

Real-Time and Accurate Collision-Free Teleoperation via Differentiable Constraint-Based Trajectory Planning

Max Grobbel^{*1}, Tristan Schneider^{*1}, Daniel Flögel¹, Sören Hohmann²

Abstract—In teleoperation, the human operator typically controls only the end-effector pose, which often leads to self-collisions of the manipulator and collisions with environmental obstacles, since joints and links are not controlled individually. A common strategy to mitigate this issue is to enhance the operator’s input using optimal-control-based trajectory planning. As derivative-based solvers require differentiable constraints, existing approaches either approximate robots and obstacles with spheres, reducing geometric accuracy, or approximate derivatives, degrading convergence and increasing computation times. We address these limitations by adapting a recent formulation of differentiable collision-avoidance constraints, based on duality in convex optimization, to the teleoperation setting. The robot is approximated with capsules and the environment with polytopes. We compare the resulting trajectory planning method against state-of-the-art techniques in simulation with varying numbers of obstacles and evaluate it on a UR5e manipulator in a real-world teleoperation test. Results show that our approach achieves lower computation times while enabling more accurate obstacle modeling, leading to smoother and collision-free end-effector teleoperation.

I. INTRODUCTION

Teleoperation of robotic manipulators is an upcoming field of research with classical applications in hazardous areas like nuclear power plant dismantling or in hard-to-reach places such as space applications [1]. Bringing human experts, e.g., surgeons, into remote places or data collection for machine learning applications are more modern applications of teleoperation [2–4].

A central component of teleoperation systems is the input device, which can be broadly classified as haptic or non-haptic. Haptic devices provide rich feedback and often replicate the kinematic chain of the controlled robot, enabling the operator to manipulate not only the end-effector but also individual robot links. However, haptic input devices are expensive, preventing widespread applications. In contrast, non-haptic input devices, e.g., camera tracking or motion tracking systems, are available for lower prices, but provide less feedback to the user, often leading to lower performance. Also, and more importantly, non-haptic input devices only allow the control of the end-effector pose. As indicated in [4], this limitation leads to a larger number of collisions of the manipulator with itself and the environment, since the operator must implicitly account for the robot’s full inverse kinematics.

¹Max Grobbel, Tristan Schneider and Daniel Flögel are with FZI - Forschungszentrum Informatik, Karlsruhe, Germany grobbel@fzi.de

²Sören Hohmann is with the Department of Electrical Engineering, Karlsruhe Institute of Technology, Karlsruhe, Germany

* equal contribution

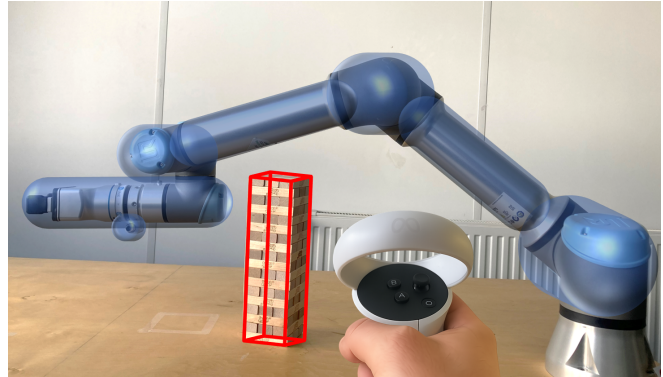


Fig. 1: Teleoperated robotic manipulator. For collision avoidance, the manipulator is approximated by capsules, while environmental obstacles are modeled as polytopes.

This shows the necessity to support the human operator in teleoperation scenarios, especially with collision avoidance. A commonly used strategy to assist the human operator in teleoperation setups is the application of shared control algorithms [5]. Those can be categorized into algorithms that generate a single set point for low-level controllers (e.g. [6]) and approaches, which use the human input as a tracking reference and plan slightly adjusted trajectories, either in task space or in joint space.

Three widely adopted approaches to trajectory planning in robotics are based on sampling methods [7], optimal control formulations [8], and learning-based algorithms [9]. While sampling- and learning-based methods generally lack inherent mechanisms for enforcing constraints, optimal control approaches explicitly exploit system dynamics, naturally accommodate state and environmental constraints, and allow for the formulation of multiple objectives, provided these objectives are expressed in a differentiable form—a requirement stemming from their reliance on derivative-based optimization solvers. Moreover, using forward kinematics within the optimal control formulation implicitly resolves the inverse kinematics and generates task-space motions [10].

Trajectory planning in teleoperation can be regarded as a specialized instance of robotic trajectory planning. The presence of a human operator in the loop introduces additional requirements, most notably real-time capabilities to ensure rapid responsiveness and the ability to adapt to time-varying reference trajectories arising from stochastic human input. Optimal control-based approaches are well suited to these demands, as they combine dynamic models with constraint handling and real-time optimization. Consequently, such

methods have been widely applied in teleoperation contexts [11–14].

Collision between the robot and its environment, including self-collisions, is avoided by enforcing a minimum distance between bodies. Optimization-based methods require differentiable constraint functions, which poses a challenge for collision avoidance: the minimum-distance computation between two bodies is itself a minimization problem and in general non-differentiable. Existing teleoperation approaches address this issue either by approximating robots and obstacles with numerous spheres [15], which leads to a trade-off between approximation accuracy and computational cost, or by employing non-differentiable distance computations with subsequently approximated gradients, as in [14].

Recent research has introduced an algorithm for differentiable collision constraints for convex bodies [16]. This approach was refined by [17], who reduced the computational demand and applied it to a path planning problem for mobile platforms, though the computation time remained above one second, limiting real-time use. The underlying idea is to reformulate the minimum-distance computation between two bodies—originally a nested, non-differentiable optimization—into a dual maximization problem, yielding a set of differentiable inequality constraints that enforce minimum separation.

The main contribution of this work is an optimal control formulation for teleoperation that contains collision-avoidance constraints for accurate geometric robot and environment models. Specifically, we extend the duality-based approach of [17] by introducing capsules as additional geometric primitives and approximating the geometry of an industrial robot manipulator with capsules (Figure 1). The resulting constraints are directly embedded into a model predictive trajectory planner, yielding a real-time capable method that ensures both self-collision avoidance and safe interaction with polytope-shaped environmental obstacles.

