cdot T_{\mathrm{D},i}} \cdot K_i \ , \tag{10}$$

where $T_{\mathrm{D},i}$ denotes the dead time and K_i describes the gain of the transfer function $G_i(s)$. To identify the dead time $T_{\mathrm{D},i}$ and the gain K_i, the MATLAB System Identification Toolbox [Lju22] is used. Table 1 summarizes the identified parameters. The longest dead

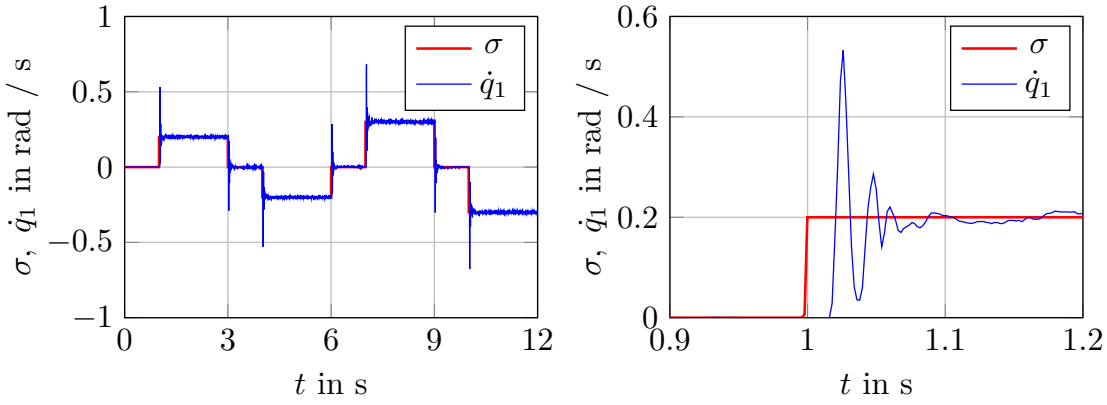
Table 4: Identified parameters of the UR3 velocity tracking controllers.

Joint Number	**Joint Name**	K_i	$T_{\mathrm{D},i}$
1	Base	0.9928	64 ms
2	Shoulder	0.9808	72 ms
3	Elbow	0.9827	72 ms
4	Wrist 1	0.9842	40 ms
5	Wrist 2	0.9833	32 ms
6	Wrist 3	0.9844	40 ms

time with 72 ms is determined for joints 2 and 3, whereas for joint 5 the shortest dead time of 32 ms is identified. As the dead times of each robot's joint controllers are non-negligible, they should be considered in the trajectory planning. In addition, non-negligible computation times of the trajectory planner (D)MPC-ELS should be considered as well due to the fact that the next control input is not always available in the next time instant, see Sec. 3.2.4. To overcome the problem of delayed system response and delayed availability of control inputs, an available sequence of control input sequence for a time instant is sent until the new optimal sequence is available.

B Model identification of an UR5e

Next, the system dynamics of the velocity tracking controllers of the UR5e manipulator are identified. As before, a test signal with step functions over a duration of 12 s are applied to each joint independently. The step response of each joint is recorded with a rate of 500 Hz. Fig2a shows the angular velocity step response of the first joint $\dot{q}_1$ and the applied test signal σ. An enlarged view of the results is presented in Fig2b. The UR5e exhibits a lower time delay but a significant overshoot of the angular velocity step response in contrast to the UR3 manipulator. As for the previous section, the step response of each joint is approximated by a zero-order system1(). For that, the dead time $T_{\mathrm{D},i}$ and the gain K_i for each joint are identified using the MATLAB System Identification Toolbox [Lju22]. The results are summarized in Table2. Similar to the results of the UR3, a slight fluctuation of the gains around 1 can be observed. In contrast, the dead times lie between 22 ms and 34 ms and are thus considerably smaller, which are almost 50 % less. This might be attributed to improved angular velocity controllers used for the robots of the e-series.



(a) Test signal with step response (b) Enlarged view

Figur2: Excitation and angular velocity step response for joint 1 of the UR5e.

Table2: Identified parameters of the UR5e velocity tracking controllers.

Joint Number	Joint Name	K_i	$T_{\mathrm{D},i}$
1	Base	1.001	24.1 ms
2	Shoulder	1.001	34.1 ms
3	Elbow	1.003	30.1 ms
4	Wrist 1	1.004	26.1 ms
5	Wrist 2	1.003	22.0 ms
6	Wrist 3	1.003	28.1 ms

Bibliography

[AA14] A. Y. Afaghani and Y. Aiyama. "Advanced-collision-map-based on-line collision and deadlock avoidance between two robot manipulators with PTP commands." In: *2014 IEEE International Conference on Automation Science and Engineering (CASE)*. 2014, pp. 1244–1251. DOI: 10.1109/coase.2014.6899486.

[AAA23] A. Y. Afaghani, J. E. Afaghani, and Y. Aiyama. "On-line Deadlock-free Planning of N-industrial-robot Arms With Independent Controllers Using Advanced Escaping Method." In: *International Journal of Control, Automation and Systems* 21 (2023), pp. 3696–3711. DOI: https://doi.org/10.1007/s12555-022-1051-2.

[AC19] V. Alcácer and V. Cruz-Machado. "Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems." In: *Engineering Science and Technology, an International Journal* 22.3 (2019), pp. 899–919. DOI: https://doi.org/10.1016/j.jestch.2019.01.006.

[Ade+18] A. Adel et al. "Design of Robotically Fabricated Timber Frame Structures." In: *Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA)*. 2018, pp. 394–403. DOI: 10.7302/9600.

[AGG17] M. H. Arbo, E. I. Grøtli, and J. T. Gravdahl. "Mid-Level MPC and 6 DOF output path following for robotic manipulators." In: *2017 IEEE Conference on Control Technology and Applications (CCTA)*. 2017, pp. 450–456. DOI: 10.1109/ccta.2017.8062503.

[AH02] 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, pp. 624–631. DOI: 10.1109/ROBOT.2002.1013428.

[Ake+18] T. Akenine-Möller et al. *Real-Time Rendering 4th Edition*. A K Peters/CRC Press, 2018, p. 1200. ISBN: 978-1-13862-700-0.

[Ake+99] P. Akella et al. "Cobots for the automobile assembly line." In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat.*

No.99CH36288C). Vol. 1. 1999, pp. 728–733. DOI: 10.1109/ROBOT.1999.770061.

[Alo+15] J. Alonso-Mora et al. "Local motion planning for collaborative multi-robot manipulation of deformable objects." In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 5495–5502. DOI: 10.1109/icra.2015.7139967.

[Ame+19] A. D. Ames et al. "Control Barrier Functions: Theory and Applications." In: *2019 18th European Control Conference (ECC)*. 2019, pp. 3420–3431. DOI: 10.23919/ecc.2019.8796030.

