Collision Avoidance in Model Predictive Control using Velocity Damper

Arthur Haffemayer, Armand Jordana, Ludovic de Matteis, Krzysztof Wojciechowski, Florent Lamiraux, Nicolas Mansard

Collision Avoidance in Model Predictive Control using Velocity Damper

Arthur Haffemayer[1,2,3], Armand Jordana[4], Ludovic de Matteïs[1], Krzysztof Wojciechowski[1], Florent Lamiraux[1], and Nicolas Mansard[1,2]

***Abstract*— We propose an advanced method for controlling the motion of a manipulator robot with strict collision avoidance in dynamic environments, leveraging a velocity damper constraint. Unlike conventional distance-based constraints, which tend to saturate near obstacles to reach optimality, the velocity damper constraint considers both distance and relative velocity, ensuring a safer separation. This constraint is incorporated into a model predictive control framework and enforced as a hard constraint through analytical derivatives supplied to the numerical solver. The approach has been fully implemented on a Franka Emika Panda robot and validated through experimental trials, demonstrating effective collision avoidance during dynamic tasks and robustness to unmodeled disturbances. An efficient open-source implementation along examples are provided here: `https://gepettoweb.laas.fr/articles/haffemayer2025.html`.**

I. Introduction

Collision avoidance with dynamic obstacles is a long-standing goal in robotics and has potential application in a variety of fields [1], [2]. Over the past few decades, various methods have been proposed to address this issue. Early approaches, such as motion planning, primarily relied on geometric techniques[3], [4]. However, those techniques are challenging to deploy online, especially during dynamic motion [5]. On one hand, neural networks have been employed to enhance motion planning [6], [7] or to learn policies with reinforcement learning [8] and imitation learning [9]. On the other hand, predicting the future robot motion avoiding the obstacle opens the way to safe control with guarantee, much needed in the industry or in human-robot interactions. This is the way we explore here.

Model Predictive Control (MPC)[10] is an appealing framework to efficiently plan online dynamic collision free motions. By formulating the collision avoidance as a hard constraint in the optimal control problem (OCP), MPC can ensure safety with the predicted trajectory serving as an explicable guarantee. Towards that goal, optimization-based technique have been explored to solve collision avoidance. Trajectory Optimization (TO) was used to filter the solution given by the motion planner [11] or could be directly used to generate optimal trajectories. [12] utilized covariant Hamiltonian solvers to address collision avoidance, while [13] introduced a cost-based formulation of the collision constraint, solved using sequential-quadratic programming.

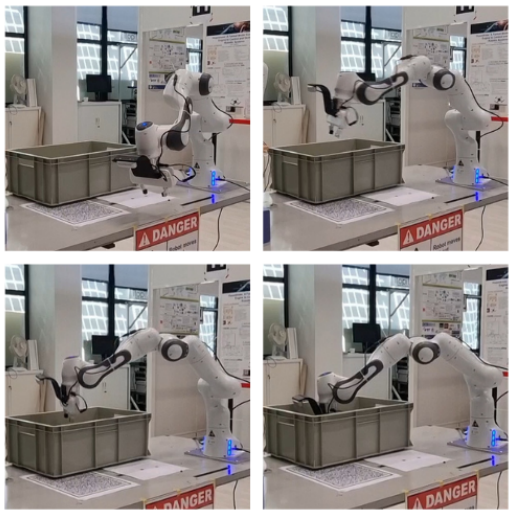


Fig. 1: Collision Avoidance in real time on a torque controlled manipulator arm using Model Predictive Control and the velocity damper constraint.

As discussed in [14], the difficulty is to build a proper barrier function to avoid self-collision with the stake of obtaining both a satisfactory behavior of the robot and an efficient solver performance to reach real-time implementation. From the first historical approach [15] of using potential fields, which still remains active [16], several formulations have been tried. The first milestones have been reached with gradient-free solvers, combining a gradient-free Model Predictive Path Integral Control (MPPI) [17] with Graphics Processing Units (GPU) to perform online collision avoidance. While this approach is capable of solving complex problems in real time, the absence of gradient information in the solver can be less efficient compared to gradient-based methods in a well-defined environment [18]. Several gradient-based OCP have also been proposed, either with soft [19], [20] or hard [21] constraints, but none reaching real-time capabilities able to unleash the MPC.

In our previous work [22], we used a distance-based constraint to perform collision avoidance using MPC. This constraint ensures that the distance between the robot and any obstacle remains greater than a predefined safety distance. While effective, this method can be overly conservative, leading to suboptimal trajectories, particularly in environments with dynamic obstacles or complex robot geome-

This work is supported by the European project AGIMUS (under GA no.101070165), ANITI (ANR-19-P3IA-0004), NERL (ANR-23-CE94-0004-02) and by the National Science Foundation grants 1932187, 2026479, 2222815 and 2315396.

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[2] Artificial and Natural Intelligence Toulouse Institute, France
[3] Continental, France
[4] Machines in Motion Laboratory, New York University, USA
*corresponding author: arthur.haffemayer@laas.fr

tries. Additionally, as the solver seeks optimal paths, it tends to saturate the constraint, meaning that the robot may approach obstacles at high speeds, posing potential risks also observed in [12], [13]. To address these limitations, the velocity damper constraint, introduced by [23], offers a more flexible alternative. Rather than imposing a strict distance threshold, the velocity damper adjusts the robot speed as it nears obstacles, allowing for smoother and more efficient motion while maintaining safety. This approach is particularly beneficial in confined spaces, allowing the robot to navigate closer to obstacles without sacrificing safety or performance. It has also been widely applied in motion planning [24], [25], aerial robotics [26], or even in humanoid locomotion [27]. However, it had yet to be implemented in an MPC framework, which implies to efficiently evaluate it and its derivates.

