Predictive Control of Cooperative Robots Sharing Common Workspace

Argtim Tika and Naim Bajcinca

***Abstract*— We present a model predictive control (MPC) algorithm for online time-optimal trajectory planning of cooperative robotic manipulators. Robotic arms sharing a common confined operational space are exposed to high interrobot collision risks. For collision avoidance, a smooth robot geometry approximation by Bézier curves is applied, utilizing velocity constraints and tangent separating planes, enabling an efficient generation of robot trajectories in real-time. The proposed optimization algorithm is validated on an experimental setup consisting of two collaborative robotic arms performing synchronous pick-and-place tasks.**

***Index Terms*— Collision avoidance, cooperative robotics, model predictive control (MPC), pick-and-place robotics, real-time trajectory planning.**

I. Introduction

SINCE their introduction, robots have significantly revolutionized industrial workplaces and continuously increased the automation level in various industrial sectors. Robot deployments involve tasks in multiple domains such as packaging, sorting, assembly, and the execution of specific functions like machine tending, welding, and painting. Many of these tasks fall under the category of pick-and-place tasks and are well-suited as benchmarking frameworks in robotics due to their widespread use in various industrial applications.

Efficient deployment of multiple cooperating robotic systems brings the promise of shorter robot cycle times and faster task execution [1]. In a shared workspace, multiple robots can perform a wider variety of tasks, either as a composition of different subtasks into a more complex task or by extending the operating space, e.g., by passing a tool or workpiece from one robot to another [2]. Some applications involving coordinated motion planning with two or more robot manipulators are given in [3] and [4]. Typically, in such settings, the robots can communicate with each other, share sensor information, and coordinate their actions and motions to jointly perform the assigned tasks. However, the complexity of these systems poses new computational challenges, especially when the robot arrangement leads to a significant overlap of robot working areas and, thus, an increased collision risk. Therefore, the setup, control, and operation of multirobot systems is still primarily an active field of research, e.g., [2], [5], [6], [7], [8].

With increased automation in the food and beverage industry, multiple robots are often used for pick-and-place operations along sorting, packing, and processing lines, supplementary to high payload industrial robots usually deployed for palletizing [9]. For fast pick-and-place task execution, primarily noncollaborative selective compliance assembly robot arm (SCARA) or delta robots with three to four degrees of freedom (DoF) are used [10]. While these highly automated systems allow for high production throughput, they tend to be product-specific and usually require significant modifications to make changes and adjustments to production lines. With the increased diversity and personalization of goods and products driven by the latest developments in industry 4.0, production in short batches becomes more attractive. Despite increasing automation, manual methods are currently still considered the only cost-effective solution for the production of small batches, characteristic of high-mix, low-volume manufacturers [11]. The increased flexibility appears to be a key element toward higher levels of operational efficiency and productivity. This can be achieved by empowering workers to work directly with automated systems or by facilitating robot integration, enabling adaptations of robotic solutions to new workspaces [12]. To this end, this work deals with the integration of 6-DoF robots into existing manual processes to increase the level of automation on existing manual sorting and packaging stations, typical for small and seasonal producers or distributors. As shown in Fig. 1, we consider as an application a fruit packing station consisting of a conveyor, fruit feeder, a vision-based detection and inspection system, and two UR5 collaborative robot arms. The packaging system is to be completed by a foil wrapping station, resulting in a confined robot setup with highly overlapping robot working areas.

The efficient and safe use of cooperating robots in a shared workspace relies on balancing the robots' workload and ensuring certain safety aspects for collision-free robot operation. This problem is addressed by introducing a hierarchical approach involving two optimization-based methods, a discrete one for task scheduling and a continuous one for point-to-point trajectory generation. For task scheduling, the discrete scheduling algorithm presented in [13] is used. This work focuses on the online trajectory generation and the algorithm implementation on an experimental setup using a robot operating system (ROS). In order to minimize the robot cycle time and increase the operating throughput, the proposed local

Manuscript received 20 June 2023; accepted 6 October 2023. This work was supported in part by the German Federation of Industrial Research Associations (AiF) within the scope of the projects CooPick, KIMKO, and KORINS, under Grant ZF4335706DB7, Grant ZF4335711PO9, and Grant ZF4335715DB9, respectively. Recommended by Associate Editor M. Oishi. *(Corresponding author: Argtim Tika.)*

The authors are with the Department of Mechanical and Process Engineering, Rheinland-Pfälzische Technische Universität (RPTU) Kaiserslautern-Landau, 67663 Kaiserslautern, Germany (e-mail: argtim.tika@mv.uni-kl.de; naim.bajcinca@mv.uni-kl.de).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCST.2023.3331525.

Digital Object Identifier 10.1109/TCST.2023.3331525

planning approach involves a centralized model predictive controller that directly generates time-optimal robot trajectories in the state space of the joint positions and velocities. Given that avoiding collisions is computationally intensive, we use an approach based on a smooth approximation of the robot's geometry using Bézier curves, which is first introduced in [14] and validated on simulations. The present work builds on these results and integrates collision avoidance as an integral part of a real-time planning model predictive control (MPC) algorithm. Experiments on two specific test cases validate the algorithm's performance. Test Case 1 is constructed to analyze the performance of the trajectory planning in scenarios with high collision potential between the robot arms. We show that, although the robotic setup with highly overlapping robot operational spaces can lead to challenging manipulation tasks, even in the worst case scenario, two robot manipulators achieve shorter cycle times than deploying a single robotic arm. Test Case 2 shows a normal operating mode of the robotic system involving task scheduling and online trajectory planning to pick and place moving objects on the conveyor.

II. Related Work

Collision-free motion planning of robotic manipulators in dynamically changing environments requires algorithms capable of planning and updating trajectories in real-time. This research area has been widely studied in the last decades, presenting different solutions and algorithms, mainly categorized as global and local planning methods. The following gives a brief overview of related work, focusing more on optimization-based approaches as more related to the presented method.

Sampling-based approaches such as the popular rapidly-exploring random tree (RRT) method [15] and the probabilistic roadmap (PRM) method [16] are, especially in global motion planning, widely used for path planning of robotic manipulators. They involve a time-consuming preprocessing stage of sampling the collision-free configuration space and generating a graph-based representation like a roadmap or a tree data structure. Sampling-based approaches are well-suited for high-dimensional configuration space and can be successfully used for path planning in multirobot systems, e.g., [17], [18]. Since these algorithms are computationally demanding, the paths are computed offline followed by an open-loop execution, making them more suitable for application in static working environments. Prioritized planning methods, such as coordination along fixed independent paths and coordination along independent roadmaps can also be applied to avoid collisions in a multirobot system, e.g., [19], [20].

Sampling-based approaches have been subject to various adaptations resulting in algorithms capable of dealing with dynamically changing environments where continuous replanning is needed. Algorithms based on RRTs for motion planning in dynamic environments are discussed in [21] (CT-RRT) and [22] (RRT^X). Dynamic roadmaps (DRMs) are introduced in [23] for real-time path planning in changing environments. Motion planning based on the composition of two separate DRMs for a dual-arm robot is presented in [24], incorporating prioritized and coordinated planning

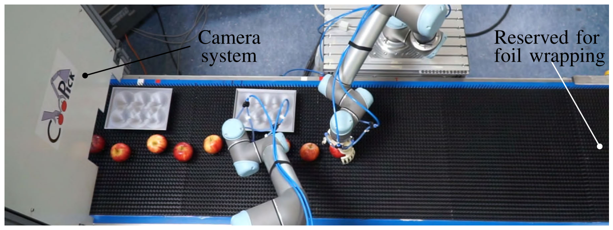

Fig. 1. Experimental setup consisting of two robotic arms inclusive of grippers, a conveyor belt with PLC, and a camera system for object detection and quality inspection.

along fixed paths or graphs for collision avoidance. To follow the generated path, a predictive path-following algorithm is proposed in [25].

One of the first and widely used local planning approaches for collision-free trajectory generation are artificial potential field methods proposed in [26]. These methods are mainly used for single-arm manipulation [27] and consist of generating a potential field with repulsive terms in the vicinity of obstacles and attracting terms at the target point. The generated artificial forces drive the robot away from the obstacles toward the goal. As an alternative to the artificial potential field approach, a local method for obstacle avoidance based on the introduction of virtual velocity dampers and the existence of separating hyperplanes is presented in [28] and described in more detail in [29] and [30]. The algorithm can also be applied to multirobot systems by representing each manipulator link by a hierarchical description with convex volumes to efficiently update the environment model as the robots move. However, the computation times achieved when applying this method are unsuitable for online applications.

Another optimization-based algorithm for collision-free robot trajectory planning involving two robot manipulators is presented in [31]. The parts of the robots are modeled by spherical shells, generating a geometric representation of the robots and their surroundings by a list of geometric primitives. This algorithm falls in the category of prioritized planning since the first robot follows a preprogrammed trajectory, and the second one has to reach a target while accounting for collision avoidance with the other robot. A multirobot trajectory optimization approach using the alternating direction method of multipliers (ADMMs) is proposed in [32]. This approach requires the existence of a strictly feasible initial trajectory, and the presented computation times are, for a setup with two robotic arms, not applicable for online applications. A local optimization approach for multiarm payload manipulation is presented in [33]. The article focuses on the teleoperation-based simultaneous guidance of multiple collaborative arms without considering any collision avoidance constraints.

CHOMP [34] and TrajOpt [35] represent two optimization-based algorithms for motion planning that can cover a wide range of robotic applications, mainly in static environments. The performance of the planners highly depends on the provided trajectory initialization. In some cases, an optimization problem needs to be solved to obtain an initialization trajectory, or multiple trajectory initialization is required to find a feasible solution. Both algorithms formulate trajectory planning as an unconstrained optimization problem penalizing the smoothness of the path and the proximity to obstacles. Implementing collision avoidance as penalties in the cost function has the drawback that the planner can converge to local minima of the cost function, which do not correspond to collision-free robot motions. These algorithms are mainly used in simulations since they rely on a geometry representation of the working environment. This poses a challenge when transferring the motion planning from simulation to reality and involves offline preprocessing. Depending on the application, additional sensors like cameras and laser scanners may be required to generate the geometry representation using voxel grids (CHOMP) or meshes (TrajOpt). In addition, CHOMP and TrajOpt are not complete planners, and postprocessing of the generated paths is required when using ROS to control a real robot. For the considered robots, this involves the motion planning framework MoveIt, adding additional processing time in the control loop. This, combined with the lack of verifiability of collision-free motion planning, makes them less suitable for real-world applications with dynamic target points where continuous replanning is required.

MPC algorithms are increasingly used in the field of robot manipulators not only for following a given path but also for point-to-point trajectory generation [36]. Considering two robotic arms, a hierarchical approach for MPC-based time-optimal planning is described in [37]. For collision avoidance, a standard approach is applied by modeling the robot's shape and obstacles using a composition of spheres and swept sphere lines. The algorithm is validated on a simple experimental setup using two planar robot arms with two DoFs each. Recent publications on MPC and dual-arm manipulation focus on cooperative object transportation without considering collision avoidance between the robot arms, e.g., [38], [39].

The listed planning algorithms encounter limitations when implementing motion planning on real applications with robots sharing a common workspace with dynamically moving target points. ADMM, for instance, requires a strictly feasible (collision-free) initial trajectory, which is difficult to provide. The PRM-based approach [24] requires offline preprocessing, and the outcome is a prioritized path planning, which requires an additional path-following algorithm to generate the control inputs for the robot arms. Experimental results presented in [31] using an interior-point optimizer also result in prioritized planning with one robot following a preprogrammed trajectory. CHOMP and TrajOpt require pre and postprocessing for an adequate environment representation and compatibility with the ROS interface. For the considered application, we present an MPC-based algorithm for time-optimal robot trajectory planning in the configuration space, which can be directly sent to the robots using a ROS velocity interface. Implementing collision avoidance as state-dependent hard constraints in the optimization problem can not prevent local minima but ensures that a computed feasible solution is, in fact, a collision-free trajectory.

III. Article Structure

The present article is structured as follows. The time-optimal MPC algorithm for online collision-free trajectory generation, including the prediction model and collision avoidance constraints, is presented in Section IV. Section V describes in detail the implementation of the algorithm on an experimental setup. The performance of the algorithm is demonstrated by experimental results presented in Section VI. A brief discussion and summary in Section VII concludes the article.

