Model predictive control under hard collision avoidance constraints for a robotic arm

Arthur Haffemayer, Armand Jordana, Médéric Fourmy, Krzysztof Wojciechowski, Guilhem Saurel, Vladimír Petrík, Florent Lamiraux, Nicolas Mansard

Model predictive control under hard collision avoidance constraints for a robotic arm

Arthur Haffemayer[1,2,3], Armand Jordana[4], Médéric Fourmy[5], Krzysztof Wojciechowski[1], Guilhem Saurel[1], Vladimir Petrik[5], Florent Lamiraux[1], and Nicolas Mansard[1,2]

***Abstract*— We design a method to control the motion of a manipulator robot while strictly enforcing collision avoidance in a dynamic obstacle field. We rely on model predictive control while formulating collision avoidance as a hard constraint. We express the constraint as the requirement for a signed distance function to be positive between pairs of strictly convex objects. Among various formulations, we provide a suitable definition for this signed distance and the analytical derivatives the numerical solver needs to enforce the constraint. The method is completely implemented on a manipulator "Panda" robot, and the efficient open-source implementation is provided along with the paper. We experimentally demonstrate the efficiency of our approach by performing dynamic tasks in an obstacle field while reacting to non-modeled perturbations.**

I. Introduction

For manipulator robots to act in a dynamic environment while achieving versatile tasks, they need to generate collision-free trajectories efficiently. While the collision constraint is one of the first that roboticists have considered, particularly in motion planning, a generic control method that would enforce collision-safe movements while enabling versatile programming of the robot motion objectives is still lacking.

Early approaches often relied on randomized motion planning, such as Rapidly Exploring Random Trees (RRT) algorithms to generate collision-free trajectories [1]. The planning phase is complemented by trajectory-following algorithms [2], ensuring precise adherence of the robot to the planned trajectory. Besides, the trajectories planned with randomized algorithms tend to have sub-optimal dynamics [1]. Trajectory optimization can then filter the random search output and produce a local optimum near the planned trajectory [3]. Several studies have led to improving the capabilities of numerical solvers to handle obstacle avoidance. In [4], a specific (covariant Hamiltonian-based) numerical algorithm optimizes the motion of various robotic systems while considering collision distances. Approach [5] works on the cost formulation to introduce the collision constraint in a more classical sequential-quadratic solver. We follow this direction by focusing on the constraint formulation to exploit off-the-shelf solvers. However, these early works could only consider the collisions in the off-line planning phase while relying on trajectory tracking to control the robot safely. This prevents safe disturbance rejection or dynamic adaptation to changes in the tasks or the motion of the nearby obstacles.

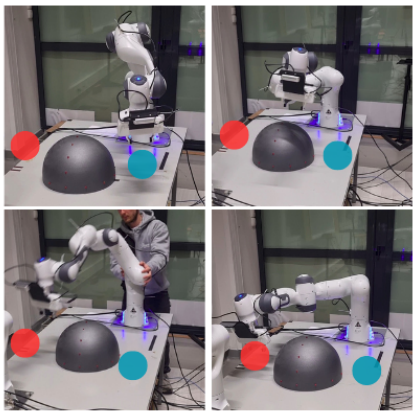


Fig. 1: Highly dynamical yet compliant torque-controlled robot avoiding a fixed obstacle with MPC.

Building on the progress in optimal control, model predictive control (MPC) has become increasingly popular in robotics due to its ability to generate complex motions online [6], [7]. To meet the computational needs of robotics, a variety of optimization algorithms tailored for optimal control have been proposed [8], [9], [10]. The first numerical algorithms used in MPC for robotics could typically not handle hard constraints. In practice, penalization functions were used for collision constraints [11], [12], [13], [14]. Most of the previous papers relied on gradient-based optimization although gradient-free (evolution strategy) has also shown interesting capabilities when using the sampling power of a GPU [14]. In both gradient-based and gradient-free approaches, the penalty implies that the solver has to find a compromise between optimized quantities (energy, time optimality, etc.) and collision avoidance. The trade-off is defined by a parameter whose value must be manually adjusted. This is a classical drawback of penalty-based MPC solvers. Compared to other constraints (e.g. torque or velocity limits), it is particularly harmful when dealing with constraints because of the lack of general margin: being conservative can easily lead to a loss of a large part of the workspace, while small violations of this constraint cannot be accepted. This makes tuning the trade-off parameter particularly delicate

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
[5] CIIRC, Czech Technical University in Prague
*corresponding author: arthur.haffemayer@laas.fr

to achieve the desired robot behavior. In some instances, researchers resort to reinforcement learning to identify the optimal set of weights [15].

