IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 28, 2022, accepted March 8, 2022, date of publication March 14, 2022, date of current version March 21, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3159233

Improving Local Trajectory Optimization by Enhanced Initialization and Global Guidance

MAXIMILIAN KRÄMER AND TORSTEN BERTRAM
Institute of Control Theory and Systems Engineering, TU Dortmund University, 44227 Dortmund, Germany
Corresponding author: Maximilian Krämer (maximilian.kraemer@tu-dortmund.de)

This work was supported by the Deutsche Forschungsgemeinschaft and Technische Universität Dortmund/TU Dortmund University through the funding program Open Access Costs.

ABSTRACT Trajectory optimization is a promising method for planning trajectories of robotic manipulators. With the increasing success of collaborative robots in dynamic environments, the demand for online planning methods grows and offers new opportunities as well as challenges for trajectory optimization. Special requirements in terms of real-time capabilities are one of the greatest difficulties. Optimizing a short planning horizon instead of an entire trajectory is one approach to reduce computation time, which nonetheless separates the optimality of local and global solutions. This contribution introduces, on the one hand, Extended Initialization as a new approach that reduces the risk of local minima and aims at improving the quality of the global trajectory. On the other hand, the particularly critical cases in which local solutions lead to standstills are mitigated by globally guiding local solutions. The evaluation performs four experiments with comparisons to Stochastic Trajectory Optimization for Motion Planning (STOMP) or Probabilistic Roadmap Method (PRM*) and demonstrates the effectiveness of both approaches.

INDEX TERMS Moving horizon planning, online trajectory optimization, local minima.

I. INTRODUCTION

Trajectory optimization enables an intuitive way to apply motion specifications to trajectory planning for robotic manipulators. It allows both, the consideration of constraints such as joint limits or collision avoidance, and the optimization of criteria that range from high-level tasks [1] over secondary objectives [2] down to the motion itself [3]. Consequently, it has long been established as a general planning method. On the other side, online planning offers a flexible and instantaneous way to deal with changes in the environment or task. The success of robots for collaborative purposes, leads to an increasing presence of people in the workspace, thus further rendering the importance of the topic. Online planning is still a challenging problem where criteria such as real-time capability and quality of the trajectory conflict. Because of its success, trajectory optimization is also a reasonable choice for online planning. However, the available planning time is limited and, depending on the dynamics of the environment, might be insufficient to simply perform trajectory optimization in a loop. For this reason, there are measures to simplify the planning problem at the price of reduced quality of the solutions.

This contribution analyzes the impact of planning short local trajectories that simplify the planning problem and proposes methods to mitigate suboptimality. The next subsection gives an overview over related methods that cope with the computational burden and negative side-effects of simplified problems in online trajectory optimization.

A. RELATED WORKS

The representation of the trajectory in an optimization problem has a relevant effect on the computation time. For example, representations such as fully discretized collocation or multiple shooting outperform single shooting by providing sparsity and improve numerical stability [4]. The sparse optimization problem is then encoded in a suitable structure like, e.g. a hypergraph [5] and solved by a sparse solver. A method to further reduce the effort exploits the separation of shooting intervals even further by distributing them over several local optimization problems and solving them via distributed optimization methods [6]. It is faster than solving with ordinary multiple shooting, however, it is most effective for full trajectories only which require knowledge about the total duration and full term predictions of the environment.

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

In dynamic environments, where spontaneous detours or evasions are necessary, the duration of a trajectory is not trivial to determine.

Automatic Differentiation [7] becomes increasingly popular and reduces the amount of expensive function evaluations compared to finite differences which have a significant impact on the total computation time because of costly distance calculations. However, it cannot easily be adapted to structural changes of the optimization problem caused by variable numbers of obstacles which is why finite differences are still commonly used.

A higher level approach to reduce the problem complexity is to devide the trajectory into several segments and perform planning of the next segment while tracking the current one [8]. It allows computation times up to the length of a segment, but also requires predictions of the environment to extend over the length of two segments, which is difficult to achieve, especially for moving people. In addition, the motion of the currently tracked segment is fixed and cannot be adjusted for changes, which is why smaller segments are desirable. However, choosing smaller segments mitigates the effectiveness of this method.

Recently, methods that utilize a fixed planning horizon which always starts from the current state become more and more popular [3], [9]–[11]. Because of their principle, they can also be called Moving Horizon Planning (MHP). A portion of the solution is tracked and the next optimization is performed similarly to Model Predictive Control. The duration of the final trajectory results implicitly even though the planning horizon is fixed. The shorter the horizon, the simpler and faster the optimization problem. This makes it the fastest method compared to the other approaches. However, the introduction of a limited planning horizon leads to a local planning problem that departs from the original problem and promotes vulnerability to local minima [3], [12], [13]. Local minima appear in the form of suboptimal global trajectories or even standstills, since a local solution (locally optimal or not) is not necessarily optimal in the sense of the global problem. A detailed problem definition follows in Section II.

The most common methods for optimization are gradient-based. Interior point [14] or sequential quadratic programming [15] approaches have become standard and efficiently exploit the sparsity of collocation or multiple shooting [5]. Their main weakness is their dependency on a proper initialization, which is a serious factor especially for nonlinear problems such as trajectory planning. Gradient-free approaches such as STOMP overcome this by improving a trajectory iteratively until convergence via random sampling and selection similar to evolutionary algorithms [16]. However, this method is often not fast enough since a complete trajectory is planned, for which the duration must be known in advance.

Suboptimal global trajectories and standstills are closely related to the structure of the environment. In [17] the authors, therefore, utilize homotopy classes for initialization of parallel planning instances so that all distinct paths are considered during planning. The best instance provides the momentary solution. This method scales poorly with the complexity of the environment and plans complete trajectories in each instance, which is only fast for a two-dimensional workspace of a mobile robot. Furthermore, the idea of homotopy planning is not effectively applicable to the one-way open kinematic chain of stationary manipulators. Another approach, therefore, reactively samples via-points to solve standstills [13]. It can resolve standstills once detected but does not proactively avoid them. Closely related is a method that traverses a list of predefined via points one by one by a local potential field approach [18]. The via points originate from a visibility graph and thus capture the whole scene in a global way [19]. However, by strictly approaching the via points one by one, the method leaves little space for the local planner to perform motion optimizations. Also closely related is a method that combines global exploration with local optimization by passing an initial global solution back and forth between a global planner and local optimization [20]. Both planners are modified from literature approaches to work in conjunction. However, due to its sequential structure, this method is not real-time capable and is limited to paths.

B. CONTRIBUTION AND OUTLINE

Based on the advantages and promising results of fully discretized and gradient-based MHP, this article presents novel methods to tackle suboptimality and local minima that arise from the simplified local planning problem. The contribution is arranged as follows:

- A detailed problem definition in Section II to form a theoretical basis and to emphasize its core aspects on which the presented methods built upon
- Two methods in Section III that take over the idea of parallel planning instances and provide an enhanced initialization to mitigate suboptimality
- One method in Section IV that extends to the idea of global via points as well as interleaving global and local planning with more flexibility of the local subproblem and faster computation of trajectories
- Comparisons in Section V with plain MHP as well as state of the art planners such as STOMP and PRM*.[1]

Section VI eventually concludes this article and gives an outlook for future work. Note, PRM* merely plans a geometric path that is still comparable from a baseline point of view, since the core problem rather lives on a geometric level. However, pure path planning methods usually do not fulfill the requirements of the online trajectory planning problem without further ado.

II. PROBLEM DEFINITION

Consider the fully discretized trajectory $\boldsymbol{\tau} \in \mathcal{T}$ that consists of states $\mathbf{x}_k \in \mathcal{X}$ and controls $\mathbf{u}_k \in \mathcal{U}$ for $k = 0, 1, \ldots, K$:

$$\boldsymbol{\tau} := \begin{pmatrix} \mathbf{x}_{0:K} := \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots, \mathbf{x}_K \\ \mathbf{u}_{0:K-1} := \mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_k, \ldots, \mathbf{u}_{K-1} \end{pmatrix}, \tag{1}$$

[1] Probabilistic Roadmap Method [21] with star strategy.

with $\mathcal{T}$ as the set of all trajectories. The sets $\mathcal{X}$ and $\mathcal{U}$ define the admissible state space of a D-DoF manipulator with respect to the constraints:

$$\mathcal{X} := \left\{ \mathbf{x} \in \mathbb{R}^D \mid \mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{x}_{\max} \right\},$$
$$\mathcal{U} := \left\{ \mathbf{u} \in \mathbb{R}^D \mid \mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max} \right\}. \tag{2}$$

The distinction between states and controls arises later from the repetitive nature of online planning, which commands the required controls instead of states. For the sake of simplicity, states $\mathbf{x}_k$ refer to the robot's joint angles and controls $\mathbf{u}_k$ refer to desired joint velocities rendering $\boldsymbol{\tau}$ to be a common joint space trajectory. Without loss of generality, we assume the robot's built-in joint velocity controllers to be sufficiently precise to attain the commanded $\mathbf{u}_k$. Note, $\mathcal{U}$ can easily be extended to also limit accelerations approximated by finite control differences.

The time law of $\boldsymbol{\tau}$ uses the uniform grid:

$$t_0 < t_1 < \ldots < t_k < \ldots < t_K = T, \quad t_{k+1} - t_k = \Delta t, \tag{3}$$

which assigns timestamps t_k to all state and control pairs.