IV. Time-Optimal Online Trajectory Planning

The considered robot manipulators each have $n = 6$ joints and are controlled in their configuration space $\mathcal{C}_r \subseteq \mathbb{R}^n, r \in \{1,2\}$, using a joint velocity control interface. Both robots are represented by the generalized coordinates $\mathbf{q}_1(t) \in \mathcal{C}_1$ and $\mathbf{q}_2(t) \in \mathcal{C}_2$, which along with the respective joint velocities span the state space

$$\mathbf{x}(t) = \left[\mathbf{x}_1^{\mathrm{T}}(t), \mathbf{x}_2^{\mathrm{T}}(t)\right]^{\mathrm{T}} = \left[\mathbf{q}_1^{\mathrm{T}}(t), \dot{\mathbf{q}}_1^{\mathrm{T}}(t), \mathbf{q}_2^{\mathrm{T}}(t), \dot{\mathbf{q}}_2^{\mathrm{T}}(t)\right]^{\mathrm{T}}. \quad (1)$$

Considering two robots and multiple tasks, a task scheduling has to be performed first to assign to each robot $r \in \{1,2\}$ a set of final joint positions $\mathbf{q}_{rf}(t)$ and velocities $\dot{\mathbf{q}}_{rf}(t)$. The task scheduling is followed by the online generation of collision-free trajectories so that the robots can safely reach the assigned tasks within the minimum possible time. Therefore, we formulate the trajectory planning problem as a time-optimal optimization problem of the form

$$\min_{\mathbf{u}(t),\, t_f} \int_{t_0}^{t_f} \mathrm{d}t \quad (2)$$

$$\text{s.t. } \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \quad (2a)$$

$$\mathbf{x}(t) \in \chi_{\text{free}}, \quad t \in \left[t_0, t_f\right] \quad (2b)$$

$$\underline{\mathbf{u}} \le \mathbf{u}(t) \le \bar{\mathbf{u}}, \quad t \in \left[t_0, t_f\right] \quad (2c)$$

$$\mathbf{x}(t_f) = \mathbf{x}_f(t_f). \quad (2d)$$

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

A. Prediction Model

Depending on the underlying low-level robot control and the used control interface, different prediction models can be used for optimization-based online robot trajectory planning. This also depends on the used planning algorithm and the resulting optimization problem's constraints. Since the robots are controlled in their configuration space, the online planning of time-optimal robot trajectories is also performed directly in the robots' configuration space, although they perform their tasks in the 3-D operational space. Moreover, the robot's geometry approximation and the collision avoidance between the robots and other potential obstacles are also defined in the 3-D workspace. The nonlinear robot forward kinematics transforms the constraints of the robot motions described in the 3-D operational space into the respective configuration space, resulting in robot joint position and velocity constraints. The use of a simplified kinematic prediction model of n double integrators for each robot, i.e.,

$$\ddot{\mathbf{q}}_r(t) = \mathbf{u}_r(t), \quad r \in \{1, 2\} \tag{3}$$

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

Considering a centralized time-optimal MPC algorithm for both robot manipulators represented by the state space $\mathbf{x}(t) \in \mathbb{R}^{4n}$ from (1), yields the overall prediction model

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{4}$$

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

$$\mathbf{A} = \operatorname{diag}\left(\begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}, \begin{bmatrix}\mathbf{0} & \mathbf{I}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\right), \quad \mathbf{B} = \operatorname{diag}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}, \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\right) \tag{5}$$

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

B. Collision Avoidance

The goal of trajectory planning is to generate robot trajectories online so that they safely reach the assigned targets in the shortest possible time t_f, i.e.,

$$\lim_{t\to t_f} ||\mathbf{x}(t) - \mathbf{x}_f(t)|| = 0, \text{ with } \mathbf{x}(t) \in \chi_{\text{free}}, \quad t \in \left[t_0, t_f\right]. \tag{6}$$

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

Performing cooperative tasks in a shared workspace with highly overlapping operating areas between the robots significantly increases the potential for interrobot collisions and imposes more complexity on trajectory planning. When formulating collision avoidance constraints, the entire robot geometry must be considered to exclude the possibility of collisions occurring between any robot parts. Therefore, each robot link is usually approximated by convex polyhedral objects, mostly spheres and ellipsoids, and between all possible interrobot links, collision avoidance conditions are defined to prohibit their intersection. Assuming five to six convex objects per robot for two robots with six joints each result in 25–36 constraints in the worst case. Moreover, in optimization-based planning algorithms with a rolling

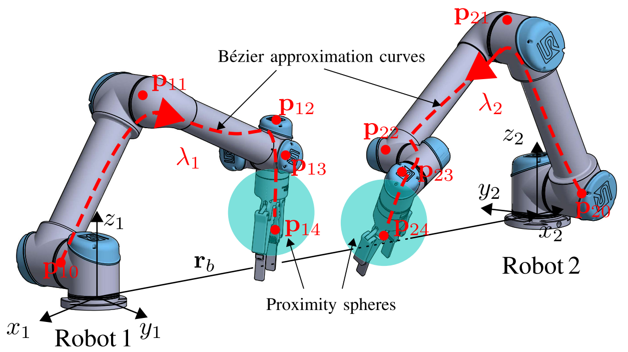

Fig. 2. Robot geometry approximation with smooth polynomial Bézier curves. Collision avoidance conditions are defined using two sliding spheres, whose positions are updated according to the minimum distance between the corresponding linkage Bézier curves.

horizon, such as MPC, the constraints are applied along the horizon, which can increase their multiplicity to several hundred. This significantly increases the computational complexity of the underlying optimization problem and poses a substantial barrier to the fast solution of the optimization problem and, thus, to the online planning of collision-free robot trajectories. To overcome this problem, we have presented in [14] a novel approach based on a smooth approximation of the robot geometry that significantly reduces the number of collision avoidance constraints. For the approximation of the robot linkage we use Bernstein basis polynomials

$$b_i^{n_r}(\lambda_r) := \begin{cases} \begin{pmatrix} n_r \\ i \end{pmatrix} \lambda_r^i (1-\lambda_r)^{n_r - i}, & \text{for } 0 \le i \le n_r \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

and a sequence of $n_r + 1$ control points $\mathbf{p}_{ri}(\mathbf{q}_r) \in \mathbb{R}^3$ per robot $r \in \{1, 2\}$, resulting in a parametric curve of the form

$$\mathbf{p}_r(\lambda_r, \mathbf{q}_r) = \sum_{i=0}^{n_r} b_i^{n_r}(\lambda_r)\mathbf{p}_{ri}(\mathbf{q}_r), \quad 0 \le \lambda_r \le 1 \tag{8}$$

known as Bézier functions [40]. Here and in the following, we suppress the time dependence of the joint variables and other related functions for better readability until further notice.

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

$$\begin{aligned} {}_0\mathbf{p}_{1i}(\mathbf{q}_1) &= \mathbf{p}_{1i}(\mathbf{q}_1) \text{ and} \\ {}_0\mathbf{p}_{2i}(\mathbf{q}_2) &= {}_0\mathbf{r}_\mathbf{b} + \mathbf{R}_{z,\frac{\pi}{2}}\mathbf{p}_{2i}(\mathbf{q}_2) \quad \forall i \in \{0, \dots, n_r\} \end{aligned} \tag{9}$$

using the SO(3) rotation matrix $\mathbf{R}_{z,(\pi/2)}$ [41].

As opposed to the standard approach in the literature, we do not apply a static convex approximation of the robot geometry but introduce a dynamic approximation scheme based on the Bézier curves by continuously computing the minimum distance between them. According to the computed minimum distance, which reflects the collision potential, we locally approximate parts of the robots by spheres and define constraints to avoid collisions, see Fig. 2. The position of the spheres is not directly linked to the robot links but rather to the approximation curves, and their location is continuously updated depending on the robot configurations, i.e., collision potential along the robots. This results in sliding collision spheres along the robot geometries following the minimal distance between the robots. Here, the center of the spheres coincides with the points where the distance between the Bézier curves is minimal.

1) Minimum Distance Computation: Existing approaches for calculating the minimum distance between two parametric curves and surfaces can be classified into root-finding and culling-based methods [42]. An overview of different global solution techniques that can be applied to find all roots of a nonlinear system of polynomial equations is given in [43]. The culling-based approach is a geometric rather than a numerical method and was originally introduced in [44] as a hierarchical framework for computing minimum distances between geometric objects. The geometric approaches is reported to be more robust and have a higher computational efficiency, especially when computing the minimum distance between high-order surfaces [44]. Given that information for the initial guess can be extracted from the control points of the curves, we formulate the computation of the minimum distance as an optimization problem

$$\boldsymbol{\lambda}^* = \arg\min_{\boldsymbol{\lambda}} \quad \|{}_0\mathbf{p}_1(\lambda_1, \mathbf{q}_1) - {}_0\mathbf{p}_2(\lambda_2, \mathbf{q}_2)\|_2$$
$$\text{s.t.} \quad 0 \le \boldsymbol{\lambda} \le 1 \tag{10}$$

with $\boldsymbol{\lambda} = [\lambda_1, \lambda_2]^{\mathrm{T}}$. To provide a good initial guess for the minimum distance, we use the heuristic approach that the closest point pair of the control points is a good approximation of the closest point pair of the underlying Bézier curves. However, this assumption does not always hold, such as when at least one edge of the control polygon, defined by connecting the control points of a Bézier curve, is not an edge of the convex hull formed by those control points. For a better approximation, the minimum distance between the two convex hulls can also be computed by applying the Gilbert–Johnson–Keerthi (GJK) algorithm [45] as used in [42] to dynamically compute the subdivision regions. Using the indices

$$\{i^*, j^*\} = \arg\min_{i,j}\{ \|{}_0\mathbf{p}_{1i}(\mathbf{q}_1) - {}_0\mathbf{p}_{2j}(\mathbf{q}_2)\|_2 \mid$$
$$i \in \{0, \dots, n_1\}, \quad j \in \{0, \dots, n_2\}\} \tag{11}$$

of the closest pair of control points, the initial guess for the optimization problem (10) is chosen as $\boldsymbol{\lambda}_0 = [i^*/n_1, j^*/n_2]^{\mathrm{T}}$. Finally, after solving the optimization problem the distance between the centers of the two spheres corresponds to the value of the cost function

$$d = \|{}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1) - {}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)\|_2\,. \tag{12}$$

2) Interrobot Collision Avoidance: Collision avoidance between the robot arms is based on velocity constraints presented in [14], which prohibit an intersection between the

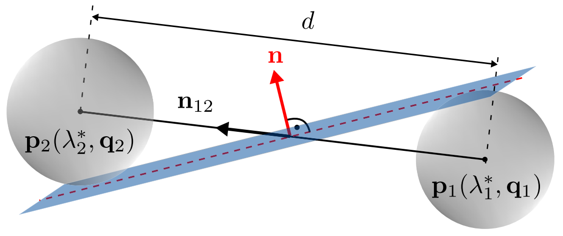
d
$\mathbf{n}$
$\mathbf{n}_{12}$
$\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)$
$\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)$

Fig. 3. Collision spheres and a tangent separating plane.

geometry approximating spheres by forcing them to slide along tangent separating planes [29]. Let $\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)$ and $\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)$ denote the centers of the spheres, see Fig. 3, and $\mathbf{v}_1(\lambda_1^*, \mathbf{q}_1)$, $\mathbf{v}_2(\lambda_2^*, \mathbf{q}_2)$ the respective velocities, the collision avoidance constraint forcing the spheres to stay on their respective side of the separating plane read

$$\mathbf{n}^{\mathrm{T}}\big(\mathbf{v}_2(\lambda_2^*, \mathbf{q}_2) - \mathbf{v}_1(\lambda_1^*, \mathbf{q}_1)\big) \ge \epsilon \tag{13}$$

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

$$\mathbf{J}_1(\mathbf{q}_1) = \frac{\partial_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)}{\partial \mathbf{q}_1}, \quad \mathbf{J}_2(\mathbf{q}_2) = \frac{\partial_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)}{\partial \mathbf{q}_2} \tag{14}$$

where ${}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)$ and ${}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)$ are the vectors from the origin of the inertial frame to the points $\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)$ and $\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)$, the constraint (13) is transformed in the robot's configuration space

$$\mathbf{n}^{\mathrm{T}}\big(\mathbf{J}_2(\mathbf{q}_2)\dot{\mathbf{q}}_2 - \mathbf{J}_1(\mathbf{q}_1)\dot{\mathbf{q}}_1\big) \ge \epsilon \tag{15}$$

and finally expressed in the state space

$$\big[\mathbf{0}^{1\times n} \;\; -\mathbf{n}^{\mathrm{T}}\mathbf{J}_1(\mathbf{x}) \;\; \mathbf{0}^{1\times n} \;\; \mathbf{n}^{\mathrm{T}}\mathbf{J}_2(\mathbf{x})\big]\mathbf{x} \ge \epsilon \tag{16}$$

with the state space vector $\mathbf{x} \in \mathbb{R}^{4n\times 1}$ from (1) and the Jacobian matrices $\mathbf{J}_1(\mathbf{x}) \in \mathbb{R}^{3\times n}$, $\mathbf{J}_2(\mathbf{x}) \in \mathbb{R}^{3\times n}$.