[And+19] J. A. E. Andersson et al. "CasADi: a software framework for nonlinear optimization and optimal control." In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36. ISSN: 1867-2957. DOI: 10.1007/s12532-018-0139-4.

[App+07] D. L. Applegate et al. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2007. ISBN: 9780691129938.

[Ard+15] M. M. G. Ardakani et al. "Real-time trajectory generation using model predictive control." In: *2015 IEEE International Conference on Automation Science and Engineering (CASE)*. 2015, pp. 942–948. DOI: 10.1109/CoASE.2015.7294220.

[Asi42] I. Asimov. "Runaround." In: *Astounding Science Fiction*. Vol. 29. Street & Smith, 1942.

[Bar+96] J. Barraquand et al. "A Random Sampling Scheme for Path Planning." In: *Robotics Research* (1996), pp. 249–264. DOI: 10.1007/978-1-4471-1021-7_28.

[BBM17] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. ISBN: 9781139061759. DOI: 10.1017/9781139061759.

[Bek06] T. Bektas. "The multiple traveling salesman problem: an overview of formulations and solution procedures." In: *Omega* 34.3 (2006), pp. 209–219. DOI: 10.1016/j.omega.2004.10.004.

[Bes+16] C. M. Best et al. "A New Soft Robot Control Method: Using Model Predictive Control for a Pneumatically Actuated Humanoid." In: *IEEE Robotics & Automation Magazine* 23.3 (2016), pp. 75–84. DOI: 10.1109/MRA.2016.2580591.

[Bet10] J. T. Betts. *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*. 2nd. Cambridge University Press, 2010. ISBN: 978-0-89871-688-7. DOI: https://doi.org/10.1137/1.9780898718577.

[Bet98] J. T. Betts. "Survey of Numerical Methods for Trajectory Optimization." In: *Journal of Guidance, Control, and Dynamics* 21.2 (1998), pp. 193–207. DOI: https://doi.org/10.2514/2.4231.

[Bez+17] J. Bezanson et al. "Julia: A Fresh Approach to Numerical Computing." In: *SIAM Review* 59.1 (2017), pp. 65–98. ISSN: 0036-1445. DOI: 10.1137/141000671.

[BGL11] J. van den Berg, S. J. Guy, and D. Lin Ming and Manocha. "Reciprocal n-Body Collision Avoidance." In: *Robotics Research*. Vol. 70. Springer, Berlin, Heidelberg. 2011, pp. 3–19. DOI: https://doi.org/10.1007/978-3-642-19457-3_1.

[BH09] P. Bosscher and D. Hedman. "Real-time collision avoidance algorithm for robotic manipulators." In: *2009 IEEE International Conference on Technologies for Practical Robot Applications*. 2009, pp. 113–122. DOI: 10.1109/TEPRA.2009.5339635.

[Bha+21] M. Bhardwaj et al. "STORM: An Integrated Framework for Fast Joint-Space Model-Predictive Control for Reactive Manipulation." In: *Conference on Robot Learning*. 2021, pp. 750–759.

[BHK13] M. Badreldin, A. Hussein, and A. Khamis. "A Comparative Study between Optimization and Market-Based Approaches to Multi-Robot Task Allocation." In: *Advances in Artificial Intelligence* 2013.1 (2013), p. 11. DOI: https://doi.org/10.1155/2013/256524.

[Bra00] G. Bradski. "The OpenCV Library." In: *Dr. Dobb's Journal of Software Tools* (2000).

[BSB00] M. Bonert, L. H. Shu, and B. Benhabib. "Motion planning for multi-robot assembly systems." In: *International Journal of Computer Integrated Manufacturing* 13.4 (2000), pp. 301–310. DOI: https://doi.org/10.1080/095119200407660.

[BSR10] R. Barták, M. A. Salido, and F. Rossi. "Constraint satisfaction techniques in planning and scheduling." In: *Journal of Intelligent Manufacturing* 21 (2010), pp. 5–15. DOI: https://doi.org/10.1007/s10845-008-0203-4.

[Cao+13] Y. Cao et al. "An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination." In: *IEEE Transactions on Industrial Informatics* 9.1 (2013), pp. 427–438. DOI: 10.1109/TII.2012.2219061.

[Car+19] A. Carron et al. "Data-Driven Model Predictive Control for Trajectory Tracking With a Robotic Arm." In: *IEEE Robotics and Automation Letters* 4.4 (2019), pp. 3758–3765. DOI: 10.1109/LRA.2019.2929987.

[Cas+09] J. Cascio et al. “Smooth proximity computation for collision-free optimal control of multiple robotic manipulators.” In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 2452–2457. DOI: 10.1109/IROS.2009.5354382.

[CCB90] C. Chang, M. J. Chung, and Z. Bien. “Collision-free motion planning for two articulated robot arms using minimum distance functions.” In: *Robotica* 8.2 (1990), pp. 137–144. DOI: https://doi.org/10.1017/S0263574700007712.

[CE17] J. Cortés and M. Egerstedt. “Coordinated Control of Multi-Robot Systems: A Survey.” In: *SICE Journal of Control, Measurement, and System Integration* 10.6 (2017), pp. 495–503.

[CFK16] W. K. Chung, L.-C. Fu, and T. Kröger. “Motion Control.” In: *Handbook of Robotics*. Springer Cham, 2016, pp. 163–194. ISBN: 978-3-319-32550-7. DOI: 10.1007/978-3-319-32552-1.

[CG15] G. Carbone and F. Gomez-Bravo. *Motion and Operation Planning of Robotic Systems: Background and Practical Approaches*. Springer Cham, 2015. ISBN: 978-3-319-34521-5. DOI: https://doi.org/10.1007/978-3-319-14705-5.

[ČGF16] M. Čáp, J. Gregoire, and E. Frazzoli. “Provably safe and deadlock-free execution of multi-robot plans under delaying disturbances.” In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 5113–5118. DOI: 10.1109/iros.2016.7759750.

[Che+16] A. Cherubini et al. “Collaborative manufacturing with physical human–robot interaction.” In: *Robotics and Computer-Integrated Manufacturing* 40 (2016), pp. 1–13. DOI: https://doi.org/10.1016/j.rcim.2015.12.007.

[Che+21] G. Chen et al. “Path planning for manipulators based on an improved probabilistic roadmap method.” In: *Robotics and Computer-Integrated Manufacturing* 72 (2021), p. 102196. DOI: 10.1016/j.rcim.2021.102196.

[Cho+05] H. Choset et al. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT press, 2005. ISBN: 9780262255912.

