Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 59-18 (2025) 379–384

Reactive Trajectory Planning for Robotic Manipulators in Shared Dynamic Workspace

Matteo Colombo * **Lorenzo Morchia** *
Andrea Maria Zanchettin * **Paolo Rocco** *

* *Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milan, Italy (e-mail: name.surname@polimi.it).*

Abstract: With the rise of collaborative robotics, the ability of a robotic manipulator to plan and adapt to dynamic and changing scenarios is taking on particular importance. Commonly used state-of-the-art motion planners are unable to satisfy such requirements, as they lack the necessary reactiveness. In this paper, we propose the adoption of nonlinear model predictive control (NMPC) to generate collision-free trajectories, which are then tracked by a low-level controller. We formulate the trajectory generation as an optimization problem in the NMPC framework, proposing a set of obstacle avoidance constraints that prioritise safety as a primary requirement. The proposed algorithm is integrated with a custom perception pipeline and tested in both simulations and on a real collaborative pick-and-place manipulation task, proving that our algorithm achieves reactive motion planning for robotic arms.

Keywords: Collaborative Robotics, Reactive Trajectory Planning, Model Predictive Control, Human-Robot Collaboration.

1. INTRODUCTION

Industrial robotic arms provide fast and precise motion that can be exploited to efficiently automate complex and repetitive tasks such as packaging, palletizing, and welding, significantly enhancing productivity. However, the setup of an industrial robotized workstation requires a significant engineering effort and the robot must be isolated from the human operators' workspace for safety reasons. The new paradigm behind collaborative robotics, on the other hand, allows humans and robots to coexist in a shared workspace, with light or no barriers between them. This new generation of robots should safely and dependably interact with human workers and provide support in flexible collaborative tasks. As such, a modern industrial robot should be able to work in cluttered and ever-changing environments. Sample-based planners (Tamizi, 2023) are unable to fulfill this flexibility requirement and adapt to sudden changes in the planning environment. This kind of planner is fast in finding feasible solutions to obstacle avoidance problems in static, cluttered scenarios but falls short in the presence of uncertainties or dynamic elements in the planning space. Variations of classical methods, such as Rapidly Exploring Random Trees (RRT, RRT*) or Probabilistic Roadmaps (PRM), have been proposed to deal with path planning in dynamic environments (Kuwata et al., 2009). However, they lack responsiveness, as they require substantial replanning effort to be effective. They are also challenging to implement in high-dimensional spaces, such as those encountered with robotic manipulators. On the other side of sampling-based planning methods, there are trajectory optimization approaches. CHOMP (Zucker et al., 2013), STOMP (Kalakrishnan et al., 2011) and TrajOpt (Schulman et al., 2013) have been proposed as fast optimization-based planning algorithms able to generate optimal smooth trajectories navigating in complex environments. Although different optimization-based approaches have been explored over the years, Model Predictive Control (MPC) (Bemporad, 2006) applied to real-time trajectory optimization for robot manipulators in dynamic scenarios is relatively underdeveloped in both research and industry. In this context, MPC iteratively solves an optimization problem over a finite time horizon T, using the current knowledge as input, and a model of the system to predict future behaviours over T. This moving horizon paradigm lends itself to a reactive trajectory replanning approach, as the dynamic obstacle avoidance requirements can be embedded in the formulation of the constraints.

In this paper, we present a motion planning framework based on Model Predictive Control, tailored towards collaborative robotic manipulators. The planner can reactively generate collision-free joint trajectories, guaranteeing obstacle avoidance in dynamic environments with unmodelled obstacles. Our planning algorithm exploits a MPC controller as a high-level controller to generate optimal, smooth and collision-free trajectories tracked by a low-level controller.

Our work is structured as follows: Section II presents an overview of the trajectory planning problem, focusing on reactive trajectory planning in a dynamic environment and details the proposed contributions. Section III discusses

⋆ This study was carried out within the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from Next-Generation EU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10-2022, PE00000004). CUP MICS D43C22003120001.

10.1016/j.ifacol.2025.10.250

the proposed formulation for our MPC, focusing on the nonlinear obstacle avoidance constraints, while Section IV presents how the proposed framework is implemented and experimentally validated.