In this work, we present a rigorous way to write the velocity damper constraint with its derivatives necessary to solve the problem. We then implement this formulation into a numerical solver to optimize the robot trajectory while dynamically avoiding collisions in real-time on an MPC scheme. We validate the effectiveness of our controller through hardware experiments on a torque-driven manipulator, performing pick-and-place tasks in an obstacle-rich environment while responding to physical perturbations introduced by a human operator. To the best of our knowledge, this is the first experimental demonstration of a torque-controlled manipulator utilizing nonlinear MPC with collision avoidance based on the velocity damper constraint, formulated as a hard constraint and solved in real time on a Franka Emika Panda robot. The results highlight the practical significance of this approach: the robot exhibits strong adaptability to dynamic perturbations, achieves precise target-reaching, and ensures compliance, particularly when interacting with a human operator, all while effectively avoiding obstacles and slowing near them if need be and not saturating the distance between the obstacle and the robot at high speed.

II. Formulation of Obstacle Avoidance Problem

A. OCP formulation

The OCP formulation of the problem is the same as the one described in the paper [22]:

$$\min_{\boldsymbol{x},\boldsymbol{u}} \sum_{t=0}^{T-1} \ell_t(\boldsymbol{x}_t, \boldsymbol{u}_t) + \ell_T(\boldsymbol{x}_T) \tag{1a}$$

$$\text{subject to } \boldsymbol{x}_{t+1} = f_t(\boldsymbol{x}_t, \boldsymbol{u}_t) \quad \forall\ 0 \le t < T, \tag{1b}$$

$$c_t(\boldsymbol{x}_t, \boldsymbol{u}_t) \ge 0, \quad \forall\ 0 \le t < T, \tag{1c}$$

$$c_T(\boldsymbol{x}_T) \ge 0, \tag{1d}$$

where $\boldsymbol{x}_t = (\boldsymbol{q}_t, \boldsymbol{v}_t)$ is the state of the robot (joints configuration $\boldsymbol{q}_t$ and joints velocity $\boldsymbol{v}_t = \dot{\boldsymbol{q}}_t$), and $\boldsymbol{u}_t = \boldsymbol{\tau}_t$ are the torques of the joints; f_t is the transition function representing the discretized robot dynamics along the horizon of length T, given an initial state $\boldsymbol{x}_0$ (e.g. measured state). The variables of the OCP are the state trajectory $\boldsymbol{x} = (\boldsymbol{x}_1, ..., \boldsymbol{x}_T)$ and the control trajectory $\boldsymbol{u} = (\boldsymbol{u}_0, \boldsymbol{u}_1, ..., \boldsymbol{u}_{T-1})$. The functions to be optimized are the running cost ℓ_t and the terminal cost ℓ_T. Functions c_t and c_T represent the hard (inequality) constraints at each time step. On the robot, this OCP is solved at each control cycle along a receding horizon.

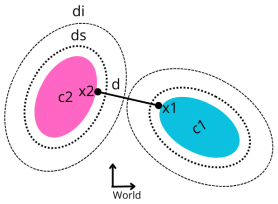


Fig. 2: Velocity damper constraint

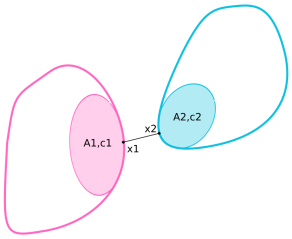


Fig. 3: Tangent ellipsoids at the closest points of two smooth collision geometries

B. Distance-based collision avoidance constraint

As mentioned in Sec. I, collision avoidance is typically enforced through inequality constraints that ensure a safe distance between the robot and any obstacles in the environment. For each relevant pair of objects $\mathcal{B}_i$, $\mathcal{B}_j$, that may be in collision and for each time discretization step $t \in \{1, \cdots, T\}$:

$$c_{t,i,j}(\boldsymbol{x}_t) = d_{ij}(\boldsymbol{x}_t) - \epsilon \ge 0\,, \tag{2}$$

where $d_{ij}(\boldsymbol{x}_t)$ is the distance between $\mathcal{B}_i$ and $\mathcal{B}_j$ when the robot is in configuration $\boldsymbol{q}_t$, and ϵ is the safety margin.

Such formulations are effective in static environments but may become overly conservative for fear of constraint saturation or not adapted when dealing with dynamic obstacles.

III. Velocity Damper Constraint

A. Definition of the Constraint

The velocity damper constraint, introduced in [23] is a method specifically designed to prevent collisions by modulating the robot velocity as it approaches an obstacle. Unlike the distance-based collision avoidance constraint defined in Sec. II-B, the velocity damper constraint takes the speed of the robot with regard to the obstacles and dynamically adjusts the robot speed based on its proximity to them. This ensures that the robot slows down as it nears a potential collision, effectively "damping" its velocity to zero when the distance to the obstacle reaches a critical threshold.

The velocity damper constraint can be expressed as:

$$\dot{d} \ge -\xi \frac{d - \epsilon}{d_i - \epsilon}, \quad \text{for } d \le d_i \tag{3}$$

Fig. 2 describes the different parameters. d_i is the influence distance, the range within which the velocity damper constraint becomes active. As the robot approaches an obstacle and enters this influence distance, the velocity damper begins to slow down the robot to prevent high-speed approach near the obstacle. ϵ is the security distance at which the robot must stop, and ξ is a passive coefficient for adjusting convergence speed. ϵ can be much smaller than the security margin ϵ of

the distance-based collision constraint because the constraint allow the robot to go arbitrarily near the limit $d = \epsilon$ in a finite time, but the component of the velocity normal to the obstacle gets very small. Appendix II in [23] proves that enforcing constraint (3) implies that $\mathcal{B}_i$ and $\mathcal{B}_j$ will never collide.