To distinguish from trajectory timestamps t_k, t_n serves as a general timestamp for $n = 0, 1, \ldots, N$. Consequently $\mathbf{q}_n := \mathbf{q}(t_n)$ is the robots actual joint configuration, with $\mathbf{q} \in \mathcal{X}_\mathrm{C}(t)$ and $\mathbf{x}_{\mathrm{f},n} := \mathbf{x}_\mathrm{f}(t_n)$ is the planning target with $\mathbf{x}_\mathrm{f} \in \mathcal{X}_\mathrm{C}(t)$ at t_n. Set $\mathcal{X}_\mathrm{C}(t) \subseteq \mathcal{X}$ contains all collision free states regarding robot self-collisions ($\mathcal{X}_\mathrm{R}(\mathbf{q}) \subseteq \mathcal{X}$), static obstacles ($\mathcal{X}_\mathrm{S}(t) \subseteq \mathcal{X}$), and dynamic obstacles ($\mathcal{X}_\mathrm{D}(t) \subseteq \mathcal{X}$):

$$\mathcal{X}_\mathrm{C}(t) = \mathcal{X}_\mathrm{R} \cap \mathcal{X}_\mathrm{S}(t) \cap \mathcal{X}_\mathrm{D}(t). \tag{4}$$

Note, $\mathcal{X}_\mathrm{S}(t)$ and $\mathcal{X}_\mathrm{D}(t)$ depend on time as the number of obstacles might change.

A. GLOBAL PLANNING PROBLEM (Offline)

A formal definition of a global optimization problem to find an optimal solution trajectory $\boldsymbol{\tau}^\star$ that starts at $\mathbf{q}_n$ and reaches $\mathbf{x}_\mathrm{d} \in \mathcal{X}_\mathrm{C}(t_n)$ while minimizing costs regarding $\mathbf{x}_k$ and $\mathbf{u}_k$ at t_n is given as:

$$\min_{\boldsymbol{\tau}, K} J(\boldsymbol{\tau}), \tag{5a}$$

subject to

$$\mathbf{x}_{k+1} - \mathbf{x}_k - \Delta t\, \mathbf{u}_k = \mathbf{0}, \tag{5b}$$
$$\mathbf{u}_k \in \mathcal{U}, \tag{5c}$$
$$\mathbf{x}_k \in \mathcal{X}_\mathrm{C}(t_k), \tag{5d}$$
$$\mathbf{x}_0 = \mathbf{q}_n, \tag{5e}$$
$$K > 0. \tag{5f}$$

Besides optimizing the number of states K it is also possible to choose a sufficiently high but fixed K or optimize Δt [22]. Solving (5) to achieve a global solution by optimization will be called Global Trajectory Optimization (GTO).

Expression (5a) utilizes the cost function:

$$J(\boldsymbol{\tau}) := \sum_{k=0}^{K} c_\mathrm{Q}(\mathbf{x}_k) + c_\mathrm{C}(\mathbf{x}_k) + c_\mathrm{R}(\mathbf{u}_k), \tag{6a}$$

with components

$$c_\mathrm{Q}(\mathbf{x}_k) := (\mathbf{x}_k - \mathbf{x}_\mathrm{d})^\mathsf{T}\, \mathbf{Q}\, (\mathbf{x}_k - \mathbf{x}_\mathrm{d}), \tag{6b}$$
$$c_\mathrm{R}(\mathbf{u}_k) := \mathbf{u}_k^\mathsf{T} \mathbf{R}\, \mathbf{u}_k, \tag{6c}$$
$$c_\mathrm{C}(\mathbf{x}_k) := \sum_{o \in \mathcal{O}} \rho(o). \tag{6d}$$

Components (6b) and (6c) state the common quadratic cost function and penalize deviations from $\mathbf{x}_\mathrm{d}$ as well as control effort with positive definite matrices $\mathbf{Q} \in \mathbb{R}^{D \times D}$ and $\mathbf{R} \in \mathbb{R}^{D \times D}$ for weighting. Although (5d) already enables collision avoidance, component (6d) keeps the robot away from obstacles in general and induces prophylactic avoidance behavior. As a robot link approaches an obstacle, the proximity function $\rho : \mathcal{O} \mapsto \mathbb{R}_0^+$ rises:

$$\rho := \begin{cases} w \left(\dfrac{d(o))}{d_\mathrm{min}} - 1 \right)^2 & d(o) < d_\mathrm{min} \\ 0 & d(o) \geq d_\mathrm{min}, \end{cases} \tag{7}$$

with weight $w \in \mathbb{R}_0^+$, activation threshold $d_\mathrm{min} \in \mathbb{R}^+$, and pairs of link and obstacle collisions $o \in \mathcal{O}$. The function $d : \mathcal{O} \mapsto \mathbb{R}$ reflects the signed distance between a collision pair o.

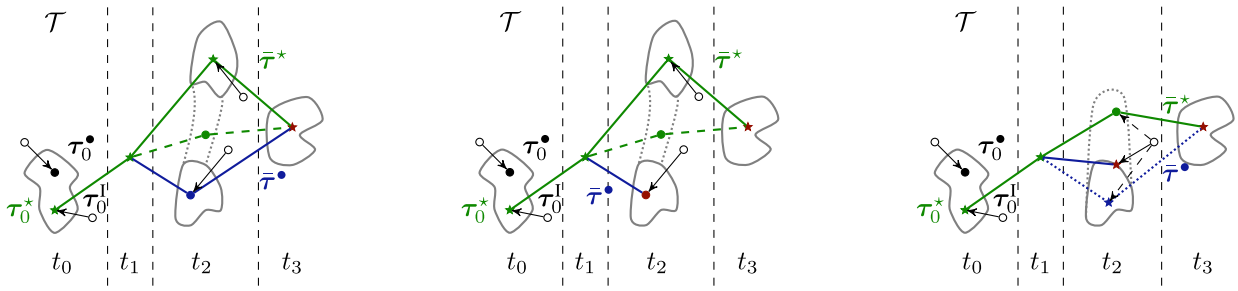
Expression (5b) relates $\mathbf{x}_k$ and $\mathbf{u}_k$ to $\mathbf{x}_{k+1}$ as a simple forward Euler integration to match the integral relation between joint angles and velocities. Equality constraint (5e) forces $\boldsymbol{\tau}^\star$ to start at the robot's state $\mathbf{q}_n$.

Solving (5) usually takes more time than available for online planning in peopled environments. Therefore, planning is performed once at t_n with $n = 0$ and $\mathbf{x}_\mathrm{d} = \mathbf{x}_{\mathrm{f},0}$. The optimality of a solution obtained by gradient-based algorithms depends on its initialization $\boldsymbol{\tau}^\mathrm{I}$. A suboptimal solution is denoted $\boldsymbol{\tau}^\bullet$ and the optimal solution is denoted $\boldsymbol{\tau}^\star$.

B. LOCAL PLANNING PROBLEM (Online)

To achieve an online-capable computation time, K is limited to a sufficiently small value. In this case, K is fixed and thus the constraint (5f) is omitted. The local problem now differs from the global one in such a way that a trajectory is sought that merely points towards $\mathbf{x}_{\mathrm{f},n}$, but does not necessarily reach it directly. Since online planning is repetitive and entangled with motion execution, $\mathbf{x}_{\mathrm{f},n}$ is approached step by step which gives this method the name of Moving Horizon Planning similar to Model Predictive Control.

Local planning is performed repeatedly at t_n for $n = 0, 1, \ldots, N$ using $\mathbf{x}_\mathrm{d} = \mathbf{x}_{\mathrm{f},n}$ and the same Δt as $\boldsymbol{\tau}$. To distinguish between trajectories of different timestamps, we use subscript notation $\boldsymbol{\tau}_n$ whenever necessary. By setting t_k for $k = 0$ equal to t_n, and considering (3), $\boldsymbol{\tau}$ effectively starts at t_n and consistently extends into the future. In each instance t_n,



(a) Case I: The Bellman Trace changes (dashed green to solid green) due to changes in the environment. Depending on the initialization at t_2, the CLT may continue suboptimal (solid blue).

(b) Case II: In the worst case, a suboptimal CLT (solid blue) ends up in a standstill (red bullet) that could have been prevented by a better initialization.

(c) Case III: If the set of feasible local solutions only offers a standstill (red star), a modification of the local problem is required. E.g., enlarging (dotted gray) the feasible set by increasing the planning horizon or changing the cost function to provide a local solution without standstill (blue star).

FIGURE 1. Illustration of three cases where local solutions affect the optimality of CLTs. The differences appear at t_2. Local solutions as well as CLTs can be optimal (stars) or suboptimal (bullets). Initializations are marked as circles.

the first element $\mathbf{u}_0$ of the optimized controls is sent to the robot resulting in the continuous-time signal $\mathbf{u}(t)$:

$$\mathbf{u}(t) := \mathbf{u}_0\big|_{t_n} = \text{const. for } t \in [t_n, t_{n+1}). \tag{8}$$

The resulting trajectory $\bar{\boldsymbol{\tau}}$ caused by the sequence of controls from local solutions is called closed-loop trajectory (CLT). It ideally matches the global solution of (5). However, this is not guaranteed as the local problem has a limited planning horizon. Suboptimality manifests in deviations from the global solution and results in, e.g. longer paths and duration or even in a standstill.

