IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 12, 2022, accepted May 4, 2022, date of publication May 20, 2022, date of current version May 31, 2022.

*Digital Object Identifier 10.1109/ACCESS.2022.3176626*

# Dynamic Collision and Deadlock Avoidance for Multiple Robotic Manipulators

**NIGORA GAFUR[1], GAJANAN KANAGALINGAM[1], ACHIM WAGNER[2], (Member, IEEE), AND MARTIN RUSKOWSKI[1,2]**

[1]Chair of Machine Tools and Control Systems, Department of Mechanical and Process Engineering, Technische Universität Kaiserslautern, 67663 Kaiserslautern, Germany
[2]German Research Center for Artificial Intelligence (DFKI), 67663 Kaiserslautern, Germany

Corresponding author: Nigora Gafur (nigora.gafur@mv.uni-kl.de)

This work was funded by the Ministry of Economics, Transport, Agriculture and Viticulture of the State of Rhineland-Palatinate through the Project "Building a Collaborative and Cooperative Robotics Platform—Kollaborative und Kooperative Robotikplattform (KoKoBot)."

**ABSTRACT** A flexible operation of multiple robotic manipulators operating in a dynamic environment requires online trajectory planning to ensure collision-free trajectories. In this work, we propose a real-time capable motion control algorithm, based on nonlinear model predictive control, which accounts for static and dynamic obstacles. The proposed algorithm is realized in a distributed scheme, where each robot optimizes its own trajectory with respect to the related objective and constraints. We propose a novel approach for collision avoidance between multiple robotic manipulators, where each robot accounts for the predicted movement of the neighboring robots. Additionally, we propose a method to reliably detect and resolve deadlocks occurring in a setup of multiple robotic manipulators. We validate our approach on pick and place scenarios involving multiple robotic manipulators operating in a common workspace in a realistic simulation environment set up in Gazebo. The robots are controlled using the Robot Operating System. Our approach scales up to 4 manipulators and computes a path for each robot in a simultaneous pick and place operation in 94% of all investigated cases without deadlock detection and 100 % of cases with the proposed deadlock resolution algorithm. In contrast, the investigated conventional path planners, such as PRM, PRM*, CHOMP and RRT-Connect, successfully plan a trajectory in at most 54% of all investigated cases for a simultaneous operation of 4 robotic manipulators hindering their application in setups of multiple manipulators.

**INDEX TERMS** Robotic manipulators, collision avoidance, distributed model predictive control, motion control, deadlock, ROS.

## I. INTRODUCTION

Modern industrial processes are increasingly dominated by shorter innovation and product life cycles, reflecting a growing demand for customized products [1]. Consequently, factory systems must become more flexible and adaptable [2], [3]. Robotic manipulators can provide such flexibility due to their complex kinematic chain. Areas of application are, for instance, assembly, disassembly or packaging lines. By operating in a shared workspace, several robotic manipulators can further increase efficiency, minimize the working area and make collaboration possible. Fig. 1 constitutes an example of four robotic manipulators sharing the same workspace and performing a pick and place task.

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng H. Zhu.

Traditionally, the collision-free trajectories of all involved robotic manipulators in industrial applications are planned for specific tasks and static environments. As robotic manipulators are generally deployed for repetitive tasks, it suffices to plan collision-free trajectories only once. In case certain parts of the production process are changed, a re-planning of collision-free trajectories and re-programming of all involved manipulators is necessary. For that reason, it is imperative to develop efficient, scalable and real-time capable motion control strategies which allow a safe and flexible operation of multiple manipulators in changing environments. Such strategies would enable, for example, an on-demand task assignment in a multi-robot setting. Furthermore, modular approaches are conceivable, where each robot may be considered as an independent module. The ability to couple and rearrange such modules in a flexible way would be

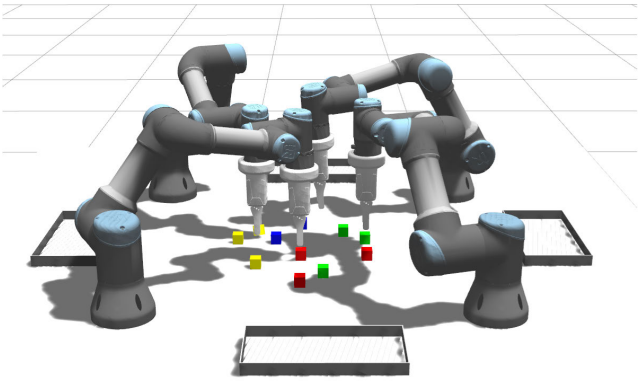


**FIGURE 1. Setup for a pick and place scenario with four collaborative UR3 manipulators.**

highly desirable from the point of view of modern production processes.

One promising concept for industrial applications is Model Predictive Control (MPC), which emerged in 1960s and was initially applied for multivariable constrained control problems in oil and chemical industries [4]. The method utilizes a model of the system dynamics to forecast its future behavior and to optimize decisions in the future. The approach allows the incorporation of physical constraints of the system, e.g., limiting input voltage or maximum speed of a considered system. Besides that, the main advantages of the MPC method are its predictive nature and online computation scheme, that enables its application in various industrial settings. In the past, MPC was primarily used for processes with slow dynamics and long sampling times due to the need to solve an optimization problem in each time step [5]. With increasing computational performance, problems with faster system dynamics can be treated, e.g., in automotive systems and robotics [6].

## II. CONTRIBUTION AND OUTLINE

This article is concerned with developing an online motion control algorithm which enables several manipulators to operate simultaneously in a common workspace. We formulate the problem of online motion control for each manipulator as an optimization problem in the joint space, which incorporates static and dynamic collision avoidance constraints. To this end, we derive a novel approach for collision avoidance between multiple robots which enables a safe robot-robot interaction. It is based on MPC to account for disturbances and uncertainties during motion control. Moreover, we use the predictive nature of MPC to exchange information between the robots and thus to account for collisions a priori. We take special care to ensure that our approach can be applied for real-world applications as an online trajectory planner. To this end, we use the concept of Distributed Model Predictive Control (DMPC) in the joint space, where each robotic manipulator is considered as an agent and shares the predicted trajectory with its neighbors once at each time step. Collision avoidance between robots is not sufficient to overcome the problem of deadlocks. To this end, we introduce a concept of how deadlocks among two or more robotic manipulators can be detected and, subsequently, resolved. Finally, there exists no approach so far, known to the authors, which combines all the mentioned contributions in one system simultaneously.

To demonstrate the efficiency of our approach, we consider a setup of multiple 6-degrees of freedom robotic manipulators in the simulation environment *Gazebo* [7], controlled by the Robot Operating System (ROS) [8]. The robotic manipulators are closely placed to each other and operate in a common workspace. Each robot is assigned several pick and place tasks. We propose a modular approach for the practical application, where each robot is placed on an independent module. Cooperation between robots is possible by coupling modules to each other. This approach has the advantage of realizing different setups of multi-robot systems depending on how many robots and what constellation of robots are required to fulfill a task. Further, we compare our approach for motion planning with sampling-based and optimization-based planners to show its efficiency. Finally, we compare computation times for different setups and draw conclusions about scalability of our approach.

The remainder of this article is organized as follows. In Section III we elaborate on existing trajectory generation methods and multi-robot planners. In Section IV the dynamic model of a robotic manipulator is introduced, followed by a formulation of the DMPC problem in Section V. A novel approach for collision avoidance is explained in detail in Section VI. Further, we introduce a new approach for deadlock detection and resolution in Section VII. Validation of our algorithm and simulation results are shown in Section VIII followed by concluding remarks in Section IX.

## III. RELATED WORK

Motion planning is still an on-going and challenging research area in robotics. In industrial applications, trajectory generation of manipulators is usually required, in addition to its feasibility, to minimize certain criteria, such as the distance traveled or traveling time, and maximizing others, such as energy efficiency or performance. It is necessary to consider dynamically changing environments to allow for a flexible operation of a manipulator. In multi-robot systems, each robot has to find a feasible path in a complex and constantly changing environment while sharing its workspace with other robots. In general, the applied methods for trajectory generation in robotic applications can be divided into two main categories: sampling-based and control-based methods.

Sampling-based methods include the widely used algorithms based on either Rapidly Exploring Random Trees (RRT's) [9] or Probabilistic Roadmaps (PRM's) [10]. The RRT method is realized as a multi-query planner, whereas the PRM method is a single-query planner [11]. The sampling-based planners are suitable for high-dimensional

configuration spaces and thus for multi-robot systems, which is the main advantage of these methods. Several sampling-based approaches exist for multi-robot motion planning, such as discrete RRT (dRRT) [12] and subdimensional expansion [13]. Recently, an asymptotically optimal extension of dRRT was introduced denoted as dRRT*, that was successfully applied for 4 robotic arms, each with 7 degrees of freedom sharing a common workspace [14]. However, the sampling-based methods are mainly applied for static environments, as the trajectories are first planned for a specific task and thereafter executed. The methods are therefore mainly used for offline trajectory planning. Further limitation of the sampling-based method includes difficulties in planning trajectories for narrow passages that often lead to jerky and unnecessary motions [15]. The Open Motion Planning Library (OMPL) [16] includes a large variety of sampling-based planners which are also integrated in the Robot Operating System (ROS) [8] framework.

Control-based planners require a more tailor-made approach depending on the type of robot. This category includes artificial potential fields [17] and optimization-based approaches [18]–[20]. Both methods search for a feasible path towards the goal based on local information from the environment. The artificial potential field method uses a potential function that induces repulsive forces against obstacles and attractive forces towards the goal. Wang *et al.* [21] applied this method for a space manipulator with multiple obstacles occupying the same workspace. Obstacles were only considered if the manipulator undercut a predefined minimal distance to the individual objects. Bosscher *et al.* [22] applied velocity damping for a cooperative motion planning of two robotic manipulators, where a trajectory was planned for each robot in advance and collisions were considered only during the execution of the trajectory. The main drawback of the potential field method is its limitation to a low-dimensional configuration space.

Optimization-based approaches are usually formulated as constrained optimization problems where a kinematic and a dynamic model of the corresponding robot are incorporated in the constraints of the optimization problem. Additionally, static as well as dynamic obstacles may be considered by adding additional constraints to the optimization problem. However, a large number of constraints can result in high computational burden. Therefore, an efficient incorporation of constraints is required, especially for multi-robot systems in a dynamic environment. The concept of MPC [23] in a receding horizon formulation is suitable for solving the trajectory generation problem for a dynamic environment by formulating an optimization problem that is solved over a prediction horizon of finite length. Exchange of information with other robots enables to account for potential collisions a priori. Further, the closed-loop control accounts for model uncertainties and disturbances.

Lam *et al.* [24], Arkadani *et al.* [25] and Belda *et al.* [26] carried out trajectory generation with MPC for a single robotic manipulator without collision avoidance. There are two possible approaches integrating collision avoidance into trajectory generation with MPC. In the first approach, the MPC algorithm itself is extended by solving the optimization problem not over the whole state space of the considered system, but only over a subset of the state space. This subset excludes all states where a collision might occur and needs to be determined a priori. This method was applied by Liu *et al.* [27] and Schoels *et al.* [28] for trajectory generation of a mobile robot, where Schoels *et al.* [28] approximated the collision-free subset by circles and Liu *et al.* [27] used polyhedra at the current state. Rösmann *et al.* [29] used a global planner to optimize trajectories of multiple mobile robots maneuvering in a low dimensional space.

The second approach to integrate collision avoidance into trajectory generation with MPC is to introduce further constraints to the optimization problem. There exist several approaches to formulate these constraints. One approach is to restrict the distance of all collision-prone object pairs, where the corresponding objects are approximated by convex bodies. Thus, for a kinematic model of a manipulator this results in a connected chain of convex bodies [22], [30], [31], where each pair of collision-prone bodies introduces an additional constraint into the optimization problem, e.g., in case of multiple manipulators or a manipulator and a human.