B. Time derivative of the distance

For efficiency, we need to provide to the numerical solver the derivative of the constraint with respect to the state $\boldsymbol{x}$ of the robot. First, let us write the distance as the norm of the vector linking the two closest points $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ belonging respectively to two convex objects $\mathcal{B}_1$ and $\mathcal{B}_2$:

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = ||\boldsymbol{x}_1 - \boldsymbol{x}_2|| \tag{4}$$

The time derivative of the distance can be written as [23]:

$$\dot{d} = \boldsymbol{n}^T(\boldsymbol{v}_{\boldsymbol{x_1}} - \boldsymbol{v}_{\boldsymbol{x_2}}) \tag{5}$$

where $\boldsymbol{n}$ is the normal vector of the direction of the closest points and $\boldsymbol{v}_{\boldsymbol{x_1}}$ and $\boldsymbol{v}_{\boldsymbol{x_2}}$ are the velocities at the points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. Note that [23] mostly gave an intuition of the proof of (4) based on geometric arguments, and a side development below will give it analytically.

C. Differentiation of the constraint

Formulating (3) in OCP (1) implies to obtain its derivatives with respect to $\boldsymbol{q}$ and $\boldsymbol{v}$. As $\dot{d}$ only depends on the normal to the collision objects, we make the hypothesis that the derivatives of $\dot{d}$ only add a dependency to the collision object curvatures around the closest points. We then proceed to the differentiation by considering the tangent ellipsoids at the closest points of each objects (see Fig. 3).

$$\begin{aligned}
\min_{\boldsymbol{x}_1, \boldsymbol{x}_2} \quad & f(\boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{1}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 && \text{(6a)}\\
\text{s.t.} \quad & \frac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{c}_1)^T A_1 (\boldsymbol{x}_1 - \boldsymbol{c}_1) = \frac{1}{2} && \text{(6b)}\\
& \frac{1}{2}(\boldsymbol{x}_2 - \boldsymbol{c}_2)^T A_2 (\boldsymbol{x}_2 - \boldsymbol{c}_2) = \frac{1}{2} && \text{(6c)}
\end{aligned}$$

Matrices A_1 and A_2 are the matrices defining the rotation and the lengths of the semi-axes of the ellipsoids. They can be written as: $A_i = R_i D_i R_i^T$, $i \in \{1, 2\}$, where the matrices D_i are the constant diagonal matrices defining the shapes of the ellipsoids and R_i belong to $SO(3)$ and define the orientation of the ellipsoids with regard to the world frame. Points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ lie, respectively, on the boundary of ellipsoids 1 and 2. Points $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are the centers of the ellipsoids. We write $\boldsymbol{\theta} = (\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{R}_1, \boldsymbol{R}_2)$ and $\boldsymbol{y} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \lambda_1, \lambda_2)$ For the given problem, the Lagrangian is defined as:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{y}, \boldsymbol{\theta}) = & \frac{1}{2}\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 \\
& + \frac{1}{2}\lambda_1 \left((\boldsymbol{x}_1 - \boldsymbol{c}_1)^T A_1 (\boldsymbol{x}_1 - \boldsymbol{c}_1) - 1\right) \\
& + \frac{1}{2}\lambda_2 \left((\boldsymbol{x}_2 - \boldsymbol{c}_2)^T A_2 (\boldsymbol{x}_2 - \boldsymbol{c}_2) - 1\right)
\end{aligned} \tag{7}$$

where λ_1 and λ_2 are the Lagrange multipliers associated with the constraints (6b) and (6c). Before discussing the limiting assumptions of (6), we first present the expression for $\dot{d}$ and its corresponding derivatives.

In the case of equality constraints, the Lagrange conditions are equivalent to the following conditions: $\mathcal{L}_{x_1}(\boldsymbol{y}^*, \boldsymbol{\theta}) = \mathcal{L}_{x_2}(\boldsymbol{y}^*, \boldsymbol{\theta}) = \mathcal{L}_{\lambda_1}(\boldsymbol{y}^*, \boldsymbol{\theta}) = \mathcal{L}_{\lambda_2}(\boldsymbol{y}^*, \boldsymbol{\theta}) = 0$, or

$$\begin{aligned}
& (\boldsymbol{x}_1 - \boldsymbol{x}_2) + \lambda_1 A_1 (\boldsymbol{x}_1 - \boldsymbol{c}_1) = 0 && \text{(8a)}\\
& (\boldsymbol{x}_2 - \boldsymbol{x}_1) + \lambda_2 A_2 (\boldsymbol{x}_2 - \boldsymbol{c}_2) = 0 && \text{(8b)}\\
& (\boldsymbol{x}_1 - \boldsymbol{c}_1)^T A_1 (\boldsymbol{x}_1 - \boldsymbol{c}_1) - 1 = 0 && \text{(8c)}\\
& (\boldsymbol{x}_2 - \boldsymbol{c}_2)^T A_2 (\boldsymbol{x}_2 - \boldsymbol{c}_2) - 1 = 0 && \text{(8d)}
\end{aligned}$$

where λ_1 and λ_2 are the Lagrange multipliers associated with the constraints (6b) and (6c).

Note that, at optimality,

$$\mathcal{L}(\boldsymbol{y}^*, \boldsymbol{\theta}) = \frac{1}{2} d^2. \tag{9}$$

To obtain $\dot{d}$ the time derivative of the distance, we assume that both ellipsoids are moving: R_i and $\boldsymbol{c}_i$ are time varying values, and that $\boldsymbol{x}_i$ minimize (6) for any time t. Let us derivate the Lagrangian with respect to time. We note $\dot{f}$ the derivative of f with respect to time.

$$\begin{aligned}
\dot{\mathcal{L}} = & \mathcal{L}_{\boldsymbol{x}_1}\, \dot{\boldsymbol{x}}_1 + \mathcal{L}_{\boldsymbol{x}_2}\, \dot{\boldsymbol{x}}_2 + \mathcal{L}_{\lambda_1}\, \dot{\lambda}_1 + \mathcal{L}_{\lambda_2}\, \dot{\lambda}_2 \\
& + \mathcal{L}_{\boldsymbol{c}_1}\, \dot{\boldsymbol{c}}_1 + \mathcal{L}_{\boldsymbol{c}_2}\, \dot{\boldsymbol{c}}_2 + \mathcal{L}_{\boldsymbol{R}_1}\, \boldsymbol{\omega}_1 + \mathcal{L}_{\boldsymbol{R}_2}\, \boldsymbol{\omega}_2
\end{aligned} \tag{10}$$

The partial derivative with respect to a rotation matrix should be understood as the tangent application, *i.e.* if $f(R)$ is a mapping from $SO(3)$ to a vector space, for any time varying rotation $R(t)$,

$$\dot{f} = f_R\, \boldsymbol{\omega}$$

where the angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ is such that $\dot{R} = [\boldsymbol{\omega}]_\times R$ with $[\boldsymbol{\omega}]_\times$ the skew-symmetric matrix i.e. for any z $[\boldsymbol{\omega}]_\times z = \boldsymbol{\omega} \times z$. See [28] for a nice and rigorous introduction about derivation on Lie groups from which we follow most of the notations.