Fig. 1 demonstrates three cases of how suboptimal CLTs occur. It shows the set of all trajectories $\mathcal{T}$ with points indicating locally suboptimal solutions $\boldsymbol{\tau}_n^\bullet$ and stars indicating locally optimal solutions $\boldsymbol{\tau}_n^\star$. Red color marks are local solutions that produce a standstill. The area is vertically divided into timestamps t_n for $n = 0, 1, 2, 3$. The gray-framed areas define subsets of $\mathcal{T}$ of all *feasible* trajectories of local optimization that start at $\mathbf{q}_n$. Circles represent initializations $\boldsymbol{\tau}_n^{\mathrm{I}}$ of the local problem with corresponding arrows pointing to the converged solution. Some local solutions (green) are part of the globally optimal solution and follow Bellman's principle of optimality [23] and some (blue) arise from local solutions that produce a suboptimal CLT. The sequence of green solutions will be called Bellman Trace. The last solution at t_3 is a standstill on $\mathbf{x}_{\mathrm{f},3}$.

a: CASE I

Assuming a proper initialization $\boldsymbol{\tau}_0^{\mathrm{I}}$, then the first two solutions follow the Bellman Trace in Fig. 1a. At t_2 a change in the environment (moved obstacle) results in a new Bellman Trace, with the old one in dashed. The dotted gray shape between both feasible sets indicates the previously joined area of feasible solutions, which is now cut in half by the obstacle. The first case of suboptimal CLTs now arises by the dependency of the local problem on its initialization $\boldsymbol{\tau}_2^{\mathrm{I}}$. It may results in the bottom (blue) solution even if the top (green) solution is locally optimal.

b: CASE II

Local suboptimality reaches its worst case when it comes to a standstill because of an obstacle blocking the path and the planner's inability to find the locally optimal solution to pass around. In Fig. 1b the sequence is equal to Fig. 1a until timestamp t_2, where now the bottom local solution produces a standstill. The second case is even worse than the first because the suboptimal CLT does not reach the target $\mathbf{x}_{\mathrm{f},3}$. Again, even if the top (green) solution is locally optimal, the problem may converge to the bottom (blue) solution when initialized poorly.

c: CASE III

In Fig. 1c, the Bellman Trace at t_2 requires a local solution that lies outside the feasible set. Consequently, the local solution converges to the boundary of the set and, in this case, becomes a standstill. This case demonstrates that even locally optimal solutions $\boldsymbol{\tau}^\star$ may not lead to a globally optimal CLT. A standstill is locally optimal e.g. if the planning horizon is too short to cover the reeving part of an evasive motion and thus leading to higher costs than stopping. One option is to increase K to extend the feasible set (dotted gray) in such a way that it includes the upper (green) solution. Then Case II applies if there are additional local minima. Since a longer horizon also means higher computation time, this is only partly applicable. Another way is to change the cost function so that a new locally optimal solution (blue) arises, which does not cause a standstill. Although the resulting sequence of local solutions no longer resembles the Bellman Trace, the goal is reached at the costs of a suboptimal CLT.

Besides improving the quality of local solutions by proper initializations $\boldsymbol{\tau}_n^{\mathrm{I}}$, it is also important to offer local solutions, which produce a (near) optimal CLT. For the former, this

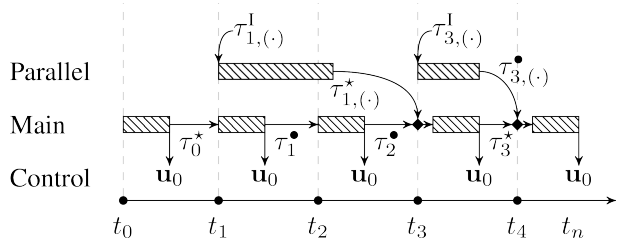


FIGURE 2. Sequence of Enhanced Initialization. The parallel instance initializes without warm start. The main instance selects either its old solution or that of the parallel instance to warm start itself, depending on the costs. The hatched bars indicate the optimization procedure and time indices of trajectories belong to when planning stated.

article utilizes an Enhanced Initialization $\boldsymbol{\tau}^{\mathrm{I}}_{n,(\cdot)}$ produced by a parallel exploring instance, which itself uses two different kinds of initializations. For the latter, we tackle the problem by offering new locally optimal solutions by adjusting the cost function of the local problem.

III. PARALLEL PLANNING AND ENHANCED INITIALIZATION

Avoiding poor initializations is not trivial as it requires knowledge of the area around a locally optimal solution. The high-dimensional search space $\mathcal{T}$ and the nonlinear constraints of (5) make explorations complicated. In addition, online planning usually exploits warm starts which eliminates explorative behavior and tends to find solutions similar to the previous one but on the other hand is important to decrease planning time as it essentially implies a replanning behavior that distributes the optimization across multiple timestamps.

To keep the benefits of warm starts and at the same time overcome its drawbacks, the presented approach explores the current feasible set by a parallel planning instance. Fig. 2 shows a sequence diagram that demonstrates how both instances intertwine:

- t_1: Both instances solve the same planning problem with the main instance using its previous solution $\boldsymbol{\tau}^{\star}_0$ and the parallel instance using an external initialization $\boldsymbol{\tau}^{\mathrm{I}}_{n,(\cdot)}$.
- t_2: The main instance warm starts with its previous solution $\boldsymbol{\tau}^{\star}_1$ since the parallel instance is still planning. It finds a suboptimal solution $\boldsymbol{\tau}^{\bullet}_2$.
- t_3: The parallel instance now offers a locally optimal solution $\boldsymbol{\tau}^{\star}_{1,(\cdot)}$ that is preferred over the previous suboptimal solution $\boldsymbol{\tau}^{\bullet}_2$ for initializing the main instance.
- t_4: The parallel solution $\boldsymbol{\tau}^{\bullet}_{3,(\cdot)}$ is suboptimal and thus the previous main solution $\boldsymbol{\tau}^{\star}_3$ is preferred.

Note that the choice for initializing the main instance is only determined by the solution with least costs. It does not necessarily have to be the locally optimal solution. The parallel instance is allowed to take more iterations than the main instance and uses dedicated techniques to generate $\boldsymbol{\tau}^{\mathrm{I}}_{n,(\cdot)}$ which are presented in the following two sections. Since the parallel solution is based on a potentially older environmental state, it does not command the robot directly but is re-optimized in the main instance. This has the additional advantage that the control can always be sent regularly.

A. INITIALIZATION BY LINEAR INTERPOLATION (L-IN)

The most general initialization for the parallel exploration instance is linear interpolation $\boldsymbol{\tau}^{\mathrm{I}}_{(\mathrm{L})}$ between $\mathbf{q}_n$ and $\mathbf{x}_{\mathrm{d}}$ for which the states and controls are:

$$\mathbf{x}_k = \mathbf{q}_n + \frac{(\mathbf{x}_{\mathrm{d}} - \mathbf{q}_n)k}{K}, \tag{9}$$

$$\mathbf{u}_k = \frac{(\mathbf{x}_{\mathrm{f},n} - \mathbf{q}_n)}{K\,\Delta t}. \tag{10}$$

It is assumed to have a solver that does not require the initialization to be feasible, as linear interpolations across the entire space usually violate constraints (5b), (5c), and (5d).

B. MIRRORED INITIALIZATION (M-IN)

A recent local solution of the main instance describes the impacts of a pushing obstacle on the trajectory. A reversed motion opens a new explorative perspective. Therefore, this more enhanced variant mirrors states $\tilde{\mathbf{x}}_k$ of the last solution $\boldsymbol{\tau}_{t_{n-1}}$ of the main instance to initialize the parallel instance.

Consider $\tilde{x}_{i,k}$ and $x_{i,\mathrm{d}}$ as the i-th components of the states $\tilde{\mathbf{x}}_k$ and $\mathbf{x}_{\mathrm{d}}$, respectively:

$$\tilde{\mathbf{x}}_k = \begin{bmatrix}\tilde{x}_{0,k} & \tilde{x}_{1,k} & \dots & \tilde{x}_{i,k} & \dots & \tilde{x}_{D-1,k}\end{bmatrix}^{\mathsf{T}}, \tag{11}$$

$$\mathbf{x}_{\mathrm{d}} = \begin{bmatrix}x_{0,\mathrm{d}} & x_{1,\mathrm{d}} & \dots & x_{i,\mathrm{d}} & \dots & x_{D-1,\mathrm{d}}\end{bmatrix}^{\mathsf{T}}, \tag{12}$$

for each joint $i = 0, 1, \dots, D-1$. Let $\tilde{\mathbf{z}}_{i,k}$ and $\tilde{\mathbf{z}}_{i,\mathrm{d}}$ be homogeneous points in the spatio-temporal domain on $\tilde{\mathbf{x}}_{0:K}$ and $\mathbf{x}_{\mathrm{f},n}$, respectively:

$$\tilde{\mathbf{z}}_{i,k} = \begin{pmatrix}t_k & \tilde{x}_{i,k} & 1\end{pmatrix}^{\mathsf{T}}, \tag{13}$$

$$\tilde{\mathbf{z}}_{i,\mathrm{d}} = \begin{pmatrix}t_K & x_{i,\mathrm{d}} & 1\end{pmatrix}^{\mathsf{T}}. \tag{14}$$