The computation of distances between two convex bodies is performed by algorithms with nested logical conditions [32]–[34]. However, the derivatives of the constraints are not smooth, which poses additional challenges to solving the underlying optimization problem. Krämer *et al.* [31] extended the collision avoidance approach from Lumelsky [32] and proposed an online motion control for one robotic manipulator in collaboration with a human. The computation times prove the efficiency of the approach, where the optimization problem is solved with a self-developed hypergraph [35] to mitigate the problem of nested logical conditions.

As an alternative to restricting the distance, virtual hyperplanes can be used to separate two collision-prone bodies. By approximating the considered objects by polyhedra and applying Farkas' lemma, collisions of the considered objects can be avoided. An implementation with multidimensional polyhedra was proposed by Gerdts *et al.* for a robotic manipulator [36]. In the work of Zhang *et al.* [37] this approach was extended so that, in addition to collision avoidance, a minimum distance between two bodies can be guaranteed. The former approach comes with the disadvantage, that for every pair of collision-prone objects, several constraints have to be added to the underlying optimization problem. Six new optimization variables have to be introduced into the optimization problem for each considered object pair. The number of additional constraints depends linearly on the number of polyhedron faces, which is computationally intractable for multi-robot systems.

The framework of MPC can be realized in a centralized or distributed scheme. The drawback of the Centralized MPC (CMPC) is the limited scalability and high

computational complexity [38]. A DMPC framework can help to split the computational burden, where each agent optimizes its own objective function [38]. This concept was already introduced for robot-human collaboration by Flad *et al.* [39]. Yanhao *et al.* [40] proposed an approach based on a distributed control for a cooperative manipulation of an object. Tika *et al.* applied CMPC [41] and DMPC [42] for a synchronous pick and place scenario for two robotic manipulators. However, the focus lies on a synchronous task completion for two robotic manipulators rather than collision avoidance. Furthermore, deadlocks are not treated in any of the mentioned works. Reliably and temporally detecting deadlocks between a group of robots is a challenging task.

## IV. DYNAMICAL MODEL

We consider a robotic manipulator with $N$ joints. The dynamical model of a manipulator can be derived by Lagrange's equations of the second kind and formulated in a matrix form as

$$\mathbf{M}(\mathbf{q}(t))\ddot{\mathbf{q}}(t) + \mathbf{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t))\dot{\mathbf{q}}(t) + \mathbf{g}(\mathbf{q}(t)) = \boldsymbol{\tau}(t), \quad (1)$$

where $\mathbf{q}(t) \in \mathbb{R}^N$ denotes the vector of generalized coordinates, which is the joint angular position vector. The inertia matrix is denoted by $\mathbf{M}(\mathbf{q}(t)) \in \mathbb{R}^{N\times N}$, $\mathbf{C}(\mathbf{q}(t), \dot{\mathbf{q}}(t)) \in \mathbb{R}^{N\times N}$ maps the angular velocities $\dot{\mathbf{q}}(t)$ to Coriolis and centrifugal torques, $\mathbf{g}(\mathbf{q}(t)) \in \mathbb{R}^N$ is the vector of gravitational and $\boldsymbol{\tau} \in \mathbb{R}^N$ denotes the vector of generalized torques.

With feedback linearization, i.e., inner-loop control, which linearizes the nonlinearity of the system, we obtain a system dynamics of $N$ double integrators, where each joint is independently controlled [43]. As a result, the dynamical model of a manipulator applied in this work admits the representation

$$\begin{bmatrix}\dot{\mathbf{q}}(t)\\ \ddot{\mathbf{q}}(t)\end{bmatrix} = \begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{q}(t)\\ \dot{\mathbf{q}}(t)\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{u}(t), \quad (2)$$

where $\mathbf{u}(t) \in \mathbb{R}^N$ denotes the control input vector, the matrix $\mathbf{I} \in \mathbb{R}^{N\times N}$ denotes the identity matrix and $\mathbf{0} \in \mathbb{R}^{N\times N}$ represents the zero matrix.

We derive a discrete-time representation of the linear system with the state vector $\mathbf{x}(t) = [\mathbf{q}^{\mathrm{T}}(t),\ \dot{\mathbf{q}}^{\mathrm{T}}(t)]^{\mathrm{T}} \in \mathbb{R}^{2N}$ in the state-space

$$\mathbf{x}^{k+1} = \mathbf{A}^{\mathrm{d}}\mathbf{x}^k + \mathbf{B}^{\mathrm{d}}\mathbf{u}^k, \quad (3)$$

where $\mathbf{A}^{\mathrm{d}} \in \mathbb{R}^{2N\times 2N}$ represents the discrete state matrix and $\mathbf{B}^{\mathrm{d}} \in \mathbb{R}^{2N\times N}$ the input matrix. Equation (3) is discretized with a sample time $T_{\mathrm{s}}$, where $(\cdot)^k$ represent discrete variables at time $t_k = k \cdot T_s$. The discrete states are denoted in the following as $\mathbf{x}_i^k = \mathbf{x}_i(t_k)$ and discrete control inputs as $\mathbf{u}_i^k = \mathbf{u}_i(t_k)$ for a manipulator $i$. The linear system in (2) describes the dynamics of a robotic manipulator in the joint space, which will be integrated as a constraint into an optimization problem. This will be discussed in more detail in Section V.

## V. DISTRIBUTED MODEL PREDICTIVE CONTROL

In the next step, we consider a system of $M$ robotic manipulators. Each manipulator $i = 1, \ldots, M$ represents an independent subsystem. The main objective of performing a cooperative task for every robotic manipulator is to safely reach the target joint state accounting for static, dynamic and self-collision constraints.

CMPC considers the system dynamics of all subsystems in a single optimization problem with respect to an aggregated objective function. However, the degrees of freedom of the centralized approach increase linearly with an increasing number of robots such that the computational costs quickly becomes inadmissible for real-time applications. In this work, we investigate DMPC where each robot is considered as an agent. For robotic manipulators working independently in a shared workspace, as e.g. pick and place tasks, the system dynamics are decoupled in states and control inputs.

### A. FORMULATION OF DMPC PROBLEM FOR A MULTI-AGENT SYSTEM

In this work, we consider a single communication iteration between the robots at each time step, where they are allowed to share their current and predicted states with all other agents.

Keeping the former in mind, we turn our attention to the formulation of the online trajectory planning problem based on DMPC. We choose the multiple shooting method for discretizing the optimization problem. The optimization problem is split equidistantly into $t_k = k \cdot T_{\mathrm{s}}$ time steps with $k = 0, \ldots, N_{\mathrm{p}}$, where $N_{\mathrm{p}}$ denotes the prediction horizon. For the sake of brevity, we choose the following notation for decision variables, i.e., states $\mathbf{x}_i^{0:N_{\mathrm{p}}} = [\mathbf{x}_i^0, \ldots, \mathbf{x}_i^{N_{\mathrm{p}}}]$ and control inputs $\mathbf{u}_i^{0:N_{\mathrm{p}}-1} = [\mathbf{u}_i^0, \ldots, \mathbf{u}_i^{N_{\mathrm{p}}-1}]$. The DMPC formulation for each involved robotic manipulator $i$ takes the following form

$$\min_{\mathbf{u}_i^{0:N_{\mathrm{p}}-1}, \mathbf{x}_i^{0:N_{\mathrm{p}}}} J_i^{\mathrm{f}}(\mathbf{x}_i^{N_{\mathrm{p}}}) + \sum_{k=0}^{N_{\mathrm{p}}-1} J_i^{\mathrm{c}}(\mathbf{x}_i^k, \mathbf{u}_i^k) \quad (4)$$

$$\text{s.t. } \mathbf{x}_i^{k+1} = \mathbf{A}_i^{\mathrm{d}}\mathbf{x}_i^k + \mathbf{B}_i^{\mathrm{d}}\mathbf{u}_i^k, \quad k = 0, \ldots, N_{\mathrm{p}} - 1, \quad (4a)$$

$$\mathbf{x}_i^0 = \mathbf{x}_i^{\mathrm{s}}, \quad (4b)$$

$$\mathbf{x}_i^k \in \mathbb{X}_i, \quad k = 0, \ldots, N_{\mathrm{p}}, \quad (4c)$$

$$\mathbf{u}_i^k \in \mathbb{U}_i, \quad k = 0, \ldots, N_{\mathrm{p}} - 1, \quad (4d)$$

$$R(\mathbf{x}_i^k) \cap \mathcal{O} = \varnothing, \quad k = 0, \ldots, N_{\mathrm{p}}, \quad (4e)$$

$$R(\mathbf{x}_i^k) \cap \mathcal{R}(\underline{\mathbf{x}}_{-i}^{*\,k}) = \varnothing, \quad k = 0, \ldots, N_{\mathrm{p}}. \quad (4f)$$

The quadratic cost function $J_i^{\mathrm{c}} : \mathbb{R}^{2N} \times \mathbb{R}^N \to \mathbb{R}$,

$$J_i^{\mathrm{c}}(\mathbf{x}_i^k, \mathbf{u}_i^k) := (\mathbf{x}_i^k - \mathbf{x}_i^{\mathrm{f}})^{\mathrm{T}}\mathbf{Q}_i^x(\mathbf{x}_i^k - \mathbf{x}_i^{\mathrm{f}}) + \mathbf{u}_i^{k\,\mathrm{T}}\mathbf{R}_i^{\mathrm{u}}\mathbf{u}_i^k + \Delta\mathbf{u}_i^{k\,\mathrm{T}}\mathbf{R}_i^{\mathrm{d}}\Delta\mathbf{u}_i^k \quad (5)$$

penalizes 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 $\Delta\mathbf{u}_i^k = \frac{\mathbf{u}_i^{k+1} - \mathbf{u}_i^k}{T_{\mathrm{s}}}$, with the positive

definite weighting matrices $\mathbf{Q}_i^{\mathrm{x}} \in \mathbb{R}^{2N\times 2N}$, $\mathbf{R}_i^{\mathrm{u}} \in \mathbb{R}^{N\times N}$ and $\mathbf{R}_i^{\mathrm{d}} \in \mathbb{R}^{N\times N}$, respectively. The terminal state cost $J_i^{\mathrm{f}} : \mathbb{R}^{2N} \to \mathbb{R}$

$$J_i^{\mathrm{f}}(\mathbf{x}_i^{N_\mathrm{p}}) := (\mathbf{x}_i^{N_\mathrm{p}} - \mathbf{x}_i^{\mathrm{f}})^{\mathrm{T}} \mathbf{Q}_i^{\mathrm{f}} (\mathbf{x}_i^{N_\mathrm{p}} - \mathbf{x}_i^{\mathrm{f}}) \tag{6}$$

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

The dynamics of manipulator $i$ is given by equation (4a), see Section IV, whereas the equation (4b) 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 equations (4c) and (4d) represent lower and upper bounds on the states and control inputs, i.e.,

$$\begin{aligned} \mathbb{X}_i &:= \{\mathbf{x}_i^k \in \mathbb{R}^{2N} \mid \mathbf{x}_{i,\min} \le \mathbf{x}_i^k \le \mathbf{x}_{i,\max}\}, \\ \mathbb{U}_i &:= \{\mathbf{u}_i^k \in \mathbb{R}^{N} \mid \mathbf{u}_{i,\min} \le \mathbf{u}_i^k \le \mathbf{u}_{i,\max}\}. \end{aligned} \tag{7}$$

The equation for joint angles (7) also accounts for self-collision constraints through limitation of the joints' angle ranges.

We formulate static and dynamic collision avoidance constraints in the task space, transforming joint positions into positions in Cartesian space using the nonlinear forward kinematics. The forward kinematics of a manipulator turns the optimization problem (4) into a non-convex one. This concerns the constraints (4e) and (4f) where forward kinematics is applied. $R(\mathbf{x}_i^k)$ denotes the interior set of Cartesian points occupied by manipulator $i$ with state $\mathbf{x}_i^k$. The trajectory vector $\underline{\mathbf{x}}^k = [\mathbf{x}_1^k, \ldots, \mathbf{x}_M^k]$ collects the trajectories of all involved robots. To account for static objects in the task space, we define the set $\mathcal{O}$ containing all interior points of all static obstacles. Indeed, constraint (4e) enforces that the intersection of $R(\mathbf{x}_i^k)$ and the obstacles $\mathcal{O}$ for state $\mathbf{x}_i^k$ is empty. To consider dynamic collision avoidance constraints, i.e., the prevention of inter-robot collisions, the short-hand notation

$$\mathcal{R}(\underline{\mathbf{x}}_{-i}^k) := \bigcup_{j=1, j\neq i}^{M} R(\mathbf{x}_j^k) \tag{8}$$

is introduced. Consequently, constraint (4f) prevents the inter-robot collision of robot $i$ with all other robots. The efficient implementation of constraints (4e) and (4f) is the topic of the subsequent section. In the following, we focus on solving the DMPC.