To solve the optimization problem and compute robot trajectories $\mathbf{x}$ satisfying the interrobot collision avoidance constraints (16), the perpendicular vector $\mathbf{n}$ must be predefined at each optimization step k, as it is not subject to the optimization. A suitable perpendicular vector $\mathbf{n}$ is computed based on a 2-D projection method, which consists of projecting the considered spheres onto 2-D in the x_0y_0-plane, thus reducing the problem to the calculation of tangential separating lines, as shown in Fig. 4. The vector normal to the separating tangent is then chosen to be equal to $\mathbf{n}_1$ or $\mathbf{n}_2$ depending on the relative robot movements. The collision avoidance constraints prevent the circles representing a 2-D projection of the spheres from intersecting, which also results in motions that do not allow the robots to move on top of each other. As already stated, projecting the spheres onto 2-D in the x_0y_0-plane results in two possible separating lines tangent to both circles, i.e., two possible perpendicular vectors $\mathbf{n} \in \{\mathbf{n}_1, \mathbf{n}_2\}$. To define the vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ perpendicular to the corresponding tangent separating lines, we conventionally use the direction of the vector between the centers of the circles

$$\mathbf{n}_{12} = \frac{{}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2) - {}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)}{d} \tag{17}$$

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

$$s = \frac{r_1 d}{r_1 + r_2}, \quad \varphi = \arcsin\left(\frac{r_1}{s}\right) \tag{18}$$

according to Fig. 4. The perpendicular vectors are then uniquely obtained by

$$\mathbf{n}_1 = \mathbf{R}^{\mathrm{T}}_{z,\frac{\pi}{2}-\varphi}\mathbf{n}_{12}, \quad \mathbf{n}_2 = \mathbf{R}^{\mathrm{T}}_{z,\varphi-\frac{\pi}{2}}\mathbf{n}_{12}. \tag{19}$$

The accurate selection between $\mathbf{n} \in \{\mathbf{n}_1, \mathbf{n}_2\}$ has an important impact on the efficiency of conflict resolution and collision avoidance. Therefore, we introduce the direction vector $\mathbf{n}_v$ of the relative movement of the proximity spheres

$$\mathbf{n}_v = \frac{\Delta_0\mathbf{p}_2(\lambda_2^*) - \Delta_0\mathbf{p}_1(\lambda_1^*)}{||\Delta_0\mathbf{p}_2(\lambda_2^*) - \Delta_0\mathbf{p}_1(\lambda_1^*)||_2} \tag{20}$$

with

$$\begin{aligned}\Delta_0\mathbf{p}_1(\lambda_1^*) &= {}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_{1f}) - {}_0\mathbf{p}_1(\lambda_1^*, \mathbf{q}_1)\\ \Delta_0\mathbf{p}_2(\lambda_2^*) &= {}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_{2f}) - {}_0\mathbf{p}_2(\lambda_2^*, \mathbf{q}_2)\end{aligned} \tag{21}$$

denoting the vectors from the position of the proximity spheres at the actual robot configurations $\mathbf{q}_1$ and $\mathbf{q}_2$ to the position of the spheres at the target configurations $\mathbf{q}_{1f}$ and $\mathbf{q}_{2f}$, respectively. This vector provides information about the spheres' relative direction when placing the robots from the actual to their final configuration. In fact, $\mathbf{n}_v$ describes the tentative relative movement of the spheres in Fig. 3 during a conflict resolution phase. Roughly speaking, we hereby invoke global information referring to the relative motion of the distance spheres over the prediction of the conflict resolution horizon. Extensive simulations and experiments indicate that this alternative over the strategy based on local, i.e., the instantaneous relative motion of the collision distance spheres is more efficient and stable in resolving the conflicts [14]. The direction vector of the relative motions of the spheres (20) is then used to choose one of the computed perpendicular vectors and apply it to the collision avoidance constraint (16). If the scalar product of the vector $\mathbf{n}_v$ with one of the perpendicular vectors is positive, the vector $\mathbf{n}$ is chosen to be equal to this vector, i.e.,

$$\mathbf{n} = \mathbf{n}_1\frac{1}{2}\left(1 + \mathrm{sgn}\left(\mathbf{n}_1^{\mathrm{T}}\mathbf{n}_v\right)\right) + \mathbf{n}_2\frac{1}{2}\left(1 + \mathrm{sgn}\left(\mathbf{n}_2^{\mathrm{T}}\mathbf{n}_v\right)\right). \tag{22}$$

Otherwise, if

$$\mathrm{sgn}\left(\mathbf{n}_1^{\mathrm{T}}\mathbf{n}_v\right)\mathrm{sgn}\left(\mathbf{n}_2^{\mathrm{T}}\mathbf{n}_v\right) \geq 0 \tag{23}$$

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

$$\begin{aligned}\mathbf{n} = \mathbf{n}_1&\frac{1}{2}\left(1 - \mathrm{sgn}\left(|\mathbf{n}_1^{\mathrm{T}}\mathbf{n}_v| - |\mathbf{n}_2^{\mathrm{T}}\mathbf{n}_v|\right)\right)\\ &+ \mathbf{n}_2\frac{1}{2}\left(1 + \mathrm{sgn}\left(|\mathbf{n}_1^{\mathrm{T}}\mathbf{n}_v| - |\mathbf{n}_2^{\mathrm{T}}\mathbf{n}_v|\right)\right).\end{aligned} \tag{24}$$

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

$$\mathbf{e}_3^{\mathrm{T}}\,{}_r\mathbf{p}_{r2}(\mathbf{q}_r) \geq d_{z1}, \quad \mathbf{e}_3^{\mathrm{T}}\,{}_r\mathbf{p}_{r4}(\mathbf{q}_r) \geq d_{z2}, \quad r \in \{1, 2\} \tag{25}$$

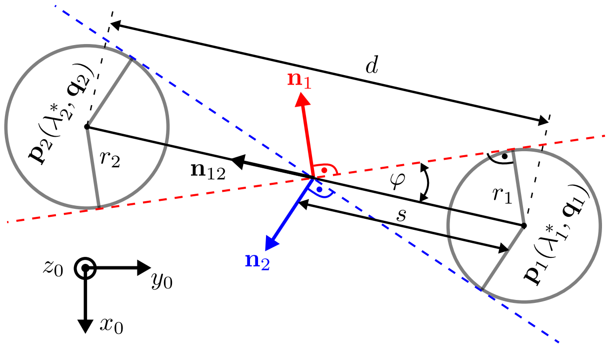

Fig. 4. Schematic illustration of separating common tangents of the circles representing the spheres projected on the x_0y_0-plane.

to limit the respective robot's working area to a certain minimum operating height and avoid collisions with the bench. Here, ${}_r\mathbf{p}_{r2}(\mathbf{q}_r)$ and ${}_r\mathbf{p}_{r4}(\mathbf{q}_r)$ denote the vectors from the origin of the respective base frame of the robots to the control points in the area of the robot wrist and end-effector, see Fig. 2. $\mathbf{e}_3^{\mathrm{T}} = [0, 0, 1]$ is the unit vector, and d_{z1}, d_{z2} are distance parameters defining the minimum operating height along the z-axis. Furthermore, to avoid collisions between the gripper and the robot's shoulder, we restrict the distance between the control points of the basis and the end-effector to a minimum distance d_s by applying the constraints

$$||{}_r\mathbf{p}_{r4}(\mathbf{q}_r) - {}_r\mathbf{p}_{r0}(\mathbf{q}_r)||_2 \geq d_s, \quad r \in \{1, 2\}. \tag{26}$$

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

$$\sin\left(\mathbf{e}_4^{\mathrm{T}}\mathbf{q}_r\right)|\cos\left(\mathbf{e}_5^{\mathrm{T}}\mathbf{q}_r\right)| \leq \varepsilon, \quad r \in \{1, 2\} \tag{27}$$

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

C. Minimum-Time MPC

To solve the optimization problem (2), we first perform the time scaling

$$\tau = \frac{t - t_0}{t_f - t_0} = \frac{t - t_0}{\Delta t_f} \tag{28}$$

mapping the time interval $t \in [t_0, t_f]$ to $\tau \in [0, 1]$ and transform the free terminal time optimization problem into a fixed terminal time problem. Without loss of generality, we assume $t_0 = 0$ and transform the dynamic model (4) in the scaled time τ. The resulting continuous time system is discretized by integration over the time step $\Delta\tau$, yielding the discrete-time dynamics

$$\mathbf{x}(k+1) = (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}(k) + \Delta\tau t_f^2\left(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\right)\mathbf{B}\mathbf{u}(k) \tag{29}$$

with the identity matrix $\mathbf{I} \in \mathbb{R}^{4n\times 4n}$ and the input $\mathbf{u}(k)$, which is constant within the time interval $\tau \in [k\Delta\tau, (k+1)\Delta\tau)$. The derivation of (29) is given in the Appendix. The time transformation is also applied to the velocity-dependent collision avoidance constraint (16), yielding

$$\delta_c \epsilon \le (1-\delta_c)\epsilon + \delta_c \frac{1}{t_f}\big[\mathbf{0}\ \ -\mathbf{n}^{\mathrm{T}}\mathbf{J}_1(\mathbf{x}(k))\ \ \mathbf{0}\ \ \mathbf{n}^{\mathrm{T}}\mathbf{J}_2(\mathbf{x}(k))\big]\mathbf{x}(k) \tag{30}$$

with the switching parameter

$$\delta_c = \begin{cases} 1, & \text{if } d \le d_{\text{in}} \\ 0, & \text{if } d > d_{\text{in}} \vee t_f^* \le t_{\min}. \end{cases} \tag{31}$$

By introducing δ_c, the constraint is activated only if the computed minimum distance d lies within the influence distance d_{in}. Otherwise, or if the robots are close to their targets, i.e., $t_f^* \le t_{\min}$, the constraint is trivially satisfied. Here, t_f^* denotes the optimal final time computed at the previous sampling time $k-1$. This is motivated by the fact that the chosen robot targets result in feasible, i.e., collision-free, configurations by applying a scheduling algorithm for robot task scheduling [13].

The time transformation must also be applied to the upper and lower bounds of the joint's velocities, yielding

$$\operatorname{diag}\big(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I}\big)\underline{\mathbf{x}} \le \mathbf{x}(k) \le \operatorname{diag}\big(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I}\big)\bar{\mathbf{x}} \tag{32}$$

with $\underline{\mathbf{x}}^{\mathrm{T}} = [\underline{\mathbf{q}}^{\mathrm{T}}, \underline{\dot{\mathbf{q}}}^{\mathrm{T}}, \underline{\mathbf{q}}^{\mathrm{T}}, \underline{\dot{\mathbf{q}}}^{\mathrm{T}}]$ and $\bar{\mathbf{x}}^{\mathrm{T}} = [\bar{\mathbf{q}}^{\mathrm{T}}, \bar{\dot{\mathbf{q}}}^{\mathrm{T}}, \bar{\mathbf{q}}^{\mathrm{T}}, \bar{\dot{\mathbf{q}}}^{\mathrm{T}}]$. Here, $\mathbf{I} \in \mathbb{R}^{n\times n}$ denotes the identity matrix, $\underline{\mathbf{q}} \in \mathbb{R}^n$, $\underline{\dot{\mathbf{q}}} \in \mathbb{R}^n$ the lower bounds, and $\bar{\mathbf{q}} \in \mathbb{R}^n$, $\bar{\dot{\mathbf{q}}} \in \mathbb{R}^n$ the upper bounds of the joint position and velocities in the time t. The initial and terminal constraints are also transformed in the scaled time τ.

