2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Synchronous Minimum-Time Cooperative Manipulation using Distributed Model Predictive Control

Argtim Tika[1] and Naim Bajcinca[2]

***Abstract*— A hierarchical algorithm involving two-layer optimization-based control policies with varying degrees of abstraction is proposed, including upper layer task scheduling and lower layer local path planning. A scenario with two robot arms performing cooperative pick-and-place tasks for moving objects is specifically addressed. The main focus of the paper lies on the bottom layer of the hierarchical control scheme, more precisely on the online generation of the synchronous robot trajectories using distributed minimum-time model predictive control (DMPC) algorithms. To this end, we introduce a decelerating coupling term in the cost functions of the individual distributed optimization algorithms to synchronize the overall robot motion. The performance of the algorithm is illustrated by extensive simulations with high-fidelity robot dynamic models.**

I. INTRODUCTION

The flexible and scalable use of multiple robot manipulators on packaging or production lines is continuously increasing. Due to the constantly growing space requirements and the high diversity of products and goods, more and more robots are used which share a common workspace and perform cooperative operations. Especially cooperative pick-and-place tasks involving multiple objects are frequently encountered. The aim is to optimally perform these repetitive tasks by using robots in order to maximize the throughput and reduce costs. One way of addressing this problem is by splitting it into the tasks of scheduling and trajectory planning. The research on robot task scheduling has been primarily focused on minimizing the cycle time by determining the optimal sequence of a set of unordered static task points in the 3D space. Similar problems are mainly modeled as an extension to the traveling salesman problem (TSP) and rely on approximate methods like genetic algorithms (GA), see [1]. In [2] a GA based method is introduced, which is adapted to take into account the multiple solutions of the inverse kinematics. [3] extends this idea to the case of a two-robot work cell. Another idea involving two robots sharing a common workspace is presented in [4]. Timed Petri nets and the uniform cell decomposition approach are used to model the robot task allocation and ensure collision-free operation.

The online point-to-point trajectory planning is a problem that is often encountered in the field of robotics. For this purpose, planning algorithms based on model prediction control (MPC) methods are increasingly used, especially in research projects. [5] proposes an MPC based approach for point-to-point trajectory generation, in which the deviation of states and control inputs are penalized. Another MPC approach is proposed in [6] in combination with an integral sliding mode control. In [7] the generation of a collision-free trajectory is formulated as a parameter optimization problem. A hierarchical control scheme for optimal task allocation and minimum-time trajectory planning involving two robots is proposed in [8]. The scheduling task leads to an integer optimization problem with linear constraints and a bilinear cost function. For the point-to-point trajectory planning a centralized MPC algorithm is introduced. Within this paper, the MPC-based planning algorithm is arranged in a distributed manner, leading to distributed model predictive control (DMPC) algorithms with coupled cost functions and constraints. The used DMPC approach offers certain advantages with regard to a scalable and flexible robot deployment. DMPC methods have already been successfully used several times for point-to-point trajectory generation of multi-agent systems, such as in the area of UAVs in [9] and [10] or in vehicle platoons in [11]. However, to the best of our knowledge, this paper presents the first results on the online synchronous trajectory planning for robot manipulators using distributed model predictive control.

This paper is organized as follows: The problem formulation along with the proposed hierarchical control architecture is addressed in Section II. Section III describes the modelling of the robot dynamics and the scheduling tasks. The synchronous distributed minimum-time MPC is presented in Section IV, followed by the simulation results in Section V. The paper is finalized by brief concluding remarks in Section VI.

II. PROBLEM FORMULATION AND CONTROL CONCEPTS

In a narrow workspace with several closely operating robots performing cooperative pick-and-place tasks, the coordinated workflow should be organized in such a way that the robots perform their tasks optimally. Therefore, two main tasks are of fundamental importance. Firstly, a resource allocation problem is raised by questioning how the task allocation for the robots is going to take place. Secondly, in order to accomplish the assigned time-depending tasks, an online trajectory planning should be employed. A highly non-convex mixed-integer nonlinear programming problem would be the result if the discrete scheduling decisions and the dynamics of the robots were to be included in a monolithic optimization framework, which would severely limit its practical online applicability. In order to cope with these difficulties, a hybrid control algorithm, as shown in Fig. 1, is employed, consisting of two layers. The upper

[1]Argtim Tika and [2]Naim Bajcinca are with the Department of Mechanical and Process Engineering, Technische Universität Kaiserslautern, Germany, argtim.tika@mv.uni-kl.de, naim.bajcinca@mv.uni-kl.de

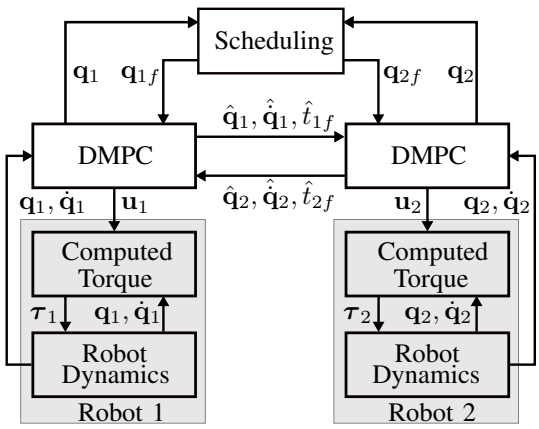


Fig. 1. Schematic illustration of the used control concept.

layer solves a scheduling problem, while on the lower layer local DMPC planning algorithms are executed. The scheduler computes the optimal task allocation by minimizing the total traversed distance. It provides the position set-points to the underlying DMPC layer, which computes the robots' optimal trajectories by minimizing the execution time of the tasks.

For the considered application, the authors presented in the previous work [8] a similar control architecture with a centralized MPC layer resulting in synchronous robot task execution. While cooperatively executing pick-and-place tasks, a synchronous robot operation appears to be useful in the coordination of the robot movements. On the one hand, this simplifies the scheduling problem, since the rescheduling for both robots takes place simultaneously and thus less frequently. On the other hand, robot coordination and collision avoidance are less elaborate and require less computing power compared to an unsynchronized approach. Of course, synchronization also entails a loss of performance, since one of the robots may work slower than it eventually could. Compared to the centralized MPC approach, time synchronization is not inherently given in the distributed case, since each robot plans its trajectory locally. Therefore, coupling in the cost functions of the distributed optimization algorithms is introduced to synchronize the robot movements in order to reach their targets simultaneously. Additionally, coupling in the constraints is introduced to impose collision avoiding restrictions. This leads to DMPC algorithms of decoupled systems with coupling in the cost functions and the constraints. For each MPC iteration, the controllers share the calculated minimum time and their predicted trajectories.