Note, that constraint (4f) establishes the coupling between the manipulators in states. Constraint (4f) implies that the optimal trajectories of all other robots, collected in $\underline{\mathbf{x}}_{-i}^{*\,k}$, is known a priori in order to solve the optimization problem (4) for manipulator $i$. To obtain

$$\underline{\mathbf{x}}_{-i}^{*\,0:N_\mathrm{p}} = [\mathbf{x}_1^{*\,0:N_\mathrm{p}}, \ldots, \mathbf{x}_{i-1}^{*\,0:N_\mathrm{p}}, \mathbf{x}_{i+1}^{*\,0:N_\mathrm{p}}, \ldots, \mathbf{x}_M^{*\,0:N_\mathrm{p}}], \tag{9}$$

for collision constraint (4f) we use an extrapolation approach [38]. Suppose

$$\hat{\underline{\mathbf{x}}}^{*\,0:N_\mathrm{p}} = [\hat{\mathbf{x}}_1^{*\,0:N_\mathrm{p}}, \ldots, \hat{\mathbf{x}}_M^{*\,0:N_\mathrm{p}}] \tag{10}$$

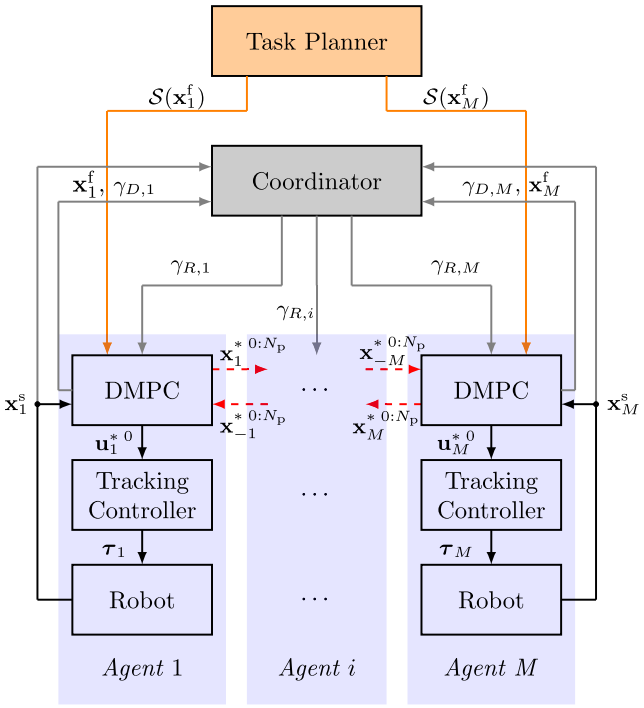



**FIGURE 2.** Control structure of collision-free online motion control for multiple robotic manipulators.

denotes the manipulators' optimal trajectories from the last converged DMPC-step. We obtain $\underline{\mathbf{x}}^{*\,0:N_\mathrm{p}}$ (and thus also $\underline{\mathbf{x}}_{-i}^{*\,0:N_\mathrm{p}}$) by shifting $\hat{\underline{\mathbf{x}}}^{*\,0:N_\mathrm{p}}$ by one time step and extrapolating the last state. In other words, for every manipulator $i = 1, \ldots, M$, we compute

$$\mathbf{x}_i^{*\,0:N_\mathrm{p}} = [\hat{\mathbf{x}}_i^{*\,1:N_\mathrm{p}}, \quad \mathbf{x}_i^{*N_\mathrm{p}}]. \tag{11}$$

where the last predicted optimal state $\mathbf{x}_i^{*N_\mathrm{p}}$ is obtained by the extrapolation of $\hat{\mathbf{x}}_i^{*\,N_\mathrm{p}}$ using the discrete system dynamics, i.e.,

$$\mathbf{x}_i^{*N_\mathrm{p}} = \mathbf{A}_i^{\mathrm{d}} \hat{\mathbf{x}}_i^{*N_\mathrm{p}} + \mathbf{B}_i^{\mathrm{d}} \mathbf{u}_i^{*N_\mathrm{p}-1}. \tag{12}$$

Note, that the equation (12) can be obtained by setting $\mathbf{u}_i^{*N_\mathrm{p}-1} = \hat{\mathbf{u}}_i^{*N_\mathrm{p}-1}$, i.e. the two last optimal inputs in the sequence $\mathbf{u}_i^{*\,0:N_\mathrm{p}-1}$ are assumed equal.

To sum up, the model predictive controller of each robot receives its current joint state and the (extrapolated) predicted joint states of neighbored robots $\underline{\mathbf{x}}_{-i}^{*\,0:N_\mathrm{p}}$ to account for collisions in the future and choose a proper control strategy to avoid them.

### B. CONTROL STRUCTURE

We propose the following control structure of our approach, illustrated in Fig. 2. First, a task planner assigns a sequence of tasks $\mathcal{S}(\mathbf{x}_i^{\mathrm{f}})$ to the robots. In general, collision avoidance alone is not sufficient to prevent deadlocks. A deadlock occurs when robots prevent each other from reaching their respective target by reaching a local minimum. Therefore, a supervisory

instance is required to coordinate the robots to resolve deadlocks. The supervisory role is taken over by a coordinator to resolve deadlocks once they are locally detected by the manipulators. Therefore, communication between the robots and the coordinator is necessary, which is indicated by gray lines in Figure 2. The coordinator receives a deadlock status denoted as $\gamma_{D,i}$ from each agent whether it is currently in a deadlock. In addition, the coordinator receives the current states $\mathbf{x}_i^{\mathrm{s}}$ and current target poses $\mathbf{x}_i^{\mathrm{f}}$ of the robots. This information is necessary to reliably detect and resolve deadlocks. If manipulators report a deadlock, the coordinator resolves it by sending an activation or deactivation status to each agent denoted as $\gamma_{R,i}$. Based on the clustering of robots into deadlock-affected and non-deadlock-affected group of robots, the manipulator closest to its target is allowed to proceed its movement, whereas the other ones receive a new target towards a neutral pose. Note, that the coordinator has no knowledge of the system dynamics of any manipulator. It relies solely upon the information shared by the robots. The DMPCs of the $M$ agents solve the problem in parallel by accounting for predicted state sequences $\underline{\mathbf{x}}_{-i}^{*\,0:N_{\mathrm{p}}}$ of the last converged DMPC step of the neighbored robots. The optimal control inputs $\mathbf{u}_i^{*\,0} = \text{const.}, i = 1, \ldots, M$ for $[t_0,\ t_1)$ are sent to the robots' underlying tracking controllers. The robots' controllers generate joint actuator torques $\boldsymbol{\tau}_i(t)$ which are applied to each robots' joints. Subsequently, the current state of a robot $\mathbf{x}_i^{\mathrm{s}}$, $i = 1, \ldots, M$ is measured and sent to the DMPC. At the same time, the obtained optimal state sequence $\mathbf{x}_i^{*\,0:N_{\mathrm{p}}}$, $i = 1, \ldots, M$ is communicated to the neighbored robots, indicated by red dashed lines in Figure 2.

## VI. COLLISION AVOIDANCE METHOD FOR MULTIPLE ROBOTIC MANIPULATORS

In the previous section, we formulated the motion control problem of $M$ robotic manipulators as $M$ coupled DMPCs. This section is dedicated to the efficient implementation of the static and dynamic collision constraints (4e) and (4f).

One of the most applied algorithms among collision avoidance methods in the literature is the Lumelsky algorithm [32]. The method approximates the robot links with line segments and introduces an algorithm to compute the minimum distance between them. The approximation of robot links as line segments is also known as line-swept sphere [22], [30], [31]. The nested logical conditions pose challenges to solving an optimization problem due to the non-smooth property of the function. Therefore, we introduce a novel approach for collision avoidance by approximating a robot's geometry by line segments and ellipsoids and derive an efficient and smooth formulation that enables the robots to safely avoid collisions.

### A. ELLIPSOID - LINE SEGMENT (ELS) APPROACH

To overcome the problem with nested logical conditions, we do not use a distance function to compute the distance between two links. Instead, we ensure at each optimization step that there is no intersection between line segments and ellipsoids formulated as hard constraints in (4f). In the following, we proceed from the perspective of a manipulator $i$ with a set of interior points in the task space denoted by $R(\mathbf{x}_i^k)$. The sets of interior points of the other robotic manipulators is designated by $\mathcal{R}(\underline{\mathbf{x}}_{-i}^{*\,k})$. We approximate the links of a manipulator $i$, for which the optimization problem is solved, by line segments and the links of the remaining manipulators by ellipsoids, we abbreviate it as ELS method. See Fig. 3 for an illustration, where the robot on the left side is approximated by 5 ellipsoids while the robot on the right is approximated by 8 line segments. By choosing proper dimensions of the ellipsoids with a suitable safety margin and thus ensuring that the lines and ellipsoids do not intersect, we assure that the constraints (4f) hold. Please note, that for every time step $k = 0, \ldots, N_{\mathrm{p}}$ pairs of ellipsoids and line segments of all involved robotic manipulators must be considered.

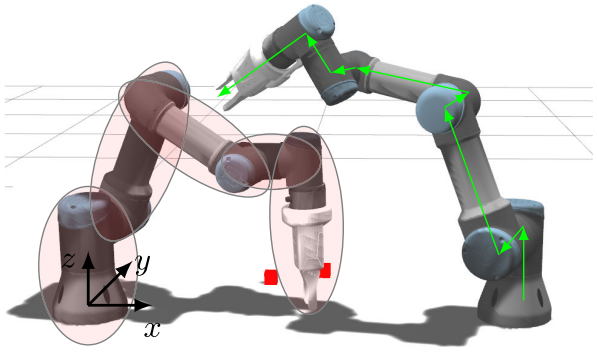



**FIGURE 3. Illustrative approximation of robots' geometry with ellipsoids and line segments from the perspective of robot *i* on the right side.**

In the following we consider collision avoidance between a robot $i$ and a robot $j$, where robot $i$ is modeled with line segments and robot $j$ with ellipsoids. A line segment $\mathbf{s}_m$ of a link $m$ is described by the equation

$$\mathbf{s}_m(\mathbf{x}_i^k) := \mathbf{b}_m(\mathbf{x}_i^k) + \alpha_m \mathbf{r}_m(\mathbf{x}_i^k), \quad \alpha_m \in [0, 1], \tag{13}$$

where vector $\mathbf{b}_m(\mathbf{x}_i^k) \in \mathbb{R}^3$ is the position vector of the basis of the considered link and vector $\mathbf{r}_m(\mathbf{x}_i^k) \in \mathbb{R}^3$ designates 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 $\alpha_m$ restricts the line segment to the length of the considered link.

For a given state $\mathbf{x}_j^k \in \mathbb{R}^{2N}$, the ellipsoid $\{\mathbf{e} \in \mathbb{R}^3 \mid H_n(\mathbf{e}, \mathbf{x}_j^k) = 1\}$ of a link $n$ is parameterized by the following quadratic equation

$$H_n(\mathbf{e}, \mathbf{x}_j^k) := (\mathbf{e} - \mathbf{e}_{0,n}(\mathbf{x}_j^k))^{\mathrm{T}} \mathbf{R}_n(\mathbf{x}_j^k) \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}}(\mathbf{x}_j^k)(\mathbf{e} - \mathbf{e}_{0,n}(\mathbf{x}_j^k)), \tag{14}$$

where $\mathbf{e} \in \mathbb{R}^3$ is a point on the ellipsoid. The centre point of the ellipsoid is denoted as

$$\mathbf{e}_{0,n}(\mathbf{x}_j^k) = \frac{1}{2}(\mathbf{b}_{n+1}(\mathbf{x}_j^k) + \mathbf{b}_n(\mathbf{x}_j^k)), \tag{15}$$

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 and the diagonal matrix

$$\mathbf{E}_n = \mathrm{diag}\left(\frac{1}{l_1^2}, \frac{1}{l_2^2}, \frac{1}{l_3^2}\right) \in \mathbb{R}^{3\times 3} \tag{16}$$

contains the squared inverse principal semi-axes $l_1, l_2, l_3 \in \mathbb{R}_{>0}$. To ensure that (4f) holds, the width of an ellipsoid should be at least twice as large as the width of a robot link and an ellipsoid should also occupy the two joints connecting the link.