Let $\mathbf{v}_i$ be a unit vector pointing from $\tilde{\mathbf{z}}_{i,0}$ to $\tilde{\mathbf{z}}_{i,\mathrm{d}}$:

$$\mathbf{v}_i = \left[\begin{pmatrix}t_K \\ x_{i,\mathrm{d}}\end{pmatrix} - \begin{pmatrix}t_0 \\ \tilde{x}_{i,0}\end{pmatrix}\right] \cdot \left\| \begin{pmatrix}t_K \\ x_{i,\mathrm{d}}\end{pmatrix} - \begin{pmatrix}t_0 \\ \tilde{x}_{i,0}\end{pmatrix} \right\|_2^{-1}. \tag{15}$$

Matrix $\mathbf{T}_i \in \mathbb{R}^{3\times 3}$ is a homogeneous transformation that transforms points $\tilde{\mathbf{z}}_{i,k}$ onto the t-axis:

$$\mathbf{T}_i = \begin{bmatrix}\mathbf{v}_i & \mathbf{v}_i^{\perp} & \begin{pmatrix}t_0 \\ \tilde{x}_{i,0}\end{pmatrix} \\ 0 & 0 & 1\end{bmatrix}, \tag{16}$$

with $\mathbf{v}_i^{\perp}$ being perpendicular to $\mathbf{v}_i$. Finally consider matrix $\mathbf{M} \in \mathbb{R}^{3\times 3}$ that mirrors along its second component:

$$\mathbf{M} = \operatorname{diag}\begin{pmatrix}1 & -1 & 1\end{pmatrix}. \tag{17}$$

The complete mirroring operation is now performed as:

$$\begin{aligned}\mathbf{Z}_i &= \begin{bmatrix}\mathbf{z}_{i,0} & \mathbf{z}_{i,1} & \dots & \mathbf{z}_{i,K}\end{bmatrix} \\ &= \mathbf{T}_i\mathbf{M}\mathbf{T}_i^{-1}\begin{bmatrix}\tilde{\mathbf{z}}_{i,0} & \tilde{\mathbf{z}}_{i,1} & \dots & \tilde{\mathbf{z}}_{i,K}\end{bmatrix}.\end{aligned} \tag{18}$$

It essentially is a transformation of $\tilde{\mathbf{z}}_{i,k}$ onto the t-axis followed by a sign flip and a backward transformation. The mirrored points $\mathbf{z}_{i,k}$ are constrained to joint limits and to a total time of $K\Delta t$. If the sequence is not of full temporal

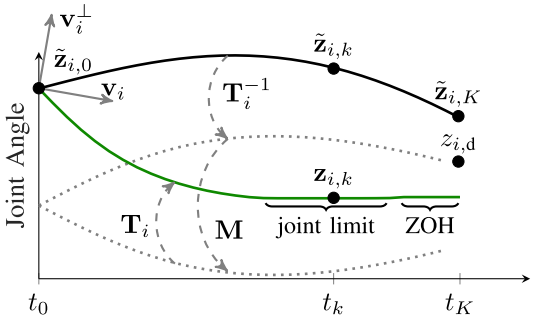


FIGURE 3. Process of spatio-temporal mirroring of a local solution along the direct connection between the first state of the local solution and the target state. The operation takes place for each dimension of the state vector. The mirrored solution (green) is subject to joint limits and extended by ZOH if needed.

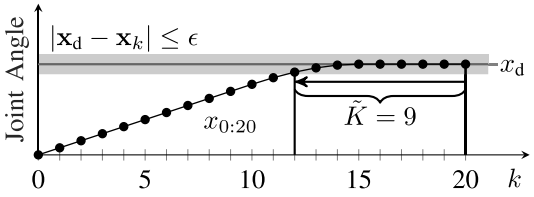


FIGURE 4. Illustration of determining the number $\tilde{K}$ of states of the local solution that lie within an ϵ region around the current target state. Counting starts at the end of the planning horizon.

length, it is continued by replicating $\mathbf{z}_{i,K}$, i.e. Zero Order Hold (ZOH). Fig. 3 shows an example of the mirroring procedure. To match the uniform grid (3) in terms of Δt and a total number of K points, the sequence is resampled into $\mathbf{x}_{0:K}$. After performing the above steps for all joints, the controls $\mathbf{u}_k$ are calculated based on finite differences and both, $\mathbf{x}_{0:K}$ and $\mathbf{u}_{0:K-1}$ are stored in $\boldsymbol{\tau}^{\mathrm{I}}_{(\mathrm{M})}$.

IV. SUB-GOAL TRACKING

To adjust the cost function and promote locally optimal solutions that avoid standstills this method swaps $\mathbf{x}_\mathrm{d} = \mathbf{x}_{\mathrm{f},n}$ with a local sub-goal $\mathbf{x}_\mathrm{d} = \mathbf{x}_{\mathrm{s},n} \in \mathcal{P}$ that is drawn from a set $\mathcal{P} \subset \mathcal{X}_\mathrm{R} \cap \mathcal{X}_\mathrm{S}(t_n)$ of candidates at t_n and reachable by the current feasible set of trajectories. This way, non-constant locally optimal solutions are preferred over standstills. Since this method acts on the reference $\mathbf{x}_\mathrm{d}$ only, it does not interfere with Enhanced Initialization introduced in the previous section.

A. SET OF CANDIDATES

By changing the target $\mathbf{x}_\mathrm{d}$ of the local planning problem, it further deviates from the global problem (5) and the CLT more likely becomes suboptimal. To limit the extent of suboptimality, $\mathcal{P}$ should be close to the globally optimal solution. Since $\mathcal{P}$ does not directly depend on time, it is purely geometric and can be expressed as a path, e.g. as a list of linearly interpolated waypoints. The approach, therefore, utilizes an optimal path planner to find the shortest path $\mathcal{P}$ in parallel to MHP.

The points, $\mathbf{p}_\mathrm{s} \in \mathcal{P}$ and $\mathbf{p}_\mathrm{g} \in \mathcal{P}$, indicate the start and goal of $\mathcal{P}$, respectively. If $\mathbf{x}_{\mathrm{f},n}$ is inside of a hypersphere with radius $\gamma \in \mathbb{R}^+$ around $\mathbf{q}_n$, sub-goal tracking is skipped ($\mathbf{x}_\mathrm{d} = \mathbf{x}_{\mathrm{f},n}$) to avoid planning trivial paths. In all other cases, the motion towards $\mathbf{x}_{\mathrm{f},n}$ is assumed to be potentially sensitive to locally optimal standstills and sub-goal tracking becomes necessary ($\mathbf{x}_\mathrm{d} = \mathbf{x}_{\mathrm{s},n}$).

If there is no previous path, or if $\mathbf{x}_{\mathrm{f},n}$ is outside of a hypersphere with radius $\delta \in \mathbb{R}^+$ around the path's current endpoint $\mathbf{p}_\mathrm{g}$, an optimal path planner generates/updates $\mathcal{P}$ based on $\mathbf{p}_\mathrm{s} = \mathbf{q}_n$ and $\mathbf{p}_\mathrm{g} = \mathbf{x}_{\mathrm{f},n}$ at t_n. The hypersphere models a hysteresis, which prevents permanent replanning, especially in the case of moving references $\mathbf{x}_{\mathrm{f},n}$. To further reduce the planning effort for moving references, multi-query planners such as PRM* should be preferred to create $\mathcal{P}$.

Besides new reference values $\mathbf{x}_{\mathrm{f},n}$, replanning $\mathcal{P}$ is also triggered by updates of $\mathcal{X}_\mathrm{S}(t_n)$. Due to the required planning time, only static (and temporary static) obstacles are considered. They often represent the largest and most bulky part of the workspace. Temporary static obstacles are dynamic obstacles that temporarily do not (or only slightly) move. The distinction can be made using background detection approaches.

B. SUB-GOAL SAMPLING

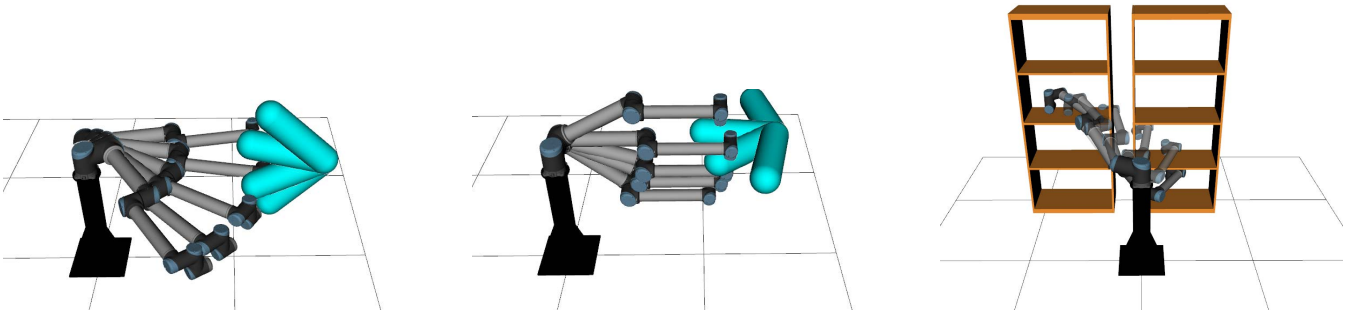
At t_n, the sub-goal $\mathbf{x}_{\mathrm{s},n}$ is drawn from $\mathcal{P}$ based on $\boldsymbol{\tau}_{n-1}$. This is done by virtually moving $\mathbf{x}_{\mathrm{s},n}$ along $\mathcal{P}$ starting at $\mathbf{p}_\mathrm{s}$. To ensure that the resulting sub-goal is within range of the local planning problem, it is moved by path length $s \in \mathbb{R}_0^+$:

$$s = \max\left\{\lambda(\tilde{K} - K_0)\Delta t,\ 0\right\}, \tag{19}$$

in which $\tilde{K} \in \mathbb{N}$ denotes the number of successive states $\mathbf{x}_k$, which are sufficiently close $\left(|\mathbf{x}_\mathrm{d} - \mathbf{x}_k| \leq \epsilon \in \mathbb{R}^+\right)$ to the current target $\mathbf{x}_\mathrm{d}$ for $k = K, K-1, \ldots, 0$. Parameter $\lambda \in \mathbb{R}^+$ can be interpreted as a pseudo-velocity at which the robot is expected to move, e.g. the robot's joint velocity limit. Together with Δt, s represents the available distance of the planning horizon before it is maxed out. Parameter $K_0 \in \mathbb{N}$ ensures that there is always some clearance. Fig. 4 demonstrates how $\tilde{K}$ is chosen.