III. Modelling

A. Robot dynamic model

The dynamic model of the considered robot manipulator is derived by means of the Lagrange formalism [12]. The robot equation of motion in matrix form can be written as

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

where $\boldsymbol{\tau} \in \mathbb{R}^n$ denotes the vector of generalized torques, $\mathbf{q} \in \mathbb{R}^n$ denotes the vector of generalized coordinates (i.e. the joint angular position), $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\mathbf{C}(\mathbf{q},\dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ represents the coefficients of the centrifugal (proportional to $\dot{q}_i^2$) and Coriolis (proportional to $\dot{q}_i\dot{q}_j, i \neq j$) forces, $\mathbf{g}(\mathbf{q}) \in \mathbb{R}^n$ is the vector of gravitational torques and $\boldsymbol{\tau}_R(\dot{\mathbf{q}}) \in \mathbb{R}^{n\times n}$ the vector of Coulomb and viscous friction torques. For the considered robots $n = 6$ holds. Additionally, a dynamic simulation model in SIMSCAPE has been developed using the CAD data. The SIMSCAPE model is mainly used for the simulation of the forward dynamics, whereas the analytically derived equations of motion are used in feedback control to implement a nonlinear dynamic inversion based controller $\boldsymbol{\tau} = \mathbf{f}(\mathbf{q},\dot{\mathbf{q}},t)$ in order to compensate the nonlinear dynamics. The nonlinear feedback law is chosen such that, the combined control system results in a linear closed loop system of n double integrators $\ddot{\mathbf{q}} = \mathbf{u}$, with the new input $\mathbf{u} \in \mathbb{R}^n$. The resulting model

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}\,, \quad \text{with}$$
$$\mathbf{x} = \begin{bmatrix}\mathbf{q}\\ \dot{\mathbf{q}}\end{bmatrix},\ \mathbf{A} = \begin{bmatrix}\mathbf{0}^{n\times n} & \mathbf{I}^{n\times n}\\ \mathbf{0}^{n\times n} & \mathbf{0}^{n\times n}\end{bmatrix} \text{ and } \mathbf{B} = \begin{bmatrix}\mathbf{0}^{n\times n}\\ \mathbf{I}^{n\times n}\end{bmatrix}, \qquad (2)$$

is used in the distributed MPC algorithms for cooperative generation of collision-free robot trajectories.

B. Modelling of the scheduling task

The goal of the scheduling layer is to allocate a set of objects $n \in \mathcal{N} = \{1,\cdots,N\}$ to a set of slots $s \in \mathcal{S} = \{1,\cdots,S\}$ while minimizing the total distance covered by the robots $r \in \mathcal{R} = \{1,\cdots,R\}$ to execute all necessary tasks. It is assumed that the number of slots is smaller than the number of objects $S \leq N$, and they are uniformly distributed on a set of classes $p \in \mathcal{P} = \{1,\cdots,P\}$, i.e. $S_1 = \{1,\cdots,S/P\}$ and $S_2 = \{S/P+1,\cdots,2\,S/P\}$ denote the set of slots located to class 1 and 2 respectively. Hence, it is necessary that both S and P are even numbers. To solve the scheduling problem, the optimization algorithm presented in [8] is used. The algorithm is based on the well established travelling salesman problem (TSP), and includes the binary variables $\mathbf{z}_b = \left[\mathbf{w}_b^T\ \mathbf{x}_b^T\right]^T$, with the binary robot to class

$$\mathbf{w}_b = [W_{11}\cdots W_{1P}\,W_{21}\cdots W_{RP}]^T\,, \qquad (3)$$

and object to slot allocation variables

$$\mathbf{x}_b = [X_{11}\cdots X_{1S}\,X_{21}\cdots X_{NS}]^T\,. \qquad (4)$$

In the following, the constraints and the cost function of the discrete optimization problem described in [8] are presented in matrix notation without going into much detail, and thus transformed into the standard form of a static optimization problem.

The robot-class assignment must ensure that each robot serves a single class and each class can only be served by a single robot. This can be expressed by the following $P+R$ equality constraints

$$\sum_{\forall r\in\mathcal{R}} W_{rp} = 1,\ \forall p \in \mathcal{P}\,, \quad \sum_{\forall p\in\mathcal{P}} W_{rp} = 1,\ \forall r \in \mathcal{R}\,, \qquad (5)$$

which can be expressed as

$$\left[\left[\mathbf{I}^{P\times P} \oplus \mathbf{1}^{1\times R}\right]\ \mathbf{0}^{P\times N\cdot S}\right]\mathbf{z}_b = \mathbf{1}^{P\times 1} \qquad (6)$$
$$\left[\left[\mathbf{1}^{1\times P} \oplus \mathbf{I}^{R\times R}\right]\ \mathbf{0}^{R\times N\cdot S}\right]\mathbf{z}_b = \mathbf{1}^{R\times 1}\,. \qquad (7)$$

Each slot in the classes has to be filled with exactly one object, but not necessarily all objects can be placed in the slots since the number of objects is assumed to be larger than the number of available slots. The first condition can be modeled by S equality and the second one by N inequality constraints

$$\sum_{\forall n \in \mathcal{N}} X_{ns} = 1,\ \forall s \in \mathcal{S}\,, \sum_{\forall s \in \mathcal{S}} X_{ns} \leq 1,\ \forall n \in \mathcal{N}\,, \quad (8)$$

leading to

$$\left[\mathbf{0}^{S\times R\cdot P}\ \left[\mathbf{I}^{S\times S} \oplus \mathbf{1}^{1\times N}\right]\right]\mathbf{z}_b = \mathbf{1}^{S\times 1} \quad (9)$$

$$\left[\mathbf{0}^{N\times R\cdot P}\ \left[\mathbf{1}^{1\times S} \oplus \mathbf{I}^{N\times N}\right]\right]\mathbf{z}_b \leq \mathbf{1}^{N\times 1}\,. \quad (10)$$