In order to ensure, that line segment $m$ and ellipsoid $n$ do not intersect, the condition

$$1 - H_n(\mathbf{s}_m(\mathbf{x}_i^k), \mathbf{x}_j^k) \leq 0, \quad \forall \alpha_m \in [0, 1] \tag{17}$$

has to hold. Alternatively, the former can be reformulated into an optimization problem, i.e., solving

$$\min_{\alpha_m} H_n(\mathbf{b}_m(\mathbf{x}_i^k) + \alpha_m \mathbf{r}_m(\mathbf{x}_i^k), \mathbf{x}_j^k) \tag{18}$$

$$s.t.\ 0 \leq \alpha_m \leq 1 \tag{18a}$$

for $\alpha_m^*$ holds. Please note, we drop further explicit reference to $\mathbf{x}_i^k$ and $\mathbf{x}_j^k$ for sake of readability. Problem (18) is solved in the following way. First, the solution $\hat{\alpha}_m \in [-\infty, \infty]$ of the unconstrained optimization problem computes to

$$\hat{\alpha}_m = -\frac{(\mathbf{b}_m - \mathbf{e}_{0,n})^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m}{\mathbf{r}_m^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m}. \tag{19}$$

The former is guaranteed to exist since $H_n(\mathbf{e})$ 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. Projecting $\hat{\alpha}_m$ onto the unit interval by the projection operator $P : (-\infty, \infty) \to [0, 1]$ gives rise to the solution $\alpha_m^*$ of (18) in closed form

$$\alpha_m^* = P\left(-\frac{(\mathbf{b}_m - \mathbf{e}_{0,n})^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m}{\mathbf{r}_m^{\mathrm{T}} \mathbf{R}_n \mathbf{E}_n \mathbf{R}_n^{\mathrm{T}} \mathbf{r}_m}\right). \tag{20}$$

Since $P$ is not continuously differentiable, we approximate $P$ by

$$\hat{P}(\alpha) = \alpha\ \Phi(\alpha) - (\alpha - 1)\ \Phi(\alpha - 1) \tag{21}$$

where $\Phi$ refers to the smooth approximation of the Heaviside function

$$\Phi(\alpha) = \frac{1}{1 + \exp(-c\alpha)} \tag{22}$$

and $c \in \mathbb{R}_{>0}$ is a scaling parameter. For $c \to \infty$ the function $\hat{P}$ converges towards $P$. Both, $\hat{P}$ and $P$ are depicted in Fig. 4 for $c = 20$. For the former parameter choice, the maximum absolute error of $\alpha_m^*$ amounts to $1.13 \cdot 10^{-2}$.

Considering static collision avoidance, formulated in equation (4e), we follow a similar approach as explained before by approximating objects with convex bodies, i.e., by spheres or ellipsoids depending on the geometry of the considered object. In our setup, the table represents a static object, so that there is a risk that the robot $i$ chooses a trajectory below or through the table to avoid another robot $j$. For this purpose it is sufficient to formulate a plane along the table and restrict

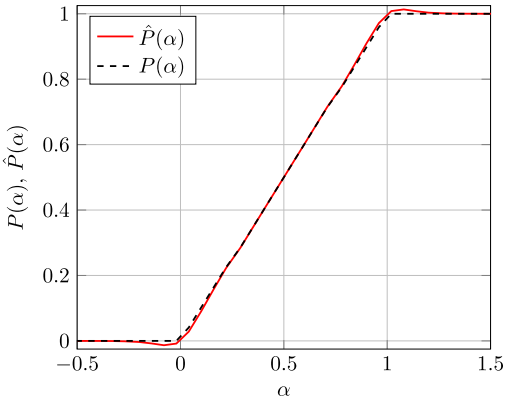



**FIGURE 4.** A comparison between the projection operator $P(\alpha)$ and the approximated function $\hat{P}(\alpha)$ with parameter $c = 20$.

the intersection of the basis of each link $\mathbf{b}_m$ of the robot $i$ with the plane by the height of the table denoted as vector $\mathbf{z} = [0\ 0\ z_{\mathrm{T}}]$, i.e.

$$\mathbf{b}_m \geq \mathbf{z} + \mathbf{z}_{\min}, \tag{23}$$

with an offset $\mathbf{z}_{\min}$. In case of the gripper, which is attached to the end effector, an additional offset equal to the length of the gripper should be considered.

### B. INTER-ROBOT COLLISION AVOIDANCE WITH ELS METHOD FOR TWO ROBOTS

In order to ensure a collision-free trajectory of a robot in a multi-robot setting, it is necessary to encompass the whole geometry of a robot, as described in the previous section VI-A. The number of collision avoidance constraints is a matter of the geometric composition of the robots and therefore setup-dependent and might be determined in a pre-processing step. In case of 2 manipulators with $N = 6$ degrees of freedom, we approximate the robot $i$ for which the DMPC problem is solved by $N_{\mathrm{L}} = 8$ line segments, starting from the basis and ending by the gripper, depicted in Fig. 3. We approximate the neighbored robot by $N_{\mathrm{E}} = 5$ ellipsoids, encapsulating the basis, subsequent three links (Shoulder, Elbow, Wrist 2) and the end-effector including the gripper, which is referred to as Wrist 3. By ensuring no intersections between any line segment with any ellipsoid, this results in formulating $N_{\mathrm{dyn}} = N_{\mathrm{L}} \cdot N_{\mathrm{E}} = 40$ collision constraints for every time step of the prediction horizon. As the ellipsoids should be chosen large enough to contain at least the diameter of the neighbored link, it is sufficient to omit the 3 short line segments connecting two joints, i.e., line segment connecting Basis and Shoulder joints, Shoulder and Elbow joints as well as Wrist 1 and Wrist 2 joints. This results in formulating $N_{\mathrm{dyn}} = 25$ collision constraints for a single time step.

However, the former only serves as an upper bound. Geometrically it is impossible to position robots in a pick & place setup, where all constraints need to be considered, as the manipulators operate in a certain distance to each other. Hence, a pre-processing step can be performed to reduce the

number of required collision constraints, depending on the considered setup. In the work at hand, each manipulator (may it be the two, three or four robot setup) is positioned on top of a flexible module which might be combined arbitrarily with other modules to form larger formations. The minimum distance between the robots dictated by the modules ensures, for example, that one robot cannot touch the base of the other robot. Furthermore, the shoulders of both robots are also not able to collide. Thus, the formulation can be reduced to $N_{\text{dyn}} = 14$ collision avoidance constraints for each time step, which are sufficient for a safe interaction between two robots with $N = 6$ degrees of freedom. Those are listed in Table 1. For instance, choosing a prediction horizon length of $N_{\text{p}} = 20$ results in a total of 280 constraints to be considered for each manipulator by the DMPC. In case that the robots are placed very close to each other, similar assumptions can be made, where certain constraints can be omitted as well.

**TABLE 1. Intersection of links for formulating collision avoidance constraints in case of two robots *j* and *i*.**

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

## VII. DETECTING AND RESOLVING DEADLOCKS FOR ROBOTIC MANIPULATORS

Deadlocks occurring in a setup of multiple robots is a well-known problem in the field of mobile robots, UAVs and robotic manipulators [44], [45]. The problem may arise if one or more robotic manipulators block each other, effectively prohibiting each other from reaching their target state. In our case, the solution of the optimization problem (4) is at a local minimum. To this end, resolving deadlocks requires a supervisory instance, i.e., a coordinator, and some sort of information exchange between the robotic manipulators and the coordinator.

We propose an approach involving a local deadlock detection, where each robotic manipulator $i$ checks by itself if it is currently in a deadlock and sends the information to the supervisory instance, i.e., the coordinator. The coordinator, as previously introduced in Section V-B, resolves an occurring deadlock. A manipulator $i$ detects a deadlock if the following two conditions are true, i.e., if the change of joint velocities over the prediction horizon is sufficiently small, meaning that manipulator $i$ is slowed down and cannot move further and, at the same time, the deviation of the robot's measured state $\mathbf{x}_i^{\text{s}}$ and the desired state $\mathbf{x}_i^{\text{f}}$ is sufficiently large

$$\left\| \dot{\mathbf{q}}_i^{*\,N_{\text{p}}} - \dot{\mathbf{q}}_i^{*\,0} \right\|_2 \leq \varepsilon_v \wedge \left\| \mathbf{x}_i^{\text{s}} - \mathbf{x}_i^{\text{f}} \right\|_2 \geq \delta_x \Rightarrow \gamma_{D,i} = 1. \quad (24)$$

If both conditions are satisfied, a deadlock is detected and deadlock parameter $\gamma_{D,i} \in \{0, 1\}$ is set to $\gamma_{D,i} = 1$ and subsequently sent to the coordinator. In case of no deadlocks, the deadlock parameter takes a value of zero. Besides, each manipulator $i$ sends its current and target states to the coordinator at each time step.

To resolve a deadlock, the coordinator classifies robots in deadlock-affected and non-deadlock-affected groups by their mutual distances. First, all robots $R_i$, $i = 1, \ldots, M$ are part of $M$ disjoint clusters $\mathcal{C}_i$, i.e., $\mathcal{C}_i = \{R_i\}$. The coordinator checks for every robot if a deadlock has been detected. In case of $\gamma_{D,i} = 1$, the coordinator moves all robots $R_j$ ($j \neq i$) which are sufficiently close to robot $R_i$ to cluster $\mathcal{C}_i$ (using forward kinematics and the provided states). The following pseudo-code Algorithm 1 summarizes these steps.

**Algorithm 1** Classification and Clustering

**for** $i = 1$ **to** $M$ **do**
  **if** $\gamma_{D,i} = 1$ **then**
    Check which robots are sufficiently close
    **for** $j = 1$ **to** $M$ **do**
      **if** $i \neq j$ & $\text{dist}(R_i, R_j) \leq d_{\min}$ **then**
        Add robot $R_j$ to the cluster $\mathcal{C}_i$
      **else**
        Robot $R_j$ remains in its own cluster
      **end if**
    **end for**
  **end if**
**end for**

In the next step, the coordinator sends, for all deadlock-affected clusters, the resolving parameter $\gamma_{\text{R},i} = 0$ to all robots in this cluster except for the robot with the smallest residual. A resolving parameter of $\gamma_{\text{R},i} = 0$ assigns the robots a new target pose $\mathbf{x}^{\text{D}}$, designated as neutral pose. The robot with the smallest residual, i.e., which is closest to its target state retains its original target state. Once this robot reaches its target state, the cluster is reset by sending all involved robots to their respective cluster $\mathcal{C}_i = \{R_i\}$ and setting their resolving parameters to $\gamma_{\text{R},i} = 1$, i.e., robots that were assigned a neutral pose are given their former targets in order to be able to finish their previous tasks. The steps are summarized as pseudo-code in Algorithm 2.

**Algorithm 2** Resolving Deadlocks

**for** $i = 1$ **to** $M$ **do**
  $k_{\min} = \arg\min_{k \in \mathcal{C}_i} \text{res}(k)$
  Send $\gamma_R = 1$ to robot $k_{\min}$
  Send $\gamma_R = 0$ to all robots in $\mathcal{C}_i \setminus k_{\min}$
  **if** $\text{res}(k_{\min}) < \varepsilon_{\text{res}}$ **then**
    Send $\gamma_R = 1$ to all robots in $\mathcal{C}_i$
    Redistribute all robots to their original clusters
  **end if**
**end for**

This clustering procedure ensures that manipulators that are not affected by a deadlock operate undisturbed in the workspace.