Recently, several algorithms have been proposed to enforce strict constraint satisfaction [16], [17], [18]. These solvers have the potential to enforce hard constraints in real-time in an MPC implementation. In particular, we consider in this paper a tailored implementation of Sequential Quadratic Programming (SQP) for optimal control problems [19]. This implementation showed promising results in hardware experiments by performing closed-loop MPC while strictly enforcing simple end-effector constraints.

In this paper, we propose to capitalize on these newly available solvers to establish a generic versatile method for controlling the robot while strictly enforcing collision avoidance. We present a simple and rigorous way to write collision avoidance as a hard constraint. We then efficiently implement this formulation into an SQP numerical scheme to optimize the robot trajectory while avoiding collisions in real-time on an MPC scheme. We demonstrate the value of this controller by hardware experiments using a torque-driven manipulator robot, performing pick-and-place-like tasks in an obstacle field while reacting to the physical perturbation of a human operator (see Fig. 1). To our knowledge, this is the first experimental demonstration of a torque-controlled manipulator arm employing nonlinear MPC with collision avoidance explicitly formulated as a hard constraint and solved using SQP. The practical implications of the results are noteworthy: the robot demonstrates versatile adaptability to perturbations during dynamic tasks, achieves precise target reaching, and ensures complete compliance, particularly when interacting with an operator, while successfully avoiding specified obstacles.

II. SQP-Based MPC

In this section, we formulate the motion generation problem as an OCP and introduce our collision constraint's abstract formulation.

A. OCP formulation

The problem is formulated as an OCP of the form :

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

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

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

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

where $\boldsymbol{x}_t = (\boldsymbol{q}_t, \boldsymbol{v}_t)$ is the robot state concatenating robot configuration $\boldsymbol{q}_t$ and velocity $\boldsymbol{v}_t = \dot{\boldsymbol{q}}_t$, and $\boldsymbol{u}_t = \boldsymbol{\tau}_t$ are the controlled joint torques; 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 optimization variables 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 costs, written here as ℓ_t for the running cost and ℓ_T for the terminal cost specify the goal-reaching task. Functions c_t and c_T represent the hard (inequality) constraints at each time step. The SQP takes state and control sequences as initial guesses and solves (1). Sec. IV-A describes these costs and constraints.

The first MPC obtained on real robot hardware [11], [7], [10] were constraint-free, i.e. not able to take into account c_t, c_T. Here we instead consider a constraint-based OCP solver and show how c_t, c_T should be chosen to enforce collision avoidance. We first discuss how the solver works, defining how c_t, c_T must be chosen.

B. SQP resolution

There exists a vast body of work providing efficient solvers tailored for optimal control [16], [17], [9], [18], [19]. These solvers typically combine the main principle of a constrained nonlinear solver [20] with some particular developments to handle the temporal structure (sparsity) of the OCP (1a). This is generally done by using the Riccati recursion to fully exploit the sparsity of the OCP, following the initial idea of the differential dynamic programming (DDP) algorithm [21].

In this work, we use the solver proposed in [19]. This solver relies on an SQP algorithm using the operator-splitting quadratic program (QP) solver OSQP [22] in the inner loop. This QP solver is in turn taking advantage of the sparse structure of the OCP by implementing a DDP recursion.

At each control cycle, the solver needs to compute one or several steps of SQP. Each step implies (i) the evaluation of the derivatives of the functions ℓ_t, ℓ_T, f_t, c_t and c_T along each point of the candidate trajectories $\boldsymbol{x}$, $\boldsymbol{u}$, (ii) the resolution of the QP in the inner loop, (iii) and a line-search or equivalent, by evaluating possibly multiple times the functions ℓ_t, ℓ_T, f_t, c_t and c_T to decide the new value of the decision variables. This implies that the obstacle constraints c_t, c_T must be written in a form that allows efficient evaluation of its value and its derivative.

III. Obstacle avoidance as a hard constraint

In this section, we formulate the obstacle avoidance constraint as a hard constraint.