We evaluate this approach in terms of computational efficiency and tracking accuracy, comparing it against state-of-the-art teleoperation methods in both simulation benchmarks and physical robot experiments.

II. RELATED WORK

Collision avoidance with the environment and with the robot itself is essential for teleoperation tasks with motion tracking devices, since the operator only controls the end-effector pose. The real-time requirements and the time-dependent tracking objective make teleoperation a distinct subproblem of trajectory planning for robotic manipulators. In the following, we first provide an overview of collision-avoidance formulations within optimal control for robotic manipulators in general, before reviewing approaches tailored to the specific requirements of teleoperation.

A. Collision avoidance for robotic manipulators

Collision-avoidance for manipulators in optimal control can be divided into five categories: joint-space formulations, and task-space formulations based on spheres, witness

points, dual collision-multipliers, or other approximation-based methods such as finite-difference or scaling techniques.

The first group formulates collision constraints in joint space [12], where each individual joint angle is restricted to remain within a predefined interval. While such constraints are straightforward to incorporate into optimization problems, the decoupled consideration of joint angles leads to a conservative approximation of the free space.

A common approach for task-space formulation is based on the approximation of all relevant bodies through spheres. Depending on the geometry and the desired accuracy, the number of spheres can become large. The distance between two spheres is computed by the Euclidean distance between the two center points and the subtraction of both radii. The spheres of all possible collision pairs need to be checked for collision. Those distance functions are differentiable with respect to the robot configuration and can therefore be directly integrated into an optimization problem [11, 13, 18].

Another approach is based on the widely used trajectory optimization algorithm of Schulman [19]. The collision avoidance relies on fast distance-computation techniques, like the Gilbert-Johnson-Keerthi (GJK) algorithm, originating in the domain of video games. These algorithms return both the minimum distance between two bodies and the corresponding closest points—also referred to as witness points [20]. Schulman’s method approximates the derivative of the distance with respect to the robot configuration (i.e., joint angles of a manipulator), resulting in a linear, differentiable approximation of the distance. The witness points are assumed to remain fixed in the local position on the bodies. The optimization problem is solved with a sequential-quadratic-programming (SQP) approach, giving the opportunity to linearize around a new solution during the SQP loop. However, the assumption of fixed witness points can lead to inaccuracies for larger configuration changes, limiting the method’s robustness in complex teleoperation scenarios.

Finite differences provide another option for estimating the derivative of non-differentiable algorithms [21] with the downsides of numerical differentiation. An alternative strategy avoids explicit distance computation by expanding all relevant objects with a scaling factor [22]. With a scaling factor larger than one, the objects are not in collision. The resulting equations are differentiable with respect to the robot configurations. As a downside, this method only enforces collision avoidance without respecting a prescribed minimum distance.

These limitations motivate more principled formulations, such as dual collision-multiplier approaches, which provide differentiable constraints while explicitly enforcing minimum distance requirements [16]. In this method, the distance computation, which is a minimization problem by definition, is transformed into a system of inequalities with the exploitation of the dual problem. While the initial formulation was validated for mobile platforms, subsequent work has extended the approach to robotic manipulators [23, 24], albeit without achieving real-time performance. A recent adaptation

[17] demonstrated promising improvements regarding the computation time for path planning of mobile robots, supporting ellipsoids and polytopes as geometry approximations for the relevant obstacles.

B. Collision avoidance in teleoperation for robotic manipulators

Although some methods for collision avoidance in teleoperation without optimal control exist (e.g. [25]), most approaches rely on optimal control, typically using geometric modeling through spheres or witness points.

For example, [13] approximate both the robot and environmental obstacles with spheres. They approximate every robot link with three spheres, which leads to a very conservative behavior. The optimization problem is solved using an SQP scheme and contains constraints to prevent collision with the environment, whereas self-collision is not addressed.

In contrast, [14] integrate both environmental and self-collision constraints into the optimization problem using the witness-point approach. This method can handle arbitrary convex geometries and leverages the Flexible Collision Library (FCL) to compute distances and witness points. The resulting optimization problem is solved with an SQP scheme. A key limitation, however, is the linear approximation of the distance function and its gradient, possibly leading to convergence issues and thus requiring a larger number of SQP iterations, leading to long computation times.

To date, no teleoperation approach has provided differentiable collision-avoidance constraints together with accurate geometric approximations that can be integrated into the optimization problem, while simultaneously ensuring real-time feasibility and handling both self-collision and environmental collision.

III. DUALITY BASED COLLISION CONSTRAINTS

To derive differentiable collision-avoidance constraints, we first introduce geometric approximations of the manipulator and the surrounding obstacles. The distance computation between two bodies—formulated as a minimization problem by definition—is then reformulated as a system of inequalities [17]. These can subsequently be integrated into the optimal control problem (OCP) as collision avoidance constraints.

A. Geometric Approximation

All bodies are mathematically formulated as subsets $\mathcal{B} \subset \mathbb{R}^3$ of the three-dimensional work space. A convex polytope \mathcal{P} can be represented as the convex hull of its vertices, which is given by

$$\mathcal{P} = \{ \mathbf{V}\boldsymbol{\varphi} \mid \boldsymbol{\varphi} \in \mathbb{R}^{N_v}, \mathbf{1}^\top \boldsymbol{\varphi} = 1, \boldsymbol{\varphi} \geq \mathbf{0} \}, \quad (1)$$

with $\mathbf{V} \in \mathbb{R}^{3 \times N_v}$ being the matrix that contains the N_v vertices as its columns [17]. A capsule \mathcal{C} is defined as the Minkowski sum

$$\mathcal{C} = \{ \mathbf{l} + \boldsymbol{\delta} \mid \mathbf{l} \in \mathcal{L}, \boldsymbol{\delta} \in \mathbb{R}^3, \|\boldsymbol{\delta}\| \leq r \} \quad (2)$$

of a line segment $\mathcal{L} \subset \mathbb{R}^3$ and a closed ball. This results in a set of all points that have an Euclidean distance to the line

segment that does not exceed a certain radius $r \in \mathbb{R}$. The line segment is formulated as a convex polytope with two vertices and the vertex matrix $\mathbf{V}_{\mathcal{L}} = [\mathbf{v}^{(1)} \quad \mathbf{v}^{(2)}]$.