The proposed synchronous approach to accomplish pick-and-place tasks involving dynamic objects implies that the robots simultaneously pick-and-place the objects. The sequence in which the slots are filled is fixed such that the robots maintain a safety distance to each other while placing the objects. However, when picking up the objects, an additional constraint should be imposed to ensure that a similar safety distance $d_{\min}$ is maintained. This can be enforced by the following constraint

$$d_{nn'} + (2 - X_{ns} - X_{n's'}) \cdot d_{\min} \geq d_{\min}, \quad \forall n, n' \in \mathcal{N}, n \neq n', s \in \mathcal{S}, s' \in \mathcal{S}_s^c. \quad (11)$$

$\mathcal{S}_s^c$ denotes the subset of slots being filled at the same time as slots $s \in \mathcal{S}$. Assuming that the first S/P and the next S/P slots from $\mathcal{S}$ are filled simultaneously, the constraint can be written as

$$\underbrace{\begin{bmatrix} X_{n1} \\ \vdots \\ X_{nS} \end{bmatrix}}_{\mathbf{x}_{nS}} + \underbrace{\left[\begin{bmatrix} \mathbf{0}^{\frac{S}{P}\times\frac{S}{P}} & \mathbf{I}^{\frac{S}{P}\times\frac{S}{P}} \\ \mathbf{I}^{\frac{S}{P}\times\frac{S}{2}} & \mathbf{0}^{\frac{S}{P}\times\frac{S}{P}} \end{bmatrix} \oplus \mathbf{I}^{(P-1)\times(P-1)}\right]}_{\mathbf{I}_c} \underbrace{\begin{bmatrix} X_{n'1} \\ \vdots \\ X_{n'S} \end{bmatrix}}_{\mathbf{x}_{n'S}} \leq \left(1 + \frac{d_{nn'}}{d_{\min}}\right)\mathbf{1}^{S\times 1}\,, \quad (12)$$

with $d_{nn'}$ denoting the euclidean distance between two objects. For any $n \in \mathcal{N}$ and $\forall\, n' \in \mathcal{N}$, (12) can be written as

$$\left[\mathbf{x}_{nS} \oplus \mathbf{1}^{(N-1)\times 1}\right] + \left[\mathbf{I}_c \oplus \mathbf{I}_n\right]\mathbf{x}_b \leq \mathbf{1}^{(N-1)S\times 1} + \frac{1}{d_{\min}}\left[\mathbf{1}^{S\times 1} \oplus \mathbf{I}_n\right]\mathbf{d}_{nN}, \quad (13)$$

with $\mathbf{d}_{nN} = [d_{n1}\, d_{n2} \cdots d_{nN}]^T$ and the matrix $\mathbf{I}_n^{(N-1)\times N}$ resulting from the identity matrix $\mathbf{I}^{N\times N}$ by removing the first row and shifting the first column vector to the column n. By this the case $n = n'$ is excluded. For instance, $n = 2$ results in

$$\mathbf{I}_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & 0 \\ : & : & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}.$$

Finally, with the identity vector $\mathbf{e}_n^{N\times 1}$ the equation (13) can be represented as

$$\left[\mathbf{0}^{(N-1)S\times RP}\left[\left[\mathbf{I}^{S\times S} \oplus \mathbf{e}_n^T\right] \oplus \mathbf{1}^{(N-1)\times 1} + \left[\mathbf{I}_c \oplus \mathbf{I}_n\right]\right]\right]\begin{bmatrix}\mathbf{w}_b \\ \mathbf{x}_b\end{bmatrix} \leq \mathbf{1}^{(N-1)S\times 1} + \frac{1}{d_{\min}}\left[\mathbf{1}^{S\times 1} \oplus \mathbf{I}_n\right]\mathbf{d}_{nN} \quad (14)$$

depending linearly on the binary optimization variables.

The cost function representing the total distance covered by the robots can be expressed as

$$f(\mathbf{z}_b) = \sum_{\forall r \in \mathcal{R}} \sum_{\forall p \in \mathcal{P}} W_{rp} \left(\sum_{\forall n \in \mathcal{N}} X_{n\hat{s}_p}\, d_{nr}\right) + \sum_{\forall n \in \mathcal{N}} \sum_{\forall s \in \mathcal{S}} X_{ns}\, d_{ns} + \sum_{\forall n' \in \mathcal{N}} \sum_{\forall p \in \mathcal{P}} \sum_{\forall s \in \mathcal{S}_p} X_{n',s+1}\, d_{n's}\,. \quad (15)$$

The bilinear part of $f(\mathbf{z}_b)$ representing the initial movements of the robots can be written as

$$\mathbf{w}_b^T \underbrace{\begin{bmatrix} \mathbf{d}_{1N}^T \oplus \mathbf{I}^{P\times P} \\ \vdots \\ \mathbf{d}_{rN}^T \oplus \mathbf{I}^{P\times P} \\ \vdots \\ \mathbf{d}_{RN}^T \oplus \mathbf{I}^{P\times P} \end{bmatrix}}_{\mathbf{D}_{RN}} \begin{bmatrix} X_{1\hat{s}_1} \\ \vdots \\ X_{N\hat{s}_1} \\ X_{2\hat{s}_2} \\ \vdots \\ X_{N\hat{s}_P} \end{bmatrix} = \mathbf{z}_b^T \mathbf{Q} \mathbf{z}_b\,, \quad (16)$$

where

$$\mathbf{Q} = \begin{bmatrix} \mathbf{D}_{RN} \\ \mathbf{0}^{NS\times NP} \end{bmatrix} \begin{bmatrix} \left[\mathbf{0}^{N\times RP}\left[\mathbf{e}_{\hat{s}_1}^T \oplus \mathbf{I}^{N\times N}\right]\right] \\ \vdots \\ \left[\mathbf{0}^{N\times RP}\left[\mathbf{e}_{\hat{s}_P}^T \oplus \mathbf{I}^{N\times N}\right]\right] \end{bmatrix},$$