A. Signed distance

We consider two objects A and B, modeled as convex and compact subsets of $\mathbb{R}^3$. We then denote by:

- A°, B° the interior sets of A and B,
- $\partial A = A \setminus A^\circ$, $\partial B = B \setminus B^\circ$ the boundaries of A and B.

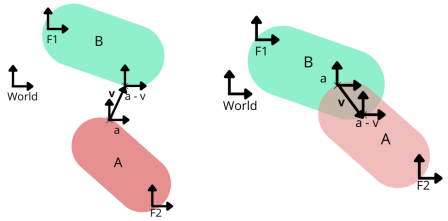
We say that A and B:

- are in collision if $A^\circ \cap B^\circ \neq \emptyset$,
- are in contact if $A^\circ \cap B^\circ = \emptyset$ and $A \cap B \neq \emptyset$,
- are collision-free if $A \cap B = \emptyset$.

If A and B are not in collision, we define the signed distance between A and B by the following optimization problem:

$$d(A, B) = \min_{\mathbf{v} \in \mathbb{R}^3} \|\mathbf{v}\|\,, \tag{2}$$

$$\text{such that } A \text{ and } B + \mathbf{v} \text{ are in contact}\,,$$



(a) Not in collision (b) Shapes in collision

Fig. 2: Definition of the signed distance and of the witness points between two convex shapes.

where $B + \mathbf{v} = \{\mathbf{b} + \mathbf{v}, \mathbf{b} \in B\}$ is the translation of B by $\mathbf{v}$. When A and B are in collision, we define the signed distance between A and B by

$$d(A, B) = -\min_{\mathbf{v} \in \mathbb{R}^3} \|\mathbf{v}\|\,, \tag{3}$$

$$\text{such that } A \text{ and } B + \mathbf{v} \text{ are in contact}\,.$$

If $\mathbf{v}$ is a minimizer of (2) or (3), and $\mathbf{a} \in A \cap B + \mathbf{v}$, we say that $\mathbf{a}$ and $\mathbf{a} - \mathbf{v}$ are witness points on A and B, respectively (see Fig. 2). If A and B are strictly convex, then the witness points are unique, they lie on the boundaries of A and B and their distance is the absolute value of $d(A, B)$.

B. Derivative of the signed distance

We now consider that A and B are strictly convex, and move with respect to each other. We denote by $T \in SE(3)$ the relative pose between the two objects. The signed distance $d(T)$ is continuous and piecewise C^1 on $SE(3)$. We denote by $\mathbf{w}_A$,$\mathbf{w}_B$ the witness points on respectively A and B. The derivative of the signed distance d with respect to the relative pose of the objects can be expressed by differentiating[1]:

$$d = \sigma\sqrt{(\mathbf{w}_A - \mathbf{w}_B | \mathbf{w}_A - \mathbf{w}_B)},$$

where $\sigma = 1$ if the objects are collision-free and -1 if they are in collision:

$$\frac{\partial d}{\partial T} = \frac{\sigma}{d}(\mathbf{w}_B - \mathbf{w}_A)(\frac{\partial \mathbf{w}_B}{\partial T} - \frac{\partial \mathbf{w}_A}{\partial T})\,, \tag{4}$$

where $\frac{\partial \mathbf{w}_A}{\partial T}$ is the tangent application of $\mathbf{w}_A$ with respect to T (and respectively for $\mathbf{w}_B$).

C. Derivative with respect to the robot configuration

Let us consider that the two bodies are attached to a joint of the robot or an obstacle of the environment. Now $T = T(\boldsymbol{q})$ only depends on the configuration $\boldsymbol{q}$ of the robot. As explained in [24, Sec.III], we can replace in (4) $\frac{\partial \mathbf{w}_A}{\partial T}$, $\frac{\partial \mathbf{w}_B}{\partial T}$ by the Jacobians of the coinciding points on $\mathbf{w}_A$ and $\mathbf{w}_B$, thus getting the Jacobian with respect to the robot configuration:

$$\frac{\partial d}{\partial \boldsymbol{q}} = \frac{\sigma}{d}(\mathbf{w}_B - \mathbf{w}_A)(\frac{\partial \mathbf{w}_B}{\partial \boldsymbol{q}} - \frac{\partial \mathbf{w}_A}{\partial \boldsymbol{q}})\,, \tag{5}$$

where now $\frac{\partial \mathbf{w}_A}{\partial \boldsymbol{q}}$ (resp. $\frac{\partial \mathbf{w}_B}{\partial \boldsymbol{q}}$) is the Jacobian of $\mathbf{w}_A$ (resp. $\mathbf{w}_B$) with respect to the robot configuration $\boldsymbol{q}$ or 0 if A (resp. B) is a static obstacle. These Jacobians are computed by forward kinematics (using the `Pinocchio` library [25]).