From the optimality condition, at $\boldsymbol{y}^*$, we have $\mathcal{L}_{\boldsymbol{x}_1} = \mathcal{L}_{\boldsymbol{x}_2} = \mathcal{L}_{\lambda_1} = \mathcal{L}_{\lambda_2} = 0$. Thus,

$$\dot{\mathcal{L}} = \mathcal{L}_{\boldsymbol{c}_1}\dot{\boldsymbol{c}}_1 + \mathcal{L}_{\boldsymbol{c}_2}\dot{\boldsymbol{c}}_2 + \mathcal{L}_{\boldsymbol{R}_1}\boldsymbol{\omega_1} + \mathcal{L}_{\boldsymbol{R}_2}\boldsymbol{\omega_2} \tag{11}$$

From (7) and (8b), we get

$$\mathcal{L}_{\boldsymbol{c}_1} = -\lambda_1 (\boldsymbol{x}_1 - \boldsymbol{c}_1)^T A_1^T = (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \tag{12a}$$

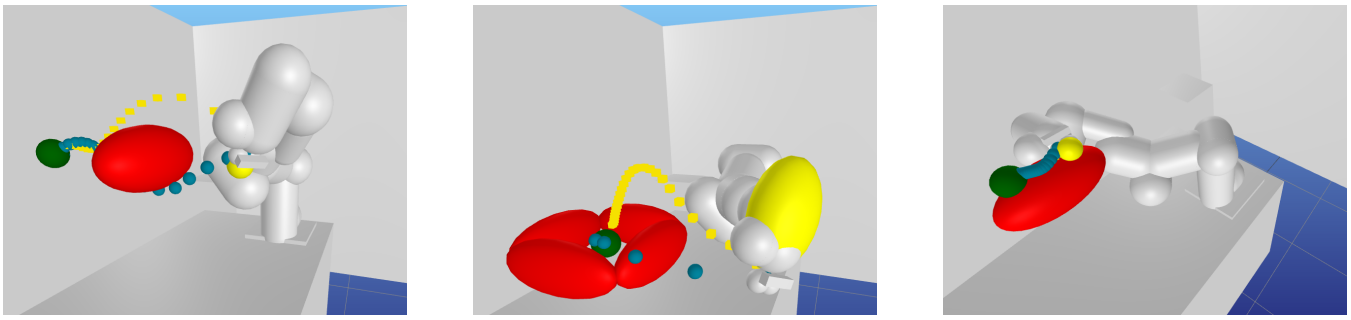
$$\mathcal{L}_{\boldsymbol{c}_2} = (\boldsymbol{x}_2 - \boldsymbol{x}_1)^T \tag{12b}$$

Recalling that $A_1 = R_1 D_1 R_1^T$ and using the differentiation rule on $SO(3)$, we get

$$\mathcal{L}_{\boldsymbol{R}_1} = \lambda_1 (\boldsymbol{x}_1 - \boldsymbol{c}_1)^T (A_1 [\boldsymbol{x}_1 - \boldsymbol{c}_1]_\times - [A_1 (\boldsymbol{x}_1 - \boldsymbol{c}_1)]_\times) \tag{13}$$

Using again Property (8b) and the Jacobi Identity $\mathbf{u}^T [\mathbf{v}]_\times = -\mathbf{v}^T [\mathbf{u}]_\times$, the expression simplifies to:

$$\begin{aligned}
\mathcal{L}_{\boldsymbol{R}_1} &= \boldsymbol{c}_1^T [\boldsymbol{x}_2 - \boldsymbol{x}_1]_\times + \boldsymbol{x}_2^T [\boldsymbol{x}_1]_\times \\
\mathcal{L}_{\boldsymbol{R}_2} &= \boldsymbol{c}_2^T [\boldsymbol{x}_1 - \boldsymbol{x}_2]_\times + \boldsymbol{x}_1^T [\boldsymbol{x}_2]_\times
\end{aligned} \tag{14}$$



(a) Scene 1: Simple collision avoidance (b) Scene 2: Goal reaching in a box (c) Scene 3: Contact-to-contact task

Fig. 4: Simulation scenes. Yellow cubes are from the velocity damper, blue spheres are from the distance-based constraint.

Finally, differentiating (9) and using (11), we obtain

$$\begin{aligned}\dot{d} = \frac{\dot{\mathcal{L}}}{d}(\boldsymbol{y},\boldsymbol{\theta}) = \frac{1}{d}((\boldsymbol{x}_1-\boldsymbol{x}_2)^T(\boldsymbol{v}_1-\boldsymbol{v}_2) \\ + (\boldsymbol{c}_1^T\left[\boldsymbol{x}_2-\boldsymbol{x}_1\right]_\times + \boldsymbol{x}_2^T\left[\boldsymbol{x}_1\right]_\times)\boldsymbol{\omega}_1 \\ + (\boldsymbol{c}_2^T\left[\boldsymbol{x}_1-\boldsymbol{x}_2\right]_\times + \boldsymbol{x}_1^T\left[\boldsymbol{x}_2\right]_\times)\boldsymbol{\omega}_2)\end{aligned} \tag{15}$$

Reformulating, we find exactly the same form as (5):

$$\begin{aligned}\dot{d} = \frac{(\boldsymbol{x}_1-\boldsymbol{x}_2)^T}{d}(\dot{\boldsymbol{c}}_1-\dot{\boldsymbol{c}}_2) \\ -(\boldsymbol{x}_1-\boldsymbol{c}_1)\times\boldsymbol{\omega}_1 + (\boldsymbol{x}_2-\boldsymbol{c}_2)\times\boldsymbol{\omega}_2) \\ = \boldsymbol{n}^T(\boldsymbol{v}_{\boldsymbol{x}_1}-\boldsymbol{v}_{\boldsymbol{x}_2})\end{aligned} \tag{16}$$

This corresponds to a complete analytical proof of the results of [23].