$\hat{s}_p$ being the slot to be filled first in class p, and the vector $\mathbf{d}_{rN} = [d_{r1}\, d_{r2} \cdots d_{rN}]^T$ of the distances between robot r and the objects $n \in \{1, \cdots, N\}$. The linear part of the cost function representing the movement from the objects to the assigned slots and back to the next selected objects can be expressed as

$$\mathbf{c}^T \mathbf{z}_b = \left[\left[\mathbf{0}^{1\times RP}\ \mathbf{d}_{NS}^T\right] + \left[\mathbf{0}^{1\times RP}\ \mathbf{d}_{NS_p}^T\right]\right]\mathbf{z}_b\,. \quad (17)$$

Finally, the discrete optimization problem can be given in this form

$$\begin{aligned} \min_{\mathbf{z}_b} \quad & \mathbf{z}_b^T \mathbf{Q} \mathbf{z}_b + \mathbf{c}^T \mathbf{z}_b \\ \text{s.t.} \quad & \mathbf{A}_e \mathbf{z}_b = \mathbf{1} \\ & \mathbf{A}_i \mathbf{z}_b \leq \mathbf{1} \\ & \mathbf{A}_s \mathbf{z}_b \leq \mathbf{b}(d_{\min}) \end{aligned} \quad (18)$$

with the equality constraints (6)-(9), the inequality constraints (10) and the safety related inequality constraints (14). The vector $\mathbf{b} \in \mathbb{R}^{N(N-1)S}$ represents the right hand side of the equation (14) $\forall\, n \in \mathcal{N}$.

IV. Synchronous distributed minimum-time MPC

According to the proposed hierarchical control scheme in Sec. II, the objective of the local model predictive controllers is the online planning of optimal robot trajectories to reach the final configuration provided by the scheduler in a minimum possible time. When generating the minimum-time robot trajectories, the final time t_f is not known a priori, resulting in a free terminal time optimization problem with the goal to minimize the final time subject to various robot-relevant kinematic and dynamic constraints. By introducing the time scaling

$$\tau = \frac{t - t_0}{t_f - t_0}\,, \tag{19}$$

the time interval $t \in [t_0, t_f]$ is mapped to $\tau \in [0, 1]$ transforming the free terminal time optimization problem into a fixed terminal time problem in the scaled time τ. With $\mathbf{q}(t) = \mathbf{q}(\tau)$,

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{q}(t) = \frac{1}{\Delta t_f}\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbf{q}(\tau), \text{ and } \quad \frac{\mathrm{d}^2}{\mathrm{d}t^2}\mathbf{q}(t) = \frac{1}{(\Delta t_f)^2}\frac{\mathrm{d}^2}{\mathrm{d}\tau^2}\mathbf{q}(\tau)\,, \tag{20}$$

where $\Delta t_f = t_f - t_0$, the system dynamics equation (2) is expressed as

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\mathbf{x}(\tau) = \mathbf{A}\mathbf{x}(\tau) + (\Delta t_f)^2\mathbf{B}\mathbf{u}(t_0 + \tau\Delta t_f)\,. \tag{21}$$

A subsequent discretization of the continuous time system

$$\begin{aligned}\mathbf{x}(\Delta\tau) =& (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}_0 + \\ & (\Delta t_f)^2\int_0^{\Delta\tau}(\mathbf{I} + \mathbf{A}(\Delta\tau - s))\mathbf{B}\mathbf{u}(t_0 + s\Delta\tau)\mathrm{d}s \\ =& (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}_0 + \\ & \Delta t_f\int_{t_0}^{t_0+\Delta\tau\Delta t_f}\left(\mathbf{I} + \mathbf{A}\left(\Delta\tau - \frac{s - t_0}{\Delta t_f}\right)\right)\mathbf{B}\mathbf{u}(s')\mathrm{d}s',\end{aligned} \tag{22}$$

yields discrete time dynamics

$$\mathbf{x}_{k+1} = (\mathbf{I} + \Delta\tau\mathbf{A})\mathbf{x}_k + \Delta\tau(\Delta t_f)^2\left(\mathbf{I} + \frac{\Delta\tau}{2}\mathbf{A}\right)\mathbf{B}\mathbf{u}_k\,, \tag{23}$$

with $\mathbf{u}_k = \text{const}$ for $\tau \in [k\Delta\tau, (k+1)\Delta\tau)$, i.e. for $t \in [t_k, t_k + \Delta\tau\Delta t_f)$. In the following, t_f is used instead of Δt_f for simpicity's sake. The main feature of the proposed DMPC is that each of the local controllers optimizes its cost function, i.e. the local robot transition time t_f, while additionally taking into account the previously achieved optimal value of the local cost function of the other controller. By coupling the cost functions, it is possible to coordinate the movements of the robots in time, so that they perform their tasks simultaneously following the synchronous approach. The time synchronization can be ensured by choosing the cost function for Robot 1 as

$$J_1(t_{1f}) = \left(t_{1f} - \hat{t}_{1f}\delta\left(\frac{\hat{t}_{2f}}{\hat{t}_{1f}}\right)\right)^2\,, \tag{24}$$

with the switch function

$$\delta\left(\frac{\hat{t}_{2f}}{\hat{t}_{1f}}\right) = \frac{2}{1 + \exp^{-\alpha\left(\frac{\hat{t}_{2f}}{\hat{t}_{1f}} - 1\right)}} - 1\,. \tag{25}$$

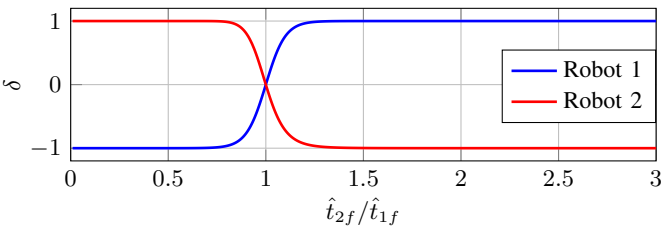


Fig. 2. Switch functions for $\alpha = 20$ to coordinate the robots movements ensuring that they reach their targets simultaneously. The switch functions show a symmetrical behavior to each other since only one of the robots, the faster one, should be slowed down without any influence on the other.