[1] see [23] for details about differentiation on $SE(3)$.

D. Integrating the constraints in the solver

For each relevant pair of objects $\mathcal{B}_i, \mathcal{B}_j$ in the robot model, that may be in collision and for each time discretization step $t \in \{1, \cdots, T\}$, we add an inequality constraint of the type:

$$c_{t,i,j}(\boldsymbol{x}_t) = d_{ij}(\boldsymbol{x}_t) - \epsilon \geq 0\,, \tag{6}$$

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

IV. Implementing a reaching task with collision avoidance

In this section, we present the OCP used in the experimental section and implement a reaching task under collision constraints. The program implements the template (1a) with c_t, c_T a vector concatenating the signed distance function introduced in (2),(3) for each relevant pair of collision bodies $\mathcal{B}_i, \mathcal{B}_j$. The dynamics f_t is computed from the forward dynamics (articulated body algorithm [26]) integrated using Euler method [27]. The cost functions ℓ_t, ℓ_T are chosen to penalize the distance to the target to reach while regularizing the state and the control, as explained below.

A. Goal-reaching task

The task is defined by the formulation of the running and terminal costs in Problem (1). For the goal-reaching task, the running and terminal costs are split into a goal-reaching term and regularization terms.

They are formulated as follows:

$$l_t(\boldsymbol{x}_t, \boldsymbol{u}_t) = \omega_{ee} l_{ee}(\boldsymbol{x}_t) + \omega_{\boldsymbol{x}} l_{\boldsymbol{x}}(\boldsymbol{x}_t) + \omega_{\boldsymbol{u}} l_u(\boldsymbol{x}_t, \boldsymbol{u}_t)\,, \tag{7a}$$

$$l_T(\boldsymbol{x}_T) = \omega_{ee} l_{ee}(\boldsymbol{x}_T) + l_{\boldsymbol{x}}(\boldsymbol{x}_T)\,, \tag{7b}$$

where $l_{ee}(\boldsymbol{x}_t)$ is the goal reaching term, $l_{\boldsymbol{x}}(\boldsymbol{x}_t)$ is the state regularization and $l_u(\boldsymbol{x}_t)$ is the control regularization. Parameters ω_i are the weights of the different costs.

1) Goal-reaching cost: The goal-reaching cost minimizes the distance between the end effector pose of the robot and the goal pose. The cost is formulated as follows:

$$\ell_{ee}(\boldsymbol{x}_t) = ||\log(T_{goal}^{-1} \cdot T_{ee}(\boldsymbol{q}_t))||^2\,, \tag{8}$$

where T_{ee}, $T_{goal} \in SE(3)$ are respectively the end effector pose obtained from forward kinematics and a pose goal. Function $\log : SE(3) \to \mathbb{R}^3$ is the SE(3) logarithm map [23].

2) State regularization cost: The state regularization cost penalizes both extreme joint configurations (i.e. far from the initial configuration) and high joint speed:

$$\ell_{\boldsymbol{x}}(\boldsymbol{x}_t) = (\boldsymbol{x}_t - \boldsymbol{x}_{init})^T Q_{\boldsymbol{x}} (\boldsymbol{x}_t - \boldsymbol{x}_{init})\,, \tag{9}$$

where $\boldsymbol{x}_{init} = (\boldsymbol{q}_{init}^T \quad 0^T)$, the initial configuration of the robot as the start of the problem.