2. RELATED WORK

Robotic motion planning techniques can be broadly classified into sampling-based and optimization-based motion planning. Sampling-based planners explore high-dimensional spaces by randomly generating valid configurations, while optimization-based planners formulate trajectory planning as a cost function minimization problem, embedding task requirements directly in the constraint formulation, which yields smoother and time-optimal trajectories at the expense of higher computational demands. Other planning approaches exist, such as Potential Field (Song and Kumar, 2002), but generally are not as successful as the aforementioned ones.

State-of-the-art sampling-based planners can deal with cluttered scenes effectively but struggle when facing tasks that require dynamic re-planning. Smoothly Rapidly exploring Random Trees (S-RRT) (Wei and Ren, 2018) increase the sampling speed of classical RRT approaches through target directional node extension, adding curvature constraints to obtain smooth paths when a new element appears in the workspace. Nonetheless, dynamic elements are not tracked, and replanning is needed at every workspace modification, which may lead to momentarily halting the robot operations. Lai and Ramos (2022) and Kuwata et al. (2009) propose to speed up replanning in sampling-based approaches by exploiting previously computed solutions. Yoshida and Kanehiro (2011) leverages RRT for replanning only if a deformation of the initial path cannot produce a feasible solution to avoid obstacles approaching the robot. Overall, this family of methods can produce feasible solutions to the dynamic replanning problem, but post-processing is usually needed to obtain a smooth trajectory from the sampled waypoints. They are also not well suited to incorporate secondary objectives, such as safety requirements for human operators. Optimization-based methods do not suffer from these problems. Zanchettin et al. (2015) developed an optimization-based algorithm that enforces real-time path safety in human-robot collaboration. The main idea is that the robot can execute its task as long as its joint velocities increase the distance between the robot and the human.

Although MPC is particularly well suited for dynamic replanning scenarios, thanks to its forward-looking approach, it is yet to be applied extensively to optimal trajectory generation and replanning for robotic arms, being mostly used to enforce robustness to a set of requirements in trajectory tracking scenarios. Furthermore, most of the proposed uses of MPC in this context are tailored to specific tasks or settings of interest.Tika and Bajcinca (2021) use a MPC algorithm to plan trajectories for two robotic manipulators in the same workspace, modelling the two robotic arms through Bezier curves to reduce computation time, but they only account for relative collision avoidance between the two robots. Mavrommati et al. (2021) explore the possibility of using a nonlinear MPC formulation to include minimum distance constraints between a manipulator and moving objects, but their formulation does not account for obstacle velocities, only slowing down the robot to increase the probability of avoiding collisions along the path. On the other hand, Krämer et al. (2020) propose a MPC scheme based on a hypergraph for online trajectory generation, also taking into account dead times and compensation of control delays. The proposed formulation, however, is only tested in simple simulation scenarios and does not consider obstacle velocities. Nevertheless, the state-of-the-art of MPC for robotic manipulators remains relatively underdeveloped and lacks a comprehensive framework, resulting in its limited use, often constrained to specific case studies, primarily tested in simulation.

With respect to the state-of-the-art:

I) We propose to integrate a simplified model and a dynamic obstacle avoidance constraint, formulated using worst-case obstacle velocity and derived from our previous work (Zanchettin et al., 2015), into a NMPC framework. This approach enables reactive, smooth, and collision-free trajectory generation for robotic manipulators operating in dynamic environments shared with humans.

II) We directly encode the target Cartesian state in the cost function, without requiring a pre-planned trajectory. Only the final target pose is provided to the algorithm, which outputs a reference to be tracked in real-time.

III) The proposed framework is tested in a simulated environment and then validated in real-world experiments, deployed on a robotic manipulator.

3. MODEL PREDICTIVE CONTROL FRAMEWORK

This Section describes the key elements of our MPC formulation. We propose a Nonlinear MPC (NMPC) that exploits nonlinear collision-avoidance constraints to enforce safe trajectory generation in dynamic environments. NMPC acts as a high-level motion planner, as shown in Figure 1, by minimising a cost function formulated directly in task space, requiring only a final target pose P_F as initialization input. No reference trajectory is needed.