D. Using the sensitivity to obtain the derivative of the speed with respect to robot state

Although $\boldsymbol{\theta}$ belongs to a Lie group, for simplicity, we will refer to its time derivative as $\dot{\boldsymbol{\theta}} = (\dot{\boldsymbol{c}}_1, \dot{\boldsymbol{c}}_2, \boldsymbol{\omega}_1, \boldsymbol{\omega}_2)$ and denote the corresponding differential $\partial\boldsymbol{\theta}$. The derivative of the velocity with respect to the state $\boldsymbol{x} = \boldsymbol{q}, \boldsymbol{v}$ of the robot can be written as:

$$\frac{\partial \dot{d}}{\partial \boldsymbol{q}} = \frac{\partial \dot{d}}{\partial \boldsymbol{\theta}}\frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{q}} + \frac{\partial \dot{d}}{\partial \dot{\boldsymbol{\theta}}}\frac{\partial \dot{\boldsymbol{\theta}}}{\partial \boldsymbol{q}} \tag{17}$$

where $\partial\boldsymbol{\theta}/\partial\boldsymbol{q}$ is the concatenation of the Jacobians of $\mathcal{B}_1$ and $\mathcal{B}_2$ and $\partial\dot{\boldsymbol{\theta}}/\partial\boldsymbol{q}$ is the derivative of the body velocities with respect to the robot configuration $\boldsymbol{q}$. These Jacobians are computed by forward kinematics, in our implementation by using the `Pinocchio` library [29].

Let us denote by $\boldsymbol{y}^*(\boldsymbol{\theta})$ the value of $\boldsymbol{y}$ that solves (7) and $\mathcal{L}^*$ the optimal value of $\mathcal{L}$. Then, from (9), we have

$$\mathcal{L}(\boldsymbol{y}^*(\boldsymbol{\theta}),\boldsymbol{\theta}) = \mathcal{L}^*(\boldsymbol{\theta}) = \frac{1}{2}d^2$$

and by differentiating:

$$\mathcal{L}_{\boldsymbol{y}}\boldsymbol{y}^*_{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}} + \mathcal{L}_{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}} = d\dot{d} = \mathcal{L}_{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}}$$

since from (8), $\mathcal{L}_{\boldsymbol{y}} = 0$, the velocity is simply $\dot{d} = \frac{1}{d}\mathcal{L}_{\boldsymbol{\theta}}\dot{\boldsymbol{\theta}}$. The derivatives of $\dot{d}$ with respect to $\dot{\boldsymbol{\theta}}$ can thus be formulated as:

$$\frac{\partial \dot{d}}{\partial \dot{\boldsymbol{\theta}}} = \frac{1}{d}\mathcal{L}_{\boldsymbol{\theta}}. \tag{18}$$

and differentiating $\dot{d} = \mathcal{L}^*_{\boldsymbol{\theta}}/\sqrt{2\mathcal{L}}$ we obtain

$$\frac{\partial \dot{d}}{\partial \boldsymbol{\theta}} = \frac{\dot{\mathcal{L}}_{\boldsymbol{\theta}}}{d} - \frac{\dot{\mathcal{L}}}{d^2}d_{\boldsymbol{\theta}} = \frac{1}{d}\mathcal{L}^*_{\boldsymbol{\theta}\boldsymbol{\theta}}\dot{\boldsymbol{\theta}} - \frac{1}{d^2}\mathcal{L}_{\boldsymbol{\theta}}\dot{d}. \tag{19}$$

with:

$$\mathcal{L}^*_{\boldsymbol{\theta}} = \begin{bmatrix}\boldsymbol{x}_1-\boldsymbol{x}_2 \\ -(\boldsymbol{x}_1-\boldsymbol{x}_2) \\ -(\boldsymbol{x}_1-\boldsymbol{x}_2)\times(\boldsymbol{x}_1-\boldsymbol{c}_1) \\ (\boldsymbol{x}_1-\boldsymbol{x}_2)\times(\boldsymbol{x}_2-\boldsymbol{c}_2)\end{bmatrix}^T \tag{20}$$

The Hessian $\mathcal{L}^*_{\boldsymbol{\theta}\boldsymbol{\theta}}$ arises from the derivative of $\mathcal{L}_{\boldsymbol{\theta}}$ at $\boldsymbol{y} = \boldsymbol{y}^*$:

$$\mathcal{L}^*_{\boldsymbol{\theta}\boldsymbol{\theta}} = \mathcal{L}_{\boldsymbol{\theta}\boldsymbol{y}}\boldsymbol{y}_{\boldsymbol{\theta}} + \mathcal{L}_{\boldsymbol{\theta}\boldsymbol{\theta}}|_{\boldsymbol{y}=cst} \tag{21}$$

where:

$$\mathcal{L}_{\boldsymbol{\theta}\boldsymbol{\theta}}|_{\boldsymbol{y}=cst} = \begin{bmatrix}0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\left[\boldsymbol{x}_1-\boldsymbol{x}_2\right]_\times & 0 & 0 & 0 \\ 0 & \left[\boldsymbol{x}_1-\boldsymbol{x}_2\right]_\times & 0 & 0\end{bmatrix} \tag{22}$$

and:

$$\mathcal{L}_{\boldsymbol{\theta}\boldsymbol{y}} = \begin{bmatrix}I & -I & 0 & 0 \\ -I & I & 0 & 0 \\ -\mathcal{X}+\mathcal{X}\mathcal{C}_1 & -\mathcal{X}\mathcal{C}_1 & 0 & 0 \\ -\mathcal{X}\mathcal{C}_2 & \mathcal{X}+\mathcal{X}\mathcal{C}_2 & 0 & 0\end{bmatrix} \tag{23}$$

with $\mathcal{X} = [\boldsymbol{x}_1-\boldsymbol{x}_2]_\times$, $\mathcal{X}\mathcal{C}_1 = [\boldsymbol{x}_1-\boldsymbol{c}_1]_\times$, $\mathcal{X}\mathcal{C}_2 = [\boldsymbol{x}_2-\boldsymbol{c}_2]_\times$. The two last columns being 0, we don't need the full $\boldsymbol{y}_{\boldsymbol{\theta}}$ but only the derivatives with respect to the ellipsoids centers. To obtain those derivatives, we use the results of [30], [31] on the sensitivity analysis. Applying the implicit function theorem on the problem, we obtain:

$$\nabla_{\boldsymbol{\theta}}y(\boldsymbol{\theta}) = \begin{bmatrix}\nabla_{\boldsymbol{\theta}}x(\boldsymbol{\theta}) \\ \nabla_{\boldsymbol{\theta}}\lambda(\boldsymbol{\theta})\end{bmatrix} = -(\mathcal{L}_{\boldsymbol{y}\boldsymbol{y}}(\boldsymbol{\theta})^{-1}\mathcal{L}_{\boldsymbol{y}\boldsymbol{\theta}}(\boldsymbol{\theta})) \tag{24}$$

where:

$$\mathcal{L}_{\boldsymbol{y}\boldsymbol{y}}(\boldsymbol{\theta}) = \begin{bmatrix}\alpha & -I_3 & \gamma & 0 \\ -I_3 & \beta & 0 & \delta \\ \gamma & 0 & 0 & 0 \\ 0 & \delta & 0 & 0\end{bmatrix} \tag{25}$$

with $\alpha = I_3 + \lambda_1 A_1$, $\beta = I_3 + \lambda_2 A_2$, $\gamma = A_1(\boldsymbol{x}_1-\boldsymbol{c}_1)$, $\delta = A_2(\boldsymbol{x}_2-\boldsymbol{c}_2)$. And:

$$\mathcal{L}_{\boldsymbol{y}\boldsymbol{\theta}}(\boldsymbol{\theta}) = \mathcal{L}_{\boldsymbol{\theta}\boldsymbol{y}}(\boldsymbol{\theta})^T \tag{26}$$

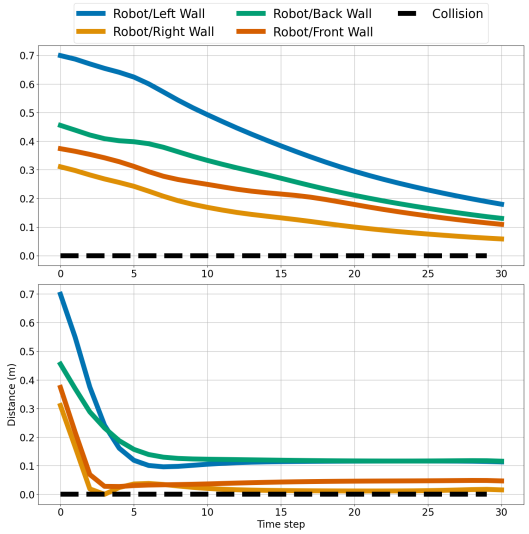


Fig. 5: Scenario 2: Distance to the obstacle (velocity damper constraint on the top and distance-based constraint on the bottom).

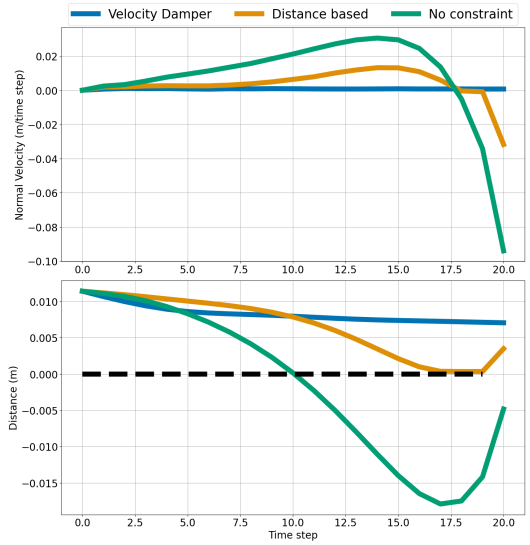


Fig. 6: Scenario 3: Normal velocity of the robot to the obstacle on the top and distance to obstacle on the bottom.

Finally, using (26), (24) and (18), (19), we obtain the derivatives of $\dot{d}$ with respect to $\boldsymbol{q}$. The derivative $\frac{d\dot{d}}{d\dot{\boldsymbol{q}}} = \frac{dd}{d\boldsymbol{q}}$ is taken from [22].

Here, we have followed the intuition that the damper for any smooth collision geometries can be obtained from only considering the normal and the curvatures around the closest points. Future work is to formally demonstrate this intuition and to exploit it for geometries that only are approximately smooth, like meshes of smooth objects. Extension to nonsmooth or nonconvex objects is another active avenue of research [32] which is out of the scope of this paper. We have already empirically validated that computing the damper from the tangent ellipsoids is effective by solving our OCP with capsules. Another second limitation is to consider equality constraints in (1), which only accurately capture the distance when the objects are not penetrating each other. It is sufficient when the OCP solver is guided by a third party (*e.g.* a motion planner or a memory of motion). If the solver also needs to explore candidate trajectories with colliding objects, then problem (1) needs to be extended to signed distance, *e.g.* using min-max optimization [23], which we also let for a future research.