The robotic manipulator is geometrically approximated using multiple capsules (Figure 1). To ensure that all potential collisions are accounted for, the real geometry of the robot links needs to be contained completely in the capsule-based collision model. Each capsule is associated with one link of the kinematic chain. The configuration-dependent locations of the center line end points are given by

$$[\mathbf{v}^{(1)} \quad \mathbf{v}^{(2)}] = \mathbf{R}_{0,i} [\tilde{\mathbf{v}}^{(1)} \quad \tilde{\mathbf{v}}^{(2)}] + \mathbf{p}_{0,i} \mathbf{1}^\top, \quad (3)$$

where $\mathbf{R}_{0,i}$ and $\mathbf{p}_{0,i}$ are the rotation matrix and translation vector describing the transformation of the corresponding i -th link with respect to a fixed global coordinate system. The vectors $\tilde{\mathbf{v}}^{(1)}$ and $\tilde{\mathbf{v}}^{(2)}$ contain the constant coordinates of the end points in the local coordinate system of the link and $\mathbf{1} = [1 \quad \dots \quad 1]^\top$ is a vector of ones.

Obstacles present in the environment are modeled using convex polytopes. Their location does not depend on the configuration. Therefore, no transformation of the vertices is necessary.

B. Minimum Distance Equivalence

The distance between two bodies \mathcal{B}_1 and \mathcal{B}_2 is defined by the optimization problem

$$\begin{aligned} \Delta(\mathcal{B}_1, \mathcal{B}_2) = \min_{\mathbf{w}, \mathbf{x}_1, \mathbf{x}_2} \quad & \|\mathbf{w}\| \\ \text{s.t.} \quad & \mathbf{w} = \mathbf{x}_1 - \mathbf{x}_2, \\ & \mathbf{x}_1 \in \mathcal{B}_1, \\ & \mathbf{x}_2 \in \mathcal{B}_2. \end{aligned} \quad (4)$$

This distance is not differentiable with respect to the pose of the bodies in many cases [22]. In order to use state-of-the-art NLP solvers, which make use of derivatives of constraint functions, a reformulation is therefore necessary.

In [17], this problem is overcome for the case of convex polytopes and ellipsoids. They prove the equivalence

$$\begin{aligned} \Delta(\mathcal{P}_1, \mathcal{P}_2) \geq \Delta_{\min} \Leftrightarrow \\ \exists \boldsymbol{\xi} \in \mathbb{R}^3, \lambda, \mu \in \mathbb{R} : -\frac{1}{4} \boldsymbol{\xi}^\top \boldsymbol{\xi} - \lambda - \mu \geq \Delta_{\min}^2, \\ \mathbf{V}_1^\top \boldsymbol{\xi} + \lambda \mathbf{1} \geq \mathbf{0}, \\ -\mathbf{V}_2^\top \boldsymbol{\xi} + \mu \mathbf{1} \geq \mathbf{0} \end{aligned} \quad (5)$$

for the minimal distance $\Delta_{\min} > 0$ of two polytopes $\mathcal{P}_1, \mathcal{P}_2$ with vertex matrices \mathbf{V}_1 and \mathbf{V}_2 .

Their proof is based on Lagrange duality. First, they reformulate the minimization problem (4) by using the squared Euclidean distance and substituting the polytope representation (1) into the constraints. By taking the dual problem of this minimization problem and simplifying, they obtain a maximization problem which, because of strong duality, yields the same distance. The decision variables of this dual problem are the newly introduced dual variables $\boldsymbol{\xi}$, λ and μ , which are called collision multipliers. Using the dual maximization problem to obtain a lower bound on the actual distance, the inequalities in (5) are obtained.

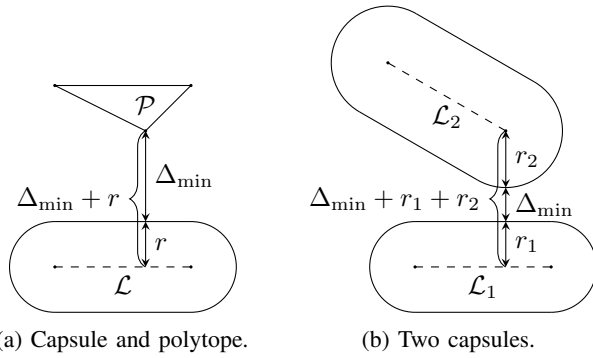


Fig. 2: Illustration of the minimal required distances.

We adapt equivalence (5) for capsules by exploiting the fact that a capsule can be treated like a convex polytope by increasing the distance threshold by the radius of the capsule and exploiting the fact that the center line is a convex polytope, i.e.,

$$\Delta(\mathcal{P}, \mathcal{C}) \geq \Delta_{\min} \Leftrightarrow \Delta(\mathcal{P}, \mathcal{L}) \geq \Delta_{\min} + r \quad (6)$$

for constraints concerning a convex polytope \mathcal{P} and a capsule \mathcal{C} (see Figure 2a), and

$$\Delta(\mathcal{C}_1, \mathcal{C}_2) \geq \Delta_{\min} \Leftrightarrow \Delta(\mathcal{L}_1, \mathcal{L}_2) \geq \Delta_{\min} + r_1 + r_2 \quad (7)$$

in the case of two capsules $\mathcal{C}_1, \mathcal{C}_2$ with respective center lines $\mathcal{L}_1, \mathcal{L}_2$ and radii r_1, r_2 (see Figure 2b).

Applying the equivalence (5) as collision-avoidance constraints into an OCP introduces five additional variables (collision multipliers) per pair of collision objects, together with five scalar inequalities for each capsule–capsule pair and $3 + N_v$ scalar inequalities for each capsule–polytope pair.