## VIII. RESULTS

### A. SIMULATION SETUP AND CONTROLLER PARAMETRIZATION

The multi-robot setup is built in the robotic simulation environment *Gazebo* [7] with simulated collaborative robotic manipulators UR3 from *Universal Robots* with $N = 6$ degrees of freedom each. *Gazebo* provides an interface to control the robots using ROS. In this work, we use the distribution *ROS Noetic*. The communication is established through the ROS action client to the *Universal Robot* ROS driver. Therefore, a velocity controller hardware interface is applied. The ROS interface allows for an easy replacement of the simulation environment in *Gazebo* by an experimental test bed.

The control algorithms are implemented in *Matlab* using *CasADi* [46] for setting up the nonlinear program for the DMPCs. The merit of *CasADi* is its automatic differentiation capability, i.e., *CasADi* computes the first and second order derivatives of the cost function and constraints using automatic differentiation. We use the interior point solver *IPOPT* [47] to solve the DMPC problems and apply *MA27* [48] to solve the underlying linear system. We set the maximum number of iterations to 1000. In addition, *CasADi* is instructed to pre-compile the optimization problems using just-in-time compilation. We choose a sampling time of $T_\mathrm{s} = 200$ ms.

The model predictive controllers run in parallel on a computer with an Intel i7-11800H CPU at 2.30 GHz using 32 GB RAM under Ubuntu 20.04. The coordinator is running on the same computer and communicates with the model predictive controllers via the UDP protocol. The trajectories between the robots are exchanged via the UDP protocol as well.

The weighting matrices of the DMPCs are chosen as

$$\mathbf{Q}_i^\mathrm{x} = \mathrm{diag}(1, 1, 1, 0.2, 0.2, 1, 1, 1, 1, 0.1, 0.1, 0.1),$$
$$\mathbf{Q}_i^\mathrm{f} = 10 \cdot \mathbf{Q}_i^\mathrm{x}, \quad \mathbf{R}_i^\mathrm{u} = \mathbf{I}^{6\times 6} \quad \text{and} \quad \mathbf{R}_i^\mathrm{d} = \mathbf{I}^{6\times 6}.$$

Joint positions and velocities of the three wrist links (connecting the last three joints) are penalized less to allow for a greater freedom of motion. The absolute values of joint velocities of an UR3 manipulator are limited to

$$[\pi, \pi, \pi, 2\pi, 2\pi, 2\pi]\,\frac{\mathrm{rad}}{\mathrm{s}}.$$

In addition, the absolute values of accelerations are limited to

$$[\pi, \pi, \pi, 2\pi, 2\pi, 2\pi]\,\frac{\mathrm{rad}}{\mathrm{s}^2}.$$

The parameters for the deadlock algorithm were chosen as follows

$$\varepsilon_v = 1.5 \cdot 10^{-3}\,\frac{\mathrm{rad}}{\mathrm{s}}, \quad \delta_x = 1.2 \cdot 10^{-2}\ \mathrm{rad}, \ d_\mathrm{min} = 0.2\ \mathrm{m}.$$

### B. INPUT DELAYS OF CLOSED-LOOP NONLINEAR MODEL PREDICTIVE CONTROL

The number of active dynamic collision constraints summarized in equation (4f) change dynamically, as not all possible collisions can occur at each time step $t_k$. The number of active collision constraints considerably affects the computation times of the non-convex optimization problem (4). However, the sampling time $T_\mathrm{s}$ cannot be increased arbitrarily, otherwise tunneling will occur resulting in undetected collisions. For this reason, we choose a sufficiently small sampling time and account for the non-negligible computation times as explained by Grüne and Pannek [49].

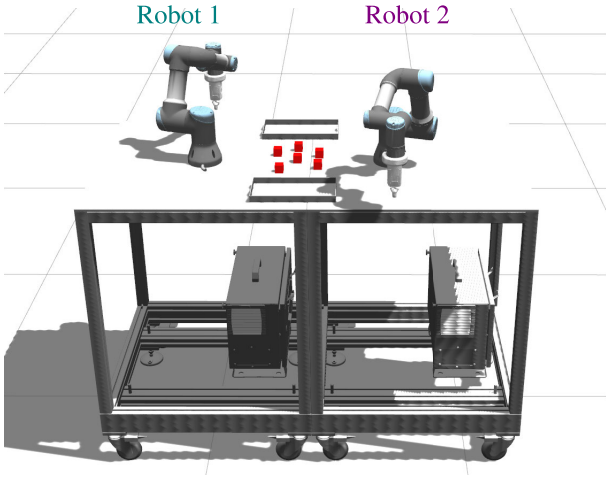



**FIGURE 5.** Simulation setup with 2 modules of UR3 robots.

### C. VALIDATION OF THE MOTION PLAN ALGORITHM

In this section, we demonstrate the flexibility of our approach by arranging the robot modules into different constellations of two, three, and four robots to study the optimal trajectories and computation times for pick and place tasks in more detail. Each manipulator is placed on top of a module of a height $z_\mathrm{T} = 1.107$ m. The task for each robot $i$ is to grasp a number of objects in a common workspace and to place them into the assigned trays. Each tray is shared by two manipulators.

In our first setup, we consider two robots, two trays and six objects as shown in Fig. 5. We consider 20 pick and place sample tasks with different object positions in the shared workspace. For each sample task the six objects are randomly placed in a common workspace via random sequential adsorption (RSA) [50] by considering additional reachability constraints of the robots. The randomly distributed objects in the common workspace can be reached by both robots. Similarly, each tray can be served by both robots and contains three slots. The robots are placed close to each other, so that inter-robot collisions are imminent. In first step, the tasks are equally distributed among the two robots by providing a sequence of setpoints to each manipulator. Each robot's task is to place three of the randomly distributed objects into assigned trays. To analyze the influence of the prediction horizon length $N_\mathrm{p}$ on performance and computation times, we choose three different prediction horizon lengths $N_\mathrm{p} \in \{10, 15, 20\}$.

For illustration purposes, several time frames are depicted in Fig. 6 for one selected sample (sample task 1) with a

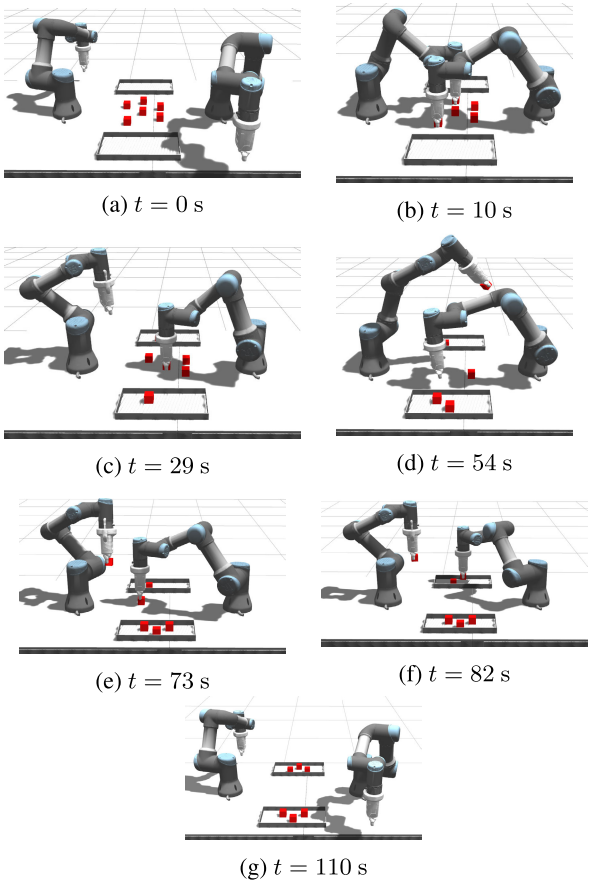



**FIGURE 6.** Selected time frames from Gazebo simulation for 2 robots for a selected sample (sample task 1).

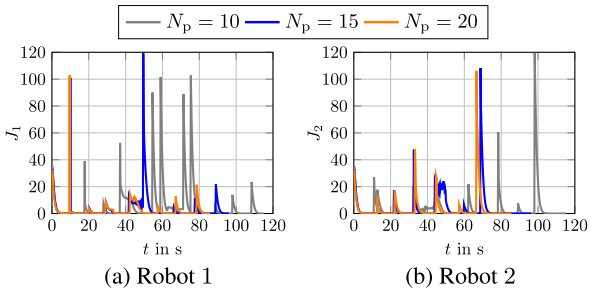



**FIGURE 7.** Cost function (4) dependence on prediction horizon $N_p$ for sample task 1.

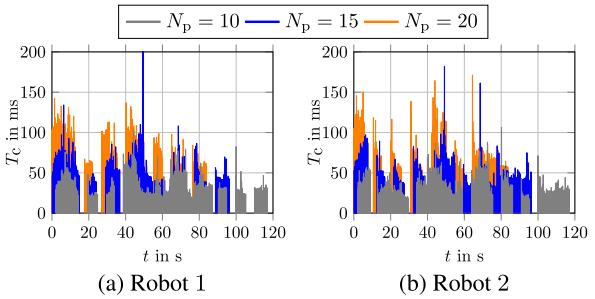



**FIGURE 8.** Comparison of computation times $T_c$ to prediction horizon $N_p$ for sample task 1.

prediction horizon length of $N_p = 20$. The initial poses of the robots are depicted in Fig. 6a. The first two objects are grasped by two robots without any problems, see Fig. 6b. In contrast, grasping the next two objects leads to a deadlock as the robots' trajectories intersect. Therefore, at time $t = 29$ s in Fig. 6c, a deadlock is resolved, where the robot on the left has been sent to a neutral pose while allowing the robot on the right to grasp its object. At time $t = 54$ s in Fig. 6d the robot on the left successfully plans an optimal and collision-free motion above its neighbor to reach its target. Later, at time $t = 73$ s in Fig. 6e another deadlock occurs, so that the robot on the left moves to its neutral pose allowing the robot on the right to grasp its object. Both robots have to serve the same tray and therefore cannot place their objects simultaneously. Therefore, the robot on the left moves to its neutral pose once again at time $t = 82$ s until the robot on the right finishes its task, shown in Fig. 6f. After finishing the assigned tasks, which took 100 s, the robots return to their initial pose, see Fig. 6g. This example demonstrates a frequent occurrence of deadlocks, where collision avoidance alone cannot solve the motion control problems.

The cost functions for both robots are provided in Fig. 7. As the cost function punishes the deviation of the current state to the desired state, it rises every time a robot receives a new target pose. Furthermore, it can be observed that the execution time, i.e., the time needed to finish all pick and place tasks, reduces with increasing prediction horizon length. For this reason, a prediction horizon length as large as possible is desired that results in faster reactions to upcoming collisions and therefore sooner actions can be taken to avoid them.

Concerning the optimality of the DMPC, we compare the joint angles with the results obtained by CMPC for different prediction horizon lengths. In Fig. 9 all joint angles are depicted for sample task 1, solved in the distributed and centralized scheme. Please note that the CMPC for $N_p = 20$ is not applicable as the computation times consistently exceed the sampling time $T_s$. Thus, we restrict to prediction horizon lengths of $N_p = 10$ and $N_p = 15$. It can be observed, that the difference between the two solutions of CMPC and the DMPC increase with time for $N_p = 10$ for both joint angles. For $N_p = 15$ the results indicate that with increasing prediction horizon length, the solution of DMPC closely follows the solution of CMPC. The corresponding computed control inputs for each joint are provided in Fig. 10.