IV. Experimentation

A. Setting up the OCP and the scenarios

To compare the two collision avoidance constraints, three different tasks of goal reaching were devised with different obstacle numbers and positions. The OCP costs are composed of:

- State regularization cost:

$$\ell_{\boldsymbol{x}}(\boldsymbol{x}_t) = (\boldsymbol{x}_t - \boldsymbol{x}_{t=0})^T Q_{\boldsymbol{x}} (\boldsymbol{x}_t - \boldsymbol{x}_{t=0}) \,, \tag{27}$$

- Goal reaching task cost:

$$\ell_{ee}(\boldsymbol{x}_t) = ||\log(T_{\text{goal}}^{-1} \cdot T_{ee}(\boldsymbol{q}_t))||^2 \,, \tag{28}$$

- Control regularization cost:

$$\ell_{\boldsymbol{u}}(\boldsymbol{x}_t, \boldsymbol{u}_t) = (\boldsymbol{u}_t - \boldsymbol{u}_{grav}(\boldsymbol{q}_t))^T Q_u (\boldsymbol{u}_t - \boldsymbol{u}_{grav}(\boldsymbol{q}_t)) \,, \tag{29}$$

with Q_x and Q_u are the coefficients matrices of the state and control penalization (hand-tuned diagonal matrices) and T_{goal} is *SE(3)* pose of the target.

The three scenarios are the following: a simple collision avoidance, a bin-picking task-like in a box and a contact-to-contact task. Fig. 4 provides an illustration of each setting. The three scenarios have the same OCP where only the weights of the costs and the pose of the target differ.

B. Comparison with distance-based Collision Avoidance

As mentioned before, a notable drawback of the distance-based collision avoidance constraint is its tendency to saturate, resulting in the robot staying very close to the collision threshold defined in the OCP, without accounting for the robot speed as it approaches the obstacle.

Through the different scenarios, we examine the collision risk between the robot end-effector and an obstacle. Fig. 4 shows the trajectories generated by the two different constraints in the different scenarios. The blue spheres represent the positions of the end-effector at each node of the trajectory found by the OCP solver with the distance-based constraint, while the yellow cubes, correspond to positions of the end-effector through each node of the trajectory found using the velocity damper constraint. The yellow ellipsoids depicted on the robot represent the shapes used to compute the collision constraints with the obstacles. To facilitate a clear comparison, the safety threshold ϵ for the distance-based constraint (2) was set to zero.

While the distance-based constraint is sufficient to prevent collision, we see that the solver produces safer trajectories with the velocity damper constraint, with non-zero distances to the obstacle. It is also clear that disregarding the distance

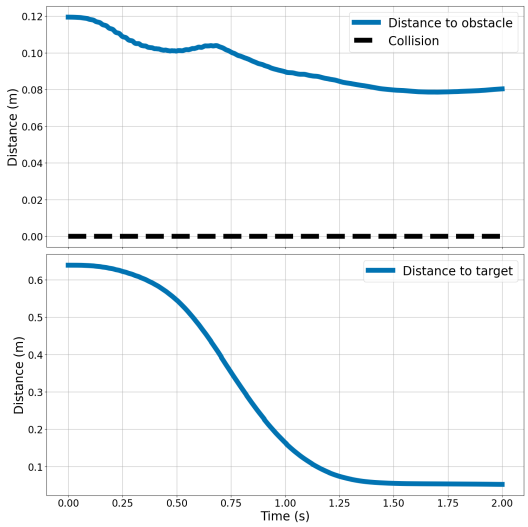


Fig. 7: Distance of the end-effector robot to the obstacle (on the top) and to the target (on the bottom).

to the obstacle leads to evident collision, as exhibited in Fig. 5.

Fig. 5 illustrates the distance between the end-effector and the obstacles composing the box over the course of the trajectory for the second scenario. In that case, we have 4 pairs of possible collisions. When only considering the distance, the optimal motion saturates the collision, as shown on the top of Fig. 5 while the velocity damper naturally leads to a motion with a nice security margin, as shown on the bottom of the Fig. 5.

Fig. 6 presents two metrics for Scenario 3: the velocity of the end-effector normal to the closest points (top) and the distance to the obstacle (bottom). In this scenario, the robot begins its motion near the obstacle and must reach a target that is also positioned close to the obstacle. As a result, the trajectory starts and ends with a minimal clearance between the robot and the obstacle. The path under the distance constraint stays near the obstacle surface (which is impractical), while the velocity damper approach allows the robot to reach the target without significantly decreasing the initial distance to the obstacle, maintaining safer clearance.

In the various scenarios tested, the distance-based constraint effectively prevents collisions, but exhibits the saturation issue discussed earlier. Moreover, Fig. 5 highlights the fast approaching velocity of the end-effector toward the obstacles. In contrast, the velocity damper constraint not only ensures collision avoidance but also allows the robot to maintain a safer distance from obstacles. Additionally, Fig. 6 shows that the velocity damper significantly reduces the robot approach speed when nearing obstacles, enhancing overall safety during operation.

C. Experimental Validation on the Real Robot

We implemented the velocity damper constraint on a real 7-degree-of-freedom torque-controlled Franka Emika Panda robot, operating in real-time using an MPC framework. The experimental setup is described as follows:

1) Experimental Setup: The experiment utilized a Franka Emika Panda robot with 7 degrees of freedom, controlled via torque inputs. The setup mirrors the one used in [22]. The MPC was implemented in C++ and executed on an AMD Ryzen 9 5950x processor running at 3.4 GHz. For constraint handling, we employed `mim-solvers` [33] with `crocoddyl` [34], as used in the simulations presented in Sec.IV-B. The robot dynamic model was characterized using the inertial parameters from [35]. Control inputs were computed at a frequency of 1 kHz, while the OCP was solved at a frequency of 100 Hz.

2) Experimental validation: The setup of the OCP is the same as the one presented in Scene 2. The end-effector of the robot has to go in and out of the box while avoiding its walls. To ensure sufficient computational margin during the solving of the OCP, the collision avoidance constraint was applied exclusively between the end-effector and the right side of the wall. This selective constraint application was chosen to balance efficiency with safety while maintaining real-time performance. The motion is illustrated in Fig. 1. Fig. 7 plots the distance of the end-effector to the obstacle. The video included highlights the highly dynamical and efficient collision avoidance while remaining compliant to external disturbance, along with movements in more complex and dynamic scenes.