IV. COLLISION FREE TRAJECTORY PLANNING

A key component of robotic control architectures is the generation of joint trajectories, which are then tracked by low-level controllers, typically PID schemes. Trajectory planning is commonly formulated as an optimal control problem (OCP), where an objective function J is minimized over a finite prediction horizon with respect to the control input \mathbf{u} . Knowledge of the physical model is incorporated in the shape of a differential equation enforced through equality constraints and additional inequality constraints on the states \mathbf{x} and inputs of the system can be added. For numerical implementation, the continuous-time problem is transcribed into a nonlinear program (NLP) under suitable assumptions. This work applies a multiple-shooting approach [26] with the control input $\mathbf{u}(t)$ defined as a piecewise constant function. The path constraints are thus only evaluated at the shooting nodes. The proposed NLP is constructed over the following subsections.

A. Motion retargeting and input prediction

In a teleoperation setting, the human input needs to be mapped from the VR controller space onto a desired pose of the end-effector in the robot space. This is done with a

relative motion retargeting as described in [27]. The user can move the VR controller freely, and only while activating a clutching mechanism, the operator’s input is mapped onto a desired pose for the robot end-effector. The desired pose at time step k in the robot space is then given with a homogeneous transformation matrix $\mathbf{T}_{\text{ref}}[k] \in \mathbb{R}^{4 \times 4}$ which contains the rotational matrix \mathbf{R}_{ref} and the translation vector \mathbf{p}_{ref} .

For a predictive planning approach, a reference trajectory $\mathbf{T}_{\text{ref}} : \{1, \dots, N\} \rightarrow \mathbb{R}^{4 \times 4}$ over the prediction horizon needs to be generated. The reference starts from step $k = 1$, since the robot states are fixed for the first step $k = 0$. A widely spread approach is based on the assumption of a movement with constant velocity [28] for translation and a constant angular velocity assumption around a constant rotation axis for rotational movements.

B. Robot Model

A robot manipulator with n_q rotational joints $\mathbf{q} = [q_0 \dots q_{n_q}]^\top$ is considered. With the assumption of sufficiently performant low-level controllers, the linear state space model

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k], \quad (8)$$

which is a double integrator, can be applied to predict the kinematic robot movements with the state transition matrix \mathbf{A} and the input mapping matrix \mathbf{B} . The discrete-time transition matrix \mathbf{A} can be computed with the matrix exponential from the continuous system. The state vector $\mathbf{x} = [\mathbf{q}^\top \dot{\mathbf{q}}^\top]^\top$ consists of the concatenation of joint angles and velocities, and the input vector $\mathbf{u} = \ddot{\mathbf{q}}$ consists of the joint accelerations. \mathbf{x} and \mathbf{u} are interpreted as sequences over a prediction horizon of length N with the respective elements $\mathbf{x}[k] \in \mathbb{R}^{2n_q}$ at time step k and $k \in \{0, \dots, N\}$, while $\mathbf{u}[k] \in \mathbb{R}^{n_q}$ is only defined up to $k = N - 1$.

C. Objective function

The main objective in a teleoperation scenario is the tracking of the given user input \mathbf{T}_{ref} . The current position $\mathbf{p}(\mathbf{q}[k])$ and orientation $\mathbf{R}(\mathbf{q}[k])$ of the end-effector are extracted from the homogeneous transformation matrix $\mathbf{T}(\mathbf{q}[k])$, which is given through the forward kinematics.

The positional tracking error can then be stated as the squared weighted error

$$l_p[k] = (\mathbf{p}[k] - \mathbf{p}_{\text{ref}}[k])^\top \cdot \mathbf{Q}_p \cdot (\mathbf{p}[k] - \mathbf{p}_{\text{ref}}[k]) \quad (9)$$

with the diagonal matrix \mathbf{Q}_p containing the elements of the weight vector $\mathbf{w}_p \in \mathbb{R}^3$ as entries. This allows the assignment of individual weights for the three translational degrees of freedom of the end-effector.

For rotational tracking, an approach based on the Frobenius norm [29] is adopted. The stage cost is formulated as

$$l_o[k] = w_o \cdot \|\mathbb{I} - \mathbf{R}[k]\mathbf{R}_{\text{ref}}[k]^\top\|_F^2 \quad (10)$$

with the scalar weight w_o . The multiplication of $\mathbf{R}[k]\mathbf{R}_{\text{ref}}[k]^\top$ becomes the identity matrix \mathbb{I} under perfect alignment of the end-effector with the desired

orientation, thus driving the cost l_o to 0. The square of the Frobenius norm is introduced here to prevent computational complexity by removing the square root operations.

A secondary set of objectives is introduced for the regularization of the optimization variables and the generation of smooth trajectories. The joint velocities $\dot{\mathbf{q}}$ and the joint accelerations $\ddot{\mathbf{q}} = \mathbf{u}$ are considered under the quadratic cost terms

$$l_a[k] = \dot{\mathbf{q}}[k]^\top \cdot \mathbf{Q}_a \cdot \dot{\mathbf{q}}[k] \quad (11)$$

$$l_b[k] = \ddot{\mathbf{q}}[k]^\top \cdot \mathbf{Q}_b \cdot \ddot{\mathbf{q}}[k] \quad (12)$$

with the diagonal weight matrices \mathbf{Q}_a and \mathbf{Q}_b

Slack variables are a common method to relax inequality constraints and to keep the optimization problem feasible. This becomes especially important in real-world applications, since through disturbances on the real system or through numerical inaccuracies, the state \mathbf{x}_0 might be slightly out of the allowed regions. This is especially likely when operating close to collision constraints, since the optimum of the optimization problem often lies on some boundaries. Slack variables push the trajectory planner back to feasible regions by a high cost. A single slack variable $s[k]$ for every time step k is included in the objective function as a third objective with the weight w_s as quadratic stage cost

$$l_s[k] = w_s \cdot s[k]^2. \quad (13)$$

All stage costs l_i are combined in the objective function as a sum of all stage costs over the prediction horizon

$$J = \sum_{k=1}^N \underbrace{l_o[k] + l_p[k]}_{\text{tracking}} + \underbrace{l_a[k] + l_b[k-1]}_{\text{regularization}} + \underbrace{l_s[k]}_{\text{slack}}. \quad (14)$$

Since the state variable is fixed through the starting constraint for the first time index $k = 0$, the sum over all stage costs starts with time index $k = 1$ and only the stage cost l_{2b} , containing the control input \mathbf{u} , is starting from $k = 0$, which is also only defined up to $k = N - 1$. The final stage cost is thus integrated into the sum without an explicit set of weights.