3.1 Model

In our framework, the controlled system is a robotic manipulator. Generally, the open-loop dynamics of a robot manipulator can be represented by a set of nonlinear

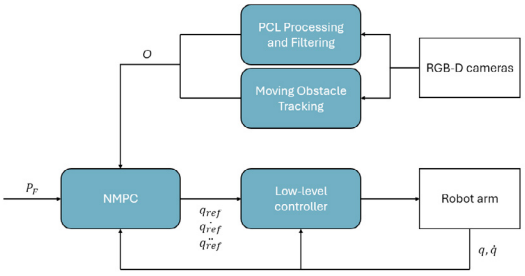

Fig. 1. Our NMPC formulation uses a simplified manipulator model to generate an optimal trajectory, which is tracked by a low-level controller. The NMPC requires feedback data from the state $[q, \dot{q}]$ of the manipulator and a set of perception data O. In addition, target Cartesian pose P_F is the input, as the cost function is formulated directly into task space.

equations of motion which describe the relationship between the forces and torques applied to the manipulator's joints. Using this nonlinear model in a MPC scenario may prove problematic and lead to high computational efforts, making the real-time applicability of the resulting controller unfeasible. Model accuracy and problem complexity need to be balanced in order to make the problem solution faster, while retaining a model that is representative enough to reflect the main dynamics relations. A double integrator is used to model the motion of each robot joint (Avanzini et al., 2018), which allows for smooth trajectories generation while maintaining system linearity (Mavrommati et al., 2021). This simplification is made possible by using the MPC controller as a high-level controller, with a lower-level regulator tracking the trajectory output. Considering the model state as $x = \left[q^\top, \dot{q}^\top\right]^\top \in \mathbb{R}^{2n}$, where q and $\dot{q}$ are joint positions and velocities respectively, the state-space formulation of the prediction model is

$$\dot{x} = \begin{bmatrix} 0^{n\times n} & I^{n\times n} \\ 0^{n\times n} & 0^{n\times n} \end{bmatrix} x + \begin{bmatrix} 0^{n\times n} \\ I^{n\times n} \end{bmatrix} u \tag{1}$$

where the system inputs $u \in \mathbb{R}^n$ are the joints acceleration $\ddot{q} \in \mathbb{R}^n$. I is an identity matrix.

3.2 Cost function

Our cost function is formulated as a nonlinear cost function in the task space of the manipulator, directly minimizing the error from target end effector (EE) pose

$$P(q(t)) = [x_{EE}, y_{EE}, z_{EE}, \phi_{EE}, \theta_{EE}, \psi_{EE}]^\top \in \mathbb{R}^6 \tag{2}$$

x_{EE}, y_{EE}, z_{EE} are the three EE Cartesian position coordinates while $\phi_{EE}, \theta_{EE}, \psi_{EE}$ represent the XYZ Euler angles defining the EE orientation. From this definition of $P(q(t))$, the overall cost function is expressed as

$$J = \Psi(x(t_F)) + \int_{t_0}^{t_F} L(x(t), u(t))\, dt \tag{3}$$

$\Psi(x(t_F))$ and $L(x(t), u(t))$ are the terminal cost and the running cost respectively:

$$\begin{aligned} \psi = &\left[P_F - P(q(t_F))\right]^\top Q_t \left[P_F - P(q(t_F))\right] \\ &+ \dot{q}^\top(t_F) Q_v \dot{q}(t_F) \end{aligned} \tag{4}$$

$$\begin{aligned} L = &\left[P_F - P(q(t))\right]^\top Q_r \left[P_F - P(q(t))\right] \\ &+ u^\top(t) Q_u u(t) \end{aligned} \tag{5}$$

where $t_F = t_0 + T$ is the final time instant, T being the prediction horizon. $Q_r, Q_t \in \mathbb{R}^{6\times 6}$ are symmetric, positive semi-definite matrices weighting the EE pose error, $Q_u \in \mathbb{R}^{n\times n}$ is the weight matrix penalizing the control effort and $Q_v \in \mathbb{R}^{n\times n}$ weights the final joint velocity. This cost function formulation in the task space is obtained by applying the nonlinear forward kinematics $k_f(\cdot) : \mathbb{R}^n \to \mathbb{R}^6$ where $P(t)$ is expressed as a function of the robot state. Through an appropriate selection of the weighting matrices, EE positions and orientations can be penalized differently according to the task that the manipulator should accomplish. Remarkably, this formulation also allows to change the desired target pose P_F in real time.