After transforming the trajectory planning problem (2) into a discrete fixed terminal time optimization problem, the resulting MPC algorithm for time-optimal robot trajectory planning can be written as a static optimization problem in the form

$$\min_{\mathbf{u}(\cdot),\, t_f} \ t_f \tag{33}$$

$$\text{s.t.}\quad \mathbf{x}(j+1\,|\,k) = (\mathbf{I}+\Delta\tau\mathbf{A})\mathbf{x}(j\,|\,k) + \Delta\tau t_f^2\Big(\mathbf{I}+\frac{\Delta\tau}{2}\mathbf{A}\Big)\mathbf{B}\mathbf{u}(j\,|\,k) \tag{33a}$$

$$\begin{aligned}&\mathbf{x}(j+1\,|\,k) \le \operatorname{diag}\big(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I}\big)\bar{\mathbf{x}}\\ &\mathbf{x}(j+1\,|\,k) \ge \operatorname{diag}\big(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I}\big)\underline{\mathbf{x}}\end{aligned} \tag{33b}$$

$$\begin{aligned}&\underline{\mathbf{u}} \le \mathbf{u}(j+1\,|\,k) \le \bar{\mathbf{u}}\\ &\underline{\dot{\mathbf{u}}} \le (\mathbf{u}(j+1\,|\,k)-\mathbf{u}(j\,|\,k))/\big(t_f\Delta\tau\big) \le \bar{\dot{\mathbf{u}}}\end{aligned} \tag{33c}$$

$$\begin{aligned}&\mathbf{x}(0\,|\,k) = \operatorname{diag}\big(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I}\big)\mathbf{x}_0(k)\\ &\mathbf{x}\big(N_p\,|\,k\big) = \operatorname{diag}\big(\mathbf{I}, t_f\mathbf{I}, \mathbf{I}, t_f\mathbf{I}\big)\mathbf{x}_f(k)\end{aligned} \tag{33d}$$

$$\begin{aligned}&\mathbf{e}_3^{\mathrm{T}}\,{}_r\mathbf{p}_{r2}(\mathbf{x}(j+1|k)) \ge d_{z1}\\ &\mathbf{e}_3^{\mathrm{T}}\,{}_r\mathbf{p}_{r4}(\mathbf{x}(j+1|k)) \ge d_{z2}\end{aligned} \tag{33e}$$

$$\begin{aligned}&\|{}_r\mathbf{p}_{r4}(\mathbf{x}(j+1|k)) - {}_r\mathbf{p}_{r0}(\mathbf{x}(j+1|k))\|_2 \ge d_s\\ &\sin\big(\mathbf{e}_4^{\mathrm{T}}\mathbf{x}(j+1|k)\big)|\cos\big(\mathbf{e}_5^{\mathrm{T}}\mathbf{x}(j+1|k)\big)| \le \varepsilon\\ &\sin\big(\mathbf{e}_{16}^{\mathrm{T}}\mathbf{x}(j+1|k)\big)|\cos\big(\mathbf{e}_{17}^{\mathrm{T}}\mathbf{x}(j+1|k)\big)| \le \varepsilon\end{aligned} \tag{33f}$$

$$\delta_c\epsilon \le (1-\delta_c)\epsilon + \delta_c\frac{1}{t_f}\big[\mathbf{0}\ \ -\mathbf{n}^{\mathrm{T}}\mathbf{J}_1(\mathbf{x}(\bar{j}+1\,|\,k))\ \ \mathbf{0}\ \ \mathbf{n}^{\mathrm{T}}\mathbf{J}_2(\mathbf{x}(\bar{j}+1\,|\,k))\big]\mathbf{x}(\bar{j}+1\,|\,k) \tag{33g}$$

where $j \in \{0,\ldots,N_p-1\}$ is the iteration index, N_p the prediction horizon, and $r \in \{1,2\}$ the index denoting the robots. In each optimization step k, the time minimizing static optimization problem is solved to generate robot trajectories from the current measured state $\mathbf{x}_0(k)$ to a desired final state $\mathbf{x}_f(k)$, as expressed by the initial and terminal constraints (33d). The scaled discrete-time dynamic model (33a) with the sampling time $\Delta\tau = 1/N_p$ is used to predict robot trajectories along the prediction horizon. The optimization is also subject to minimum/maximum bounds on the states (33b), the inputs and their time derivatives (33c), and further constraints for collision-free trajectory planning. The constraints to avoid collisions with the workbench (33e) and between the robot's links (33f) are always active along the entire prediction horizon. Here, $\mathbf{e}_4^{\mathrm{T}}\mathbf{x}(\cdot) = q_{14}(\cdot)$, $\mathbf{e}_5^{\mathrm{T}}\mathbf{x}(\cdot) = q_{15}(\cdot)$, and $\mathbf{e}_{16}^{\mathrm{T}}\mathbf{x}(\cdot) = q_{24}(\cdot)$, $\mathbf{e}_{17}^{\mathrm{T}}\mathbf{x}(\cdot) = q_{25}(\cdot)$ represent the variables of the robot's joints four and five, see (27). The interrobot collision avoidance constraints (33g), on the other side, are applied only along a part of the prediction horizon N_p, as denoted by the index $\bar{j} = j \in \{0,\ldots,N_c\}$ with $N_c < N_p$. This is motivated by the fact that robot trajectories are iteratively replanned at each optimization step k. Therefore, it is sufficient to ensure collision-free planning along a section of the prediction horizon that is sufficiently larger than the control horizon.

The scaled time interval $\tau \in [0,1]$ is subdivided into N_p equidistant time intervals $\Delta\tau$, which are mapped by $\Delta t = \Delta\tau t_f^*$ into increasingly tighter Δt-intervals as time propagates, since the computed optimal final time t_f^* reduces with each optimization step. In early iterations, the time resolution of the computed trajectories is lower and increases after each MPC iteration, resulting in sequentially finer robot trajectories. This has the advantage that, toward the end, when the robot approaches the target, the time resolution of the trajectories is higher, leading to more accurate planning required to reach the target points.

V. Implementation

The experimental setup consists of two UR5 collaborative robotic arms equipped with pneumatic grippers, a camera system, a conveyor belt inclusive programmable logic controller (PLC), and a Host-PC where the algorithms are implemented, see Fig. 1. The planned time-optimal trajectories are sent to the robot controllers via transmission control protocol/internet protocol (TCP/IP) communication protocol using ROS melodic [46]. The resulting system and implementation architecture are shown in Fig. 5. The scheduler gets information about the position of objects ${}_0\mathbf{p}_o$, slots ${}_0\mathbf{p}_s$ (relative to the inertial frame), and the actual conveyor speed v_c, and computes feasible robot target points, which are transformed into final robot joint positions $\mathbf{q}_{rf}$ and velocities $\dot{\mathbf{q}}_{rf}$. ROS Hardware Interface along with the input–output robot interface is used to close or open the grippers, activate the camera system, and set the reference velocity $v_{c,\text{ref}}$ for the PLC.

With the measured actual robot state and the given final target state, the trajectories are predicted along the prediction horizon using (29) and provided as an initial guess to the MPC algorithm to initialize the motion planning. The input vector $\mathbf{u}(k)$ needed to predict trajectories is approximated

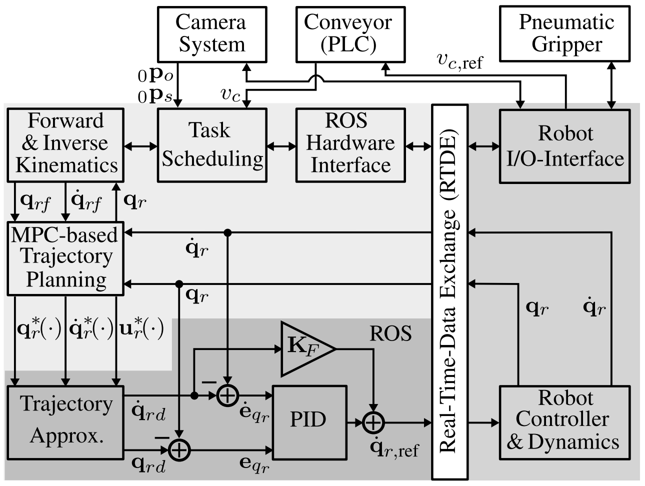

Fig. 5. System architecture of the considered experimental setup. The algorithms for task scheduling and minimum-time trajectory planning, inclusive of forward and inverse kinematics, are implemented on a Host-PC. The generated trajectories are approximated by quintic polynomials and forwarded to the low-level robot controller via a PID-feedback controller in combination with a feedforward term using ROS and a velocity control interface.

by a bang–bang acceleration/deceleration profile, typical for time-optimal planning. The computed minimum-time joint trajectories consisting of position $\mathbf{q}_r^*(\cdot) = [\mathbf{q}_r^*(0), \ldots, \mathbf{q}_r^*(N_p)]$, velocity $\dot{\mathbf{q}}_r^*(\cdot) = [\dot{\mathbf{q}}_r^*(0), \ldots, \dot{\mathbf{q}}_r^*(N_p)]$, and acceleration $\mathbf{u}_r^*(\cdot) = [\mathbf{u}_r^*(0), \ldots, \mathbf{u}_r^*(N_p)]$ data points are forwarded, along with a time vector $t^* = [0, \ldots, t_f^*]$, to ROS Control to finally generate reference points for the low-level robot controllers. In this work, *joint velocity interface* in conjunction with *joint velocity controller* is used by ROS Control to send and receive control commands to the underlying robot controllers. More specifically, *joint trajectory controller* is used for executing joint-space trajectories on a group of joints. Since we iteratively plan the trajectories, we use the *action interface*, which grants the possibility of replacing trajectories at each optimization step when the new ones are available.

Given a set of position, velocity, and acceleration data points to be reached at specific time instants, the controller performs a spline interpolation using quintic polynomials. As shown in Fig. 5 the used velocity controller provides a closed-loop control structure using a proportional integral derivative (PID) controller in combination with a feedforward term, given by

$$\dot{\mathbf{q}}_{r,\text{ref}}(t) = \mathbf{K}_F\dot{\mathbf{q}}_{rd}(t) + \mathbf{K}_P\mathbf{e}_{q_r}(t) + \mathbf{K}_I\int_0^t \mathbf{e}_{q_r}(\tilde{t})\mathrm{d}\tilde{t} + \mathbf{K}_D\dot{\mathbf{e}}_{q_r}(t)$$

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

To realize the feedback loops needed for MPC and the ROS velocity controller, fast reading of the current position and velocity of the robot joints is crucial. Measuring and reading the necessary feedback data requires time, depending on the update frequency of the UR controller and ROS itself. The control and update frequency of the robot controller is 125 Hz, providing sufficient performance for online robot trajectory planning. However, when using ROS to access the measurement data via ROS messages, the performance of the reading

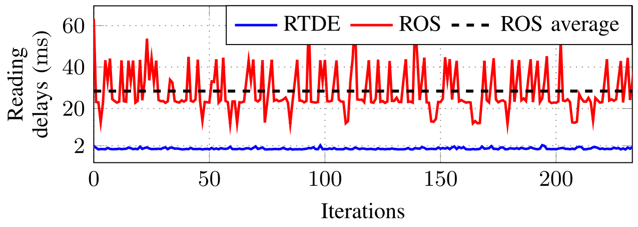

Fig. 6. Direct comparison of reading time delays coming from RTDE and ROS while performing pick-and-place tasks.

TABLE I
TUNED PID PARAMETERS

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

time is poor, causing time delays between 15 and 60 ms per robot, as shown in Fig. 6. To overcome this problem, a direct synchronization for data exchange between the UR controller and the Host-PC is established using a real-time data exchange (RTDE) interface over a standard TCP/IP connection, without breaking any real-time properties of the UR controller [47]. Although the ROS driver uses the RTDE interface for data exchange, the processing overhead when using ROS messages causes significant delays far greater than the delays resulting from RTDE, see Fig. 6.

The feedforward PID controller is provided by the ROS Driver in conjunction with the velocity control interface. The PID parameters are tuned using sinusoidal reference trajectories following some basic guidelines described in [41] since the default PID values do not guarantee a good tracking performance. The sinusoidal reference trajectories were generated considering the position and velocity limits of the robots in order not to exceed physical limitations and to avoid self-collision between the links. Therefore, optimization-based trajectories can also be considered, similar to those introduced by Tika et al. [48] for robot dynamic parameter estimation. To evaluate the tracking performance of the tuned parameters shown in Table I, we let the robot perform pick-and-place tasks and thus track trajectories other than those used during the tuning process. As shown in Fig. 7 for the position and in Fig. 8 for the velocity of the second robot joint, the chosen parameters result in a very good tracking performance with nonsignificant tracking errors. The tracking performance of the other robot joints is similarly good, so we refrain from a graphical representation of the trajectories.

The algorithms are implemented in Python on a standard Host-PC with an Intel Core i7-8700 Processor and a 3.20 GHz clock rate using Ubuntu 18.04.6 LTS with a real-time kernel. The modeling of minimum-distance computation problem (10) and the MPC problem (33) is done using CasADi [49], and the resulting nonlinear optimization problems are solved by applying the Interior Point OPTimizer (IPOPT) [50] with the linear solver MA57 for the solution of the underlying linear system for step computations. The maximum number of iterations is limited to 100 with an overall acceptable relative convergence tolerance of 10^{-6}.