D. Collision Constraints

Collision avoidance is included in the optimization problem as a set of inequality constraints. In the following, equations for self-collision and environmental collision avoidance are derived.

For self-collisions, a set $\mathcal{I}_s \subset \{0, \dots, N_s - 1\}^2$ of relevant collision pairs is considered. It contains all tuples (i, j) where collisions between capsules \mathcal{C}_i and \mathcal{C}_j of the robot collision model should be avoided. Some capsule pairs do not need to be considered for collision avoidance, since they kinematically cannot come into contact (e.g., shoulder and elbow).

For each relevant pair $(i, j) \in \mathcal{I}_s$ and each time step $k \in \{1, \dots, N\}$, the collision multipliers $\gamma_{s,ij}[k] =$

$[\boldsymbol{\xi}_{s,ij}[k]^\top \lambda_{s,ij}[k] \mu_{s,ij}[k]]^\top \in \mathbb{R}^5$ are introduced as decision variables, along with a constraint function

$$\mathbf{g}_{s,ij}(\mathbf{q}, \gamma_{s,ij}) = \begin{bmatrix} \frac{1}{4} \boldsymbol{\xi}_{s,ij}^\top \boldsymbol{\xi}_{s,ij} + \lambda_{s,ij} + \mu_{s,ij} + (\Delta_{\min} + r_i + r_j)^2 \\ -\mathbf{V}_{\mathcal{L}_i}(\mathbf{q})^\top \boldsymbol{\xi}_{s,ij} - \lambda_{s,ij} \mathbf{1} \\ \mathbf{V}_{\mathcal{L}_j}(\mathbf{q})^\top \boldsymbol{\xi}_{s,ij} - \mu_{s,ij} \mathbf{1} \end{bmatrix}. \quad (15)$$

Here, r_i, r_j are the radii of the capsules \mathcal{C}_i and \mathcal{C}_j , respectively, and $\mathbf{V}_{\mathcal{L}_i}(\mathbf{q}), \mathbf{V}_{\mathcal{L}_j}(\mathbf{q}) \in \mathbb{R}^{3 \times 2}$ are their vertex matrices containing the end points of their center lines as columns. These matrices depend on the configuration \mathbf{q} as given in (3). The dependence on the time step k is omitted for compactness.

With Equation 15, the inequality $\mathbf{g}_{s,ij}(\mathbf{q}[k], \gamma_{s,ij}[k]) \leq \mathbf{0}$ for all $k \in \{1, \dots, N\}$ guarantees avoidance of self-collision. The interpretation is as follows: Every feasible point of the planning problem corresponds to a configuration trajectory \mathbf{q} for which the distance requirement $\Delta(\mathcal{C}_i, \mathcal{C}_j) \geq \Delta_{\min}$ is satisfied at all times, due to the equivalences (5) and (7). Conversely, these equivalences guarantee that for every configuration trajectory satisfying the distance requirement, there exist collision multipliers for which $\mathbf{g}_{s,ij}(\mathbf{q}[k], \gamma_{s,ij}[k]) \leq \mathbf{0}$ is satisfied, i.e., this inequality does not restrict the trajectory more than necessary to guarantee the distance requirement.

The avoidance of collisions with environmental obstacles is treated analogously by introducing the collision multipliers $\gamma_{o,ij}[k] = [\boldsymbol{\xi}_{o,ij}[k]^\top \lambda_{o,ij}[k] \mu_{o,ij}[k]]^\top \in \mathbb{R}^5$ and the constraint function

$$\mathbf{g}_{o,ij}(\mathbf{q}, \gamma_{o,ij}) = \begin{bmatrix} \frac{1}{4} \boldsymbol{\xi}_{o,ij}^\top \boldsymbol{\xi}_{o,ij} + \lambda_{o,ij} + \mu_{o,ij} + (\Delta_{\min} + r_i)^2 \\ -\mathbf{V}_{\mathcal{L}_i}(\mathbf{q})^\top \boldsymbol{\xi}_{o,ij} - \lambda_{o,ij} \mathbf{1} \\ \mathbf{V}_j^\top \boldsymbol{\xi}_{o,ij} - \mu_{o,ij} \mathbf{1} \end{bmatrix} \quad (16)$$

for every $(i, j) \in \mathcal{I}_o \subset \{0, \dots, N_s - 1\} \times \{0, \dots, N_o - 1\}$ that corresponds to a relevant pair of a capsule \mathcal{C}_i of the robot model and a convex environmental obstacle polytope \mathcal{P}_j with vertex matrix \mathbf{V}_j .

To improve efficiency, the surface of the table that the manipulator is mounted on is not modeled as a polytope, but instead as a horizontal plane that all capsules need to stay above. The corresponding constraint functions are given by

$$\mathbf{g}_{t,i}(\mathbf{q}) = -([0 \ 0 \ 1] \mathbf{V}(\mathbf{q}))^\top + (\Delta_{\min} + r_i) \mathbf{1} \quad (17)$$

for each $i \in \{0, \dots, N_s - 1\}$, requiring no collision multipliers.

The entire collision constraint function $\mathbf{g}(\mathbf{x}, \boldsymbol{\gamma})$ is obtained by vertically concatenating all $\mathbf{g}_{s,ij}$ with $(i, j) \in \mathcal{I}_s$, all $\mathbf{g}_{o,ij}$ with $(i, j) \in \mathcal{I}_o$, and all $\mathbf{g}_{t,i}$ with $i \in \{0, \dots, N_s - 1\}$, with $\boldsymbol{\gamma}$ being the vector of all collision multipliers, which is also constructed by vertical concatenation.