3.3 Constraints

Inequality constraints are used to enforce joint limits on the positions and velocities of the manipulator and to prevent self-collisions of the robot links

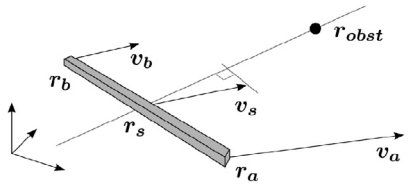

Fig. 2. A generic link l of a robotic arm modelled, as a rigid beam.

$$q_{min} \le q(t) \le q_{max} \tag{6}$$

$$\dot{q}_{min} \le \dot{q}(t) \le \dot{q}_{max} \tag{7}$$

$$\ddot{q}_{min} \le u(t) \le \ddot{q}_{max} \tag{8}$$

The purpose of limiting the state and the accelerations is twofold: enforcing that the generated trajectory is feasible for the robot and generating curvature-constrained trajectories. A second set of constraints, derived from our previous work (Ragaglia et al., 2018), enforces dynamic obstacle avoidance requirements by imposing a minimum separation distance between a dynamic body and the robotic manipulator. Given the challenges behind monitoring the velocities of moving elements in real-time, a worst-case approach is adopted by considering the maximum allowable speed of a dynamic obstacle. At any time instant t, the speed of the manipulator that guarantees dynamic collision avoidance is represented by

$$v_r(t) \cdot T_s \le \max\left(0, d(t) - \Delta\right) \tag{9}$$

$v_r(t)$ represents the signed robot speed for a generic point along the link in the direction of the obstacle, and T_s is the worst-case scenario stopping time. d is the distance between the obstacle and the robot link l closest to it. $\Delta > 0$ is a tunable clearance parameter that considers obstacle velocity, robot dimension and other sources of uncertainties to provide a further layer of safety when replanning the trajectory accounting for new dynamic obstacles. In our formulation, Δ is computed as

$$\begin{aligned} \Delta = v_o^w \cdot T_s + S_{min} \quad &\text{for moving obstacles} \\ \Delta = S_{min} \quad &\text{for static obstacles} \end{aligned}$$

where v_o^w is the worst-case obstacle velocity and S_{min} the actual clearing distance between robot and obstacle. Equation (9) can be elaborated to obtain the set of inequality constraints (Zanchettin et al., 2015):

$$T_s E_{j,l} \dot{q} \le f_{j,l} \tag{10}$$

with

$$E_{j,l} = \begin{bmatrix} (r_{ob,j} - r_{a,l})^\top J_{a,l} \\ \left((r_{ob,j} - r_{a,l})^\top J_{b,l} - (r_{b,l} - r_{a,l})^\top J_{a,l}\right) \end{bmatrix}$$

$$f_{j,l} = \left(\max(0, d_{j,l} - \Delta_j)\right)^2 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$J_{a,l}$ and $J_{b,l}$ are the Jacobians of points $r_{a,l}$ and $r_{b,l}$ of the manipulator link l (shown in Figure 2), respectively, while $d_{j,l} = \min_s\left(\|r_{ob,j} - r_{s,l}\|\right)$ represents the distance between l and a generic obstacle in the workspace. While the inequality constraint represented by (10) is linear with respect to $\dot{q}$, is a nonlinear constraint with respect to the joint states q. At every timestep t_s, constraint (10) must be verified between each link l_i in the robot and the j-th obstacle in the environment, resulting in $2 \cdot n_L \cdot n_O \cdot n_t$ constraints to be satisfied. n_L, n_O and n_t are the