3) Control regularization cost: The control regularization cost is defined to keep the controls close to the torque compensating for gravity at a given configuration:

$$\ell_{\boldsymbol{u}}(\boldsymbol{x}_t, \boldsymbol{u}_t) = (\boldsymbol{u}_t - \boldsymbol{u}_{grav}(\boldsymbol{q}_t))^T Q_u (\boldsymbol{u}_t - \boldsymbol{u}_{grav}(\boldsymbol{q}_t))\,, \tag{10}$$

where $\boldsymbol{u}_{grav}(\boldsymbol{x}_t)$ is the torque compensating for gravity at configuration $\boldsymbol{q}_t$.

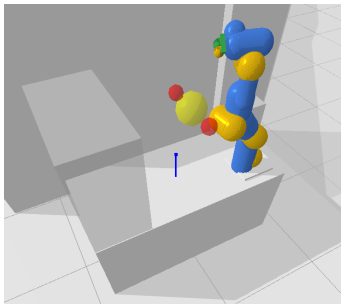


Fig. 3: Scene arrangement and capsule decomposition.

B. Receding horizon scheme for a pick-and-place task

We use the costs formulated in the previous subsection to implement an MPC scheme for reaching two target points alternately. The horizon is then composed of temporal stages: initially reaching goal G_1, then to the second goal G_2 before returning to G_1, and so forth. To simplify the implementation, we maintain a constant horizon in the OCP solver for a short duration with goal G_1 and then promptly switch to G_2, and so on. This approximation follows a receding horizon pattern on the cycle G_1, G_2 while ensuring optimal robot behavior. The collision constraint is always enforced.

V. Experimentation

This section experimentally evaluates the performances of our approach in comparison to previous methods and on the real hardware, on a Franka Emika Panda (see Fig. 1).

A. Comparison with penalization

We first propose a comparative analysis in simulation between two frequently used solvers in MPC, SQP [19] and Feasibility-driven Differential Dynamic Programming (FDDP) [28]. We scrutinize their collision-handling capabilities and assess the smoothness of their behaviors—key facets influencing their efficacy in real-world robotic applications.

1) Comparison setup: To compare the two methods, we devised a scene where the end-effector is required to reach one designated target and then another, all while moving around a stationary obstacle placed in its trajectory, as illustrated in Fig. 1.

The geometry of the robot is decomposed into capsules (see Fig. 3). Since only the last three links are in proximity to the obstacle in the trajectory, we selectively add in the OCP, as denoted by (1), only the collision constraints between the obstacle and those last three links.

As mentioned earlier, FDDP and unconstrained SQP (USQP) cannot incorporate hard constraints. Therefore, we include the collision term as a quadratic barrier by adding the following term to ℓ_t and ℓ_T:

$$\ell_B(\boldsymbol{q}) = \begin{cases} (d(\boldsymbol{q}) - \epsilon)^2 & \text{if } d(\boldsymbol{q}) \leq \epsilon\,, \\ 0 & \text{otherwise}\,. \end{cases}$$

We used the same collision margin $\epsilon = 0.5\ cm$ when including $d(\boldsymbol{q})$ as a penalty (in FDDP and USQP) or as a constraint (in our constrained SQP (CSQP)).

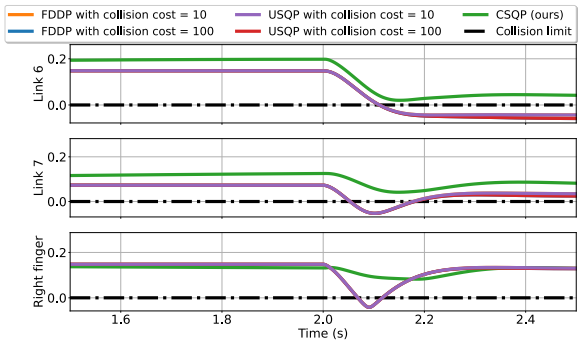


Fig. 4: Minimal distance (in meters) between the obstacle and the closest links of the robot.

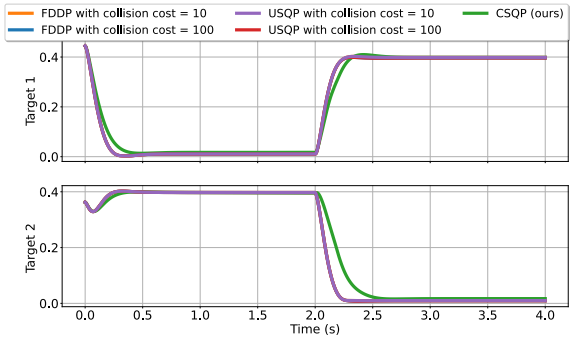


Fig. 5: Distance (in meters) between the end effector and the two targets.