Here t_{1f} describes the optimization variable and $\hat{t}_{1f}, \hat{t}_{2f}$ are the previously calculated optimal values of the cost functions of the respective controllers. The cost function for Robot 2 is defined analogously. If Robot 1 requires less time to reach its target than Robot 2, i.e., $\hat{t}_{2f} > \hat{t}_{1f}$, then δ becomes positive for the Robot 1 and negative for Robot 2, see Fig. 2. In the current optimization step, $\delta > 0$ implies that the minimum point of the cost function (24) is shifted to $\hat{t}_{1f}\delta$. Consequently, depending on δ, the faster robot is required to have a minimum time t^*_{1f} that is not far from the minimum time $\hat{t}_{1f}$ of the preceding optimization step. The faster robot is thus slightly slowed down in each optimization step as long as the optimal values of the cost functions do not converge towards the same point. On the other hand, $\delta < 0$ in the cost function $J_2(t_{2f})$ of the second robot does not influence the optimization, therefore, Robot 2 continues its attempt to reach the target as quickly as possible.

Note that in the proposed distributed approach, the controllers exchange the optimal values of their cost functions and the resulting optimal trajectories at each optimization step. While the calculated optimal times are needed in the cost functions, the predicted trajectories are used in the constraints to establish the conditions for avoiding collisions between the robots. This results in a DMPC algorithm of the decoupled systems with coupled cost functions and constraints. At each iteration k of the optimization, each controller optimizes its own set of inputs keeping the information needed from the other controller constant as previously received.

The optimization problem for Robot 1 reads:

$$\begin{aligned}\min_{\mathbf{u}_1}\quad & J_1(t_{1f}) \\ \text{s.t.}\quad & \mathbf{x}_1(j+1\,|\,k) = \mathbf{f}(\mathbf{x}_1(j\,|\,k),\, \mathbf{u}_1(j\,|\,k),\, t_{1f}^2) \\ & \begin{bmatrix}\mathbf{q}_{\min}\\ \dot{\mathbf{q}}_{\min}\, t_{1f}\end{bmatrix} \le \mathbf{x}_1(j+1\,|\,k) \le \begin{bmatrix}\mathbf{q}_{\max}\\ \dot{\mathbf{q}}_{\max}\, t_{1f}\end{bmatrix} \\ & \mathbf{u}_{\min} \le \mathbf{u}_1(j\,|\,k) \le \mathbf{u}_{\max} \\ & d_{fr}(\mathbf{x}_1(j\,|\,k), \hat{\mathbf{x}}_2(j\,|\,k-1)) \ge r_{\min} \\ & d_{z_1}(\mathbf{x}_1(j\,|\,k)) \ge h_{z_1,\min} \\ & \mathbf{x}_1(0\,|\,k) = \begin{bmatrix}\mathbf{q}_{10}\\ \dot{\mathbf{q}}_{10}\, t_{1f}\end{bmatrix} \\ & \mathbf{x}_1(N_p\,|\,k) = \begin{bmatrix}\mathbf{q}_{1f}(k)\\ \dot{\mathbf{q}}_{1f}(k)\, t_{1f}\end{bmatrix}\end{aligned} \tag{26}$$

where $j \in \{0, \dots, N_p - 1\}$ is the iteration index, and N_p the prediction horizon. The optimization problem for Robot 2 is defined analogously. The sampling time of the scaled discrete time dynamics $\mathbf{x}_1(j+1 \,|\, k)$ from (23) is chosen as $\Delta\tau = 1/N_p$.

Due to the recurrent trajectory planning, the introduced time transformation (19) is applied in each optimization iteration k, mapping the time interval $t \in [t_{k,0},\, t_{k,f}]$ to $\tau \in [0,\, 1]$. The scaled time τ is accordingly divided into equidistant intervals, i.e., $\tau \in \{0, 1/N_p, 2/N_p, \dots, (N_p-1)/N_p, 1\}$. Moreover, the control inputs $\mathbf{u}_1(j \,|\, k)$ are uniformly distributed over τ by $\Delta\tau$, and are therefore constant over the time interval $\Delta t = t_{1f}^* \Delta\tau$, see (23), where t_{1f}^* refers to the underlying optimal solution of (26). As time t_k propagates, the $\Delta\tau$ intervals are mapped to increasingly tighter Δt-intervals because t_{1f}^* reduces with each optimization step. The closer the robot approaches the target, the smaller the solution t_f^* gets, while the number of control points remains the same. This leads to a finer prediction towards the end, which, at some point, leads to the case $\Delta\tau t_f^* < T_s$, in which there is more than one control input per sample T_s (robot control sampling period). It is possible to provide only the first control input to the plant when $\Delta\tau t_f^* \geq T_s$. Once the sample time has been undercut (i.e. $\Delta\tau t_f^* < T_s$), an equivalent input is calculated by weighted averaging of all input variables occurring within the sampling period. For this purpose, the weights

$$\eta_F = \left\lfloor \frac{T_s}{\Delta\tau t_{1f}^*} \right\rfloor \quad \text{and} \quad \eta_R = \frac{T_s - \eta_F \Delta\tau t_{1f}^*}{\Delta\tau t_{1f}^*} \tag{27}$$

are calculated first. Herein η_F is the number of inputs occurring within T_s and η_R the remaining value, i.e., the length of the last input, which is not entirely within T_s. $\lfloor \cdot \rfloor$ denotes the integer (floor) operator. For $\eta_F < N_p$ the equivalent input can be calculated as follows

$$\mathbf{u}_{1,eq}(k) = \frac{1}{\eta_F + \eta_R} \left(\sum_{i=1}^{\eta_F} \mathbf{u}_1^*(i \,|\, k) + \eta_R \mathbf{u}_1^*(\eta_F + 1 \,|\, k) \right). \tag{28}$$

The desired final state $\mathbf{x}_1(N_p \,|\, k)$ in (26), which, according to the time optimal setting, has to be reached in each optimization step k, i.e. at $\tau = 1$, is calculated from the information provided by the scheduler. Since the scheduler operates in a global Cartesian coordinate system, the position of the target ${}_s\mathbf{p}_t$ (objects or slots) has to be transformed into the local robot coordinates using the homogeneous transformation

$$\begin{bmatrix} {}_r\mathbf{p}_t \\ 1 \end{bmatrix} = \mathbf{T}_r^s \begin{bmatrix} {}_s\mathbf{p}_t \\ 1 \end{bmatrix}. \tag{29}$$