At the first planning cycle ($n = 0$), there is no previous solution and thus $\mathbf{x}_{\mathrm{s},0} = \mathbf{p}_\mathrm{s} = \mathbf{q}_0$. The resulting solution leads to $\tilde{K} = K + 1$ since all states lie on $\mathbf{q}_0$ and no motion is necessary. In the subsequent cycle ($n = 1$), $\mathbf{x}_{\mathrm{s},1}$ virtually moves along $\mathcal{P}$ according to (19) and the robot starts moving according to (8). This process repeats for all subsequent cycles. The combination of $\mathcal{P}$ starting at $\mathbf{q}_n$ and choosing $\mathbf{x}_{\mathrm{s},n}$ so that it is robustly embedded in the previous solution $\boldsymbol{\tau}_{n-1}$ ensures that $\mathbf{x}_{\mathrm{s},n}$ lies within the planning horizon of the next planning instance.

To ensure that $\mathbf{x}_{\mathrm{s},n}$ does not collide with a dynamic obstacle, s is reduced gradually until $\mathbf{x}_{\mathrm{s},n} \in \mathcal{X}_\mathrm{D}(t_n)$. In the worst case, a big-enough dynamic obstacle constantly pushes the robot back. However, usually the dynamic obstacle is either classified as background eventually and $\mathcal{P}$ passes it, it moves out of the way by itself, or the horizon reaches around it.



(a) Scenario A: Vertically moving dynamic obstacle interferes with the robot during horizontal motion.

(b) Scenario B: Horizontally moving dynamic obstacle interferes with the robot during vertical motion.

(c) Scenario C: Pick (left shelf) and place (right shelf) task in a static environment.

FIGURE 5. Description of the scenarios for evaluation.

In particular, the latter is encouraged by sub-goal tracking as it preserves the flexibility of the local planner and imposes fewer requirements on the path quality compared to classical path tracking approaches.

V. EVALUATION

The evaluation comprises four experiments based on three scenarios shown in Fig. 5. The first two show a dynamic obstacle disturbing the robot during a horizontal and vertical motion, respectively and are utilized in Experiments I – III. The third scenario is static and includes two shelves from which the robot picks and places a fictional object. It is utilized in Experiment IV only. The obstacles and robot links are modeled by swept-spheres, which allow for efficient distance calculation in (4) and (7) [24]. More details on the exact structure of the scenarios and the robot's start and target configurations can be found in Appendix A.

To exploit sparsity and efficiently maintain changing constraint dimensions that arise from dynamic environments, the optimization problem of MHP is transformed into a hypergraph representation [5]. MHP is embedded in a custom C++ ROS framework that handles the interface to the environment and the computation of distances, constraints, and costs. Its performance under multiple dynamic obstacles in the form of a person has been demonstrated by Krämer et al. [3]. The framework utilizes the interior point method IPOpt [14] to solve the planning problem as well as any parallel instances using the linear solver MA27 from [25]. The parameters for this are summarized separately for all experiments in Appendix B.

The robot is a Universal Robot 10 ($D = 6$), which is simulated in Gazebo to simplify setting up, running, and analyzing the experiments. However, the motion commands generated by MHP can also be sent directly to the real robot, which is demonstrated for Experiment IV in the accompanying video. Both, the simulation and planning framework run on a computer equipped with an Intel i7-8700 CPU at 3.2 GHz using 32GB RAM RAM at 2666MHz under Ubuntu 18.04.

A. EXPERIMENT I (Scenarios A + B)

The first experiment investigates the influence of linear and mirrored initializations on the resulting quality of the local solution of MHP. For this, the costs of these local solutions are compared with the costs of a baseline solution. The Baseline is MHP initialized with $\boldsymbol{\tau}^{\mathrm{I}}_{(\mathrm{L})}$ at the beginning and warm starting afterwards. The variants L-IN and M-IN do not use warm start and always re-initialize with $\boldsymbol{\tau}^{\mathrm{I}}_{(\mathrm{L})}$ or $\boldsymbol{\tau}^{\mathrm{I}}_{(\mathrm{M})}$, respectively. All variants solve the same local problem and use the parameter given in Appendix B. For better comparability, the robot ignores incoming motion commands and remains in its initial configuration.

Fig. 6 shows cost values of all three variants along with the pitch angle of the obstacle in Scenario A. In the beginning, the path in front of the robot is free and the costs correspond to a local trajectory pointing directly to the target. All variants share almost identical costs up to $t = 1\,\mathrm{s}$ and thus also share the same solution. The costs start to increase as the intruding obstacle starts pushing away the local solutions. The costs of the Baseline keep increasing until the obstacle stops at $t = 2\,\mathrm{s}$ since the local solution warm starts and thus lacks explorative capabilities. The other two variants feature exploratory abilities due to their type of initializations and are able to find a better local solution, which eventually lies on the level of a direct trajectory. Instead of being pushed downwards entirely, the local solutions of these variants pass the obstacle on the upside. Besides a clear improvement to the Baseline, there are also differences within the type of initializations. For example, M-IN finds the locally better solution earlier than L-IN.

Fig. 7 shows the costs for Scenario B in the same way. Again, in the beginning, the path is clear for a trajectory that points directly to the target. Due to the intruding obstacle, the costs start to increase for all three variants. For the Baseline, they rise to their maximum value at $t = 1.75\,\mathrm{s}$ and then fall when the obstacle has passed, and finally stay constant after the obstacle stops. However, the costs still do not reach the locally optimal costs as for variant L-IN and M-IN. Both perform equally in this scenario. Their initially stronger increase

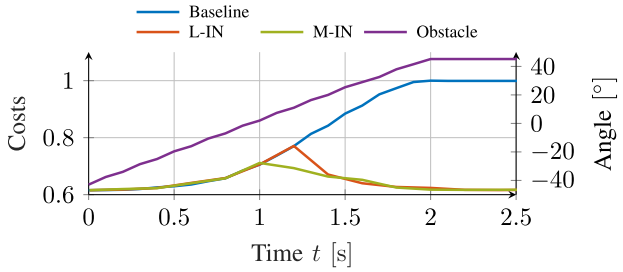


FIGURE 6. Normalized costs of the local solutions of Baseline, L-IN, and M-IN as well as the pitch angle of the obstacle for Scenario A in Experiment I.

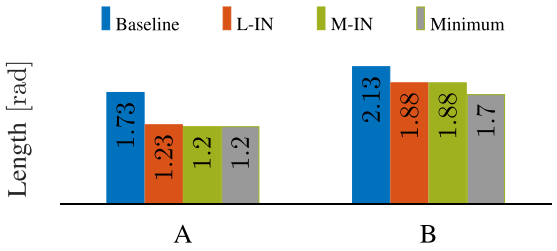


FIGURE 8. Spatial lengths of the CLTs for Experiment II in Scenario A and B. The minimum length corresponds to the shortest distance between start and goal configuration.

compared to the main instance until $t = 1.2\,\mathrm{s}$ indicates that these initializations can also lead to a locally worse solution than Baseline. However, this is not critical in normal operation when Enhanced Initialization as shown in Fig. 2 is enabled, since the parallel solution is only used to initialize the main instance if it constitutes lower costs.

B. EXPERIMENT II (SCENARIOS A + B)

Since the first experiment demonstrated the effectiveness of solving a local problem initialized by L-IN or M-IN instead of warm starting, this experiment investigates the effects on the CLT when the main instance is initialized by parallel solutions according to Fig. 2. Consequently, this experiment deals with the following variants: the Baseline variant without Enhanced Initialization, which initializes itself at the beginning via $\boldsymbol{\tau}^{\mathrm{I}}_{(\mathrm{L})}$ and subsequently via warm start, and two variants with a parallel instance initialized by $\boldsymbol{\tau}^{\mathrm{I}}_{(\mathrm{L})}$ or $\boldsymbol{\tau}^{\mathrm{I}}_{(\mathrm{M})}$ used for enhanced initialization of the main instance. Their parameters are given in Appendix B. To generate a CLT the robot no longer ignores the motion commands. This experiment utilizes the length of the CLTs as a measure to compare their quality.

Fig. 8 compares the spatial lengths of the CLTs for Scenario A and B. In Scenario A, the Baseline clearly produces the longest distance (1.73 rad) compared to L-IN (1.23 rad) and M-IN (1.2 rad), since it avoids the obstacle in below and moves in front of it. There is hardly any difference between both variants with Enhanced Initialization in this scenario when it comes to path length. Both almost match the shortest distance. However, inspecting the second joint's CLT in Fig. 9 shows differences in terms of when a locally better solution is found. Up to $t = 1.3\,\mathrm{s}$, all three variants tend to avoid the obstacle downwards before variant M-IN selects a locally better solution upwards. Variant L-IN tends to avoid the obstacle downwards for a little longer before it also selects a new solution. It can be seen that the distance already traveled in joint 2 is revised which reduces the fluidity of the robot's motion. Variant M-IN, successfully recognizes a locally better solution earlier and passes the obstacle below with less directional changes.