The trade-off between the reaching cost ℓ_t and the barrier ℓ_B is parameterized by a weighting factor whose value is manually tuned. For the penalization method to accurately account for collisions, the weights assigned to the collision cost must be set to a significantly high value. However, setting those weights too high causes instabilities in the state and control of the robot. Hence, different sets of weights for the collision cost are tested (ranging from 10 to 100).

Excluding the collision constraints, the OCP problem remains identical. Both are implemented using the `crocoddyl` OCP library [27], either using the FDDP solver implemented in the package, or the unconstrained SQP solver from the add-on `mim-solvers` [19].

2) Comparison results: We compare the behavior in a simulation of the two solvers. The results are summarized by Fig. 4 and 5.

The distances between the robot and the obstacles are plotted in Fig. 4. While the CSQP manages to enforce a strict positive distance at all times, the FDDP and USQP fail and reach a net collision, regardless of the penalty tuning. If implementing it in a real scenario, more tuning of the penalty and the security margin would be necessary, while implementing the avoidance with a strict constraint immediately provides a safe and guaranteed behavior.

Fig. 5 plots the distance with the two successive targets during the first pick-and-place cycle. The solutions given by the penalized MPC overlap because they reach nearly

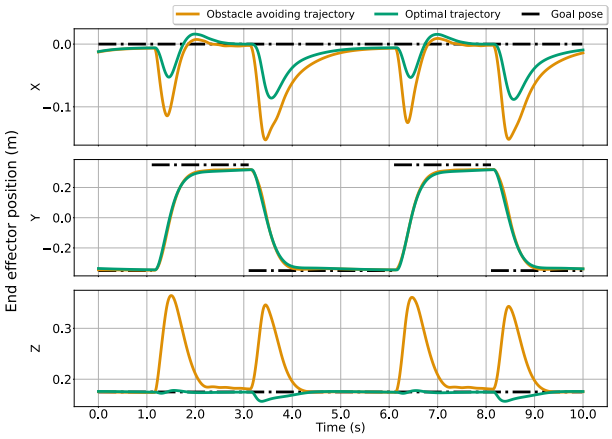


Fig. 6: Cartesian pose of the end effector.

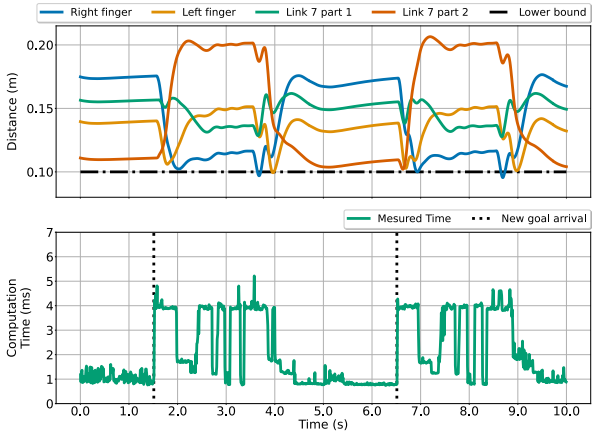


Fig. 7: Computation time compared to the distance obstacle / selected links for five collision avoidance constraints.

the same optimal solution. All trajectories are smooth and efficiently reach the targets, independently of the implementation of the avoidance. The penalized MPC is marginally faster to reach the objective, which is explained by the shortcuts taken by the robot while colliding with the obstacle. There are no side effects of using the hard constraint in the quality of the solution found by the CSQP.

B. Experimentation on the real robot

We then experimentally validate the OCP with a constrained optimization solver using the following setup.

1) Experimental setup: We used a 7 degrees of freedom torque-controlled Franka Emika Panda robot. Our MPC controller was implemented in C++ and we ran it on an AMD Ryzen 9 5950x @ 3.4 GHz. The constrained solver used is the same as the simulations in V-A CSQP from mim-solvers [19]. To characterize the dynamics of the robot, we use the inertial parameters from [29]. The torque controlling the robot is computed at 1 kHz but the OCP is only solved at 100 Hz.