E. Nonlinear Optimization Problem

The objective function (14) is minimized with respect to the system states \mathbf{x} , system input \mathbf{u} , the slack variables s

and the collision multipliers γ over the prediction horizon, resulting in the NLP

$$\begin{aligned} \min_{\mathbf{x}, \mathbf{u}, \gamma, s} \quad & J(\mathbf{x}, \mathbf{u}, s, \mathbf{T}_{\text{ref}}) \\ \text{s.t.} \quad & \mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k] \quad \forall k \in \{0, \dots, N-1\} \\ & \mathbf{x}_{\min} \leq \mathbf{x}[k] \leq \mathbf{x}_{\max} \quad \forall k \in \{1, \dots, N\} \\ & \mathbf{u}_{\min} \leq \mathbf{u}[k] \leq \mathbf{u}_{\max} \quad \forall k \in \{0, \dots, N-1\} \\ & \mathbf{g}(\mathbf{x}[k], \gamma[k]) - s[k]\mathbf{1} \leq \mathbf{0} \quad \forall k \in \{1, \dots, N\} \\ & s[k] \geq 0 \quad \forall k \in \{1, \dots, N\} \\ & \mathbf{x}[k=0] = \mathbf{x}_0. \end{aligned} \quad (18)$$

The slack variable $s[k]$ is subtracted from the inequality constraint \mathbf{g} . While $s[k]$ is also constrained to always be positive, this allows single entries of $\mathbf{g}[k]$ to become positive, violating the collision constraints, at a high cost in the objective function. To reduce the number of optimization variables, only one slack variable is defined and applied to all inequality constraints for each time step k .

In addition to the collision constraints, minimal and maximal values are imposed on joint angles \mathbf{q} , angular velocities $\dot{\mathbf{q}}$, and angular accelerations \mathbf{u} . The collision multipliers γ don't appear in the objective function, but still need to be considered by the applied solver to find collision-free trajectories.

V. EXPERIMENTAL SETUP

We evaluate our proposed method for collision avoidant trajectory planning in teleoperation on a UR5e manipulator. First it is compared against the two common approaches with sphere approximation [13] and witness points [14] in a simulation with an increasing number of obstacles, to test for scalability. Secondly, we test our method in a real teleoperation setting by trying to deliberately cause collisions. We check for computation time, distance to obstacles and tracking accuracy.

The NLP Equation 18 is implemented with Python and CasADi [30] and the open source NLP solver IPOPT [31], and solved at a fixed sample rate of 20 Hz for a given reference trajectory \mathbf{T}_{ref} and the current state \mathbf{x}_0 . The CasADi built-in just-in-time compilation has been utilized with the highest optimization level, so that all computationally demanding tasks are run as compiled binaries. The utilized hardware is an AMD Ryzen 7950X CPU. The implementation is publicly available at: <https://github.com/GiantMGG/robot-optimal-trajectory-planning>.

A. Implementation of Baseline Approaches

The first baseline method [14] is implemented with several modifications. As the original work does not fully describe the quadratic approximation of the objective, and since a nonlinear solver was used, we employ IPOPT without quadratic approximation. This yields reliable convergence at the cost of higher computation times. To ensure recursive feasibility between SQP iterations, slack variables are added

to the collision constraints and penalized quadratically in the objective. The UR5e is approximated with capsules, consistent with our approach. For a fair comparison, the reference is adapted from a constant setpoint to a constant-velocity prediction.

The second baseline, based on sphere approximations (e.g., [13]), is directly integrated into the optimization problem Equation 18 and aligned with the capsule formulation for comparability. The robot is represented by 29 spheres distributed along the centerlines of the corresponding capsules. Collision constraints for both self- and environment interactions are expressed as distance functions between relevant sphere pairs, using the sum of their radii. Environmental obstacles are modeled with three spheres in simulation and five spheres in the real-robot experiments.

B. Introduction of Testcases

All three systems are evaluated in a simplified simulation, where the robot state is assumed to follow the planned trajectories exactly, without dynamics or disturbances. Three 20–30 s scenarios with pre-recorded references are defined: (i) free arm jogging, (ii) end-effector motion into the robot base to trigger self-collisions, and (iii) motion to trigger collisions with the environment. Each scenario is varied by the number of cuboid obstacles (0, 1, 3, 6, 9, 12). The obstacles are either modeled as polytopes with 8 vertices, or with 3 spheres. To reduce peripheral effects on computation time, each configuration is run four times. This test case analyzes the scalability of the approaches with increasing number of collision constraints.

A second test case for evaluation is run on an UR5e robot. A larger cuboid, modeled with 5 spheres or polytope with 8 vertices, serves as a single obstacle (Figure 3a). The operator's task is to try to touch the obstacle with parts of the robot from multiple sides and angles and to induce self-colliding motion. This testcase shows the applicability to a real robot in a teleoperation scenario, which closes the overall control loop including the human operator, allowing for analysis of the tracking and collision avoidance performance.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

A. Evaluation in Simulation

The results of the simulative study, with emphasize on computation time, are summarized in Table I. The free arm jogging scenario is consistently solved with the lowest computation times across all methods, indicating that the other two scenarios indeed activate the collision constraints and therefore impose higher computational demand.

Among the evaluated methods, NMPC-MP exhibits the highest computation time but shows better scalability with the number of obstacles. The original publication employed a quadratic programming (QP) solver, which may reduce computation times considerably; however, we were unable to achieve reliable convergence in our implementation. This may be attributed to the original Schulman approximation [19], which assumes one witness point to remain fixed in

TABLE I: Computation times (ms) for three scenarios with varying numbers of obstacles, averaged over four runs.

Scenario	Method	0 Obstacles		1 Obstacle		3 Obstacles		6 Obstacles		9 Obstacles		12 Obstacles	
		mean±std	max	mean±std	max	mean±std	max	mean±std	max	mean±std	max	mean±std	max
(i) Arm jogging	NMPC-MP [14]	151±0.3	168	195±0.3	210	301±0.3	326	453±2.0	477	623±2.7	654	797±3.1	849
	Ours	15±0.3	25	22±0.3	49	46±0.5	76	114±0.4	186	196±0.9	493	316±1.3	552
	Spheres [13]	20±0.3	36	29±0.3	46	58±1.1	99	116±1.4	170	202±1.1	269	301±0.8	392
(ii) Self-collision	NMPC-MP [14]	155±0.9	198	195±0.0	245	299±1.3	360	467±2.6	544	640±2.0	739	809±4.5	1006
	Ours	14±0.0	28	24±0.3	39	62±0.6	127	107±0.8	275	210±1.8	345	398±2.6	2884
	Spheres [13]	25±0.0	49	34±0.3	57	84±0.9	206	130±0.9	214	305±2.1	1639	448±1.9	2481
(iii) Obstacles	NMPC-MP [14]	166±0.0	197	209±0.5	258	314±0.9	352	488±1.6	593	681±3.4	759	881±3.0	1011
	Ours	14±0.3	22	26±0.0	40	59±0.9	89	124±1.2	250	231±0.8	664	377±2.8	850
	Spheres [13]	21±0.3	37	36±0.4	55	75±1.1	117	148±1.3	232	248±0.4	347	366±1.2	534

space, an assumption that does not hold in the context of self-collision avoidance.