The results of Scenario B are similar to Scenario A except that now the second and third joints perform the actual motion and the first joint is used for evasion. Fig. 10 shows again a tendency in the wrong direction before variant M-IN and L-IN find a new locally better solution. The Baseline stays with the initial tendency and rides out the obstacle.

C. EXPERIMENT III (SCENARIOS A + B)

The previous results demonstrated that Enhanced Initialization leads to better local solutions. However, this does not necessarily imply that they are also part of the global solution. The third experiment therefore extends the previous one by comparing the similarity between local solutions and global solutions generated by three global methods. Besides GTO, these methods are STOMP and the path planner PRM*, which both are sampling-based. GTO, STOMP, and PRM* have been integrated into the same C++ framework as MHP for maximum comparability. They share the same objective

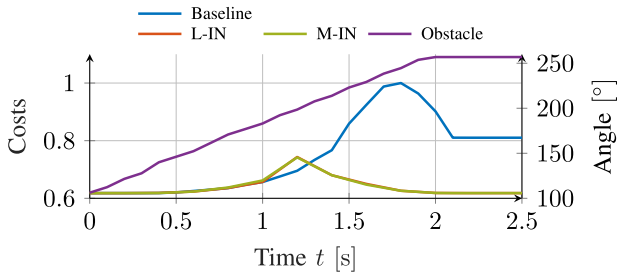


FIGURE 7. Normalized costs of the local solutions of Baseline, L-IN, and M-IN as well as the yaw angle of the obstacle for Scenario B in Experiment I.

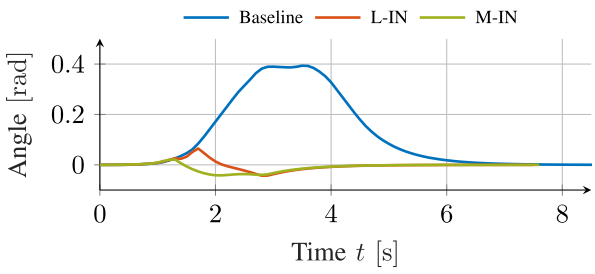


FIGURE 9. CLTs for the second joint in Scenario A for Baseline, L-IN, and M-IN.

TABLE 1. Distance between local and global solutions at certain timestamps. Bold values indicate the global solution with the shortest path length. Lower values mean better similarity.

		$t=0$			$t=0.5$			$t=1$			$t=1.5$			$t=2$			$t=2.5$			$t=3$		
		GTO	STOMP	PRM*	GTO	STOMP	PRM*	GTO	STOMP	PRM*	GTO	STOMP	PRM*	GTO	STOMP	PRM*	GTO	STOMP	PRM*	GTO	STOMP	PRM*
A	Baseline	0.18	0.21	**0.03**	0.27	**0.98**	2.06	0.86	**1.36**	6.39	9.69	**9.28**	9.74	8.24	8.02	**8.00**	0.07	1.28	**2.73**	0.03	0.66	**3.33**
	L-IN	0.18	0.21	**0.03**	0.72	**1.07**	5.15	7.35	**1.37**	3.41	9.68	**9.33**	9.94	0.16	0.31	**0.77**	0.01	0.18	**0.20**	0.01	0.17	**0.04**
	M-IN	0.19	0.22	**0.02**	0.39	**0.97**	3.43	7.28	**1.37**	3.54	0.35	**0.93**	3.49	0.17	0.11	**0.26**	0.02	0.19	**0.05**	0.01	0.16	**0.03**
B	Baseline	**0.00**	0.40	0.00	0.47	0.35	**0.97**	0.98	**1.01**	7.64	14.46	**12.73**	15.14	9.85	7.62	**7.24**	6.50	3.60	**3.53**	2.77	2.38	**2.38**
	L-IN	**0.00**	0.40	0.00	0.25	0.57	**1.26**	1.03	**0.97**	7.51	14.50	**12.79**	12.17	1.38	2.21	**2.56**	1.40	0.89	**1.03**	1.88	0.79	**0.75**
	M-IN	**0.00**	0.40	0.00	0.45	0.35	**0.99**	1.10	**1.01**	7.97	14.45	**12.76**	14.84	1.36	2.29	**2.27**	1.39	0.92	**1.08**	1.88	0.81	**0.76**

functions, constraints, and distance calculation framework if needed. GTO is formally closest to solving the global problem, but itself depends on initialization, which in this case is $\boldsymbol{\tau}^{\mathrm{I}}_{(\mathrm{L})}$. STOMP is taken from [26], with custom joint limit clipping and the sampling matrix from the original publication [16]. It is not fully equivalent to GTO due to a fixed trajectory duration. PRM* is taken from the Open Motion Planning Library [27]. With sufficient planning time, it is asymptotically optimal in finding the shortest path, but it is purely geometric. To cover a wide range of global solutions, all three global methods are part of the evaluation. The parameters for the MHP variants are the same as in the previous experiment and those for STOMP and PRM* can be found in Appendix B.

A local solution is optimal in the sense of the global problem if it is a substrategy of the globally optimal solution [23]. Since in all three scenarios the obstacle is dynamic and the robot is moving, the globally optimal solution also varies. For this reason, global solutions are determined at different timestamps ($t = 0\,\mathrm{s}, 0.5\,\mathrm{s}, \ldots, 3\,\mathrm{s}$) based on the current state of the robot and obstacle. The similarity between the local solution and the global solutions of the same timestamp is considered as a measure of quality. The closer a local solution is to the globally optimal solution of its timestamp, the more it conforms to the local optimal solution. The similarity measure is inspired by the Frechet and Hausdorff distance and is defined in (20) in Appendix C. Lower values mean more similar.

Table 1 shows the distances between local solutions of Baseline, L-IN, and M-IN from Experiment II and global solutions. Due to the differences in the global methods, the shortest global solution is marked in bold. STOMP often generates shorter paths than PRM* when large obstacles obstruct the direct path, since PRM* uses a limited planning time of 15 s in which it cannot always generate enough samples. GTO and STOMP usually generate very similar path lengths, often differing only because of their ways of optimizing the motion planning problem.

For the first three timestamps in Scenario A, the local solutions of all three variants share approximately the same but increasing distance. At $t = 1.5\,\mathrm{s}$, the local solution of M-IN is significantly closer to the global solution than the other two variants. Fig. 11 demonstrates how the local solution of L-IN plans below (positive angle) the obstacle, while M-IN follows the global solution of STOMP above (negative angle) the obstacle. Back to Table 1 and one timestamp

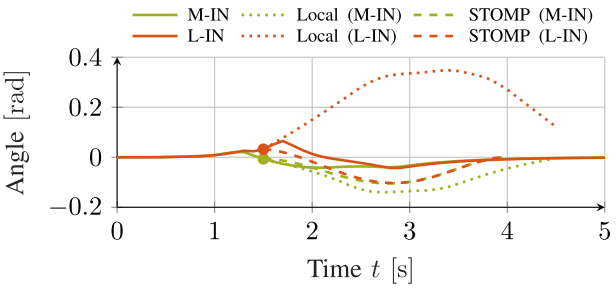


FIGURE 11. Second joint angle of the CLTs for L-IN and M-IN along with their local solutions and global solutions from STOMP at $t = 1.5\,\mathrm{s}$ in Scenario A.

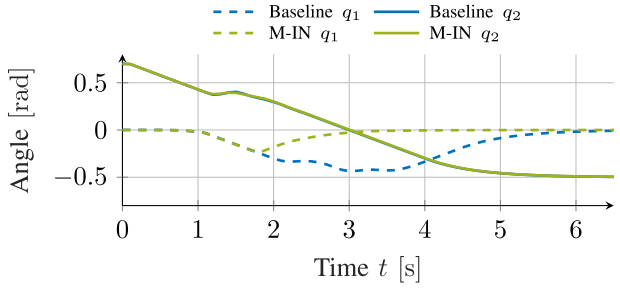


FIGURE 10. CLTs for the first two joints in Scenario B for Baseline and M-IN.

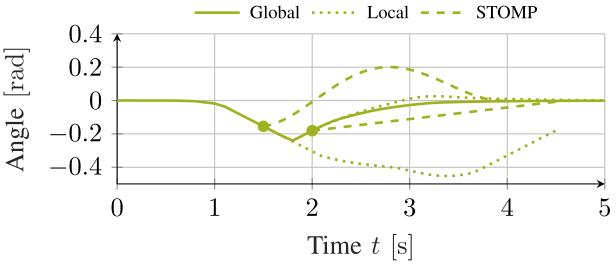


FIGURE 12. First joint angle of the CLTs for L-IN and M-IN along with their local solutions and global solutions from STOMP at $t = 1.5\,\mathrm{s}$ and $t = 2\,\mathrm{s}$ in Scenario B.