Contrary to previous works [30], we are not using the Ricatti gains to increase the frequency of the control update, as (i) manipulator robots are less sensitive to control delays than legged robots and (ii) the equivalence between Ricatti gains and policy derivatives exhibited in [30] is not straight forward enough to generalize to an OCP with inequality constraints[2].

2) Experimental validation: We set up the same configuration as in the simulation (see 1) with the MPC described in Section IV-B. An obstacle, represented by a half-sphere on the table, is positioned along the path of the optimal trajectory without an obstacle, and two targets, G_1 and G_2, are placed on each side of the obstacle. In this scenario, the solver is tasked with finding an alternative trajectory.

The MPC parameters were set to $N_h = 20$ number of nodes and a $\delta = 50\ ms$, time steps of the solver (optimization over a horizon of 1 s). The safety margin for the collisions, expressed as the lower bound threshold, for the distance constraint is fixed at 10 cm. The results are summarized by Fig. 6, 7, Table I, and the accompanying video.

Fig. 6 plots the two trajectories of the end-effector, with and without considering the obstacle. Both trajectories are smooth but clearly illustrate the disturbance induced by the obstacle.

Fig. 7 illustrates the minimal distance between the obstacle and the links designated as collision pairs. Below that, the corresponding time taken to solve the OCP is presented. Two distinct phases are observable: the initial phase occurs when the robot is at the targeted position and stationary, resulting in solutions in less than 1 ms, as illustrated in Table I. The second phase starts with the robot receiving the new target. On average, the solver finds a solution between 3.5 ms to 4 ms however, it occasionally extends to 7 ms when multiple links of the robot are close to collision with the obstacle.

Given that the time limit for finding a solution is 10 ms, and considering the marginal difference in the maximum time the solver takes to find a solution, increasing the number of collision pairs might seem feasible.

Nevertheless, as demonstrated in the video [31], adding more collision pairs is not relevant as only 5 pairs (both fingers, the hand, and the two parts of the link 7) are in collision in the optimal trajectory for this problem.

However, we can see in Fig. 7 that the right finger violates the constraint by 5 mm. At times, the solver reaches its iteration limits and fails to find a solution that prevents collisions. Nonetheless, the solution provided by the solver is sufficiently effective as a suitable warm start, thereby mitigating further collision risks.

Despite the fast motion of the end effector and the abruptness of the change of targets, the robot successfully avoids the obstacle and stays within safe boundaries. In conclusion, this experiment and the examples presented in the accompanying video demonstrate the potential of obstacle avoidance with MPC and hard constraints.

[2]in particular as the policy becomes nondifferentiable at constraint saturation.

Collision pairs	First phase *(ms)*	Second phase *(ms)*	Max *(ms)*
1	0.75 ± 0.08	3.54 ± 0.18	6.58
2	0.73 ± 0.06	3.53 ± 0.17	6.47
3	0.76 ± 0.06	3.74 ± 0.17	7.11
4	0.79 ± 0.06	3.73 ± 0.15	6.95
5	0.83 ± 0.06	3.97 ± 0.13	7.38

TABLE I: Computational time statistics.

VI. Conclusion

This paper introduces an effective method for controlling robot motion with strict collision avoidance in a versatile task and obstacle setting. We relied on an OCP solver able to enforce hard constraints and formulated collision avoidance as a smooth signed distance function that we want to keep positive. After selecting a suitable definition, we provide analytical derivatives for numerical solver enforcement.

The method is implemented on a torque-controlled manipulator, and an efficient open-source implementation accompanies this paper [31]. Experimental trials demonstrate the efficiency of the approach to achieve a dynamic task while handling non-modeled perturbations. This work contributes a practical solution for robot motion control, offering both a novel methodology and an open-source implementation for the research community.

Our current endeavor involves expanding the applicability of our controller to encompass more intricate scenarios, characterized by an increased number of obstacles and their varying geometric complexities. This expansion introduces notable challenges, notably in optimizing the numerical efficiency of the solver to address the inherent complexity of nonconvex situations effectively.

References

[1] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," in *IEEE ICRA*, 1999. [Online]. Available: https://ieeexplore.ieee.org/document/770022

[2] C. Samson, "Control of chained systems application to path following and time-varying point-stabilization of mobile robots," *IEEE transactions on Automatic Control*, vol. 40, no. 1, pp. 64–77, 1995.