V. Conclusion

In this paper, we revisit the velocity damper constraint in the context of MPC. Analytical derivatives were provided and implemented to ensure efficient integration with numerical solvers. The approach was deployed on a torque-controlled Franka Emika Panda robot. Experimental results confirm the advantages of the velocity damper constraint compared to the traditional distance-based control. More specifically, the proposed approach maintains safer distances from obstacles during dynamic tasks. This work reaffirms the practical value of the velocity damper constraint in modern robotics, bridges the gap with MPC and offers an updated open-source implementation for the research community.

Future work will aim to extend this approach to all convex primitives and to more complex environments, including multiple dynamic obstacles and more intricate geometries, while optimizing solver performance to manage the computational complexity of challenging scenarios efficiently.

References

[1] B. Siciliano and O. Khatib, *Handbook of Robotics*. Springer, 2008.
[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.
[3] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *IEEE International Conference on Robotics and Automation (ICRA)*, 1999.
[4] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, may 2006.
[5] J. Kuffner and S. LaValle, "RRT-connect: An efficient approach to single-query path planning," *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.
[6] R. Glasius, A. Komoda, and S. C. A. M. Gielen, "Neural Network Dynamics for Path Planning and Obstacle Avoidance," *Neural Networks*, vol. 8, pp. 125–133, Jan. 1995.

[7] S. X. Yang and M. Meng, "An efficient neural network approach to dynamic robot motion planning," *Neural Networks*, vol. 13, no. 2, pp. 143–148, Mar. 2000.

[8] T. Jurgenson and A. Tamar, "Harnessing Reinforcement Learning for Neural Motion Planning," *Robotics: Science and Systems (RSS)*, 2019.

[9] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From Perception to Decision: A Data-driven Approach to End-to-end Motion Planning for Autonomous Ground Robots," *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[10] D. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, Nov. 2014.

[11] A. El Khoury, F. Lamiraux, and M. Taïx, "Optimal motion planning for humanoid robots," *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[12] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[13] J. Schulman *et al.*, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, Aug. 2014.

[14] C. Khazoom, D. Gonzalez-Diaz, Y. Ding, and S. Kim, "Humanoid Self-Collision Avoidance Using Whole-Body Control with Control Barrier Functions," *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[15] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *IEEE International Conference on Robotics and Automation (ICRA)*, 1985.

[16] X. Xu, Y. Hu, J. Zhai, L. Li, and P. Guo, "A novel non-collision trajectory planning algorithm based on velocity potential field for robotic manipulator," *International Journal of Advanced Robotic Systems*, vol. 15, no. 4, Jul. 2018, sAGE Publications.

[17] 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 Minimum-Jerk Robot Motion Generation," *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[18] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[19] A. Escande, S. Miossec, and A. Kheddar, "Continuous gradient proximity distance for humanoids free-collision optimized-postures," *IEEE-RAS 7th International Conference on Humanoid Robots (Humanoids)*, 2007.

[20] S. Miossec, K. Yokoi, and A. Kheddar, "Development of a software for motion optimization of robots - application to the kick motion of the hrp-2 robot," *2006 IEEE International Conference on Robotics and Biomimetics*, 2006.

[21] A. El Khoury, F. Lamiraux, and M. Taïx, "Optimal motion planning for humanoid robots," *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[22] A. Haffemayer, A. Jordana, M. Fourmy, K. Wojciechowski, G. Saurel, V. Petrik, F. Lamiraux, and N. Mansard, "Model Predictive Control Under Hard Collision Avoidance Constraints for a Robotic Arm," *21st International Conference on Ubiquitous Robots (UR)*, 2024.

[23] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," *IEEE International Conference on Robotics and Automation (ICRA)*, 1987.

[24] S. Stavridis and Z. Doulgeri, "Bimanual Assembly of Two Parts with Relative Motion Generation and Task Related Optimization," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[25] S. Stavridis, P. Falco, and Z. Doulgeri, "Pick-and-place in dynamic environments with a mobile dual-arm robot equipped with distributed distance sensors," *IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, 2021.

[26] W. Zhang and H. Wen, "Motion planning of a free-flying space robot system under end effector task constraints," *Acta Astronautica*, vol. 199, pp. 195–205, Oct. 2022.

[27] F. Kanehiro, W. Suleiman, F. Lamiraux, E. Yoshida, and J.-P. Laumond, "Integrating dynamics into motion planning for humanoid robots," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[28] J. Solà, J. Deray, and D. Atchuthan, "A micro Lie theory for state estimation in robotics," Dec. 2021. [Online]. Available: http://arxiv.org/abs/1812.01537

[29] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The Pinocchio C++ library : A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," *2019 IEEE/SICE International Symposium on System Integration (SII)*, pp. 614–619, 2019.

[30] A. V. Fiacco, "Chapter 2 Basic Sensitivity and Stability Results," in *Mathematics in Science and Engineering*, ser. Introduction to Sensitivity and Stability Analysis in Nonlinear Programming. Elsevier, Jan. 1983, vol. 165, pp. 8–64.

[31] G. Giorgi and C. Zuccotti, "A Tutorial on Sensitivity and Stability in Nonlinear Programming and Variational Inequalities under Differentiability Assumptions," 2018.

[32] L. Montaut, Q. L. Lidec, A. Bambade, V. Petrik, J. Sivic, and J. Carpentier, "Differentiable collision detection: a randomized smoothing approach," *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[33] A. Jordana, S. Kleff, A. Meduri, J. Carpentier, N. Mansard, and L. Righetti, "Stagewise implementations of sequential quadratic programming for model-predictive control," *IEEE Transactions on Robotics (T-RO)*, 2023.

[34] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control," *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[35] C. Gaz, M. Cognetti, A. Oliva, P. Giordano, and A. Luca, "Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 4147 – 4154, 2019.