[Chr+13] P. D. Christofides et al. “Distributed model predictive control: A tutorial review and future research directions.” In: *Computers & Chemical Engineering* 51 (2013), pp. 21–41. DOI: https://doi.org/10.1016/j.compchemeng.2012.05.011.

[CJS08] J. Cortés, L. Jaillet, and T. Siméon. “Disassembly Path Planning for Complex Articulated Objects.” In: *IEEE Transactions on Robotics* 24.2 (2008), pp. 475–481. DOI: 10.1109/TRO.2008.915464.

[Coe02] C. A. C. Coello. "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art." In: *Computer Methods in Applied Mechanics and Engineering* 191.11–12 (2002), pp. 1245–1287. DOI: https://doi.org/10.1016/S0045-7825(01)00323-1.

[CR79] C. R. Cutler and B. L. Ramaker. "Dynamic matrix control – A computer control algorithm." In: *Proceedings of the 1980 Joint Automatic Control Conference* 17 (1979), p. 72. DOI: 10.1109/JACC.1980.4232009.

[CWP96] J. E. Colgate, W. Wannasuphoprasit, and M. A. Peshkin. "Cobots: Robots for Collaboration With Human Operators." In: *ASME International Mechanical Engineering Congress and Exposition*. 1996, pp. 433–439. DOI: https://doi.org/10.1115/IMECE1996-0367.

[DB15] A. Dobson and K. E. Bekris. "Planning representations and algorithms for prehensile multi-arm manipulation." In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2015, pp. 6381–6386. DOI: 10.1109/iros.2015.7354289.

[DB17] R. N. Darmanin and M. K. Bugeja. "A review on multi-robot systems categorised by application domain." In: *2017 25th Mediterranean Conference on Control and Automation (MED)*. 2017, pp. 701–706. DOI: 10.1109/med.2017.7984200.

[Dec03] R. Dechter. *Constraint Processing*. Morgan Kaufmann, 2003. ISBN: 978-1-55860-890-0. DOI: https://doi.org/10.1016/B978-1-55860-890-0.X5000-2.

[Dek+18] S. A. Deka et al. "Robust and Safe Coordination of Multiple Robotic Manipulators: An Approach Using Modified Avoidance Functions." In: *Journal of Intelligent & Robotic Systems* 90 (2018), pp. 419–435. DOI: https://doi.org/10.1007/s10846-017-0699-y.

[DH20] P. B. g. Dohmann and S. Hirche. "Distributed Control for Cooperative Manipulation With Event-Triggered Communication." In: *IEEE Transactions on Robotics* 36.4 (2020), pp. 1038–1052. DOI: 10.1109/TRO.2020.2973096.

[Die+06] M. Diehl et al. "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control." In: *Fast Motions in Biomechanics and Robotics*. Ed. by M. Diehl and K. Mombaur. Vol. 340. Springer, Berlin, Heidelberg, 2006, pp. 65–93. ISBN: 978-3-540-36118-3. DOI: https://doi.org/10.1007/978-3-540-36119-0_4.

[Die+11] A. Dietrich et al. "Extensions to reactive self-collision avoidance for torque and position controlled humanoids." In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 3455–3462. DOI: 10.1109/ICRA.2011.5979862.

[Dij59] E. W. Dijkstra. "A note on two problems in connexion with graphs." In: *Numerische Mathematik* 1 (1959), pp. 269–271. DOI: 10.1007/BF01386390.

[Din+18] Y. Ding et al. "Model predictive control and its application in agriculture: A review." In: *Computers and Electronics in Agriculture* 151 (2018), pp. 104–117. DOI: 10.1016/j.compag.2018.06.004.

[Dog+19] M. Dogar et al. "Multi-robot grasp planning for sequential assembly operations." In: *Autonomous Robots* 43 (2019), pp. 649–664. DOI: https://doi.org/10.1007/s10514-018-9748-z.

[Drg+20] J. Drgoňa et al. "All you need to know about model predictive control for buildings." In: *Annual Reviews in Control* 50 (2020), pp. 190–232. DOI: 10.1016/j.arcontrol.2020.09.001.

[Dui+16] N. van Duijkeren et al. "Path-following NMPC for serial-link robot manipulators using a path-parametric system reformulation." In: *2016 European Control Conference (ECC)*. 2016, pp. 477–482. DOI: 10.1109/ecc.2016.7810330.

[EH16] S. Erhart and S. Hirche. "Model and Analysis of the Interaction Dynamics in Cooperative Manipulation Tasks." In: *IEEE Transactions on Robotics* 32.3 (2016), pp. 672–683. DOI: 10.1109/TRO.2016.2559500.

[Eva93] J. W. Evans. "Random and cooperative sequential adsorption." In: *Reviews of modern physics* 65.4 (1993), p. 1281. DOI: 10.1103/RevModPhys.65.1281.

[Fau+17] T. Faulwasser et al. "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot." In: *IEEE Transactions on Control Systems Technology* 25.4 (2017), pp. 1505–1511. DOI: 10.1109/TCST.2016.2601624.

[Fen+16] S. Feng et al. "Robust dynamic walking using online foot step optimization." In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016, pp. 5373–5378. DOI: 10.1109/iros.2016.7759790.

[FGW02] A. Forsgren, P. E. Gill, and M. H. Wright. "Interior Methods for Nonlinear Optimization." In: *SIAM review* 44.4 (2002), pp. 525–597. DOI: https://doi.org/10.1137/S0036144502414942.

[FH86] E. Freund and H. Hoyer. "Pathfinding in multi-robot systems: Solution and applications." In: *Proceedings. 1986 IEEE International Conference on Robotics and Automation*. Vol. 3. IEEE, 1986, pp. 103–111. DOI: 10.1109/robot.1986.1087653.

[FIN04] A. Farinelli, L. Iocchi, and D. Nardi. "Multirobot systems: a classification focused on coordination." In: *IEEE Transactions on Systems, Man, and Cy-*

bernetics, Part B (Cybernetics) 34.5 (2004), pp. 2015–2028. DOI: 10.1109/TSMCB.2004.832155.

[Fla+12] F. Flacco et al. "A depth space approach to human-robot collision avoidance." In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 338–345. DOI: 10.1109/ICRA.2012.6225245.

[FT87] 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*. 1987, pp. 1152–1159. DOI: 10.1109/ROBOT.1987.1087982.

[Gaf+22a] N. Gafur et al. "Dynamic Collision and Deadlock Avoidance for Multiple Robotic Manipulators." In: *IEEE Access* 10 (2022), pp. 55766–55781. DOI: 10.1109/access.2022.3176626.

[Gaf+22b] N. Gafur et al. *Dynamic path planning and reactive scheduling for a robotic manipulator using NMPC*. Accessed: 2024-08-18. 2022. URL: https://www.youtube.com/watch?v=J6H2kAi4zmc.