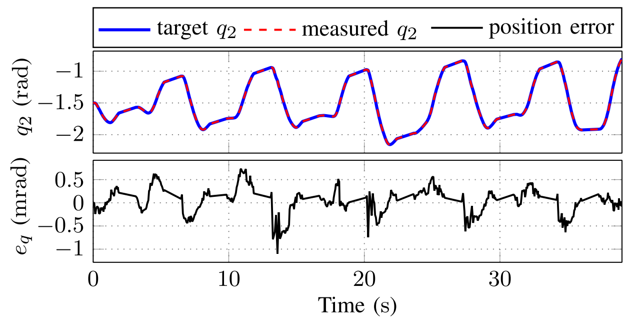

Fig. 7. Comparison between the target and the measured position of robot joint two and depiction of the resulting position tracking error.

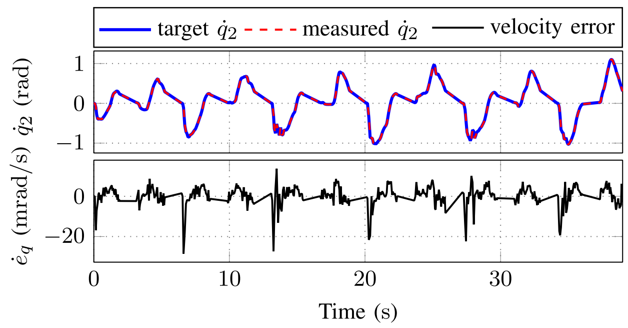

Fig. 8. Comparison between the target and the measured velocity of robot joint two and depiction of the resulting velocity tracking error.

VI. Experimental Results

To evaluate the performance of the proposed algorithm for online planning of collision-free robot trajectories, we conduct experiments considering two test cases with the experimental setup shown in Fig. 1, which results in a robot arrangement with considerable overlap of their operational areas. Test Case 1 is mainly constructed to test the performance of the collision avoidance constraints and does not reflect a normal operating mode of the packing line. Here, we consider 12 objects and two trays with six slots each, distributed as shown in Fig. 9. Test Case 2, on the other side, reflects a rather normal operating mode of the packing line with objects and trays transported by the conveyor belt, see Fig. 1.

A. Test Case 1

In order to compute feasible task points for the robots, scheduling is performed following the idea presented in [13], which consists of assigning a sequence of objects and trays to each robot by minimizing the Euclidean distance covered by the robot end-effectors. The scheduling algorithm is subject to constraints for robot-task assignment and additional constraints to ensure that two simultaneously picked objects do not lie very close to each other and the final robot configurations do not result in overlapping robot geometries, i.e., the tasks points $\mathbf{q}_{fr}$ are feasible. Performing task scheduling results in the task sequence shown in Fig. 10(b), with objects one to six and Tray 1 assigned to Robot 2, and objects seven to 12 along with Tray 2 assigned to Robot 1. To test the performance of the collision avoidance constraints the scheduling algorithm is further constrained resulting in the task sequence shown in Fig. 10(a), where the robot-object assignment remains the same with Robot 1 filling Tray 1 and Robot 2 filling Tray 2.

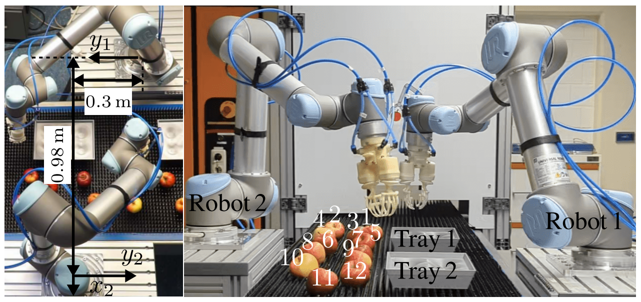

Fig. 9. Experimental setup for Test Case 1: The robots have to pick up six objects each and place them in the trays.

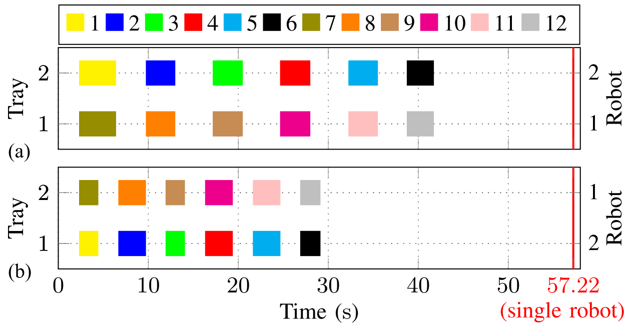

Fig. 10. Test Case 1: Task scheduling for two robots, 12 objects, and two trays. The white space between the colored bars indicates the time needed by the robots to pick up the next objects. (a) Test Case 1a: The objects $\{7, 8, 9, 10, 11, 12\}$ and Tray 1 are assigned to Robot 1, and the objects $\{1, 2, 3, 4, 5, 6\}$ along with Tray 2 to Robot 2, leading to tasks with high collision potential. (b) Test Case 1b: The objects $\{7, 8, 9, 10, 11, 12\}$ and Tray 2 are assigned to Robot 1, and the objects $\{1, 2, 3, 4, 5, 6\}$ along with Tray 1 to Robot 2, leading to tasks with low collision potential. The red vertical line denotes the time it would take for a single robot to perform the pick-and-place tasks.

The assigned object-tray combination leads in this case to overlaps in the paths of the robot end-effectors and thus to motions with high interrobot collision potential.

A visual representation of the experimental results for Test Case 1a is shown in Fig. 11. The scene succession Fig. 11(a)–(e) represents an entire pick-and-place sequence starting from the robot's initial position, picking up the assigned objects 1 and 7, and placing them in the respective slots. On their way to the objects and the respective slots, the robots sidestep each other to avoid collisions, as shown in Fig. 11(b) and (d). After placing the first objects, the robots go to the following assigned objects 2 and 8, according to the schedule shown in Fig. 10(a), and place them in the respective slots of the trays, and so forth. The other images, Fig. 11(f)–(l) represents snapshots of collision avoidance motions of the robots when they are on their way to the objects or slots. This clearly shows that collision avoidance is performed so that the robot arms do not move across each other. Fig. 11(m)–(p) shows the location of the geometry approximating spheres corresponding to the robot configurations in Fig. 11(b), (d), (h), and (l), respectively.

Therefore, we built in MATLAB/Simulink a dynamic simulation model of the experimental setup using Simscape-Multibody and robot CAD data with dynamic parameters identified in [48], and let the robots track position, velocity, and acceleration trajectories, recorded while performing the experiment. The position of the spheres is then visualized using recorded optimal curve parameters λ_1^* and λ_2^*,

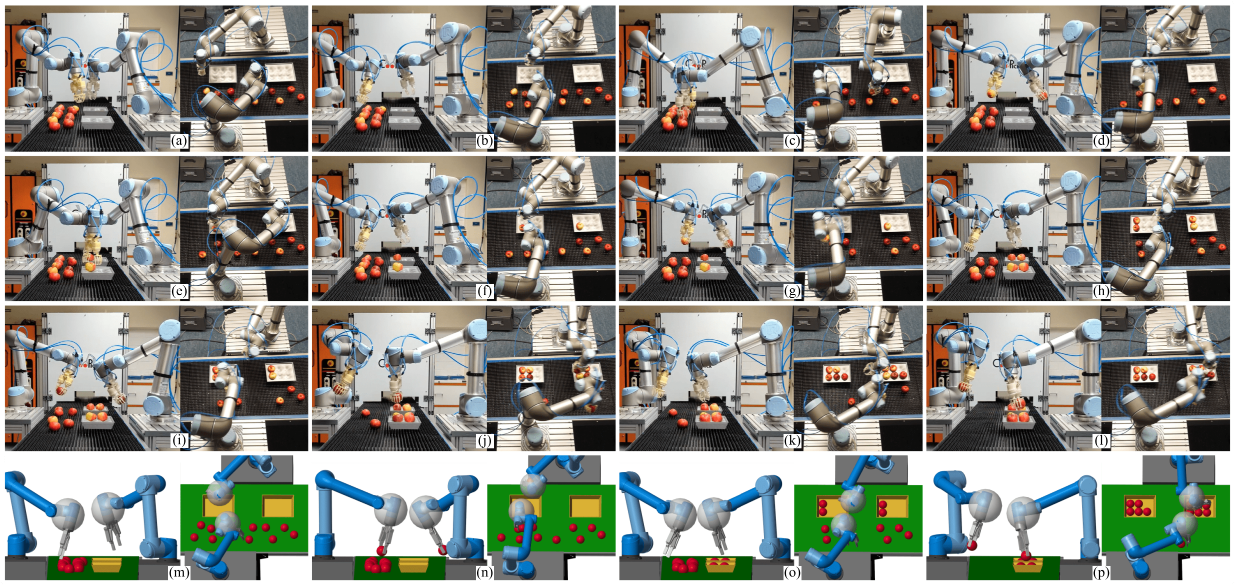

Fig. 11. Pictures show a side and top view of the experiment for Test Case 1a, where the robots perform simultaneous pick-and-place tasks with high collision potential. Scenes (a)–(e) show an entire pick-and-place cycle starting from the start position (a), picking up the assigned objects (c), and placing them in the respective slots (e) (Robot 1 Object 7 in Slot 1 and Robot 2 Object 1 in Slot 7). The robots avoid collisions on the way to the objects and slots, as shown by scenes (c) and (d), respectively. The other snapshots (f) through (l) also show conflict resolution and collision avoidance. The Simulation scenes (m)–(p) display the position of the spheres corresponding to the robot configurations shown in (b), (d), (h), and (l), respectively. A video of the experiment can be found under the following link: https://youtu.be/8qSCSVmFoW0. (a) Initial robot position. (b) Collision avoidance: going to objects 1 & 7. (c) Picking up objects 1 & 2. (d) Collision avoidance: going to slots 1 & 7. (e) Placing objects 1 & 7. (f) Collision avoidance: going to objects 2 & 8. (g) Collision avoidance: going to slots 2 & 8. (h) Collision avoidance: going to objects 3 & 9. (i) Collision avoidance: going to slots 3 & 9. (j) Collision avoidance: going to slots 5 & 11. (k) Collision avoidance: going to objects 6 & 12. (l) Collision avoidance: going to slots 6 & 12. (m) Sphere's position corresponding to b). (n) Sphere's position corresponding to d). (o) Sphere's position corresponding to h). (p) Sphere's position corresponding to l).

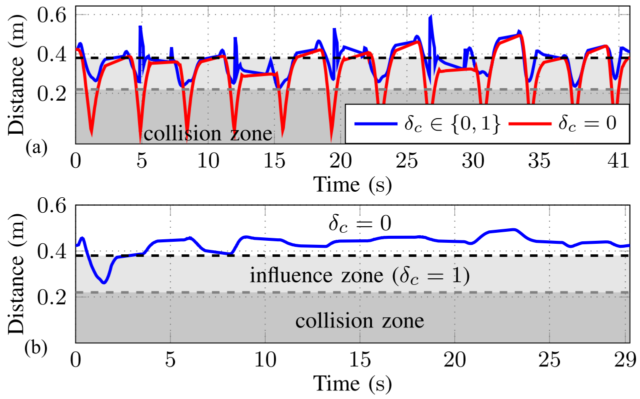

Fig. 12. Distance between the centers of the spheres for Test Case 1. (a) Test Case 1a: For $\delta_c \in \{0, 1\}$, the collision avoidance constraints are active within the influence zone, i.e., $d \leq d_{\text{in}}$, in (31), resulting in no overlapping of the spheres at any time. $\delta_c = 0$ denotes the case where the collision constraints are deactivated. In this case, the distance between the spheres undercuts the safety distance, defined by the sum of the radii of the proximity spheres, which would result in collisions between the robots. (b) Test Case 1b: At the beginning, the interrobot collision avoidance constraints prohibit the minimum distance from undercutting the critical minimum value. After that, the robots perform tasks with low collision potential, which is also reflected in the time course of the minimum distance.

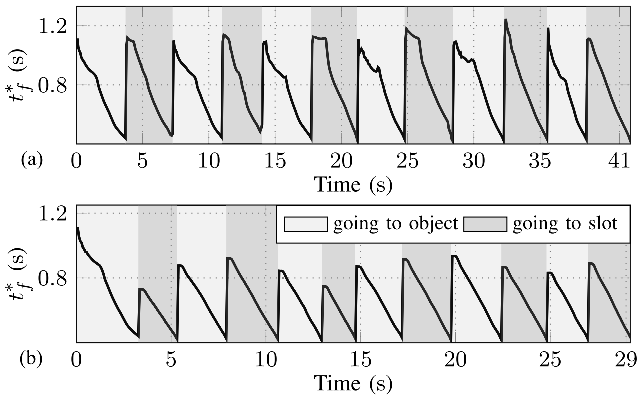

Fig. 13. Computed minimum final time representing the value of the cost function of the optimization problem (33) for Test Case 1a shown in (a) and Test Case 1b shown in (b).

computed during the experiment by solving the optimization problem (10).