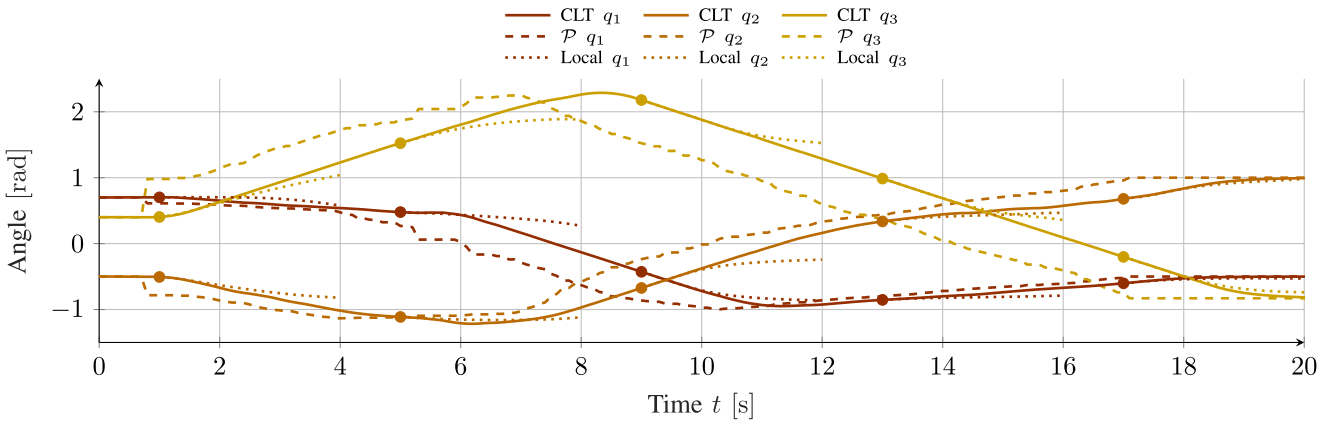


FIGURE 13. Superposition of the CLT of M-IN+ST and $\mathcal{P}$ with local solutions at $t = 0\,\text{s}, 0.5\,\text{s}, \ldots, 3\,\text{s}$ for the first three joints in Scenario C.

further, the behavior is similar for variant L-IN. The Baseline almost always shows the worst similarity. This reinforces the results from Experiment I that mirroring tends to find a better local solution earlier on Scenario A than linear interpolation, with evidence that it additionally contributes to the global optimal solution due to its similarity to the optimal[2] strategy.

For the first three timestamps in Scenario B, the local solutions of all three variants are close to the global solution. At $t = 1.5\,\text{s}$, they all deviate to a similar but significant extent. The reason is that at this point the global solutions have just changed direction and both, L-IN and M-IN are still following the old direction. This can also be seen in Fig. 12 as a comparison between timestamps $t = 1.5\,\text{s}$ and $t = 2\,\text{s}$. It shows the superposition of the local solution and CLT of M-IN together with the global solution of STOMP for the first joint. At $t = 1.5\,\text{s}$ the local solution passes the obstacle in front when STOMP already suggests to pass behind. At $t = 2\,\text{s}$, the local solution follows that of STOMP. Although M-IN usually tends to find a better solution earlier than L-IN, the delay time between mirrored initialization of the parallel instance followed by optimization and re-initialization of the main instance (see Fig. 2) can take up to two planning cycles, which is crucial in this case. For the remaining timestamps in Table 1, all variants get closer to the global solution as the obstacle passed by, whereby M-IN is slightly better than L-IN. The Baseline also improves but still has the least similarity.

D. EXPERIMENT IV (SCENARIO C)

The fourth experiment deals with sub-goal tracking from Section IV and investigates to what extent it can improve the CLT in terms of standstills and how much it differs to solutions from global methods. Since the Baseline of the previous experiments has already proven to be inferior, it no longer appears in this experiment. Instead, variant M-IN without sub-goal tracking and a new variant M-IN+ST with sub-goal tracking are compared. M-IN+ST uses PRM* without smoothing and any post-processing to generate $\mathcal{P}$. Both variants are compared with three global methods GTO, STOMP, and PRM* from the previous experiment. The parameters are given in Appendix B. The experiment uses Scenario C with two shelves.

Table 2 summarizes the results of this experiment. Variant M-IN is not able to drive out of the left shelf to reach the target and runs into a standstill. The same applies to STOMP, which was not able to determine a valid trajectory even though it was initialized using PRM*. Variant M-IN+ST, on the other hand, is able to move out of the left cabinet and successfully reaches the bottom shelf of the right cabinet due to sub-goal tracking. However, the resulting length of 10.85 rad is higher than that of the global methods PRM* (7.25 rad) and GTO (6.07 rad). Fig. 13 overlays the CLT of M-IN+ST and $\mathcal{P}$ with local solutions at five different timestamps for the first three joints. The joint angle of $\mathcal{P}$ corresponds to the determined sub

TABLE 2. Results for experiment IV.

		M-IN	M-IN+ST	GTO[a]	STOMP[a]	PRM*
Success		×	✓	✓	×	✓
Final length	[rad]	-	10.85	6.07	-	7.25
Length of $\mathcal{P}$	[rad]	-	10.81	-	-	-
Calc. Time	[s]	0.08	0.08	22.57[b]	27.13[b]	14.80

[a] Initialized with PRM*
[b] Excluding planning time of PRM*

TABLE 3. Start and goal configurations.

	A		B		C	
Joint	$\mathbf{q}_0$	$\mathbf{x}_\text{f}$	$\mathbf{q}_0$	$\mathbf{x}_\text{f}$	$\mathbf{q}_0$	$\mathbf{x}_\text{f}$
1	−1	0.2	0	0	0.7	−0.5
2	0	0	0.7	−0.5	−0.5	1.0
3	0	0	−0.7	0.5	0.4	−0.83
4	0	0	0	0	0.07	−0.15
5	0	0	0	0	2.28	1.05
6	0	0	0	0	0	0

[2]Optimal in terms of the shortest global solution of GTO, STOMP, and PRM*.

TABLE 4. Parameters for trajectory optimization and solver. Only deviations from the baseline column on the left are explicitly given. Equal values are replaced by •.

		Exp. I			Exp. II			Exp. III	Exp. IV		
Parameter	Baseline	L-IN	M-IN	Baseline	Main	Parallel (L-IN,M-IN)	GTO	Main	Parallel (M-IN,M-IN+ST)	GTO
K	30	•	•	•	•	•	200	•	•	200
Δt	0.1 s	•	•	•	•	•	0.05 s	•	•	0.05 s
$\mathbf{x}_{\max}, \mathbf{x}_{\min}$[a]	± 3.1 rad	•	•	•	•	•	•	•	•	•
$\mathbf{u}_{\max}, \mathbf{u}_{\min}$[a]	± 0.3 rad s^{-1}	•	•	•	•	•	•	•	•	•
$\mathbf{Q}$	diag$(2, \ldots, 2)$	•	•	•	•	•	•	•	•	•
$\mathbf{Q}_K$	diag$(200, \ldots, 200)$	•	•	•	•	•	•	•	•	•
$\mathbf{R}$	diag$(1, \ldots, 1)$	•	•	•	•	•	•	•	•	•
Relative tolerance[b]	0.001	•	•	•	•	•	•	•	•	•
Iterations[b]	30	50	50	•	50	50	200	•	50	200
Solve time[b]	80 ms	∞	∞	•	∞	∞	∞	•	∞	∞

[a] Equal for all joints.
[b] The solver stops when one of the threshold values is reached.

references $\mathbf{x}_{s,n}$. The local solutions usually reach these values because of a sufficient margin of $K_0 = 8$. The figure further indicates how the resulting global solution corresponds to a smoothed and optimized version of $\mathcal{P}$, which shows how the flexibility of optimization is successfully exploited in this method compared to simply tracking $\mathcal{P}$. However, it is not able to provide a shorter path than initially given by $\mathcal{P}$.

GTO finds the shortest solution, after it was initialized with PRM*. A linear interpolation is too far away from the solution to be found by the optimizer in the highly nonlinear search space. PRM* provides a slightly longer solution than GTO since it terminates after 25 s and cannot optimize a trajectory in a gradient-based and thus objective-based manner. Note that GTO and PRM* are mainly used for references and therefore not tuned for speed. Both methods may generate the solution in a shorter time than given in Table 2. In this experiment, however, the focus lies on the rough order of magnitude that demonstrates the exclusion of GTO and PRM* as online planners.

VI. CONCLUSION AND OUTLOOK

This article presents approaches to improve local solutions in MHP via both, advanced initialization and sub-goal tracking. Optimization with linear or mirrored initialization without warm start already provides local solutions with less final costs. Enhanced Initialization is a method to combine the advantages of warm starts with the exploratory property of a parallel instance. It successfully finds locally better solutions and improves CLTs in terms of final path length (Case I). The results show that M-IN tends to find locally better solutions earlier than L-IN. Additionally, Enhanced Initialization using M-IN is most often closer to the shortest global solution than L-IN or normal MHP. However, there is still room for improving M-IN as it does not always switch to a better and locally close-to-optimal solution in time and also tends to slightly move in the wrong direction first. This is partially limited by design as for mirroring in the right direction it requires a prior tendency in the wrong direction. Another reason is the delay measured from the initialization and optimization of the parallel instance via re-optimization in the main instance.

The changes in the cost function by following a dynamic sub-goal successfully establish local solutions that prevent standstills (Case II/III). A primitive path planned in parallel with MHP is sufficient as the set of sub-goal candidates. The resulting motion is smooth and except for the goal state completely subject to optimization of MHP.