The sphere-based approach and our collision-multiplier formulation yield comparable results, with our method showing slightly lower mean and maximum computation times. The resulting joint trajectories are also very similar across both methods. In contrast, NMPC-MP produces less smooth trajectories, since its system model does not account for joint accelerations.

With a prediction horizon of $N = 10$ steps, the sphere-based approach already struggles in the presence of a single obstacle, whereas our approach still maintains the cycle time of 50 ms. It should also be noted that the approximation of the cuboids with three spheres is rather inaccurate.

Neither the sphere-based method nor our approach scales well with an increasing number of obstacles and thus cannot be applied in real-time teleoperation with a horizon of $N = 10$ in cluttered environments. For every new obstacle and each capsule of the robot model, our approach introduces 11 additional constraints and 5 optimization variables per time step. Depending on the robot link (some approximated by up to 7 spheres), as many as 21 constraints may be added, since each cuboid is approximated by three spheres.

B. Evaluation on Real Manipulator

The collision-avoiding behavior of the planner is shown in Figure 3 and Figure 4. The trade-off between tracking performance and collision avoidance appears in Figure 3b and Figure 4b, and is also evident in the 3D plots (Figure 3a, Figure 4a), where the operator-driven reference passes through the obstacle while the end-effector remains collision-free. The upper plots in Figure 3c and Figure 4c show the distances between gripper and obstacle (first 20 s), as well as between two links driven close together during teleoperation (last 20 s). These distances, computed from exact meshes, confirm that no collisions occur. A video of the experimental evaluation is available at: <https://youtu.be/OR5T9Hy2fdE>.

Compared to the sphere-based approach, the more accurate geometric modeling of robot and obstacles allows the manipulator to move closer to obstacles, reducing unnecessary evasive maneuvers (Figure 3a vs. Figure 4a). For computation time, our method remains between 20 ms and 30 ms for most of the trajectory, with peaks up to 40 ms during obstacle avoidance—still below the control cycle of 50 ms (see lower plot of Figure 3c). The sphere-based approach exhibits longer runtimes, with two peaks exceeding 50 ms.

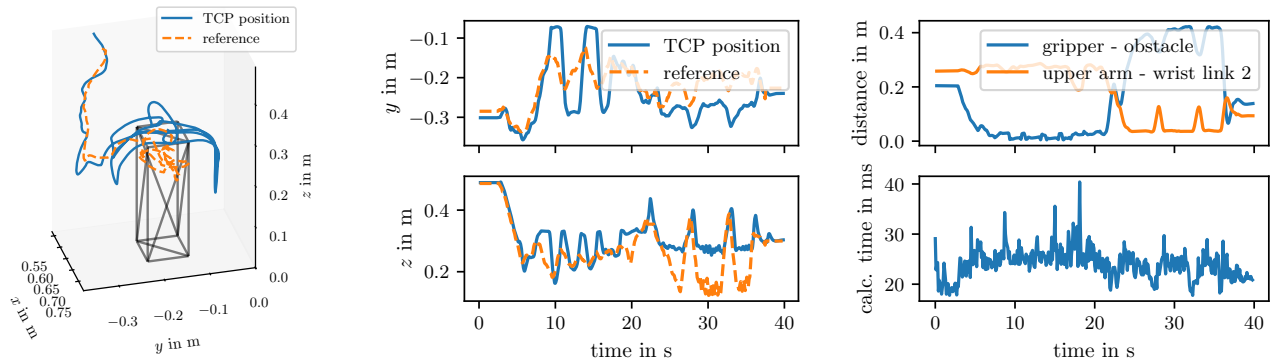
In both methods, computation time increases near obstacles or self-collisions, when avoidance is required.

VII. CONCLUSION

We proposed a method that combines differentiable collision constraints with accurate geometric modeling using capsules and polytopes. The approach achieves lower computation times than state-of-the-art methods while providing more precise obstacle representations. We validated the method in simulation with varying obstacle densities and on a UR5e manipulator in a real-world teleoperation study, demonstrating smoother and collision-free control. Although developed for teleoperation, the approach is directly transferable to general robotic applications. Future work will explore SQP-based methods to further reduce computation time.