The computed minimum distance between the centers of the proximity spheres in Test Case 1 is shown in Fig. 12. The dark gray area marks the region where the spheres overlap, i.e., where the robots would collide. The light gray area denotes the region (influence distance) where the collision avoidance constraints are active when enabled. It is evident that collisions would always occur between the robots without collision conditions. The proposed collision avoidance constraints ensure safe robot operation as they prevent the minimum distance between the robots from falling below a certain safety distance.

To further evaluate the performance of the algorithm, we analyze the computed minimum final time t_f^* shown in Fig. 13 and the joint position, velocity, and acceleration trajectories for Robot 1 shown in Figs. 14 and 15, respectively. The trajectories are shown only for Robot 1 and half of the experiment duration since the trajectories for Robot 2 and the remaining tasks have similar profiles and do not provide additional information. For better readability, we display only the measured joint position and velocity, since, as shown in Figs. 7 and 8, the error between the target and the measured joint position respective velocity is very small.

The robot manipulators share the same cost function, i.e., the same final time, resulting in a synchronous pick-and-place task execution. The computed optimal time shown in Fig. 13

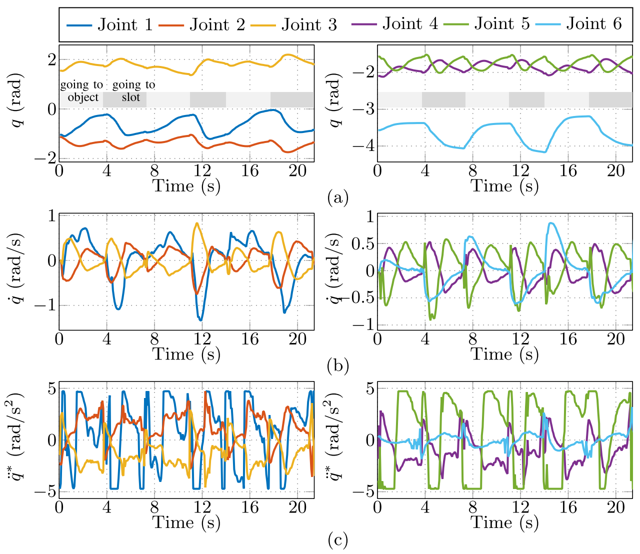

Fig. 14. Measured joint positions, velocities, and computed accelerations for Robot 1 in Test Case 1a. The light and dark gray areas mark the time span when the robot is going to an object, respectively, slot. (a) Measured joint positions. (b) Measured joint velocities. (c) Computed joint accelerations.

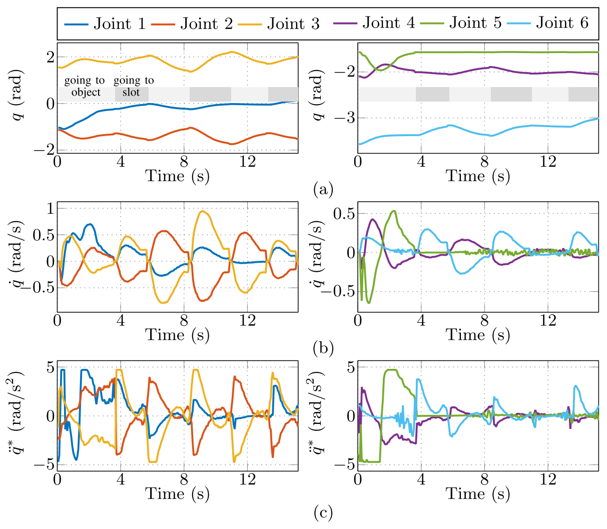

Fig. 15. Measured joint positions, velocities, and computed accelerations for Robot 1 in Test Case 1b. The light and dark gray areas mark the time span when the robot is going to an object, respectively, slot. (a) Measured joint positions. (b) Measured joint velocities. (c) Computed joint accelerations.

has 12 maximum and 12 minimum peaks, representing each the beginning and the end of task execution (picking or placing an object), respectively. This corresponds to picking six objects each and placing them in the respective slots. Accordingly, the minimum time is maximal at the beginning denoting the start of the trajectory planning, and decreases continuously as the robots approach their targets with each sampling step (MPC iteration). The slope of the decreasing cost function varies with time, and at some points, a local increase compared to the previous value can also be observed. This occurs especially in Test Case 1a during a conflict resolution and collision avoidance situation, which causes the robots to take more time to achieve their goals. A local change in the course of the computed minimum time is also reflected in the profile of the minimum distance and the robot trajectories, especially the velocity and acceleration. It can be observed that whenever the slope of the decreasing cost function becomes smaller or locally positive, the distance between the robots usually increases as well. As a reaction to this, there are sometimes sudden changes in the velocity and acceleration in order to avoid collisions, but the robot trajectories remain smooth, see Fig. 14.

In Test Case 1b, starting from the initial position Robot 1 will pick up Object 7, and Robot 2 Object 1. This is the same as in Test Case 1a and corresponds to the image sequence in Fig. 11(a)–(d). However, Robot 1 then places its assigned object in Slot 7 (Tray 2) and Robot 2 in Slot 1 (Tray 1), see Fig. 10(b). Compared to Case 1a, this leads to nonoverlapping in the paths of the robot end-effectors and thus to low collision risk. Due to different scheduling, robots need to avoid each other only during the execution of the first tasks when moving from the initial points to the first objects. This can also be observed in the time course of the minimum distance and the optimal final time shown in Figs. 12(b) and 13(b), respectively. While executing the first task (going to the first objects), the value of the minimum distance enters the influence zone ($d \leq d_{\text{in}}$) activating the constraints and thus prohibiting the minimum distance from falling below the critical value. The computed optimal time for the first task is also higher and the decreasing slope is changing with time. The remaining task points do not lead to robot motions with high collision potential as the robots do not get close to each other. In this case, the robots need less time to reach the assigned goals resulting in shorter transition times and faster overall mission completion. The entire experiment lasts about 29.3 s, which is 12.5 s faster compared to Test Case 1a. Deploying a single robot to execute the same tasks, i.e., fill both trays, results in an overall task execution time of about 57.2 s. Thus, depending on the task planning, two robots can perform the entire task between 15.4 and 27.9 s faster than a single robot.

A correlation between the interrobot collision avoidance constraints and the resulting time evolution of the trajectories in Figs. 14 and 15 is evident. When avoiding collisions, the wrists of the robots are more involved in the trajectory planning, as the spheres are mainly located in the wrist and end-effector area, as shown in Fig. 11. The acceleration profile of at least one robot joint exhibits a time characteristic that is typical for time-optimal trajectory planning. In the trajectory planning problem, the terminal constraints impose that all robot joints reach their target angles simultaneously. Given that, for the considered robots, all joints have the same speed characteristics, the robot joints farthest from their targets or those who are more involved in avoiding collisions determine the time needed by the robots to reach the final configurations, i.e., the optimal value of the final time t_f^*. In Figs. 14 and 15, it is shown that these joints exhibit bang–bang acceleration profiles. While executing the first task in Test Case 1b, the fifth robot joint reacts quickly to avoid collisions, which is also reflected in the velocity and acceleration profile. During the execution of the remaining tasks with low collision potential, the fifth robot joint does not change, keeping the orientation of the gripper perpendicular to the task points. In this case, joints two and three are the dominant ones since they have to cover a larger distance and exhibit higher velocity and acceleration values.

TABLE II
PARAMETERS

Description	Parameter	Value	Unit
Prediction horizon	N_p	10,15,20	
Collision-free horizon	N_c	5,8,10,12,14	
Influence distance	d_{in}	0.38	m
Sphere radius	$r_1 = r_2$	0.11	m
Self collision angular parameter	ε	0.15	
Self collision distance parameter	d_{z1}	0.10	m
Self collision distance parameter	d_{z2}	0.35	m
Self collision distance parameter	d_s	0.18	m
Velocity parameter	ϵ	$0.24\,t_f^*$	m/s
Minimum time	$t_{\min}$	0.2	s
Conveyor speed	v_c	5	cm/s
Joint velocity limits	$\underline{\dot{q}}, \bar{\dot{q}}$	$\mp\pi$	rad/s
Joint acceleration limits	$\underline{u}, \bar{u}$	$\mp 3\pi/2$	rad/s^2
Joint jerk limits	$\underline{\dot{u}}, \bar{\dot{u}}$	∓ 5	rad/s^3

The parameters used during the experiments are listed in Table II. For the presented results, a prediction horizon of $N_p = 10$ is chosen, and the constraints for interrobot collision avoidance (33g) are applied along its first half, i.e., $N_c = 5$. The velocity parameter ϵ is chosen to be updated dynamically using $\epsilon = 0.24\,t_f^*$, depending on the value of the minimum final time t_f^* (cost function) computed in the previous planning iteration. The value of the optimal final time may locally increase during a collision avoidance phase as the robots slow down their motions, leading to a higher time to reach the targets. In this case, a larger ϵ-value is beneficial to resolve a conflict situation. On the contrary, in less critical scenarios with low collision potential, the final time decreases faster, also relaxing the interrobot collision avoidance constraints by reducing ϵ. Considering the shape and size of the robot's links, the radius of the geometry approximating spheres is chosen to be constant and equal for both robots. However, a varying sphere radius can also be considered as a function of the path variable λ.

The choice of the prediction horizons N_p is related to the expected maximum value for the cost function $t_{f,\max}$, which for the considered tasks lies between 0.6 and 1.2 s. In the scaled time τ, the prediction horizon covers the entire period until the final time point and is subdivided into N_p equidistant intervals $\Delta\tau$. The interval boundaries serve as control points, reshaping the trajectories after each MPC iteration. The aim is to achieve a control point distribution at $t_{f,\max}$ in time steps of about 100 ms, i.e.,

$$\Delta t_{\max} = \Delta\tau\, t_{f,\max} = 100\,\text{ms}\,, \quad \text{with} \quad \Delta\tau = \frac{1}{N_p}\,. \tag{34}$$

Considering on average $t_{f,\max} \approx 1$ s, the prediction horizon is chosen as

$$N_p = \lfloor \frac{t_{f,\max}}{\Delta t_{\max}} \rfloor = 10 \tag{35}$$

with $\lfloor\cdot\rfloor$ denoting the integer (floor) operator. Since the time interval $\Delta t = \Delta\tau t_f^*$ is becoming shorter with decreasing t_f^*, the collision-free horizon N_c has to be sufficiently large to ensure collision-free operation along the entire control horizon t_c, which is mainly determined by the computation time. Here, we are referring to the minimum time $t_{\min} = 200$ ms which denotes that for $t_f^* < t_{\min}$ the robots are near their target points (approximately two control horizons far). Therefore, for $t_c \leq \Delta t_{\max}$, it is sufficient to ensure that at least one control horizon is collision-free, i.e., $\Delta\tau t_{\min} N_c \geq \Delta t_{\max}$, yielding

$$N_c \geq \frac{\Delta t_{\max}}{t_{\min}} N_p\,. \tag{36}$$

Choosing $N_c = N_p/2$ satisfies (36) for the used time parameters.

TABLE III
EXPERIMENTAL RESULTS FOR TEST CASE 1

Horizon	$N_p = 10$ $N_c = 5$	$N_p = 15$ $N_c = 8$	$N_p = 20$ $N_c = 10$
max solver time (ms) (MPC / min. distance)	a) 66 / 13 b) 58 / 10	a) 77 / 16 b) 62 / 14	a) 86 / 19 b) 74 / 17
$t^*_{f,\max}$ (s)	a) 1.25 b) 1.12	a) 1.30 b) 1.14	a) 1.27 b) 1.17
Experiment duration (s)	a) 41.8 b) 29.3	a) 42.8 b) 29.7	a) 43.2 b) 29.9
average solver iterations (MPC / min. distance)	a) 13/6 b) 8/3	a) 14/6 b) 8/3	a) 15/6 b) 8/3

We have performed experiments using different values for the prediction horizon N_p, the collision-free N_c horizon, and the velocity parameter ϵ. The results are summarized in Tables III and IV. The computation time for the MPC-based trajectory planning problem (33) and the minimum distance calculation (10) increases with the increased horizon. In Test Case 1a, the computation time is higher since the collision-avoidance constraints significantly influence the computation time due to their complexity and nonlinearity. This can be seen in Fig. 16, where high fluctuations in the MPC computation time are evident. For $N_p = 20$ and $N_c = 10$, the total computation time sometimes exceeds 100 ms being slightly larger than the requested maximum control horizon. In this case, $t_c > \Delta t_{\max}$, so $N_c > N_p/2$ should be chosen to ensure collision-free trajectory planning above $\Delta t_{\max}$ toward the end of the trajectory planning when $t_f^* \approx t_{\min}$. However, increasing N_c will also increase the computation time. In general, toward the end of the trajectory planning the computation time is shorter as the robots are before t_f^* approaches $t_{\min}$ outside the danger zone for interrobot collisions. In Test Case 1b, the computation time remains below 100 ms and, as shown in Fig. 17, reaches its maximum value at the beginning. This is expected since the robots only need to avoid collisions in the first task and subsequently perform tasks with low collision potential. For the other performance parameters, such as the maximum value of the cost function $t^*_{f,\max}$ and the experiment duration, there is not much difference when performing experiments with larger horizons. The velocity parameter ϵ is chosen by trial and error while performing the experiments. A smaller ϵ-value may lead to robots coming closer to each other, resulting in robots requiring more time to resolve conflict situations. A larger ϵ, on the other hand, pushes the robots farther from each other. Consequently, both increasing and decreasing ϵ can result in longer task execution times as shown in Table IV when comparing to the results in Table III for $\epsilon = 0.24 t_f^*$ and $N_p = 15$, $N_c = 8$.