The obtained position vector ${}_r\mathbf{p}_t$ relative to the origin of the base frame of robot r is subsequently used to calculate the desired robot end configuration. To describe the target orientation relative to the robots base frame, Euler angles ${}_r\mathbf{\Psi}_t = [\phi, \theta, \psi]^T$ are used. The pose of the end effector to reach the desired target is then given by

$$\boldsymbol{\varrho}_r(\mathbf{q}_f) = \begin{bmatrix} {}_r\mathbf{p}_t \\ {}_r\mathbf{\Psi}_t \end{bmatrix}. \tag{30}$$

Using (30) and the robot's forward kinematic equations, the final robot configuration $\mathbf{q}_f$ is calculated by analytically solving the inverse kinematic problem [13]. The desired final robot configuration $\mathbf{q}_f(k)$ changes over time as the targets, which have to be reached, move along the $y-$axis with a constant velocity ${}_s\mathbf{v}_t = [0\, v_b\, 0]^T$ (w.r.t. the global coordinate frame), with v_b denoting the velocity of the conveyor belt. Therefore, in each optimization step, the desired position of the end effector is updated depending on the target (object or slot) position. At the first optimization step, the desired position of the end effector is required to be equal to the initial target position, i.e.

$$\mathbf{q}_f(k) = \boldsymbol{\varrho}_r^{-1} \left(\begin{bmatrix} {}_r\mathbf{p}_t(k) \\ {}_r\mathbf{\Psi}_t \end{bmatrix} \right). \tag{31}$$

For this, the minimum transition time t_f^* is computed, which will be used in the following step to have a better estimation of the target's position

$$\mathbf{q}_f(k) = \boldsymbol{\varrho}_r^{-1} \left(\begin{bmatrix} {}_r\mathbf{p}_t(k) + t_f^*(1-\Delta\tau)\, {}_r\mathbf{v}_t \\ {}_r\mathbf{\Psi}_t \end{bmatrix} \right). \tag{32}$$

The conveyor belt movements do not affect the orientation ${}_r\mathbf{\Psi}_t$ of the target relative to the robot's base coordinate system. To finally compute the desired robot's joint velocities, the end effector Jacobian $\mathbf{J}_{re}(\mathbf{q})$ is used. Thus, the joint velocities are then found via the inverse Jacobian

$$\dot{\mathbf{q}}_f(k) = \mathbf{J}_{re}(\mathbf{q}_{1f})^{-1} \begin{bmatrix} {}_r\mathbf{v}_t \\ \mathbf{0}^{3\times 1} \end{bmatrix}. \tag{33}$$

It should be noted that the proposed algorithm is only simultaneously investigated and validated within the scope of this paper. If implemented in a real system, the support by a camera is necessary to grab and place the objects.

The presence of static and dynamic obstacles in the working area of a manipulator limits its full ability and operating range. Therefore, when using optimization-based algorithms for online trajectory planning, it is reasonable to formulate the collision avoidance conditions as motion constraints in the optimization problem, see [14]. Three different types of collision avoidance constraints are taken into consideration in this work. Firstly, self-collision constraints are formulated for the relevant joints by limiting their movements between $\mathbf{q}_{\min}$ and $\mathbf{q}_{\max}$. Secondly, a constraint $d_{z_1}(\mathbf{x}_1(j \,|\, k))$ with a minimum operating height in the $z-$coordinate is considered to avoid collisions between the robot and the conveyor belt. Finally, the constraints to avoid collisions between the two robots are imposed. For the sake of simplicity, these constraints are formulated with respect to the robot end effectors by utilizing the concept of a distance function $d_{fr}(\mathbf{x}_1(j \,|\, k), \hat{\mathbf{x}}_2(j \,|\, k-1))$. Here, $\hat{\mathbf{x}}_2$ refers to the optimal trajectory of the second robot calculated in the previous MPC step. This information is needed to calculate the distance between the robots as a function of $\mathbf{x}_1$ and is kept constant during an MPC iteration k.

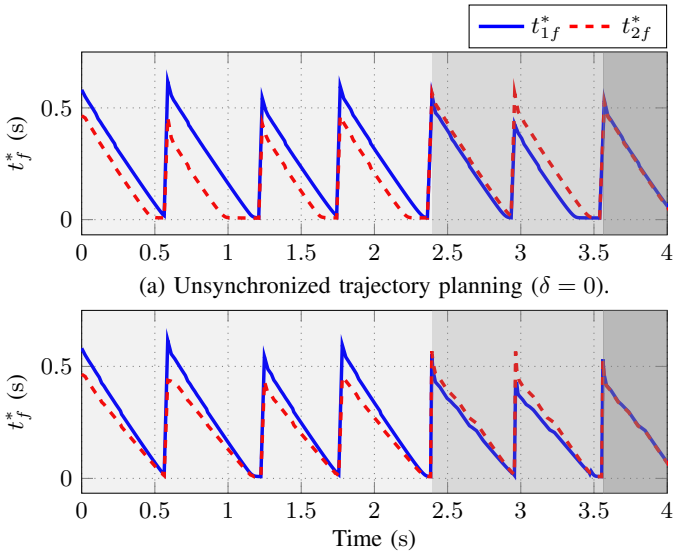
V. Simulation Results

The proposed hierarchical control algorithm is implemented and evaluated on a dynamic simulation model using MATLAB/SIMULINK. The dynamic simulation model consists of the dynamics of the robots and the computed torque controllers. The arrangement of the robots next to each other is such that their working areas overlap considerably. In the context of this paper, the focus lies mainly on the online planning of synchronous robot trajectories using distributed MPCs of dynamically decoupled systems with coupling in the cost function and the constraints. As the results of the scheduling layer are discussed in detail in [8], they will not be addressed further in this paper. The scheduling is performed iteratively after each placed object and solved using the GUROBI solver. The information provided by the scheduler is used on the underlying DMPC layer to compute the desired robot end configuration and joint velocities, i.e. to build the terminal constraints $\mathbf{x}_r(N_p \,|\, k)$, (26). The resulting optimization problems for trajectory generation are then solved for both robots independently using the full discretization approach and applying the IPOPT solver. Following this approach, the unknown states, control inputs, as well as the final time are merged into a vector of optimization variables