[Gaf+22c] N. Gafur et al. "Dynamic path planning and reactive scheduling for a robotic manipulator using nonlinear model predictive control." In: *2022 30th Mediterranean Conference on Control and Automation (MED)*. 2022, pp. 604–611. DOI: 10.1109/med54222.2022.9837147.

[Gar+14] S. Garrido-Jurado et al. "Automatic generation and detection of highly reliable fiducial markers under occlusion." In: *Pattern Recognition* 47.6 (2014), pp. 2280–2292. DOI: https://doi.org/10.1016/j.patcog.2014.01.005.

[Gar+21] C. R. Garrett et al. "Integrated task and motion planning." In: *Annual Review of Control, Robotics, and Autonomous Systems* 4 (2021), pp. 265–293. DOI: https://doi.org/10.1146/annurev-control-091420-084139.

[GJK88] E. Gilbert, D. Johnson, and S. Keerthi. "A fast procedure for computing the distance between complex objects in three-dimensional space." In: *IEEE Journal on Robotics and Automation* 4.2 (1988), pp. 193–203. DOI: 10.1109/56.2083.

[GLK18] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. "Ffrob: Leveraging symbolic planning for efficient task and motion planning." In: *The International Journal of Robotics Research* 37.1 (2018), pp. 104–136. DOI: https://doi.org/10.1177/0278364917739114.

[GLS21] J. S. Grover, C. Liu, and K. Sycara. "Deadlock Analysis and Resolution for Multi-robot Systems." In: *Algorithmic Foundations of Robotics XIV. WAFR 2020. Springer Proceedings in Advanced Robotics*. Ed. by S. LaValle et al.

Vol. 17. Springer, Cham, 2021, pp. 294–312. ISBN: 978-3-030-66723-8. DOI: https://doi.org/10.1007/978-3-030-66723-8_18.

[GP17] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer Cham, 2017, pp. XIV, 456. ISBN: 978-3-319-46023-9. DOI: https://doi.org/10.1007/978-3-319-46024-6.

[Gur23] L. Gurobi Optimization. *Gurobi Optimizer Reference Manual*. 2023. URL: https://www.gurobi.com.

[GVG23] T. Gold, A. Völz, and K. Graichen. "Model Predictive Interaction Control for Robotic Manipulation Tasks." In: *IEEE Transactions on Robotics* 39.1 (2023), pp. 76–89. DOI: 10.1109/TRO.2022.3196607.

[GWR24] N. Gafur, A. Wagner, and M. Ruskowski. *Experimental study of robot-robot cooperation*. Accessed: 2024-08-18. 2024. URL: https://www.youtube.com/watch?v=S65-6tGfBIw.

[GYR21] N. Gafur, V. Yfantis, and M. Ruskowski. "Optimal scheduling and non-cooperative distributed model predictive control for multiple robotic manipulators." In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 390–397. DOI: 10.1109/IROS51168.2021.9636118.

[Had+08] S. Haddadin et al. "Collision Detection and Reaction: A Contribution to Safe Physical Human-Robot Interaction." In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 3356–3363. DOI: 10.1109/iros.2008.4650764.

[Haj+18] I. Hajizadeh et al. "Adaptive Model Predictive Control for Nonlinearity in Biomedical Applications." In: *IFAC-PapersOnLine* 51.20 (2018), pp. 368–373. DOI: 10.1016/j.ifacol.2018.11.061.

[Han+12] C. Hansen et al. "Enhanced approach for energy-efficient trajectory generation of industrial robots." In: *2012 IEEE International Conference on Automation Science and Engineering (CASE)*. 2012, pp. 1–7. DOI: 10.1109/coase.2012.6386343.

[Har+23] V. N. Hartmann et al. "Long-Horizon Multi-Robot Rearrangement Planning for Construction Assembly." In: *IEEE Transactions on Robotics* 39.1 (2023), pp. 239–252. DOI: 10.1109/TRO.2022.3198020.

[Hay86] S. Hayati. "Hybrid position/Force control of multi-arm cooperating robots." In: *Proceedings. 1986 IEEE International Conference on Robotics and Automation*. IEEE, 1986, pp. 82–89. DOI: 10.1109/robot.1986.1087650.

[HDA17] S. Haddadin, A. De Luca, and A. Albu-Schäffer. "Robot Collisions: A Survey on Detection, Isolation, and Identification." In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1292–1312. DOI: 10.1109/TRO.2017.2723903.

[He+22] K. He et al. "Visibility Maximization Controller for Robotic Manipulation." In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 8479–8486. DOI: 10.1109/LRA.2022.3188430.

[Hen98] M. A. Henson. "Nonlinear model predictive control: current status and future directions." In: *Computers & Chemical Engineering* 23.2 (1998), pp. 187–202. DOI: https://doi.org/10.1016/S0098-1354(98)00260-9.

[HFD11] B. Houska, H. J. Ferreau, and M. Diehl. "ACADO toolkit—An open-source framework for automatic control and dynamic optimization." In: *Optimal Control Applications and Methods* 32.3 (2011), pp. 298–312. ISSN: 1099-1514. DOI: 10.1002/oca.939.

[HL17] M. Hassan and D. Liu. "Simultaneous area partitioning and allocation for complete coverage by multiple autonomous industrial robots." In: *Autonomous Robots* 41 (2017), pp. 1609–1628. DOI: https://doi.org/10.1007/s10514-017-9631-3.

[HLL19] Q. Huang, J. Lan, and X. Li. "Robotic Arm Based Automatic Ultrasound Scanning for Three-Dimensional Imaging." In: *IEEE Transactions on Industrial Informatics* 15.2 (2019), pp. 1173–1182. DOI: 10.1109/TII.2018.2871864.

[Hog84] N. Hogan. "Impedance Control: An Approach to Manipulation." In: *1984 American Control Conference*. 1984, pp. 304–313. DOI: 10.23919/acc.1984.4788393.

[HWL21] Y. He, M. Wu, and S. Liu. "A cooperative optimization strategy for distributed multi-robot manipulation with obstacle avoidance and internal performance maximization." In: *Mechatronics* 76 (2021), p. 102560. DOI: https://doi.org/10.1016/j.mechatronics.2021.102560.

[Int21] International Organization for Standardization. *ISO 8373:2021 - Robotics – Vocabulary*. 2021. URL: https://www.iso.org/standard/75539.html%C2%A8.

[JN01] M. Jager and B. Nebel. "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots." In: *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No.01CH37180)*. Vol. 3. 2001, pp. 1213–1219. DOI: 10.1109/iros.2001.977148.