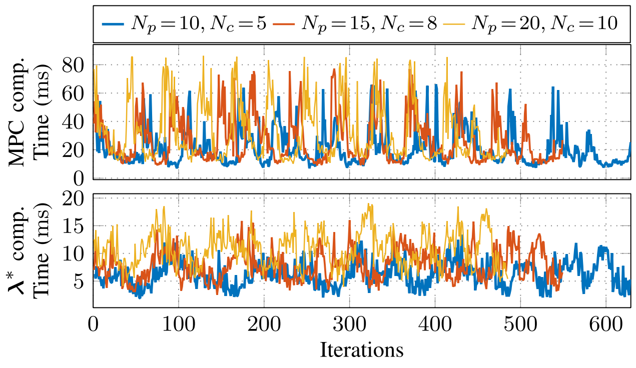

Fig. 16. Computation time for the MPC-based trajectory planning problem and the minimum distance calculation for three different horizon lengths in Test Case 1a.

TABLE IV
EXPERIMENTAL RESULTS FOR TEST CASE 1a, WITH N_p= 15, N_c= 8

Velocity parameter ϵ	$0.10\,t_f^*$	$0.18\,t_f^*$	$0.36\,t_f^*$
max solver time (ms) (MPC / min. distance)	76 / 16	73 / 17	73 / 24
$t^*_{f,\max}$ (s)	1.20	1.22	1.22
Experiment duration (s)	45.6	44.42	44.8
average solver iterations (MPC / min. distance)	14/6	14/6	14/6

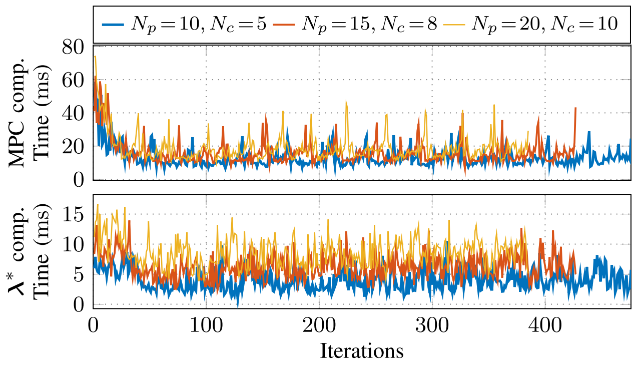

Fig. 17. Computation time for the MPC-based trajectory planning problem and the minimum distance calculation for three different horizon lengths in Test Case 1b.

To analyze the effect of N_c, we performed experiments with a fixed prediction horizon $N_p = 15$ and $N_c \in \{10, 12, 14\}$. The experimental results are summarized in Table V for the more critical Test Case 1a. Increasing N_c increases the number of collision avoidance constraints, resulting in higher computation times. For $N_c \geq 12$ the total computation time is above 100 ms. With the selected values for N_c also toward the end of the trajectory planning, the collision-free horizon is longer than the maximum computation time, i.e., $\Delta\tau t_{\min} N_c \geq t_c$ holds. A longer horizon N_c does not affect the obtained maximum value of the cost function $t^*_{f,\max}$. However, we see that with increased N_c the experiment duration increases by about 1–2 s. Applying the collision avoidance constraints along a longer horizon decreases the slope and the convergence rate of t_f^*, leading to slightly longer overall task execution times.

When collisions are not a concern, the interior point solver requires only a few iterations to converge to a feasible solution. The number of iterations increases if interrobot collisions pose an issue. In this case, the solver requires, on average, 14 iterations to converge to a feasible solution for the trajectory planning problem and six iterations for the minimum distance calculation.

TABLE V
EXPERIMENTAL RESULTS FOR TEST CASE 1a AND N_p= 15

Horizon	$N_c = 10$	$N_c = 12$	$N_c = 14$
max solver time (ms) (MPC / min. distance)	80 / 19	85 / 22	94 / 24
$t^*_{f,\max}$ (s)	1.25	1.25	1.26
Experiment duration (s)	44.1	45.6	46.1
average solver iterations (MPC / min. distance)	14/6	15/6	14/6

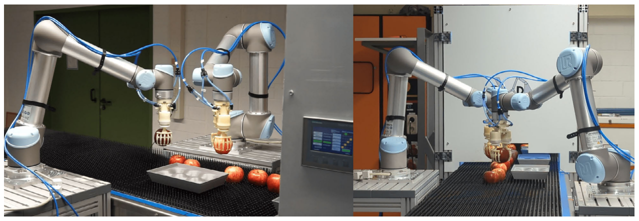

Fig. 18. Test Case 2: Pick-and-place tasks on a conveyor belt. A video of the experiment can be found under the following link: https://youtu.be/07viYkl-v1Y.

B. Test Case 2

The proposed control structure addresses the integration of robotic manipulators into existing manual sorting or packing processes. Therefore, with Test Case 2, we aim to demonstrate an application similar to such processes and let the robots perform pick-and-place tasks with objects transported by a conveyor belt, see Fig. 18. During the experiment, the robots fill two trays each, corresponding to picking up a total of 24 objects and placing them in the slots of the trays. The estimation of the pose of the objects is done by applying the deep learning algorithm presented in [51]. For the pick-and-place task execution, scheduling is performed first to assign a tray and a sequence of objects to a robot. As mentioned, scheduling is performed in the Cartesian space by minimizing the Euclidean distance covered by the robot end-effectors. Since objects and trays are transported by a conveyor belt, new task points are constantly added, requiring an iterative task scheduling after an object is placed in its respective slot.

Moving task points requires continuously updating the desired final robot state $\mathbf{x}_f(k)$ in (33d). Let ${}_0\mathbf{p} \in \{{}_0\mathbf{p}_o, {}_0\mathbf{p}_s\}$ denote the position of a task point and ${}_0\mathbf{p}(k)$ its current from the camera system captured position value relative to the inertial frame. The final target point position is then continuously updated using the prediction

$$ {}_0\mathbf{p}_f(k) = {}_0\mathbf{p}(k) + {}_0\mathbf{v}_c(k)\, t_f^*(1 - \Delta\tau) \quad (37) $$

with ${}_0\mathbf{v}_c(k) = [0, -v_c(k), 0]^\top$ representing the current measured speed vector of the conveyor relative to the inertial frame. Here, $t_f^*(1 - \Delta\tau)$ models the remaining time period the robots need to reach the assigned targets. Applying inverse kinematics and inverse differential kinematics to (37) and ${}_0\mathbf{v}_c$, along with a desired end-effector orientation, the final position and velocities of the robot joints are calculated to update the desired final state vector $\mathbf{x}_f(k)$.

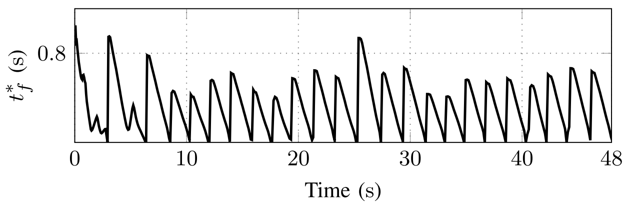

Fig. 19. Computed minimum final time for Test Case 2 representing the value of the cost function of the optimization problem (33).

TABLE VI
EXPERIMENTAL RESULTS FOR TEST CASE 2

Horizon	$N_p = 10$ $N_c = 5$	$N_p = 15$ $N_c = 8$	$N_p = 20$ $N_c = 10$
max solver time (ms) (MPC / min. distance)	53 / 11	64 / 15	74 / 18
max t_f^* (s)	0.93	0.97	1.04
Experiment duration (s)	48.1	48.4	48.6
average solver iterations (MPC / min. distance)	8 / 3	8 / 3	8 / 3

From the performance of the task execution, Test Case 2 is similar to Test Case 1b, as the robots usually perform tasks with low collision potential. This can be also seen in the course of the computed final time t_f^* in Fig. 19 and the experimental results summarized in Table VI. A local increase of the optimal final time t_f^* toward the end of the execution of the first two tasks is related to the position prediction (37) of the task points. More precisely, this is due to the changes in the belt speed, which are higher at the beginning due to friction effects and decrease with time, so that the speed converges to a steady state and maintains the desired value of 5 cm/s nearly constant. The trajectories of the robots are not explicitly shown because they are similar to those in Fig. 15. The computation time shown in Fig. 20 for the entire experiment duration and three different horizon lengths never exceeds 100 ms.

The entire experiment with two robots lasts about 48.1 s. We performed the same experiment deploying a single robot manipulator to fill four trays, resulting in an overall experiment duration of about 84.2 s. The double-arm solution being 36.1 s faster than the single-robot-arm solution is a promising indicator that efficient deployment of multiple robot arms can result in an improved performance in terms of robot cycle time and overall working throughout.

VII. Discussion and Conclusion

As an optimization-based planner, the performance of the proposed approach relies besides the problem formulation itself on the chosen parameters of the algorithm and the numerical solver. Using a longer prediction horizon is beneficial for trajectory planning in terms of increased time resolution. However, this requires a longer collision-checking horizon, increasing the dimension of the optimization problem and, thus, the computation time. A longer computation time is reflected in increased control horizon and overall robot task execution time as it reduces the convergence rate of the cost function.

Overall, the solver converges within a few iterations toward a feasible solution, but the iteration evaluation is computationally costly. The number of iterations varies between 6–40 for the MPC problem and 2–12 for the minimum distance computation. The experimental results obtained with different horizon lengths show that, although the average number of iterations does not considerably increase with the horizon, the computation time increases. This comes from the collision avoidance formulation as nonlinear state-dependent constraints but also from the terminal constraints. The exact satisfaction of the equality constraints at the end of the prediction horizon may require high computational effort.

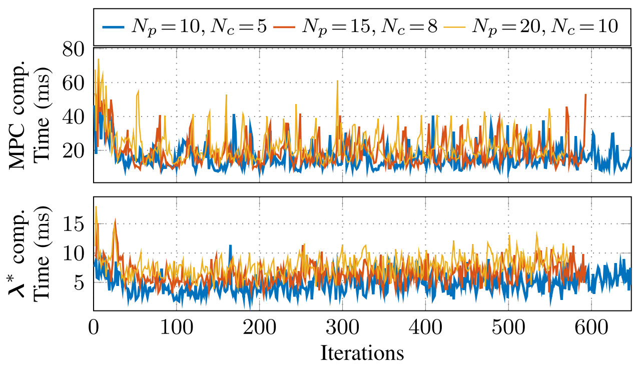

Fig. 20. Computation time for the MPC-based trajectory planning problem and the minimum distance calculation for three different horizon lengths in Test Case 2.

The success rate of the planner depends mainly on the difficulty of the tasks regarding interrobot collisions, the horizon lengths, and the velocity parameter ϵ, which directly influences the collision avoidance constraints. Limiting the number of solver iterations to 100, we performed experiments considering the execution of 100 randomly chosen pick-and-place tasks similar to Test Case 1a in terms of difficulty level. With ϵ varying between $0.1t_f^*$ to $0.35t_f^*$, for $N_p = 15$ and $N_c = 8$ the success planning rate varies from 86% at worst to 94% at best. In all cases where the solver fails to find an optimal solution, the maximum number of iterations is reached, so the trajectory execution is aborted. For smaller ϵ, this is usually the case when the robots get very close to each other. Consequently, the robot's speed is very low, and the solver has difficulties resolving the conflict situation within a time span applicable for online trajectory planning. Contrary, a larger ϵ might bring the robots outside the influence zone defined by (31), deactivating the collision-avoidance constraints. Reentering the influence zone generates repulsive forces pushing the robots again outside the influence zone. Here, the trajectory planning results in high-velocity motions with robots leaving and entering the influence zone without being able to pass each other. The solver always converges to a feasible solution within a few iterations for pick-and-place tasks with low interrobot collision risks since the trajectory planning problem, in this case, is simple.