$$\begin{aligned}\boldsymbol{\zeta}_r(k) = [&\boldsymbol{x}_r^T(0 \,|\, k) \;\cdots\; \boldsymbol{x}_r^T(N_p \,|\, k) \\ &\mathbf{u}_r^T(0 \,|\, k) \;\cdots\; \mathbf{u}_r^T(N_p - 1 \,|\, k) \quad t_{rf}(k)]^T \quad ,\end{aligned}$$

with the index r denoting Robot 1 and Robot 2. After each MPC iteration k, the robots share their predicted trajectories and the computed minimum time with each other. The sampling time is set to $T_s = 8\,\mathrm{ms}$ according to the control frequency of the considered robot manipulators. The prediction horizon and the belt velocity are chosen as $N_p = 15$ and $v_b = 0.1\,\mathrm{m/s}$. Choosing a shorter prediction horizon, as would be desirable for computational reasons, will reduce the solution space of the optimization problem due to the terminal constraints. On the other hand, a broader prediction horizon, which might be advantageous with respect to the collision avoidance constraints, would increase the computational effort. Thereby, our chosen value $N_p = 15$ is found to be a good trade-off.

At the beginning of the optimization, i.e. at the beginning of the trajectory planning, the calculated minimum time is long enough and decreases with each MPC step, as the robot approaches its target with each applied control input. The minimum final time becomes nearly zero when the robot reaches the target, triggering the re-execution of the planning algorithm, and thus leading the robot to the next destination. As can be seen in Fig. 3, this results in a sawtooth-like progression of the optimal time over the simulation duration. According to the results, Robot 1 needs more time to reach the first target than Robot 2, i.e. $t_{1f}^* > t_{2f}^*$. In case the robots' trajectories are not synchronized, i.e. $\delta = 0$, Robot 2 will reach its first target faster than Robot 1. This is due to the fact that for the selected targets, the second robot has to cover a shorter distance compared to the first one. Robot 2 will reach the first object after approximately $0.46\,\mathrm{s}$ and the corresponding slot after $1\,\mathrm{s}$, whereas Robot 1 after $0.58\,\mathrm{s}$ and $1.2\,\mathrm{s}$, respectively, see Fig. 3 (a). Since it is required that the robots perform their tasks simultaneously, the faster robot is gradually slowed down until the final times become equal. This is done by coupling the cost functions of the trajectory generation optimization algorithms, as described in Sec. IV. Synchronizing the trajectories in time, results in a lower slope of the optimal time curve of the slower robot compared to the uncoordinated case, see Fig. 3 (b). As long as the calculated optimal times of the two robots differ, the faster robot is slowed down slightly in each step until the final times converge towards the same value. Nevertheless,



(a) Unsynchronized trajectory planning ($\delta = 0$).

(b) Synchronized trajectory planning ($\delta \neq 0$).

Fig. 3. Comparison of the calculated minimum times t_f^* of the two robots for both cases, unsynchronized (a) and synchronized (b). As a result of the time synchronization, the difference between the calculated optimal times decreases with each optimization step until they finally equalize. Due to the synchronization, the robots will reach their targets simultaneously.

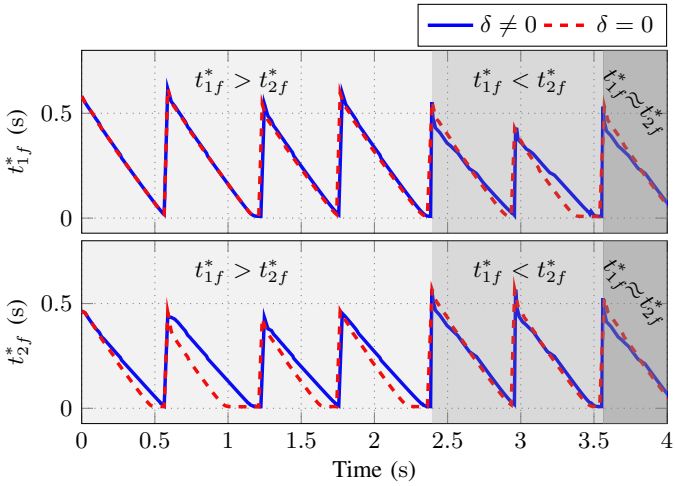
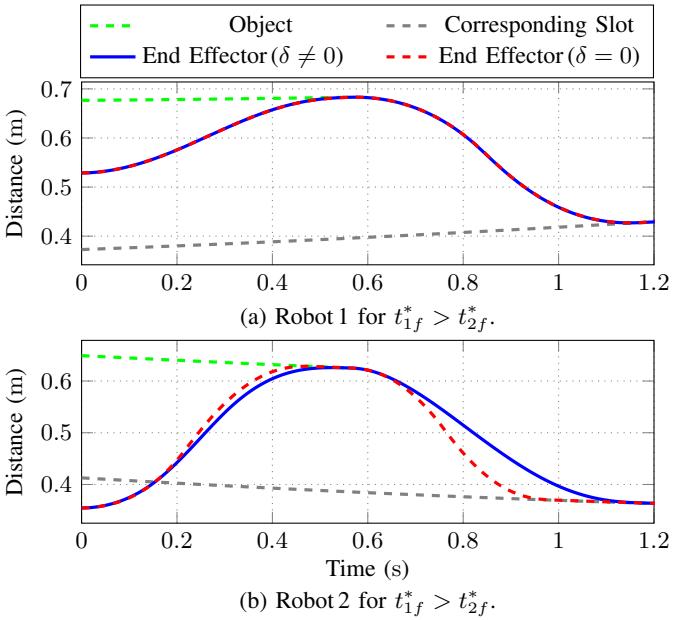


Fig. 4. Comparison between the unsynchronized ($\delta = 0$) and synchronized ($\delta \neq 0$) case for each robot individually. For $t_{1f}^* > t_{2f}^*$, the time synchronization does not affect the slower Robot 1 and vise-versa in case of Robot 2 for $t_{2f}^* > t_{1f}^*$. The time synchronization ensures that the optimal time of the faster robot changes more slowly in each optimization step until it matches the time of the slower robot. For $t_{1f}^* \approx t_{2f}^*$, i.e. $\delta \approx 0$, both robots try to reach their targets as fast as possible (see also Fig. 3).