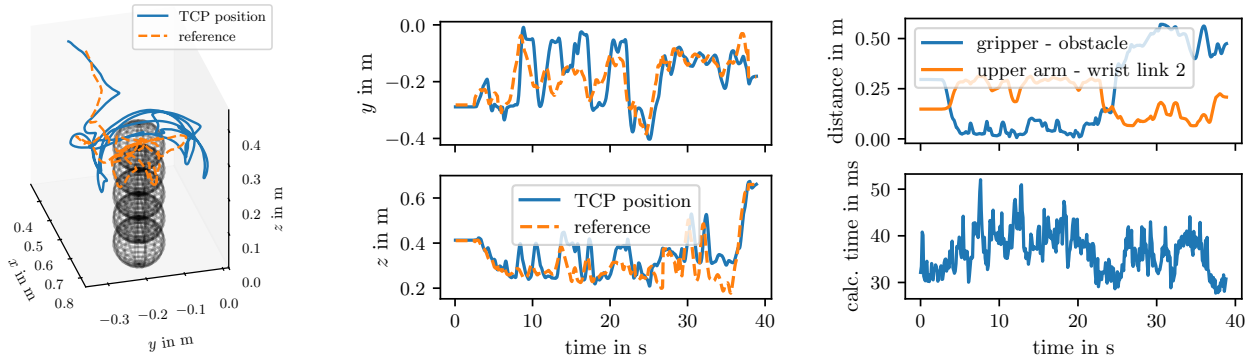
REFERENCES

- [1] P. F. Hokayem and M. W. Spong, "Bilateral teleoperation: An historical survey," *Automatica*, vol. 42, no. 12, 2006.
- [2] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer International Publishing, 2016.
- [3] S. Lichiardopol, "A survey on teleoperation," *Technische Universitat Eindhoven, DCT report*, vol. 20, 2007.
- [4] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, "GELLO: A General, Low-Cost, and Intuitive Teleoperation Framework for Robot Manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2024.
- [5] G. Li, Q. Li, C. Yang, Y. Su, Z. Yuan, and X. Wu, "The Classification and New Trends of Shared Control Strategies in Telerobotic Systems: A Survey," *IEEE Trans. Haptics*, vol. 16, no. 2, 2023.
- [6] D. Rakita, B. Mutlu, and M. Gleicher, "A Motion Retargeting Method for Effective Mimicry-based Teleoperation of Robot Arms," in *Proc. 2017 ACM/IEEE Int. Conf. Human-Robot Interaction (HRI)*, 2017.
- [7] M. Elbanhawi and M. Simic, "Sampling-Based Robot Motion Planning: A Review," *IEEE Access*, vol. 2, 2014.
- [8] Z. Zhao, S. Cheng, Y. Ding, Z. Zhou, S. Zhang, D. Xu, and Y. Zhao, "A Survey of Optimization-Based Task and Motion Planning: From Classical to Learning Approaches," *IEEE/ASME Trans. Mechatronics*, vol. 30, no. 4, 2025.
- [9] J. Wang, T. Zhang, N. Ma, Z. Li, H. Ma, F. Meng, and M. Q.-H. Meng, "A survey of learning-based robot motion planning," *IET Cyber-Systems and Robotics*, vol. 3, no. 4, 2021.
- [10] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen, "Implementation of Nonlinear Model Predictive Path-Following Control for an Industrial Robot," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, 2017.
- [11] R. Lima, S. Saha, N. George, V. Vakharia, S. Parab, S. Gaonkar, V. Vatsal, and K. Das, "Teleoperated Omni-Directional Dual Arm Mobile Manipulation Robotic System With Shared Control for Retail Store," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, 2024.
- [12] "Shared Autonomy Control for Slosh-Free Teleoperation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2023.
- [13] M. Rubagotti, T. Taunyazov, B. Omarali, and A. Shintemirov, "Semi-Autonomous Robot Teleoperation With Obstacle Avoidance via Model Predictive Control," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, 2019.



(a) Trajectory of end-effector position. (b) End-effector position in y and z direction. (c) Distances and calculation time.

Fig. 3: Trajectory recording with our proposed method.



(a) Trajectory of end-effector position. (b) End-effector position in y and z direction. (c) Distances and calculation time.

Fig. 4: Trajectory recording with robot and obstacle approximation through spheres.

- [14] “NMPC-MP: Real-time Nonlinear Model Predictive Control for Safe Motion Planning in Manipulator Teleoperation,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021.
- [15] B. Sundaralingam, S. K. S. Hari, A. Fishman, C. Garrett, K. Van Wyk, V. Blukis, A. Millane, H. Oleynikova, A. Handa, F. Ramos, N. Ratliff, and D. Fox, “CuRobo: Parallelized Collision-Free Robot Motion Generation,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2023.
- [16] X. Zhang, A. Liniger, and F. Borrelli, “Optimization-Based Collision Avoidance,” *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, 2021.
- [17] C. Dietz, S. Albrecht, A. Nurkanović, and M. Diehl, “Efficient Collision Modelling for Numerical Optimal Control,” in *Proc. Eur. Control Conf. (ECC)*, 2023.
- [18] T. Stoyanov, R. Krug, A. Kiselev, D. Sun, and A. Loutfi, “Assisted Telemanipulation: A Stack-Of-Tasks Approach to Remote Manipulator Control,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2018.
- [19] 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. (RSS)*, 2013.
- [20] A. Haffemayer, A. Jordana, M. Fourmy, K. Wojciechowski, G. Saurel, V. Petrik, F. Lamiriaux, and N. Mansard, “Model Predictive Control Under Hard Collision Avoidance Constraints for a Robotic Arm,” in *Proc. Int. Conf. Ubiquitous Robots (UR)*, 2024.
- [21] M. Krämer, C. Rösman, F. Hoffmann, and T. Bertram, “Model predictive control of a collaborative manipulator considering dynamic obstacles,” *Optimal Control Applications and Methods*, vol. 41, no. 4, 2020.
- [22] K. Tracy, T. A. Howell, and Z. Manchester, “Differentiable Collision Detection for a Set of Convex Primitives,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2023.
- [23] S. He, B. Jin, T. Chen, Y. Xu, D. Li, X. Zhang, and T. Zou, “Model and Collision Avoidance for Motion Planning of Rigid-Soft Robot With Continuous Expression and Input Mapping,” *IEEE Trans. Autom. Sci. Eng.*, vol. 22, 2025.
- [24] S. Lin, C. Hu, J. Yu, and Y. Liang, “Batch Iterative Dual Optimization for Collision-Free Robot Motion Generation,” *IEEE Trans. Ind. Informat.*, vol. 21, no. 4, 2025.
- [25] X. Wang, C. Yang, H. Ma, and L. Cheng, “Shared control for teleoperation enhanced by autonomous obstacle avoidance of robot manipulator,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2015.
- [26] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing, 2022.
- [27] M. Grobbel, D. Flögel, P. Rigoll, and S. Hohmann, “Disentangling coordinate frames for task specific motion retargeting in teleoperation using shared control and VR controllers,” in *Proc. IEEE Conf. Telepresence*, 2025.
- [28] R. Chipalkatty, G. Droge, and M. B. Egerstedt, “Less Is More: Mixed-Initiative Model-Predictive Control With Human Inputs,” *IEEE Transactions on Robotics*, vol. 29, no. 3, 2013.
- [29] D. Q. Huynh, “Metrics for 3D Rotations: Comparison and Analysis,” *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, 2009.
- [30] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi: A software framework for nonlinear optimization and optimal control,” *Mathematical Programming Computation*, vol. 11, no. 1, 2019.
- [31] A. Wächter, “Short Tutorial: Getting Started With Ipopt in 90 Minutes,” 2009.