Future work analyzes the influence and opportunities of environmental predictions as well as initializations transformed from operational space. Especially for the former, some promising approaches have emerged in the past to predict the motion of humans that can be incorporated into methods and references in a prescient way. Initializations that originate from operational space might have the ability tackle standstills directly on a more intuitive level and at the same time focus the analysis more on Case II.

APPENDIX A
SCENARIOS

The obstacle of Scenarios A and B is a line-swept sphere (radius of 0.1 m and line length of 0.5 m) with the line's origin located at (1.6 0 0.9) m. The rotation starts at $-45°$ and $-103°$ and transits within 2 s to 45° and 103°, respectively. The center of the shelves footprint of Scneario C are located at (1.15 −0.3 0.025) m and (1.15 0.74 0.025) m for the left and right shelf, respectively. They are both of 0.8 m length, 0.4 m depth, and 2 m height. There is 0.43 m of vertical clearance between each layer. The shelves are modeled by rectangle-swept spheres. The start and goal configurations of each scenario are given in Table 3.

APPENDIX B
PARAMETERS

The sets in (4) are based on the minimum distance $\hat{d}_{\min} = 0.01$ m. The repelling potentials (7) become active when the distance falls below $d_{\min} = 0.1$ m and use a weight of $w = 50$. Furthermore, Table 4 shows the parameters of all variants of trajectory optimization. Table 5 contains the

parameters for sub-goal tracking and Tables 6 and 7 contain parameters of the global methods STOMP and PRM*, respectively.

TABLE 5. Parameters for subgoal tracking.

Parameter	M-IN+ST
δ	0.08 rad
γ	0.1 rad
ϵ	0.1 rad
λ	0.4
K_0	8

TABLE 6. Parameters for STOMP. Only deviations from the left column are explicitly given. Equal values are replaced by •.

Parameter	Exp. III (Ref.)	Exp. IV (Ref.)
Samples	250	100
Iterations[a]	150+5	200+5
Δt	0.01 s	0.1 s
Covariance factor	0.1	0.01
Cost sensitivity (h)	10	•
Rollouts	50	•

[a] Normal iterations + additional iterations after solution is valid.

TABLE 7. Parameters for PRM*. Only deviations from the left column are explicitly given. Equal values are replaced by •.

Parameter	Exp. III (Ref.)	Exp. IV (Ref.)	Exp. IV (M-IN+ST)
Time limit	15 s	25 s	75 ms
Validity resolution	0.01 rad	•	•

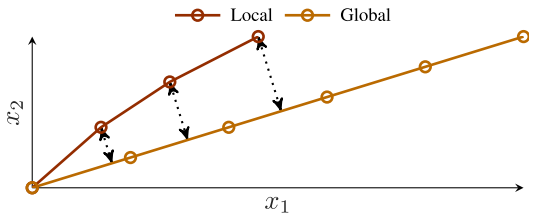


FIGURE 14. Illustration for calculating the distance between points of a short local solution to linear segments of a longer global solution as a measure of similarity.

APPENDIX C
SIMILARITY MEASURE

Consider a local solution $\mathbf{x}_{0:K}$ and a global solution $\hat{\mathbf{x}}_{0:M}$. Then the following metric defines a distance between both paths:

$$\ell = \sum_{k=0}^{K} \min \mathrm{d}(\mathbf{x}_k, \hat{\mathbf{x}}_{0:M}), \tag{20}$$

with $\mathrm{d}(\mathbf{x}_k, \hat{\mathbf{x}}_{0:M})$ denoting a distance function between points $\mathbf{x}_k$ and linearly interpolated segments of $\hat{\mathbf{x}}_{0:M}$. It is adapted from the Frechet and Hausdorff distance to account for usually shorter local solutions. Fig. 14 shows an example of the metric for a two dimensional state space with $K = 3$ and $M = 5$.

REFERENCES

[1] S. Saha and A. A. Julius, "Task and motion planning for manipulator arms with metric temporal logic specifications," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 379–386, Jan. 2018.
[2] M. Krämer, F. I. Muster, C. Rösmann, and T. Bertram, "An optimization-based approach for elasticity-aware trajectory planning of link-elastic manipulators," *Mechatronics*, vol. 75, May 2021, Art. no. 102523.
[3] M. Krämer, C. Rösmann, F. Hoffmann, and T. Bertram, "Model predictive control of a collaborative manipulator considering dynamic obstacles," *Optim. Control Appl. Methods*, vol. 41, no. 4, pp. 1211–1232, 2020.
[4] J. T. Betts, "Survey of numerical methods for trajectory optimization," *J. Guid., Control, Dyn.*, vol. 21, no. 2, pp. 193–207, Mar./Apr. 1998.
[5] C. Rösmann, M. Krämer, A. Makarow, F. Hoffmann, and T. Bertram, "Exploiting sparse structures in nonlinear model predictive control with hypergraphs," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2018, pp. 1332–1337.
[6] C. Wang, J. Bingham, and M. Tomizuka, "Trajectory splitting: A distributed formulation for collision avoiding trajectory optimization," 2021, *arXiv:2111.01899*.
[7] W. Li and R. Xiong, "Dynamical obstacle avoidance of task-constrained mobile manipulation using model predictive control," *IEEE Access*, vol. 7, pp. 88301–88311, 2019.
[8] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proc. 21st Int. Conf. Automated Planning Scheduling*, 2013, pp. 207–215.
[9] G. B. Avanzini, A. M. Zanchettin, and P. Rocco, "Constrained model predictive control for mobile robotic manipulators," *Robotica*, vol. 36, no. 1, pp. 19–38, Jan. 2018.
[10] M. Wang, J. Luo, and U. Walter, "A non-linear model predictive controller with obstacle avoidance for a space robot," *Adv. Space Res.*, vol. 57, no. 8, pp. 1737–1746, 2016.
[11] A. Zube, "Cartesian nonlinear model predictive control of redundant manipulators considering obstacles," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Mar. 2015, pp. 137–142.
[12] M. Kramer, R. J. Velasco-Guillen, P. Beckerle, and T. Bertram, "Comparing online robot joint space trajectory optimization for task space applications," in *Proc. IEEE/ASME Int. Conf. Adv. Intell. Mechatronics (AIM)*, Jul. 2021, pp. 223–230.
[13] S. Byrne, W. Naeem, and R. S. Ferguson, "Efficient local sampling for motion planning of a robotic manipulator," in *Advances in Autonomous Robotics* (Lecture Notes in Computer Science), vol. 7429. Berlin, Germany: Springer, 2012, pp. 164–175.
[14] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, May 2006.
[15] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Proc. Robot., Sci. Syst. IX*, 2013, pp. 1–10.
[16] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 4569–4574.
[17] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robot. Auton. Syst.*, vol. 88, pp. 142–153, Feb. 2017.
[18] M. Imran and F. Kunwar, "A hybrid path planning technique developed by integrating global and local path planner," in *Proc. Int. Conf. Intell. Syst. Eng. (ICISE)*, Jan. 2016, pp. 118–122.
[19] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Commun. ACM*, vol. 22, no. 10, pp. 560–570, 1979.
[20] A. Kuntz, C. Bowen, and R. Alterovitz, "Interleaving optimization with sampling-based motion planning (IOS-MP): Combining local optimization with global exploration," 2016, *arXiv:1607.06374*.
[21] 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. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[22] B. Magyar, N. Tsiogkas, J. Deray, S. Pfeiffer, and D. Lane, "Timed-elastic bands for manipulation motion planning," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3513–3520, Oct. 2019.
[23] R. E. Bellman, *Dynamic Programming*. New York, NY, USA: Dover, 2003.
[24] V. J. Lumelsky, "On fast computation of distance between line segments," *Inf. Process. Lett.*, vol. 21, no. 2, pp. 55–61, Aug. 1985.
[25] Harwell Subroutine Library. *A Collection of Fortran Codes for Large Scale Scientific Computation*. Accessed: May 5, 2021. [Online]. Available: http://www.hsl.rl.ac.uk/
[26] ROS-Industrial. *Stomp_Ros, Commit 201c16c*. [Online]. Available: https://github.com/ros-industrial/stomp_ros
[27] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, Dec. 2012.

MAXIMILIAN KRÄMER was born in Lünen, Germany, in April 1990. He received the B.Sc. degree in computer science and the M.Sc. degree in electrical engineering and information technology from TU Dortmund University, Germany, in 2014 and 2016, respectively, where he is currently pursuing the Dr.-Ing. degree with the Institute of Control Theory and Systems Engineering. His research interests include human–robot-interaction, online trajectory planning, and nonlinear model predictive control.

TORSTEN BERTRAM received the Dipl.-Ing. and Dr.-Ing. degrees in mechanical engineering from the Gerhard Mercator Universität Duisburg, Duisburg, Germany, in 1990 and 1995, respectively. In 1990, he joined the Department of Mechanical Engineering, Gerhard Mercator Universität Duisburg, as a Research Associate. During 1995–1998, he was a Subject Specialist with the Corporate Research Division, Bosch Group, Stuttgart, Germany. In 1998, he returned to Gerhard Mercator Universität Duisburg, as an Assistant Professor. In 2002, he became a Professor with the Department of Mechanical Engineering, Technische Universität Ilmenau, Ilmenau, Germany. Since 2005, he has been a member of the Department of Electrical Engineering and Information Technology, TU Dortmund University, Dortmund, Germany, and as a Professor of systems and control engineering. His research interests include control theory and computational intelligence and their application to mechatronics, service robotics, and automotive systems.