(a) Robot 1 for $t^*_{1f} > t^*_{2f}$.

(b) Robot 2 for $t^*_{1f} > t^*_{2f}$.

Fig. 5. Distances of the end effectors, the assigned objects, as well as the corresponding slots. All distances are shown relative to the same coordinate system to simplify the comparison. (a): The slower Robot 1, continues planning its trajectory regardless of the time synchronization targeting to finish the task as fast as possible. (b): Due to the time synchronization, the faster Robot 2 reaches its targets at the same time as the slower Robot 1.

this does not affect the slower robot, as it must not be slowed down any further. This can be seen by the comparison between the synchronized ($\delta \neq 0$) and the unsynchronized ($\delta = 0$) trajectory planning for both robots as shown in Fig. 3 and Fig. 4. For the slower robot (Robot 1 for $t^*_{1f} > t^*_{2f}$ and Robot 2 for $t^*_{2f} > t^*_{1f}$) the outcome of the optimization for the synchronized and the unsynchronized planning is the same. For the faster robot, on the other hand, the decrease of the final time becomes smaller due to the synchronization of the trajectory planning. The difference between the optimal final times (t^*_{1f}, t^*_{2f}) of the robots becomes smaller with each iteration until they finally approach to zero simultaneously. In case the robots need about the same time to reach their targets, i.e., $t^*_{1f} \approx t^*_{2f}$, the synchronization has no influence on the optimal final times, and both robots try to achieve their goals as fast as possible.

The effect of the synchronization on the robot movements can also be observed by considering all distances relevant to a pick-and-place scenario, as shown in Fig. 5 for both, the synchronized ($\delta \neq 0$) and unsynchronized ($\delta = 0$) case. For Robot 1 the course of the end effector from the starting position to the first object and afterward to the corresponding slot is the same in both cases, since the synchronization has no impact on the slower robot, see Fig. 5 (a). On the contrary, the end effector of the faster Robot 2 shows a different behavior due to the synchronization, see Fig. 5 (b). In the unsynchronized way, Robot 2 would reach its targets earlier. Especially when reaching the slot, one can see that the robot could place the object after about 1 s. In the synchronous case, however, the robot reaches the slot after about 1.2 s, and thus at the same time as Robot 1.

VI. CONCLUSIONS

An optimization-based algorithm is presented, which consists of a central scheduling layer for the robot task allocation and an underlying distributed layer of communicating DMPC algorithms for trajectory generation. The cost functions of the resulting distributed optimization problems are coupled via a switch function, imposing a synchronous trajectory planning for the robots. The proposed approach is demonstrated in simulations showing the coupling term's influence on synthetizing synchronized robot motions by progressively decelerating the faster robot and not influencing the slower one. Future work will consider the real-time implementation of the suggested framework in an experimental setup. The validation in a realistic environment will also provide an opportunity to analyze the effects of disturbances and communication delays on the closed-loop performance, in particular regarding the time synchronization and the robustness of the DMPC algorithms.

REFERENCES

[1] Q. Zhang and M.-Y. Zhao, "Minimum time path planning of robotic manipulator in drilling/spot welding tasks," *Journal of Computational Design and Engineering*, vol. 3, no. 2, pp. 132–139, 2016.

[2] P. T. Zacharia and N. A. Aspragathos, "Optimal robot task scheduling based on genetic algorithms," *Robotics and Computer-Integrated Manufacturing*, vol. 21, no. 1, pp. 67–79, 2005.

[3] E. K. Xidias, P. T. Zacharia, and N. A. Aspragathos, "Time-optimal task scheduling for two robotic manipulators operating in a three-dimensional environment," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 224, no. 7, pp. 845–855, 2010.

[4] S. Sutdhiraksa and R. Zurawski, "Scheduling robotic assembly tasks using petri nets," in *Proceedings of IEEE International Symposium on Industrial Electronics*, pp. 459–465, IEEE, 1996.

[5] M. M. G. Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, "Real-time trajectory generation using model predictive control," in *2015 IEEE Int. Conference on Automation Science and Engineering (CASE)* (Q.-S. Jia, ed.), (Piscataway, NJ), pp. 942–948, IEEE, 2015.

[6] G. P. Incremona, A. Ferrara, and L. Magni, "Mpc for robot manipulators with integral sliding modes generation," *IEEE/ASME Transactions on Mechatronics*, vol. PP, pp. 1–1, 02 2017.

[7] M. Schlemmer and G. Gruebel, "Real-time collision- free trajectory optimization of robot manipulators via semi-infinite parameter optimization," *The International Journal of Robotics Research*, vol. 17, no. 9, pp. 1013–1021, 1998.

[8] A. Tika, N. Gafur, V. Yfantis, and N. Bajcinca, "Optimal scheduling and model predictive control for trajectory planning of cooperative robot manipulators," *21th World Congress of the International Federation of Automatic Control (IFAC), Berlin, Germany*, July 2020.

[9] A. Richards and J. How, "Decentralized model predictive control of cooperating uavs," vol. 4, pp. 4286 – 4291 Vol.4, 01 2005.

[10] C. Luis, M. Vukosavljev, and A. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, 01 2020.

[11] Y. Zheng, S. Li, K. Li, F. Borrelli, and J. Hedrick, "Distributed model predictive control for heterogeneous vehicle platoons under unidirectional topologies," *IEEE Transactions on Control Systems Technology*, 08 2016.

[12] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot dynamics and control*. Hoboken, N.J.: Wiley, 2006.

[13] K. P. Hawkins, "Analytic inverse kinematics for the universlal robots ur-5/ur10 arms," *Technical report, Georgia Institute of Technology*, 07 2013.

[14] B. Faverjon and P. Tournassoud., "A local based approach for path planning of manipulators with a high number of degrees of freedom. in proceedings.," *1987 IEEE International Conference on Robotics and Automation*, pp. 1152–1159, 1987.