[Joc20] G. Jocher. *Ultralytics YOLOv5*. Version 7.0. 2020. DOI: 10.5281/zenodo.3908559. URL: https://github.com/ultralytics/yolov5.

[JTT01] P. Jiménez, F. Thomas, and C. Torras. "3D collision detection: a survey." In: *Computers & Graphics* 25.2 (2001), pp. 269–285. DOI: https://doi.org/10.1016/S0097-8493(00)00130-8.

[Kab76] W. Kabsch. "A solution for the best rotation to relate two sets of vectors." In: *Acta Crystallographica Section A* 32.5 (1976), pp. 922–923. DOI: 10.1107/S0567739476001873.

[Kal+11] M. Kalakrishnan et al. "STOMP: Stochastic trajectory optimization for motion planning." In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 4569–4574. DOI: 10.1109/ICRA.2011.5980280.

[Kan+10] F. Kanehiro et al. "Integrating geometric constraints into reactive leg motion generation." In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 4069–4076. DOI: 10.1109/IROS.2010.5651634.

[Kav+96] L. Kavraki et al. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces." In: *IEEE Transactions on Robotics and Automation* 12.4 (1996), pp. 566–580. DOI: 10.1109/70.508439.

[KF11] S. Karaman and E. Frazzoli. "Sampling-based algorithms for optimal motion planning." In: *The International Journal of Robotics Research* 30.7 (2011), pp. 846–894. DOI: 10.1177/0278364911406761.

[KH04] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator." In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, pp. 2149–2154. DOI: 10.1109/IROS.2004.1389727.

[Kha85] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots." In: *Proceedings. 1985 IEEE International Conference on Robotics and Automation*. 1985, pp. 500–505. DOI: 10.1109/ROBOT.1985.1087247.

[KHE15] A. Khamis, A. Hussein, and A. Elmogy. "Multi-robot Task Allocation: A Review of the State-of-the-Art." In: *Cooperative Robots and Sensor Networks 2015*. Ed. by A. Koubâa and J. Martínez-de Dios. Vol. 604. Springer, Cham, 2015, pp. 31–51. ISBN: 978-3-319-18299-5. DOI: https://doi.org/10.1007/978-3-319-18299-5_2.

[KK92] J.-O. Kim and P. Khosla. "Real-time obstacle avoidance using harmonic potential functions." In: *IEEE Transactions on Robotics and Automation* 8.3 (1992), pp. 338–349. DOI: 10.1109/70.143352.

[KKK16] M. D. Killpack, A. Kapusta, and C. C. Kemp. "Model predictive control for fast reaching in clutter." In: *Autonomous Robots* 40 (2016), pp. 537–560. DOI: 10.1007/s10514-015-9492-6.

[KL00] J. Kuffner and S. LaValle. "RRT-connect: An efficient approach to single-query path planning." In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*. Vol. 2. 2000, pp. 995–1001. DOI: 10.1109/robot.2000.844730.

[KL16] L. E. Kavraki and S. M. LaValle. "Motion Planning." In: *Springer Handbook of Robotics. Springer Handbooks*. Ed. by B. Siciliano and O. Khatib. Springer, Cham, 2016, pp. 139–162. ISBN: 978-3-319-32550-7. DOI: https://doi.org/10.1007/978-3-319-32552-1_7.

[Kne+13] R. A. Knepper et al. "IkeaBot: An autonomous multi-robot coordinated furniture assembly system." In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 855–862. DOI: 10.1109/icra.2013.6630673.

[Krä+20] M. Krämer et al. "Model predictive control of a collaborative manipulator considering dynamic obstacles." In: *Optimal Control Applications and Methods* 41.4 (2020), pp. 1211–1232. DOI: https://doi.org/10.1002/oca.2599.

[Krö10] T. Kröger. *On-Line Trajectory Generation in Robotic Systems: Basic Concepts for Instantaneous Reactions to Unforeseen (Sensor) Events*. Springer Berlin, Heidelberg, 2010. ISBN: 9783642051753. DOI: 10.1007/978-3-642-05175-3.

[KW10] T. Kröger and F. M. Wahl. "Online Trajectory Generation: Basic Concepts for Instantaneous Reactions to Unforeseen Events." In: *IEEE Transactions on Robotics* 26.1 (2010), pp. 94–111. DOI: 10.1109/TRO.2009.2035744.

[Lar+99] E. Larsen et al. *Fast Proximity Queries with Swept Sphere Volumes*. Tech. rep. Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.

[LaV06] S. M. LaValle. *Planning Algorithms*. USA: Cambridge University Press, 2006. ISBN: 978-0-521-86205-9.

[Lee+14] D.-H. Lee et al. "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. DOI: 10.1109/iccas.2014.6988044.

[Lee+22] D.-H. Lee et al. "Peg-in-Hole Assembly With Dual-Arm Robot and Dexterous Robot Hands." In: *IEEE Robotics and Automation Letters* 7.4 (2022), pp. 8566–8573. DOI: 10.1109/LRA.2022.3187497.

[LG87] B. H. Lee and C. S. G. Lee. "Collision-Free Motion Planning of Two Robots." In: *IEEE Transactions on Systems, Man, and Cybernetics* 17.1 (1987), pp. 21–32. DOI: 10.1109/TSMC.1987.289330.

[Lib18] H. S. Library. *A collection of Fortran codes for large-scale scientific computation*. 2018. URL: http://www.hsl.rl.ac.uk/.

[Lju22] L. Ljung. *System Identification Toolbox*. The MathWorks, Inc. 2022.

[LK01] S. M. LaValle and J. J. Kuffner. "Rapidly-Exploring Random Trees: Progress and Prospects." In: *Algorithmic and Computational Robotics*. AK Peters/CRC Press, 2001, pp. 303–307.

[LK14] T. Lozano-Pérez and L. P. Kaelbling. "A constraint-based method for solving sequential manipulation planning problems." In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 3684–3691. DOI: 10.1109/iros.2014.6943079.

[LMG11] D. Lam, C. Manzie, and M. Good. "Application of Model Predictive Contouring Control to an X-Y Table." In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 10325–10330. DOI: https://doi.org/10.3182/20110828-6-IT-1002.01260.

[Log+18] M. Logothetis et al. "A Model Predictive Control Approach for Vision-Based Object Grasping via Mobile Manipulator." In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, pp. 1–6. DOI: 10.1109/IROS.2018.8593759.

[Lum85] V. J. Lumelsky. "On fast computation of distance between line segments." In: *Information Processing Letters* 21.2 (1985), pp. 55–61. DOI: https://doi.org/10.1016/0020-0190(85)90032-8.

[LW79] T. Lozano-Pérez and M. A. Wesley. "An algorithm for planning collision-free paths among polyhedral obstacles." In: *Communications of the ACM* 22.10 (1979), pp. 560–570. DOI: https://doi.org/10.1145/359156.359164.