The computation times for every time step for sample task 1 are depicted in Fig. 8. As expected, the computation time increases with an increasing prediction horizon length $N_p$. Grasping as well as placing procedures are performed without solving the DMPC, as once the robot reaches its target it moves down or up within a short time interval. The gaps in the computation times correspond to grasping and placing procedures. To get a better impression of computation times of the DMPC and the CMPC for all 20 sample tasks, the mean computation times as well as the standard deviations

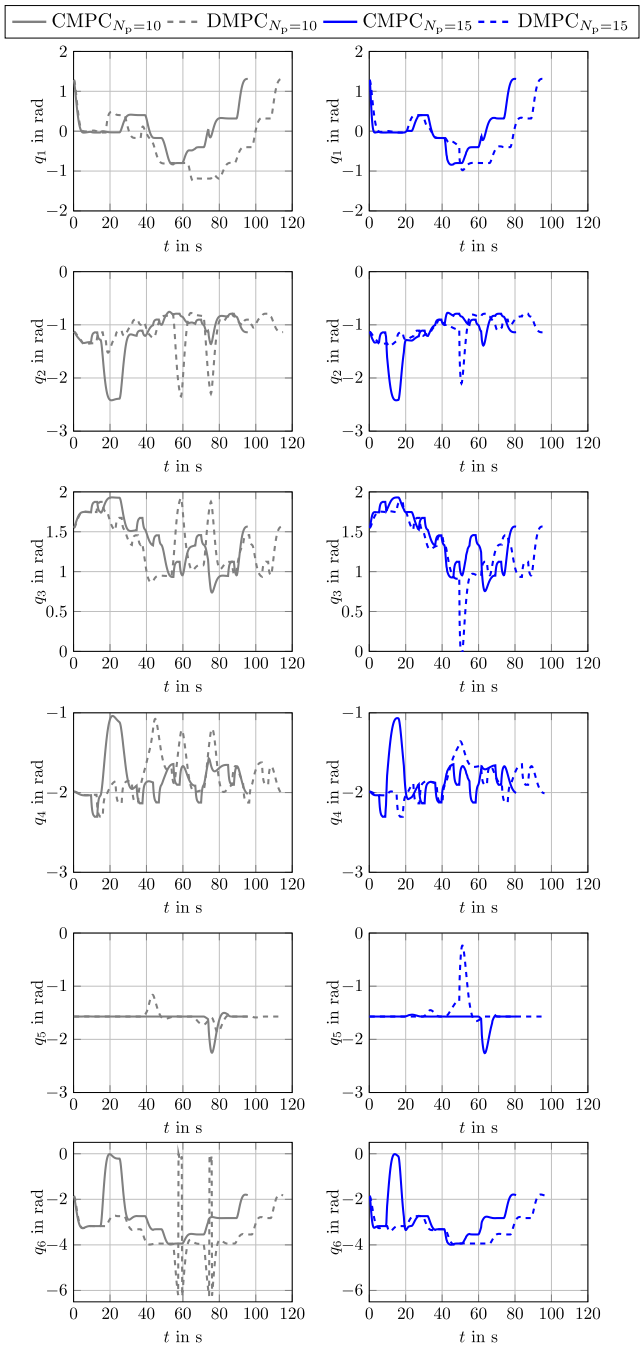
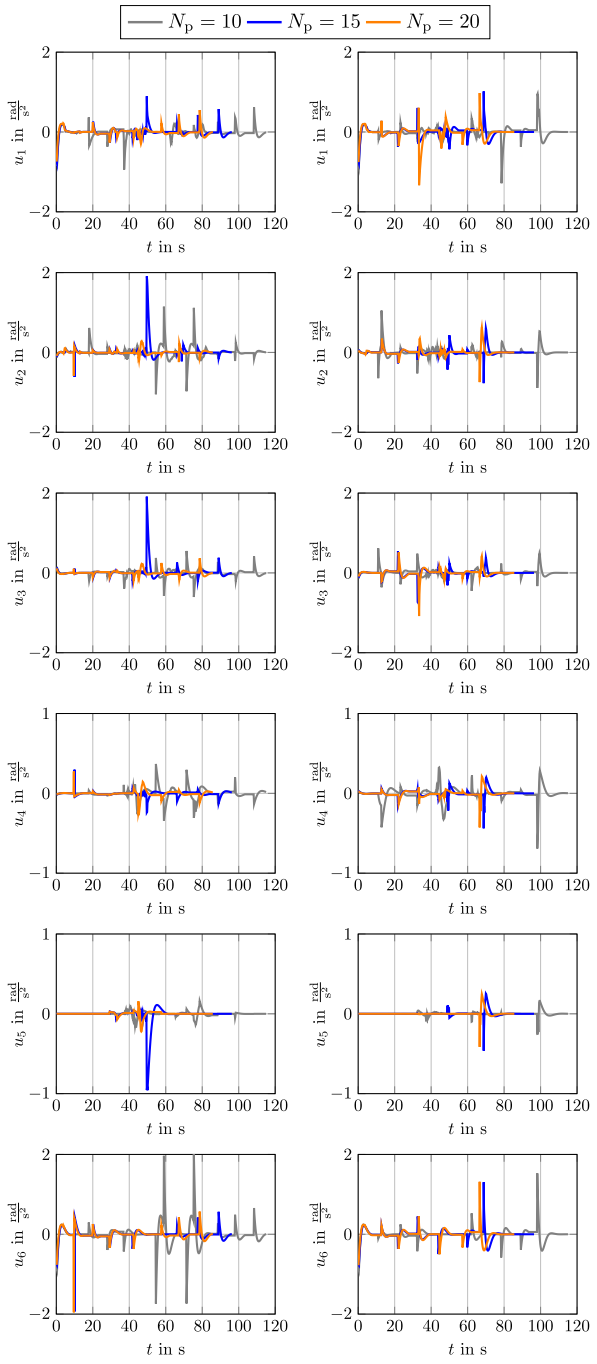



**FIGURE 9.** Comparing solutions from distributed and centralized MPC for all joint angles with different prediction horizon lengths for Robot 1 (sample task 1).




(a) Robot 1 (b) Robot 2

**FIGURE 10.** Computed control inputs of all joint angles from DMPC for different prediction horizon lengths for Robot 1 and Robot 2 (sample task 1).

are summarized in Table 2. The left column corresponds to the robot on the left in the simulation environment and the right column corresponds to robot on the right, see Fig. 6. In case of the DMPC the mean computation times increase superlinearly with an increasing prediction horizon length which might be attributed to the direct solver used by *IPOPT*. Importantly, the standard deviation increases in the same fashion as well. Compared to the computation times obtained by the CMPC, a speed-up factor of more than 2.5 has been achieved by solving the problem in a distributed scheme. As mentioned earlier, the CMPC for $N_p = 20$ computation times are omitted.

### D. BENCHMARK PROBLEMS WITH OMPL PLANNERS AND CHOMP PLANNER

We compare our approach with several sampling-based methods integrated in OMPL, such as RRT-Connect, PRM, PRM*

**TABLE 2. Benchmark results for 20 sample tasks in a setup with 2 robots.**

| Planner | $N_p$ | mean($T_c$)± std($T_c$) in ms | $T_e$± std($T_e$) in s | L± std(L) in m | s± std(s) in $\frac{rad}{s}$ | $S_r$ |
|---|---|---|---|---|---|---|
| DMPC | 10 | $32.88 \pm 14.34$ $31.84 \pm 18.24$ | $122.36 \pm 21.61$ | $5.02 \pm 1.35$ $5.80 \pm 1.58$ | $27.40 \pm 7.24$ $28.33 \pm 6.57$ | 95% |
| | 15 | $57.47 \pm 24.41$ $59.89 \pm 26.53$ | $113.55 \pm 24.35$ | $5.03 \pm 1.25$ $5.41 \pm 1.47$ | $25.62 \pm 6.67$ $26.20 \pm 8.27$ | 95% |
| | 20 | $75.99 \pm 34.76$ $78.66 \pm 34.21$ | $108.04 \pm 14.41$ | $5.01 \pm 1.01$ $5.29 \pm 0.85$ | $25.67 \pm 6.18$ $24.09 \pm 2.84$ | 95% |
| CMPC | 10 | $85.82 \pm 30.63$ | $110.57 \pm 16.57$ | $4.83 \pm 1.14$ $5.23 \pm 1.06$ | $26.79 \pm 6.58$ $26.81 \pm 3.25$ | - |
| | 15 | $125.15 \pm 53.54$ | $96.87 \pm 11.13$ | $4.86 \pm 1.06$ $4.90 \pm 1.01$ | $27.72 \pm 6.40$ $24.69 \pm 4.12$ | - |
| CHOMP | - | $4643 \pm 1129.90$ | $266.66 \pm 56.30$ | $4.93 \pm 1.04$ $6.71 \pm 1.97$ | $48.80 \pm 26.48$ $49.78 \pm 19.64$ | 43% |
| PRM | - | $299.50 \pm 357.34$ | $70.47 \pm 13.10$ | $7.30 \pm 2.46$ $8.10 \pm 2.90$ | $45.04 \pm 11.10$ $48.74 \pm 9.58$ | 75% |
| PRM* | - | $10066 \pm 29.81$ | $308.59 \pm 56.61$ | $5.73 \pm 1.47$ $5.79 \pm 1.55$ | $38.72 \pm 7.92$ $43.98 \pm 6.95$ | 76% |
| RRTC | - | $53 \pm 15.93$ | $65.60 \pm 12.08$ | $6.19 \pm 1.98$ $7.43 \pm 2.76$ | $42.27 \pm 8.85$ $47.00 \pm 9.57$ | 72% |

and the optimization-based method CHOMP with regard to execution time, pathlength, planning (computation) time, smoothness and success rate for a setup of 2 robots. The execution time $T_e$ designates the time which is needed to finish the pick and place tasks by both robots. The path length $L$ describes the total distance traveled by the end effector of one robot to finish the assigned tasks. The time $T_c$ corresponds to the planning time to compute the trajectory from initial to the target pose for the investigated OMPL planners. For the MPC approach, time $T_c$ is referred to computation time which is needed to perform a single MPC step. The trajectory smoothness $s = \int_{T_e} ||\ddot{\mathbf{q}}(t)||\mathrm{d}t$ describes the Euclidean norm of the integral of joint accelerations. Success rate $S_r$ specifies percentage of how many sample tasks were successfully planned for 2 robots simultaneously by a planner. Table 2 summarizes the results of 20 sample tasks for the chosen planners, where the mean values and standard deviations for the selected metrics are provided. Note, RRT-Connect is abbreviated as RRTC in the Table 2. The results are also visualized in Fig. 11 to provide a comparison between the planners. The results for Robot 1 (left column in Table 2, respectively for each metric) are visualized in Fig. 11.

The sampling-based methods guarantee a probabilistic completeness, i.e., a planner is guaranteed to find a solution with a given probability in a finite amount of time. Planners such as PRM* need certain amount of time to plan a trajectory which is to be specified by the user. Not restricting the planning time results in a not terminating algorithm. The benchmark methods plan the trajectories for the two robots at the same time and send them to the robots executing the trajectories simultaneously. To ensure comparability between the benchmark methods and the DMPC regarding success rate, we integrate the same deadlock resolution procedure as for the DMPC, described in Section VII. In other words, once a planner fails to find a solution for both robots, e.g., simultaneous picking or placing is not feasible due to otherwise occurring collisions (geometrically infeasible poses), one robot is sent to its neutral pose, so that the other one can grasp or place an object. Comparing the success rate of the planners, CHOMP manages to find a path for 2 robots simultaneously in only 43% of the considered sample tasks. In other words, in 57% of sample tasks only one robot at the same time could operate in a shared workspace. Although the pathlength keeps up with the results of MPC, it should be noted that in half of the cases the path could not be planned, thus the robot could not travel as much as in the case of MPC. The execution time is the second longest compared to the other planners, followed by the second longest planning time. CHOMP plans jerky trajectories resulting in high standard deviations of trajectory smoothness meaning high fluctuation of joint velocities. In addition, we encountered several collisions with the CHOMP algorithm between the robots resulting in a poor performance. This can be explained by the fact that collision avoidance constraints are incorporated as soft constraints in the objective function and thus a collision-free trajectory cannot be guaranteed.