[3] A. El Khoury, F. Lamiraux, and M. Taïx, "Optimal motion planning for humanoid robots," in *2013 IEEE ICRA*, May 2013. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6631013

[4] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *2009 IEEE ICRA*, May 2009. [Online]. Available: http://ieeexplore.ieee.org/document/5152817/

[5] 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. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364914528132

[6] M. Neunert *et al.*, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE RAL*, 2018.

[7] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-Frequency Nonlinear Model Predictive Control of a Manipulator," in *2021 IEEE ICRA*, May 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9560990/

[8] F. Farshidian, M. Neunert, A. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *IEEE ICRA*, 2017.

[9] R. Verschueren *et al.*, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, 2022.

[10] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," in *IEEE ICRA*, 2021.

[11] E. Dantec *et al.*, "Whole body model predictive control with a memory of motion: Experiments on a torque-controlled talos," in *IEEE ICRA*, 2021.

[12] M. Gaertner, M. Bjelonic, F. Farshidian, and M. Hutter, "Collision-Free MPC for Legged Robots in Static and Dynamic Scenes," Mar. 2021, arXiv:2103.13987 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2103.13987

[13] J.-R. Chiu, J.-P. Sleiman, M. Mittal, F. Farshidian, and M. Hutter, "A collision-free mpc for whole-body dynamic locomotion and manipulation," in *IEEE ICRA*, 2022.

[14] B. Sundaralingam *et al.*, "cuRobo: Parallelized Collision-Free Minimum-Jerk Robot Motion Generation," 2023, arXiv:2310.17274 [cs]. [Online]. Available: http://arxiv.org/abs/2310.17274

[15] E. D'Elia, J.-B. Mouret, J. Kober, and S. Ivaldi, "Automatic Tuning and Selection of Whole-Body Controllers," in *IEEE IROS*, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9981058

[16] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IROS*, 2019.

[17] G. Frison and M. Diehl, "Hpipm: a high-performance quadratic programming framework for model predictive control," *IFAC-PapersOnLine*, vol. 53, no. 2, 2020.

[18] W. Jallet, A. Bambade, E. Arlaud, S. El-Kazdadi, N. Mansard, and J. Carpentier, "Proxddp: Proximal constrained trajectory optimization," *Subm. IEEE TRO*, 2023. [Online]. Available: https://inria.hal.science/hal-04332348/document

[19] A. Jordana, S. Kleff, A. Meduri, J. Carpentier, N. Mansard, and L. Righetti, "Stagewise implementations of sequential quadratic programming for model-predictive control," *Subm. IEEE TRO*, 2023. [Online]. Available: https://laas.hal.science/hal-04330251v1/document

[20] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 2006.

[21] D. Q. Mayne, "Differential dynamic programming–a unified approach to the optimization of dynamic systems," in *Control and dynamic systems*. Elsevier, 1973, vol. 10, pp. 179–254.

[22] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An Operator Splitting Solver for Quadratic Programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020, arXiv:1711.08013 [math]. [Online]. Available: http://arxiv.org/abs/1711.08013

[23] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *CoRR*, vol. abs/1812.01537, 2018. [Online]. Available: http://arxiv.org/abs/1812.01537

[24] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *IEE ICRA*, 1987.

[25] J. Carpentier *et al.*, "The Pinocchio C++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *International Symposium on System Integration (SII)*, 2019.

[26] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[27] C. Mastalli *et al.*, "Crocoddyl: An efficient and versatile framework for multi-contact optimal control." IEEE ICRA, 2020.

[28] C. Mastalli, W. Merkt, J. Marti-Saumell, H. Ferrolho, J. Sola, N. Mansard, and S. Vijayakumar, "A Feasibility-Driven Approach to Control-Limited DDP," 2022, arXiv:2010.00411 [cs, eess]. [Online]. Available: http://arxiv.org/abs/2010.00411

[29] 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 RAL*, vol. PP, 2019.

[30] E. Dantec, M. Taïx, and N. Mansard, "First order approximation of model predictive control solutions for high frequency feedback," *IEEE RAL*, vol. 7, no. 2, 2022.

[31] A. Haffemayer, "Colmpc: Collision Avoidance for MPC." [Online]. Available: https://gepettoweb.laas.fr/articles/haffemayer2024.html