[LX19] W. Li and R. Xiong. "Dynamical Obstacle Avoidance of Task- Constrained Mobile Manipulation Using Model Predictive Control." In: *IEEE Access* 7 (2019), pp. 88301–88311. DOI: 10.1109/ACCESS.2019.2925428.

[Mad+21] Á. Madridano et al. "Trajectory planning for multi-robot systems: Methods and applications." In: *Expert Systems with Applications* 173 (2021), p. 114660. DOI: https://doi.org/10.1016/j.eswa.2021.114660.

[Mar+23] T. Marcucci et al. "Motion planning around obstacles with convex optimization." In: *Science Robotics* 8.84 (2023), eadf7843. DOI: 10.1126/scirobotics.adf7843.

[Mar18] A. Marino. "Distributed Adaptive Control of Networked Cooperative Mobile Manipulators." In: *IEEE Transactions on Control Systems Technology* 26.5 (2018), pp. 1646–1660. DOI: 10.1109/TCST.2017.2720673.

[Mat+17] J. Matschek et al. "Force Feedback and Path Following using Predictive Control: Concept and Application to a Lightweight Robot." In: *IFAC-PapersOnLine* 50.1 (2017), pp. 9827–9832.

[Mav+21] A. Mavrommati et al. "An Application of Model Predictive Control to Reactive Motion Planning of Robot Manipulators." In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. 2021, pp. 915–920. DOI: 10.1109/case49439.2021.9551432.

[May+00] D. Q. Mayne et al. "Constrained model predictive control: Stability and optimality." In: *Automatica* 36.6 (2000), pp. 789–814. DOI: 10.1016/S0005-1098(99)00214-9.

[MBF18] J. A. Marvel, R. Bostelman, and J. Falco. "Multi-Robot Assembly Strategies and Metrics." In: *ACM Computing Surveys* 51.1 (2018), p. 14. DOI: 10.1145/3150225.

[MH07] J. M. Maciejowski and M. Huzmezan. *Predictive Control: With Constraints*. Prentice Hall, Harlow, 2007. ISBN: 9780201398236.

[MPP09] A. Mucherino, P. J. Papajorgji, and P. M. Pardalos. "k-Nearest Neighbor Classification." In: *Data Mining in Agriculture*. Springer New York, 2009, pp. 83–106. ISBN: 978-0-387-88615-2. DOI: 10.1007/978-0-387-88615-2_4.

[MS13] A. Montaño and R. Suárez. "An on-line coordination algorithm for multi-robot systems." In: *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*. 2013, pp. 1–7. DOI: 10.1109/etfa.2013.6648032.

[Nav+16] M. Naveau et al. "A Reactive Walking Pattern Generator Based on Nonlinear Model Predictive Control." In: *IEEE Robotics and Automation Letters* 2.1 (2016), pp. 10–17. DOI: 10.1109/LRA.2016.2518739.

[OL89] P. A. O'Donnell and T. Lozano-Pérez. "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. DOI: 10.1109/robot.1989.100033.

[Oll+22] A. Ollero et al. "Past, Present, and Future of Aerial Robotic Manipulators." In: *IEEE Transactions on Robotics* 38.1 (2022), pp. 626–645. DOI: 10.1109/TRO.2021.3084395.

[Pan+21] T. Pan et al. "A General Task and Motion Planning Framework For Multiple Manipulators." In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 3168–3174. DOI: 10.1109/iros51168.2021.9636119.

[Par+08] D.-H. Park et al. "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields." In: *Humanoids 2008 – 8th IEEE-RAS International Conference on Humanoid Robots*. 2008, pp. 91–98. DOI: 10.1109/ichr.2008.4755937.

[Par+17] S. Parascho et al. "Cooperative Fabrication of Spatial Metal Structures." In: *Fabricate 2017* (2017), pp. 24–29. DOI: https://doi.org/10.2307/j.ctt1n7qkg7.7.

[Pet+19] K. H. Petersen et al. "A review of collective robotic construction." In: *Science Robotics* 4.28 (2019), eaau8479. DOI: 10.1126/scirobotics.aau8479.

[PPM12] C. Park, J. Pan, and D. Manocha. "ITOMP: Incremental Trajectory Optimization for Real-Time Replanning in Dynamic Environments." In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 22. 1. Association for the Advancement of Artificial Intelligence (AAAI), 2012, pp. 207–215. DOI: 10.1609/icaps.v22i1.13513.

[QB03] S. Qin and T. A. Badgwell. "A survey of industrial model predictive control technology." In: *Control Engineering Practice* 11.7 (2003), pp. 733–764. DOI: https://doi.org/10.1016/S0967-0661(02)00186-7.

[Rat+09] N. Ratliff et al. "CHOMP: Gradient optimization techniques for efficient motion planning." In: *2009 IEEE International Conference on Robotics and Automation*. 2009, pp. 489–494. DOI: 10.1109/ROBOT.2009.5152817.

[Rav+20] A. A. Ravankar et al. "HPPRM: Hybrid Potential Based Probabilistic Roadmap Algorithm for Improved Dynamic Path Planning of Mobile Robots." In: *IEEE Access* 8 (2020), pp. 221743–221766. DOI: 10.1109/ACCESS.2020.3043333.

[RHB15] C. Rösmann, F. Hoffmann, and T. Bertram. "Planning of multiple robot trajectories in distinctive topologies." In: *2015 European Conference on Mobile Robots (ECMR)*. 2015, pp. 1–6. DOI: 10.1109/ECMR.2015.7324179.

[RHK15] L. Rupert, P. Hyatt, and M. D. Killpack. "Comparing Model Predictive Control and input shaping for improved response of low-impedance robots." In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. 2015, pp. 256–263. DOI: 10.1109/humanoids.2015.7363544.

[RK92] E. Rimon and D. Koditschek. "Exact robot navigation using artificial potential functions." In: *IEEE Transactions on Robotics and Automation* 8.5 (1992), pp. 501–518. DOI: 10.1109/70.163777.

[RM09] J. B. Rawlings and D. Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, LLC, 2009. ISBN: 9780975937709.

[Rös+18] C. Rösmann et al. "Exploiting Sparse Structures in Nonlinear Model Predictive Control with Hypergraphs." In: *2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. 2018, pp. 1332–1337. DOI: 10.1109/AIM.2018.8452378.

[RSS16] E. J. Rodríguez-Seda, D. M. Stipanović, and M. W. Spong. "Guaranteed Collision Avoidance for Autonomous Systems with Acceleration Constraints and Sensing Uncertainties." In: *Journal of Optimization Theory and Applications* 168 (2016), pp. 1014–1038. DOI: https://doi.org/10.1007/s10957-015-0824-7.