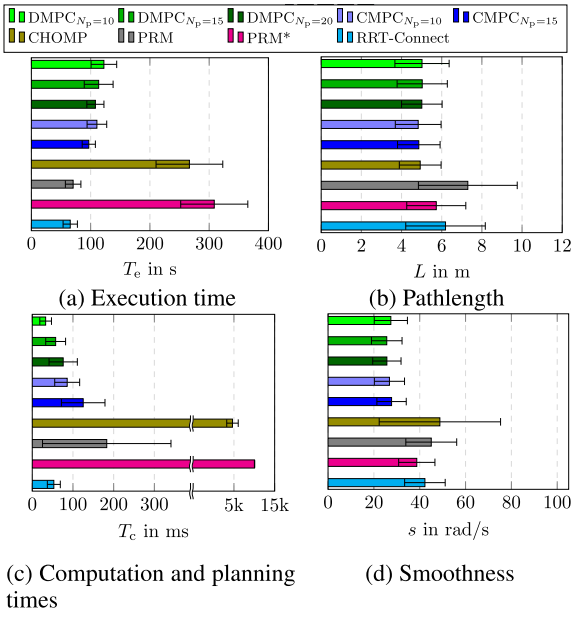



**FIGURE 11. Comparison of MPC approach with some selected OMPL planners for 20 sample tasks in a setup with 2 robots for Robot 1.**

PRM and RRT-Connect have a success rate of 75% and 72%, respectively. The execution times are the smallest one compared to CHOMP and RRT*. However, shorter planning times do not necessarily lead to shorter path legnth. As can be seen from Fig. 11b, the pathlengths for both planners are longer than for CHOMP and PRM*. In contrast to other planners, planning time of PRM shows a large standard deviation. PRM and RRT-Connect plan trajectories of similar smoothness.

The planner PRM* leads to the highest planning time as it fully utilizes the given (maximum) planning time of 10 s and then uses the best result up to that point. This results also in the highest execution time. However, it provides the highest success rate of 76%. As PRM* creates a dense graph, it leads to smooth paths compared to other non-optimal sampling-based planners. The pathlength of PRM* slightly exceeds the values of DMPC and thus belongs to the shortest one compared to the investigated OMPL planners and CHOMP.

From the obtained results, the execution time of DMPC and CMPC tend to decrease with increasing prediction horizon. It is considerably shorter compared to CHOMP and PRM*. The pathlength does not considerably change with increasing prediction horizon in case of DMPC and CMPC. However, the computation time increases superlinearly with increasing prediction horizon as more constraints have to be accounted for. The MPC schemes compute the smoothest trajectories compared to the considered OMPL planners and CHOMP. The success rate in case of DMPC describes the number of time steps, where both robots perform pick and place tasks simultaneously, i.e., the remaining time steps the robots were resolving a deadlock. The comparison to other planners show that the DMPC approach finds an optimal path in 95% for simultaneous pick and place tasks, independent of the prediction horizon length, and leads to better results for almost all metrics except for execution time, where RRT-Connect outperforms. In addition, the MPC approach has the benefit of reacting to dynamically changing environments. In other words, once the one robot's target pose changes the other robot's target is not influenced and it can still re-plan its own motion at any time. Therefore, MPC approach provides flexibility by not defining the changes beforehand, which is not the case with the planners studied here and thus are only limited to offline planning.

### E. SCALABILITY OF THE DMPC APPROACH

Subsequently, we study setups of three and four robot modules to investigate how the computation times scale with an increasing number of robots. The setup with three modules, each comprising a single robot, are set up in a row, where the outer robots cooperate with the robot in the middle, illustrated in Fig. 12. The robot in the middle is performing pick and place tasks in two different workspaces. For sake of brevity, two time steps from a selected sample are illustrated in Fig. 13, where deadlock and collision avoidance occurred. The pick and place tasks were successfully performed by the three robots for all sample tasks.

As the last setup, we consider four modules with four manipulators and 12 objects in the shared workspace, depicted in Fig. 14. In this case, not all objects are reachable by all robots and each tray can only be served by two robots. As before, we study several samples with randomly placed object. The robots perform the pick and place tasks for 12 objects into the four provided trays. In Fig. 15, two distinct time steps are illustrated showing the robots performing the pick and place tasks.

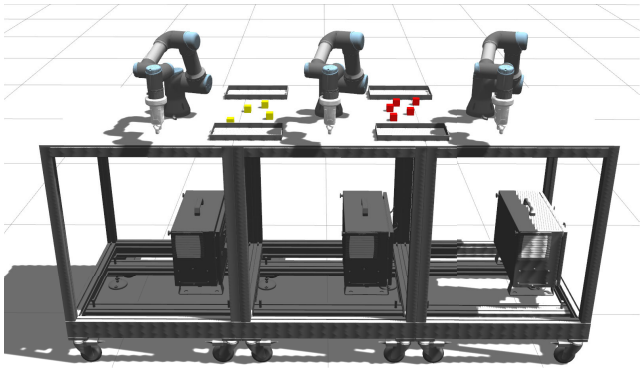
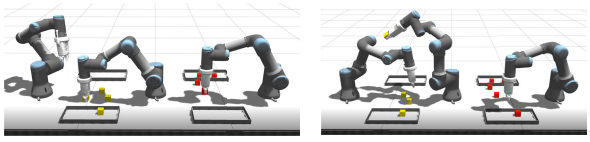


**FIGURE 12.** Simulation setup with 3 modules of UR3 robots.



(a) $t = 33$ s (b) $t = 42$ s

**FIGURE 13.** Selected time frames from Gazebo simulation for 3 robots.

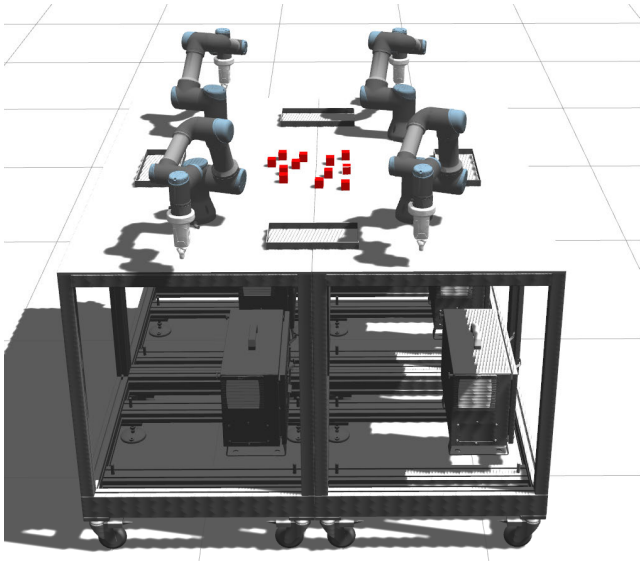
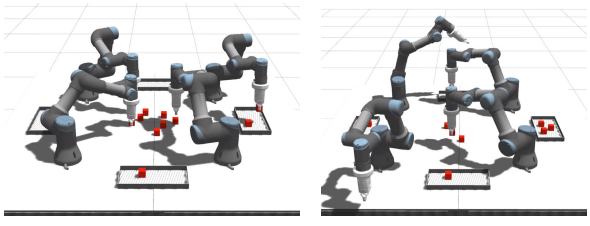


**FIGURE 14.** Simulation setup with 4 modules of UR3 robots.



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

**FIGURE 15.** Selected time frames from Gazebo simulation for 4 robots.

We compare mean computation times for the setups of two, three and four robots and their dependence on the prediction horizon length. From Fig. 16 it can be observed that

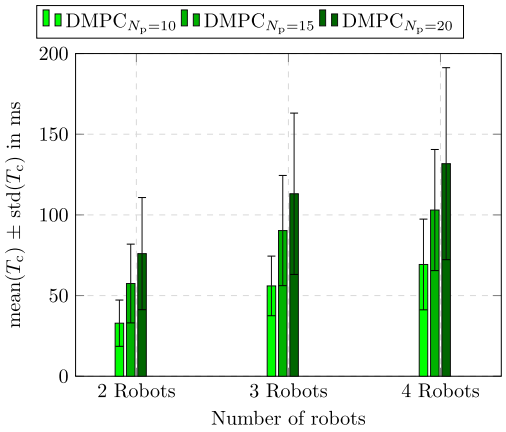



**FIGURE 16.** Scaling of computation times with the number of robots.

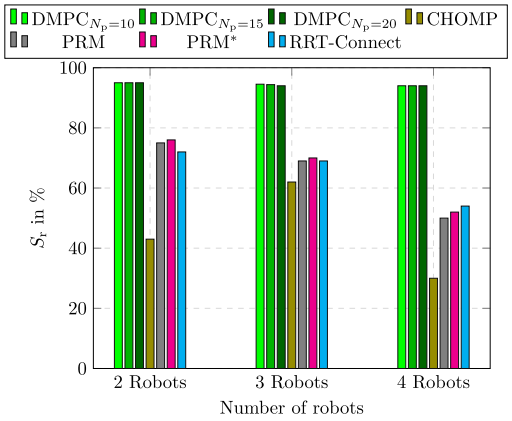



**FIGURE 17.** Scaling of DMPC and OMPL planners regarding success rate with the number of robots.

computation times rise with an increasing number of robots. The mean computation times for prediction horizon lengths $N_p \in \{10, 15\}$ do not exceed 100 ms for 2 and 3 robots. Moreover, the mean computation times remain under 200 ms even for 4 robots with $N_p = 20$. As previously shown for a setup of 2 robots, prediction horizon length of $N_p = 15$ is sufficient as the solution closely follows the solution of the CMPC.

Finally, we assess how the OMPL planners and CHOMP scale with number of robots with regard to success rate. The results in Fig. 17 show that CHOMP delivers better results for the linear setup of 3 robots, see Fig. 12 than for 2 robots which can be attributed to the fact that the robot in the middle interacts with the outer robots in an alternating manner. However, for the setup of 4 robots, where all the robots share a common workspace, the performance of CHOMP considerably deteriorates going from 3 to 4 robots. For the setup of 4 robots, CHOMP manages to find a solution for all four robots in one third of times. The success rates of the considered OMPL planners decrease almost linearly with the number of robots. In contrast, DMPC approach delivers a success rate of approximately 94 % regardless of the number of robots.

## IX. CONCLUSION

In this work, we introduced a novel motion control algorithm for multiple robotic manipulators. The manipulators were set up as modules which were combined into larger structures of 2, 3 and 4 robots. Each robot planned its own collision-free trajectory using Distributed Model Predictive Control (DMPC) that accounted for static and dynamic obstacles. The framework required a communication between the manipulators for safe interaction with each other. We proposed a new approach for collision avoidance between multiple robotic manipulators. Each robot was approximated by line segments, while the other surrounding robots were approximated by ellipsoids. Excluding intersections between line segments and ellipsoids ensures a collision-free trajectory of each robot. This formulation allowed for a computation of the optimal trajectories in real-time. In a setup of multiple robotic manipulators, deadlocks may occur, which is a well-known problem in robotics. In our approach each manipulator detects if it is currently in a deadlock. Based on this information, the introduced coordinator resolves occurring deadlocks without interrupting the motion of not deadlock-affected robots.

The motion control algorithm was validated on different constellations of robotic modules for two, three and four manipulators performing pick and place tasks. The setup was built in the simulation environment *Gazebo* and controlled by ROS. We observed that for cases where robots have to serve the same tray or have to pick objects very close to each other, deadlocks consistently occurred. However, in each case, the manipulators reliably detected deadlocks and the coordinator successfully resolved them. Concerning the optimality of our approach, we compared trajectories of the proposed DMPC approach with the centralized solution. The results showed, that with longer prediction horizon, the difference between the solutions decreases and the distributed solution closely follows the centralized one. Finally, a comparison of computation times with centralized MPC as a benchmark showed, that a considerable speed-up is achieved by solving the problem in a distributed scheme. Finally, we compared our approach with the well-known sampling-based (RRT-Connect, PRM, PRM*) and optimization-based planners (CHOMP). In simple static cases, the new DMPC approach shows better performance in most cases concerning execution time, pathlength and smoothness of the trajectories compared to the investigated OMPL planners and CHOMP. Moreover, with the DMPC approach, robots can perform pick and place tasks simultaneously in 94% of sample tasks regardless of the number of robots. The comparison with other planners showed, that the success rate to find a feasible path for each robot decreases with the number of robots, e.g. CHOMP succeeds to find a plan in only 30% of cases while the OMPL planners do not exceed a success rate of 54% in case of 4 robots. The main

advantage of the new DMPC approach is to bring in more functionality and flexibility for dynamic environments, i.e., each robot can freely move and change its target pose on the fly without requiring a replanning for all robots simultaneously. In other words, each robot is capable of reacting upon path changes of other robots at each time step, which is not the case with the investigated planners. Finally, we showed by comparing computation times obtained for different numbers of robots, that our framework scales to multiple robotic manipulators.