Prior to developing the geometry approximation using Bézier curves, we performed simulations approximating the robot geometry by circles and ellipsoids. We applied velocity constraints in the form of virtual velocity dampers to avoid intersections between the approximating convex primitives. Velocity constraints were used since, for the proposed time-optimal planning, they have the advantage of the time to target t_f, a subject of minimization, inherently appearing in the constraints due to time scaling. Even after reducing the number of the approximating convex geometries to a minimum by using three ellipsoids for the shoulder, elbow, and robot wrist three, including the gripper, and a circle in the area of wrists one and two, the achieved computation times were not suitable for real-time implementation in an experimental setup as considered in this article.

In the effort to generate time-optimal and real-time robot trajectories for robotic arms performing pick-and-place tasks in a shared work environment, a continuous approximation of the robot geometry by Bézier curves is introduced at the cost of a coarser geometry approximation by moving proximity spheres, without compromising the safe robot use. To this end, an efficient collision avoidance approach has been formulated by defining velocity restrictions along tangent separating planes as inequality constraints of an MPC problem. Thereby, the underlying predictive character of the algorithms is of eminent interest as potential collisions can be timely anticipated and, thus, avoided in advance. This fact underpins the key role and necessity of predictive control strategy in cooperative robotics. From the implementation perspective, however, this invokes online real-time calculations of trajectory adaptations, which also has been successfully demonstrated in this work using an experimental setup with two robotic arms performing pick-and-place tasks on a conveyor belt. For the considered application, we have shown that using two robot arms in a tight shared workspace leads, even in the worst case, to an improvement in cycle times compared to deploying a single robot to execute the same tasks. This improvement is greater through coordinated task and trajectory planning, enabling a more efficient robot operation. Future work will include a monolithic integration of task scheduling within the framework of a mixed-integer predictive algorithm and its extension to address asynchronous robot task execution.

Appendix

A. Time Discretization of the Prediction Model

The solution of the linear time-invariant system (4) is given in the form

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

by successively applying Picard's iteration method with the state-transition matrix

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

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

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

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

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{\Phi}(\Delta\tau)\mathbf{x}(k)\\ &\quad + \big(\Delta t_f\big)^2 \int_{k\Delta\tau}^{(k+1)\Delta\tau} \mathbf{\Phi}\big((k+1)\Delta\tau - \tilde{\tau}\big)\mathbf{B}\mathbf{u}\big(t_0 + \tilde{\tau}\Delta t_f\big)\mathrm{d}\tilde{\tau}\\ &= \mathbf{\Phi}(\Delta\tau)\mathbf{x}(k)\\ &\quad + \Delta t_f \int_{t_0+k\Delta\tau\Delta t_f}^{t_0+(k+1)\Delta\tau\Delta t_f} \mathbf{\Phi}\big((k+1)\Delta\tau - \big(\tilde{t} - t_0\big)/\Delta t_f\big)\mathbf{B}\mathbf{u}(k)\mathrm{d}\tilde{t}\end{aligned}$$

yielding the discrete time dynamics (29).

References

[1] R. Shome, K. Solovey, J. Yu, K. Bekris, and D. Halperin, "Fast, high-quality two-arm rearrangement in synchronous, monotone tabletop setups," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 3, pp. 888–901, Jul. 2021.

[2] H. Ha, J. Xu, and S. Song, "Learning a decentralized multi-arm motion planner," in *Proc. Conf. Robot. Learn. (CoRL)*, 2020, pp. 1–12.

[3] S. Mirrazavi, N. Figueroa, and A. Billard, "Coordinated multi-arm motion planning: Reaching for moving objects in the face of uncertainty," in *Robotics: Science and Systems*. Ann Arbor, MI, USA: Robotics: Science and Systems XII, 2016.

[4] Y. Gan, J. Duan, M. Chen, and X. Dai, "Multi-robot trajectory planning and position/force coordination control in complex welding tasks," *Appl. Sci.*, vol. 9, no. 5, p. 924, Mar. 2019.

[5] J. Chen et al., "Cooperative task and motion planning for multi-arm assembly systems," 2022, *arXiv:2203.02475*.

[6] M. Wagner, P. Heß, S. Reitelshöfer, and J. Franke, "Cooperative processing with multi-robot systems," in *Proc. IEEE Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2017, pp. 663–669.

[7] C. Wong, S. Shackleford, D. Potter, J.-P. Richardson, L. McDermott, and J. Nolan, "Robotic task sequencing and motion coordination for multiarm systems," *IEEE/ASME Trans. Mechatronics*, vol. 27, no. 6, pp. 5275–5286, Dec. 2022.

[8] S. Zhang and F. Pecora, "Online sequential task assignment with execution uncertainties for multiple robot manipulators," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 6993–7000, Oct. 2021.

[9] H. J. Bückenhüskes and G. Oppenhäuser, "DLG trend report: Robots in the food and beverage industry," *DLG Lebensmittel*, vol. 9, no. 6, pp. 16–17, 2014.

[10] S. D. Han, S. W. Feng, and J. Yu, "Toward fast and optimal robotic pick-and-place on a moving conveyor," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 446–453, Apr. 2020.

[11] S. Berger and B. Armstrong, "The puzzle of the missing robots," *MIT Case Studies Social Ethical Responsibilities Comput.*, Jan. 2022. [Online]. Available: https://mit-serc.pubpub.org/pub/puzzle-of-missing-robots

[12] L. Sanneman, C. Fourie, and J. A. Shah, "The state of industrial robotics: Emerging technologies, challenges, and key research directions," *Found. Trends® Robot.*, vol. 8, no. 3, pp. 225–306, 2021.

[13] 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)*, Las Vegas, NV, USA, Oct. 2020, pp. 7675–7681.

[14] A. Tika and N. Bajcinca, "Model predictive control based cooperative robot manipulation and collision avoidance in shared workspaces," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2021, pp. 702–709.

[15] S. M. Lavalle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*. Wellesley, MA, USA: A K Peters, 2001, pp. 293–308.

[16] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Mar. 1996.

[17] 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.

[18] 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.

[19] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2002, pp. 624–631.

[20] D.-H. Lee, D.-H. Kim, J. Y. Lee, and C.-S. Han, "Collision-free coordination of two dual-arm robots with assembly precedence constraint," in *Proc. 14th Int. Conf. Control, Autom. Syst. (ICCAS)*, Oct. 2014, pp. 515–520.

[21] Y.-C. Tsai and H.-P. Huang, "Motion planning of a dual-arm mobile robot in the configuration-time space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 2458–2463.

[22] M. Otte and E. Frazzoli, "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning," *Int. J. Robot. Res.*, vol. 35, no. 7, pp. 797–822, Jun. 2016.

[23] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *Int. J. Robot. Res.*, vol. 21, no. 12, pp. 999–1030, Dec. 2002.

[24] A. Völz and K. Graichen, "Composition of dynamic roadmaps for dual-arm motion planning," in *Proc. IEEE Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2017, pp. 1242–1248.

[25] A. Völz and K. Graichen, "A predictive path-following controller for continuous replanning with dynamic roadmaps," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3963–3970, Oct. 2019.

[26] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1985, pp. 500–505.

[27] C. C. B. Viturino, U. de Melo Pinto Junior, A. G. S. Conceição, and L. Schnitman, "Adaptive artificial potential fields with orientation control applied to robotic manipulators," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9924–9929, 2020.

[28] P. Tournassoud, "A strategy for obstacle avoidance and its application to multi-robot systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, Apr. 1986, pp. 1224–1229.

[29] P. Tournassoud, "On motion coordination," INRIA, Rennes, France, Tech. Rep., RR-0549, Jul. 1986.

[30] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *Proc. IEEE Int. Conf. Robot. Autom.*, Oct. 1987, pp. 1152–1159.

[31] P. Bosscher and D. Hedman, "Real-time collision avoidance algorithm for robotic manipulators," in *Proc. IEEE Int. Conf. Technol. Practical Robot Appl.*, Woburn, MA, USA, Nov. 2009, pp. 113–122.

[32] R. Ni, Z. Pan, and X. Gao, "Robust multi-robot trajectory optimization using alternating direction method of multiplier," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 5950–5957, Jul. 2022.

[33] F. Kennel-Maushart, R. Poranne, and S. Coros, "Manipulability optimization for multi-arm teleoperation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 3956–3962.

[34] M. Zucker et al., "CHOMP: Covariant Hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, nos. 9–10, pp. 1164–1193, Aug. 2013.

[35] J. Schulman et al., "Motion planning with sequential convex optimization and convex collision checking," *Int. J. Robot. Res.*, vol. 33, no. 9, pp. 1251–1270, Aug. 2014.

[36] 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.

[37] S. A. Homsi, "Online generation of time-optimal trajectories for industrial robots in dynamic environments," M.S. thesis, Université Grenoble Alpes, Grenoble, France, Mar. 2016.

[38] C. Zhou, M. Lei, L. Zhao, Z. Wang, and Y. Zheng, "TOPP-MPC-based dual-arm dynamic collaborative manipulation for multi-object nonprehensile transportation," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2022, pp. 999–1005.

[39] N. Dehio, Y. Wang, and A. Kheddar, "Dual-arm box grabbing with impact-aware MPC utilizing soft deformable end-effector pads," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 5647–5654, Apr. 2022.

[40] P. Bézier, *The Mathematical Basis of the UNISURF CAD System*. London, U.K.: Butterworth, 1986.

[41] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Dynamics and Control*. Hoboken, NJ, USA: Wiley, 2006.

[42] J.-W. Chang, Y.-K. Choi, M.-S. Kim, and W. Wang, "Computation of the minimum distance between two Bézier curves/surfaces," *Comput. Graph.*, vol. 35, no. 3, pp. 677–684, Jun. 2011.

[43] N. M. Patrikalakis and T. Maekawa, *Shape Interrogation for Computer Aided Design and Manufacturing*. 1st ed. Cham, Switzerland: Springer, 2009.

[44] D. E. Johnson and E. Cohen, "A framework for efficient minimum distance computations," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1998, pp. 3678–3684.

[45] 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. 4, no. 2, pp. 193–203, Apr. 1988.

[46] *Robotic Operating System*, Stanford Artif. Intell. Lab., Stanford, CA, USA, 2018.

[47] *RTDE: Real-Time Data Exchange*, Universal Robots, Odense, Denmark, 2019.

[48] A. Tika, J. Ulmen, and N. Bajcinca, "Dynamic parameter estimation utilizing optimized trajectories," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Aug. 2020, pp. 7300–7307.

[49] 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.

[50] A. Wächter and L. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, pp. 25–57, Mar. 2006, doi: 10.1007/s10107-004-0559-y.

[51] L. A. Giefer, J. D. Arango Castellanos, M. M. Babr, and M. Freitag, "Deep learning-based pose estimation of apples for inspection in logistic centers using single-perspective imaging," *Processes*, vol. 7, no. 7, p. 424, Jul. 2019.

Argtim Tika received the Diploma degree (Dipl.-Ing.) in automation and control from the Faculty of Electrical Engineering and Information Technology, Technische Universität Wien (TU WIEN), Vienna, Austria, in 2016.

Since 2016, he has been a Research Assistant at the Department of Mechanical and Process Engineering, Rheinland-Pfälzische Technische Universität (RPTU) Kaiserslautern-Landau, Kaiserslautern, Germany. His current research interests include modeling, optimization, and control of robotic systems.

Naim Bajcinca graduated in theoretical physics and electrical engineering from the University of Prishtina, Prishtina, Kosova. He received the Ph.D. degree in robust control from the Technical University of Berlin (Institute of Robotics and Mechatronics), Berlin, Germany, and the German Aerospace Research Center (DLR), Oberpfaffenhofen, Germany, in 2006.

He worked as a Research Associate with the Max-Planck Institute for Dynamics of Complex Technical Systems, Magdeburg, Germany, prior to accepting a Full Professor position at the Department of Mechanical and Process Engineering, RPTU Kaiserslautern-Landau, Kaiserslautern, Germany. His research interests in control theory include stability analysis and control design in robust and optimal control, hybrid dynamical systems, networked control systems, stochastic control, and data-driven and learning-based control. His work on applied control comprises various domains of engineering and life sciences, including robotics, human–machine interaction, embedded and cyber–physical systems, autonomous systems, power systems, production systems, process engineering, and systems biology.