[RSS17] T. Rybus, K. Seweryn, and J. Z. Sasiadek. "Control System for Free-Floating Space Manipulator Based on Nonlinear Model Predictive Control (NMPC)." In: *Journal of Intelligent & Robotic Systems* 85 (2017), pp. 491–509. DOI: DOIhttps://doi.org/10.1007/s10846-016-0396-2.

[Rus+20] M. Ruskowski et al. "Production Bots für Production Level 4: Skill-basierte Systeme für die Produktion der Zukunft." In: *atp magazin* 62.9 (2020), pp. 62–71. ISSN: 2190-4111. DOI: 10.17560/atp.v62i9.2505.

[Ryb18] T. Rybus. "Obstacle avoidance in space robotics: Review of major challenges and proposed solutions." In: *Progress in Aerospace Sciences* 101 (2018), pp. 31–48. DOI: https://doi.org/10.1016/j.paerosci.2018.07.001.

[Sch+13] J. Schulman et al. "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization." In: *Robotics: Science and Systems IX* (2013). DOI: 10.15607/rss.2013.ix.031.

[Sch+20] T. Schoels et al. "CIAO*: MPC-based Safe Motion Planning in Predictable Dynamic Environments." In: *IFAC-PapersOnLine* 53.2 (2020), pp. 6555–6562. DOI: https://doi.org/10.1016/j.ifacol.2020.12.072.

[Sch+21] M. Schwenzer et al. "Review on model predictive control: An engineering perspective." In: *The International Journal of Advanced Manufacturing Technology* 117 (2021), pp. 1327–1349. DOI: 10.1007/s00170-021-07682-3.

[Sci+20] N. Scianca et al. "MPC for Humanoid Gait Generation: Stability and Feasibility." In: *IEEE Transactions on Robotics* 36.4 (2020), pp. 1171–1188. DOI: 10.1109/TRO.2019.2958483.

[Sho+20] R. Shome et al. "drrT*: Scalable and informed asymptotically-optimal multi-robot motion planning." In: *Autonomous Robots* 44.3 (2020), pp. 443–467. DOI: https://doi.org/10.1007/s10514-019-09832-9.

[SHV06] M. W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Vol. 33. 5. Wiley, 2006, p. 608. ISBN: 978-1-119-52404-5. DOI: 10.1108/ir.2006.33.5.403.1.

[Sic+09] B. Siciliano et al. *Robotics: Modelling, Planning and Control*. Springer London, 2009. ISBN: 978-1-84628-641-4. DOI: https://doi.org/10.1007/978-1-84628-642-1.

[SL02] G. Sanchez and J.-C. Latombe. "Using a PRM planner to compare centralized and decoupled planning for multi-robot systems." In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. Vol. 2. IEEE. IEEE, 2002, pp. 2112–2119. DOI: 10.1109/robot.2002.1014852.

[ŞMK12] I. A. Şucan, M. Moll, and L. E. Kavraki. "The Open Motion Planning Library." In: *IEEE Robotics & Automation Magazine* 19.4 (2012), pp. 72–82. DOI: 10.1109/MRA.2012.2205651. URL: https://ompl.kavrakilab.org.

[SN11] R. L. A. Shauri and K. Nonami. "Assembly manipulation of small objects by dual-arm manipulator." In: *Assembly Automation* 31.3 (2011), pp. 263–274. DOI: https://doi.org/10.1108/01445151111150604.

[SS00] L. Sciavicco and B. Siciliano. *Modelling and Control of Robot Manipulators*. Springer London, 2000, pp. XXIV, 378. ISBN: 978-1-85233-221-1. DOI: https://doi.org/10.1007/978-1-4471-0449-0.

[SSH15] 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." In: *Algorithmic Foundations of Robotics XI. Springer Tracts in Advanced Robotics*. Vol. 107. 5. Springer, Cham, 2015, pp. 591–607. ISBN: 978-3-319-16594-3. DOI: https://doi.org/10.1177/0278364915615688.

[Sta18] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: https://www.ros.org.

[Str04] M. Strandberg. "Augmenting RRT-planners with local trees." In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*. Vol. 4. 2004, pp. 3258–3262. DOI: 10.1109/ROBOT.2004.1308756.

[Sug+09] K. Sugihara et al. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley & Sons, 2009, p. 696. ISBN: 978-0-470-31785-3.

[Sul+17] W. R. Sultana et al. "A review on state of art development of model predictive control for renewable energy applications." In: *Renewable and Sustainable Energy Reviews* 76 (2017), pp. 391–406.

[SY21] P. Shi and B. Yan. "A Survey on Intelligent Control for Multiagent Systems." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51.1 (2021), pp. 161–175. DOI: 10.1109/TSMC.2020.3042823.

[SZ21] Y. Shi and K. Zhang. "Advanced model predictive control framework for autonomous intelligent mechatronic systems: A tutorial overview and perspectives." In: *Annual Reviews in Control* 52 (2021), pp. 170–196. DOI: 10.1016/j.arcontrol.2021.10.008.

[SZP18] F. Suárez-Ruiz, X. Zhou, and Q.-C. Pham. "Can robots assemble an IKEA chair?" In: *Science Robotics* 3.17 (2018), eaat6385. DOI: 10.1126/scirobotics.aat6385.

[Tal+18] R. Tallamraju et al. "Decentralized MPC based Obstacle Avoidance for Multi-Robot Target Tracking Scenarios." In: *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2018, pp. 1–8. DOI: 10.1109/ssrr.2018.8468655.

[TB21] 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. DOI: 10.23919/ecc54610.2021.9654910.

[TB24] A. Tika and N. Bajcinca. "Predictive Control of Cooperative Robots Sharing Common Workspace." In: *IEEE Transactions on Control Systems Technology* 32.2 (2024). DOI: 10.1109/TCST.2023.3331525.

[TGB22] A. Tika, F. Gashi, and N. Bajcinca. "Robot Online Task and Trajectory Planning using Mixed-Integer Model Predictive Control." In: *2022 European Control Conference (ECC)*. 2022, pp. 2005–2011. DOI: 10.23919/ecc55457.2022.9838243.

[Tha+22] S. Thakar et al. "A Survey of Wheeled Mobile Manipulation: A Decision-Making Perspective." In: *Journal of Mechanisms and Robotics* 15.2 (2022), p. 020801. DOI: https://doi.org/10.1115/1.4054611.

[THR22] S. Taheri, P. Hosseini, and A. Razban. "Model predictive control of heating, ventilation, and air conditioning (HVAC) systems: A state-of-the-art review." In: *Journal of Building Engineering* 60 (2022), p. 105067. DOI: 10.1016/j.jobe.2022.105067.