In future, we plan to realize the proposed approach on an experimental testbed for at least two robotic manipulators performing assembly and disassembly tasks. So far, we investigated our approach for a homogeneous group of robots. However, it would be also desirable to extend the approach to a non-homogeneous robotic setup. Our approach can further be extended to collaboration tasks with a human and the presence of other dynamic obstacles in the workspace.

## ACKNOWLEDGMENT

The authors would like to thank the Ministry of Economics, Transport, Agriculture and Viticulture of the State of Rhineland-Palatinate for financial support within the project ''Building a collaborative and cooperative robotics platform-KoKoBot''.

## REFERENCES

[1] J. Hermann, A. David, A. Wagner, and M. Ruskowski, ''Considering interdependencies for a dynamic generation of process chains for production as a service,'' *Proc. Manuf.*, vol. 51, pp. 1454–1461, Jan. 2020.

[2] M. Ruskowski, A. Herget, J. Hermann, W. Motsch, P. Pahlevannejad, A. Sidorenko, S. Bergweiler, A. David, C. Plociennik, J. Popper, K. Sivalingam, and A. Wagner, ''Production bots für production level 4: Skill-basierte systeme für die produktion der zukunft,'' *ATP Magazin*, vol. 62, no. 9, pp. 62–71, 2020.

[3] S. Wrede, O. Beyer, C. Dreyer, M. Wojtynek, and J. Steil, ''Vertical integration and service orchestration for modular production systems using business process models,'' *Proc. Technol.*, vol. 26, pp. 259–266, Jan. 2016.

[4] M. Morari and J. H. Lee, ''Model predictive control: Past, present and future,'' *Comput. Chem. Eng.*, vol. 23, nos. 4–5, pp. 667–682, May 1999.

[5] S. J. Qin and T. A. Badgwell, ''A survey of industrial model predictive control technology,'' *Control Eng. Pract.*, vol. 11, no. 7, pp. 733–764, 2003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0967066102001867

[6] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, ''Learning-based model predictive control: Toward safe learning in control,'' *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 3, no. 1, pp. 269–296, May 2020.

[7] N. Koenig and A. Howard, ''Design and use paradigms for gazebo, an open-source multi-robot simulator,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, Sep./Oct. 2004, pp. 2149–2154.

[8] Stanford Artificial Intelligence Laboratory. *Robotic Operating System*. Accessed: May 23, 2018. [Online]. Available: https://www.ros.org

[9] S. M. LaValle and S. A. Hutchinson, ''Optimal motion planning for multiple robots having independent goals,'' *IEEE Trans. Robot. Autom.*, vol. 14, no. 6, pp. 912–925, Dec. 1998.

[10] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars, ''Probabilistic roadmaps for path planning in high-dimensional configuration spaces,'' *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[11] L. E. Kavraki and S. M. LaValle, ''Motion planning,'' in *Springer Handbook of Robotics*. Cham, Switzerland: Springer, 2016, pp. 139–162.

[12] 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,'' *Int. J. Robot. Res.*, vol. 35, no. 5, pp. 501–513, Apr. 2016.

[13] G. Wagner and H. Choset, ''Subdimensional expansion for multirobot path planning,'' *Artif. Intell.*, vol. 219, pp. 1–24, Feb. 2015.

[14] R. Shome, K. Solovey, A. Dobson, D. Halperin, and K. E. Bekris, ''dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning,'' *Auto. Robots*, vol. 44, nos. 3–4, pp. 443–467, Mar. 2020.

[15] R. Geraerts and M. H. Overmars, ''Creating high-quality paths for motion planning,'' *Int. J. Robot. Res.*, vol. 26, no. 8, pp. 845–863, Aug. 2007.

[16] I. A. Şucan, M. Moll, and L. E. Kavraki, ''The open motion planning library,'' *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012. [Online]. Available: https://ompl.kavrakilab.org

[17] O. Khatib, ''Real-time obstacle avoidance for manipulators and mobile robots,'' in *Autonomous Robot Vehicles*. New York, NY, USA: Springer, 1986, pp. 396–404.

[18] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, ''STOMP: Stochastic trajectory optimization for motion planning,'' in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4569–4574.

[19] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, ''CHOMP: Gradient optimization techniques for efficient motion planning,'' in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 489–494.

[20] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, ''Finding locally optimal, collision-free trajectories with sequential convex optimization,'' *Robot., Sci. Syst.*, vol. 9, no. 1, pp. 1–10, 2013.

[21] M. Wang, J. Luo, and U. Walter, ''A non-linear model predictive controller with obstacle avoidance for a space robot,'' *Adv. Space Res.*, vol. 57, no. 8, pp. 1737–1746, 2016.

[22] P. Bosscher and D. Hedman, ''Real-time collision avoidance algorithm for robotic manipulators,'' *Ind. Robot, Int. J.*, vol. 38, no. 2, pp. 186–197, 2011.

[23] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, ''Constrained model predictive control: Stability and optimality,'' *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[24] D. Lam, C. Manzie, and M. C. Good, ''Multi-axis model predictive contouring control,'' *Int. J. Control*, vol. 86, no. 8, pp. 1410–1424, 2013.

[25] M. M. G. Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, ''Real-time trajectory generation using model predictive control,'' in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2015, pp. 942–948.

[26] K. Belda and O. Rovny, ''Predictive control of 5 DOF robot arm of autonomous mobile robotic system motion control employing mathematical model of the robot arm dynamics,'' in *Proc. 21st Int. Conf. Process Control (PC)*, Jun. 2017, pp. 339–344.

[27] C. Liu, C. Lin, and M. Tomizuka, ''The convex feasible set algorithm for real time optimization in motion planning,'' *SIAM J. Control Optim.*, vol. 56, no. 4, pp. 2712–2733, Jun. 2018.

[28] T. Schoels, P. Rutquist, L. Palmieri, A. Zanelli, K. O. Arras, and M. Diehl, ''CIAO: MPC-based safe motion planning in predictable dynamic environments,'' *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6555–6562, 2021.

[29] C. Rösmann, F. Hoffmann, and T. Bertram, ''Planning of multiple robot trajectories in distinctive topologies,'' in *Proc. Eur. Conf. Mobile Robots (ECMR)*, Sep. 2015, pp. 1–6.

[30] J. Cascio, M. Karpenko, Q. Gong, P. Sekhavat, and I. M. Ross, ''Smooth proximity computation for collision-free optimal control of multiple robotic manipulators,'' in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 2452–2457.

[31] M. Krämer, C. Rösmann, F. Hoffmann, and T. Bertram, ''Model predictive control of a collaborative manipulator considering dynamic obstacles,'' *Optim. Control Appl. Methods*, vol. 41, no. 4, pp. 1211–1232, Jul. 2020.

[32] V. J. Lumelsky, ''On fast computation of distance between line segments,'' *Inf. Process. Lett.*, vol. 21, no. 2, pp. 55–61, Aug. 1985.

[33] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, ''A fast procedure for computing the distance between complex objects in three-dimensional space,'' *IEEE J. Robot. Autom.*, vol. RA-4, no. 2, pp. 193–203, Apr. 1988.

[34] E. Rimon and S. P. Boyd, ''Obstacle collision detection using best ellipsoid fit,'' *J. Intell. Robot. Syst.*, vol. 18, no. 2, pp. 105–126, 1997.

[35] C. Rösmann, M. Krämer, A. Makarow, F. Hoffmann, and T. Bertram, ''Exploiting sparse structures in nonlinear model predictive control with hypergraphs,'' in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2018, pp. 1332–1337.

[36] M. Gerdts, R. Henrion, D. Hömberg, and C. Landry, ''Path planning and collision avoidance for robots,'' *Numer. Algebra, Control Optim.*, vol. 2, no. 3, pp. 437–463, 2012.

[37] X. Zhang, A. Liniger, and F. Borrelli, ''Optimization-based collision avoidance,'' *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, May 2021.

[38] P. D. Christofides, R. Scattolini, D. M. de la Peña, and J. Liu, ''Distributed model predictive control: A tutorial review and future research directions,'' *Comput. Chem. Eng.*, vol. 51, pp. 21–41, Apr. 2013.

[39] M. Flad, L. Fröhlich, and S. Hohmann, "Cooperative shared control driver assistance systems based on motion primitives and differential games," *IEEE Trans. Human-Mach. Syst.*, vol. 47, no. 5, pp. 711–722, Oct. 2017.
[40] Y. He, M. Wu, and S. Liu, "An optimisation-based distributed cooperative control for multi-robot manipulation with obstacle avoidance," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9859–9864, 2021.
[41] A. Tika, N. Gafur, V. Yfantis, and N. Bajcinca, "Optimal scheduling and model predictive control for trajectory planning of cooperative robot manipulators," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9080–9086, 2021.
[42] A. Tika and N. Bajcinca, "Synchronous minimum-time cooperative manipulation using distributed model predictive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 7675–7681.
[43] B. Siciliano, O. Khatib, and T. Kröger, *Springer Handbook of Robotics*, vol. 200. Heidelberg, Germany: Springer, 2008.
[44] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct./Nov. 2001, pp. 1213–1219.
[45] R. Tallamraju, S. Rajappa, M. J. Black, K. Karlapalem, and A. Ahmad, "Decentralized MPC based obstacle avoidance for multi-robot target tracking scenarios," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Aug. 2018, pp. 1–8.
[46] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.
[47] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, May 2006.
[48] Harwell Subroutine Library. (2018). *A Collection of Fortran Codes for Large-Scale Scientific Computation*. [Online]. Available: http://www.hsl.rl.ac.uk
[49] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control*. Cham, Switzerland: Springer, 2017.
[50] J. W. Evans, "Random and cooperative sequential adsorption," *Rev. Mod. Phys.*, vol. 65, no. 4, p. 1281, 1993.

**NIGORA GAFUR** received the M.Sc. degree in mechanical engineering from the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, in 2017. She is currently pursuing the Ph.D. degree in mechanical engineering with the Department of Mechanical and Process Engineering, Technische Universität Kaiserslautern (TUK).

Her current research interests include motion control in robotics and model predictive control.

**GAJANAN KANAGALINGAM** is currently a Graduate Student and a Student Research Assistant at the Chair of Machine Tools and Control Systems, Department of Mechanical and Process Engineering, Technische Universität Kaiserslautern (TUK).

**ACHIM WAGNER** (Member, IEEE) is currently the Head of Research of the Department Innovative Factory Systems, German Research Center for Artificial Intelligence, and a Lecturer in the field of automation technology and computer science at the Technische Universität Kaiserslautern. After graduating in information technology, he worked as a Scientist and a Research Group Leader in the research fields of electrical materials engineering, medical and rehabilitation robotics, autonomous mobile robots, and dependable systems at the Universities of Saarland, Mannheim and Heidelberg. His current research interests include fault-tolerant autonomous production systems and intelligent human-technology systems.

**MARTIN RUSKOWSKI** received the Diploma degree in electrical engineering and the Dr.-Ing. degree in mechanical engineering from Leibniz University Hannover, Hannover, Germany, in 1996 and 2004, respectively.

He is currently the Head of the Innovative Factory Systems Research Department, German Research Center for Artificial Intelligence (DFKI), and the Chair of the Department of Machine Tools and Control Systems, Technische Universität Kaiserslautern (TUK). Since 2017, he has been the Head of the Board of the Technology Initiative SmartFactory KL. His major research interests include development of innovative control concepts for automation, artificial intelligence in automation technology, and industrial robots as machine tools. Prior to these positions, he held several management positions at industrial firms, most recently as the Vice President of Global Research and Development at KUKA Industries Group.