[Tik+20] A. Tika et al. "Optimal Scheduling and Model Predictive Control for Trajectory Planning of Cooperative Robot Manipulators." In: *IFAC-PapersOnLine* 53.2 (2020), pp. 9080–9086. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2020.12.2136.

[Tou86] P. Tournassoud. "A strategy for obstacle avoidance and its application to mullti-robot systems." In: *Proceedings. 1986 IEEE International Conference*

on Robotics and Automation. IEEE, 1986, pp. 1224–1229. DOI: 10.1109/robot.1986.1087543.

[Ume91] S. Umeyama. "Least-squares estimation of transformation parameters between two point patterns." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (1991), pp. 376–380. DOI: 10.1109/34.88573.

[Vaz+14] S. Vazquez et al. "Model Predictive Control: A Review of Its Applications in Power Electronics." In: *IEEE Industrial Electronics Magazine* 8.1 (2014), pp. 16–31. DOI: 10.1109/MIE.2013.2290138.

[VDS11] L. Van den Broeck, M. Diehl, and J. Swevers. "Model predictive control for time-optimal point-to-point motion control." In: *IFAC Proceedings Volumes* 44.1 (2011), pp. 2458–2463. DOI: https://doi.org/10.3182/20110828-6-IT-1002.01784.

[Vil+18] V. Villani et al. "Survey on human–robot collaboration in industrial settings: Safety, intuitive interfaces and applications." In: *Mechatronics* 55 (2018), pp. 248–266. DOI: https://doi.org/10.1016/j.mechatronics.2018.02.009.

[VRW07] A. N. Venkat, J. B. Rawlings, and S. J. Wright. "Distributed Model Predictive Control of Large-Scale Systems." In: *Lecture Notes in Control and Information Sciences*. Springer Berlin Heidelberg, 2007, pp. 591–605. ISBN: 9783540726982. DOI: 10.1007/978-3-540-72699-9_50.

[WAE17] L. Wang, A. D. Ames, and M. Egerstedt. "Safety Barrier Certificates for Collisions-Free Multirobot Systems." In: *IEEE Transactions on Robotics* 33.3 (2017), pp. 661–674. DOI: 10.1109/TRO.2017.2659727.

[War89] C. Warren. "Global path planning using artificial potential fields." In: *Proceedings, 1989 International Conference on Robotics and Automation*. Vol. 1. 1989, pp. 316–321. DOI: 10.1109/ROBOT.1989.100007.

[WB06] A. Wächter and L. T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming." In: *Mathematical Programming* 106.1 (2006), pp. 25–57. ISSN: 1436-4646. DOI: 10.1007/s10107-004-0559-y.

[WC15] G. Wagner and H. Choset. "Subdimensional expansion for multirobot path planning." In: *Artificial Intelligence* 219 (2015), pp. 1–24. DOI: 10.1016/j.artint.2014.11.001.

[WLW16] M. Wang, J. Luo, and U. Walter. "A non-linear model predictive controller with obstacle avoidance for a space robot." In: *Advances in Space Research* 57.8 (2016), pp. 1737–1746. DOI: https://doi.org/10.1016/j.asr.2015.06.012.

[Woo09] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2009, p. 488. ISBN: 978-0-470-51946-2.

[XZA10] E. K. Xidias, P. T. Zacharia, and N. A. Aspragathos. "Time-Optimal Task Scheduling for Two Robotic Manipulators Operating in a Three-Dimensional Environment." In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 224.7 (2010), pp. 845–855. DOI: https://doi.org/10.1243/09596518JSCE949.

[YCH19] H. J. Yoon, S. Y. Chung, and M. J. Hwang. "Shadow Space Modeling and Task Planning for Collision-free Cooperation of Dual Manipulators for Planar Task." In: *International Journal of Control, Automation and Systems* 17 (2019), pp. 995–1006. DOI: https://doi.org/10.1007/s12555-018-0236-1.

[YSV21] L. Yan, T. Stouraitis, and S. Vijayakumar. "Decentralized Ability-Aware Adaptive Control for Multi-Robot Collaborative Manipulation." In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 2311–2318. DOI: 10.1109/LRA.2021.3060379.

[ZA05] P. T. Zacharia and N. A. Aspragathos. "Optimal robot task scheduling based on genetic algorithms." In: *Robotics and Computer-Integrated Manufacturing* 21.1 (2005), pp. 67–79. DOI: https://doi.org/10.1016/j.rcim.2004.04.003.

[ZA15] J. Zhou and Y. Aiyama. "On-line collision avoidance system for two PTP command-based manipulators with distributed controller." In: *Advanced Robotics* 29.4 (2015), pp. 239–251. DOI: https://doi.org/10.1080/01691864.2014.985610.

[Zha+20] Z. Zhang et al. "Two Hybrid End-Effector Posture-Maintaining and Obstacle-Limits Avoidance Schemes for Redundant Robot Manipulators." In: *IEEE Transactions on Industrial Informatics* 16.2 (2020), pp. 754–763. DOI: 10.1109/TII.2019.2922694.

[Zha+21] X. Zhang et al. "Task-Space Decomposed Motion Planning Framework for Multi-Robot Loco-Manipulation." In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 8158–8164. DOI: 10.1109/icra48506.2021.9560902.

[Zho+16] C. Zhou et al. "A generic optimization-based framework for reactive collision avoidance in bipedal locomotion." In: *2016 IEEE International Conference on Automation Science and Engineering (CASE)*. 2016, pp. 1026–1033. DOI: 10.1109/COASE.2016.7743516.

[Zhu+24] T. Zhu et al. "Real-Time Dynamic Obstacle Avoidance for Robot Manipulators Based on Cascaded Nonlinear MPC With Artificial Potential Field." In:

IEEE Transactions on Industrial Electronics 71.7 (2024), pp. 7424–7434. DOI: 10.1109/TIE.2023.3306405.

[ZJW23] J. Zhang, L. Jin, and Y. Wang. "Collaborative Control for Multimanipulator Systems With Fuzzy Neural Networks." In: *IEEE Transactions on Fuzzy Systems* 31.4 (2023), pp. 1305–1314. DOI: 10.1109/TFUZZ.2022.3198855.

[ZLB21] X. Zhang, A. Liniger, and F. Borrelli. "Optimization-Based Collision Avoidance." In: *IEEE Transactions on Control Systems Technology* 29.3 (2021), pp. 972–983. DOI: 10.1109/TCST.2019.2949540.

[Zub15] A. Zube. "Cartesian nonlinear model predictive control of redundant manipulators considering obstacles." In: *2015 IEEE International Conference on Industrial Technology (ICIT)*. 2015, pp. 137–142. DOI: 10.1109/ICIT.2015.7125089.