Algorithm 1 *NMPC Algorithm*

Generate Static obstacles set $\mathcal{O}_s$
Initialize: t_0, t_s, T, $x(t_0)$, Q_r, Q_t, Q_v, Q_u, $\mathcal{O}_d$

1: **Input**: current target pose $P_F(t_0)$
2: **while** P_F not reached **do**
3: **if** MovingObstacles() **then**
4: Update obstacle set $\mathcal{O}_d$;
5: updateMovingObstaclePose();
6: Update (11) from $\mathcal{O} \leftarrow \mathcal{O}_s \cup \mathcal{O}_d$;
7: **end if**
8: **if** New P_F **then**
9: $P_F \leftarrow P_F(t_0)$;
10: **end if**
11: $x(t_0) = x_0 \leftarrow$ updateCurrentState();
12: $u_{opt}(t)|_{t_0}^{t_0+T}, x_{opt}(t)|_{t_0}^{t_0+T} = \min\left(J(x(t), u(t), P_F)\right)$,
13: subject to (1), (6), (7), (8) and (11), $\forall t \in [t_0, t_0{+}T]$;
14: **if** ConstraintViolated() **then**
15: STOP(), **break**;
16: **end if**
17: $x_{ref} \leftarrow x_{opt}(t_0 + t_s)$;
18: $\ddot{q}_{ref} \leftarrow u_{opt}(t_0 + t_s)$;
19: Track x_{ref}, $\ddot{q}_{ref}$ using low-level controller;
20: $t_0 \leftarrow t_0 + t_s$;
21: **end while**

number of links, obstacles and timesteps. This yields a potential high number of constraints that scale with the number of obstacles in the workspace. This, however, can be mitigated through environmental clustering techniques to be applied in the perception module of the overall control scheme, shown in Figure 1.
The set of nonlinear inequality constraints can hence be formulated as

$$G(q(t)) \cdot \dot{q}(t) - b(t) \leq 0 \tag{11}$$

by collecting the terms in (10), where $G(q(t)) \in \mathbb{R}^{2n_L n_O \times n}$ and $b(t) \in \mathbb{R}^{2n_L n_O}$.

3.4 Control algorithm

The proposed NMPC control scheme from Figure 1 is described in Algorithm 1. The control problem to be solved is discretized into N timesteps of t_s seconds each. At every t_s, the optimization problem is solved with updated constraints and input states. A strict requirement for the algorithm to be applicable is that t_c should be much smaller than t_s, to guarantee that the low-level controller can properly track the computed trajectory, as discussed in depth in Section 4. The obstacle set $\mathcal{O} = \mathcal{O}_s \cup \mathcal{O}_d$ is divided into subsets of static and dynamic obstacles $\mathcal{O}_s$ and $\mathcal{O}_d$. After initialization, only $\mathcal{O}_d$ is updated at every iteration. $\mathcal{O}$ is used in the optimization problem to enforce the inequality constraints (9). The robot stops in case of a constraint violation, indicating that the next collision-free waypoint is unavailable (line 15). This strategy is used as a further safety layer, to prevent any harm to the operator working in proximity of the robot.

4. EXPERIMENTAL VALIDATION

The proposed control algorithm is first validated in simulation and then tested on an ABB GoFa 5 collaborative robot in different scenarios. We use ROS as middleware, while the NMPC algorithm is converted into C++ using the MATLAB code generator with the SQP solver. The full pipeline runs on a personal computer with a 1.30GHz Intel i7 processor and 16 Gb of RAM.

4.1 Perception module

The complexity of our obstacle avoidance constraints scales proportionally to n_O, since $2{\cdot}n_L{\cdot}n_O{\cdot}n_t$ is the number of inequality constraints. An appropriate representation of the environment should be adopted to guarantee the system's responsiveness. In our real-world implementation, we used a pair of Kinect2 cameras to generate a PointCloud (PCL) representing the working environment. Using the raw PCL for distance and collision checking computations is impractical, as it results in a set of inequalities having the same dimension as the number of PCL points. An interesting approach explored for dynamic obstacle avoidance in other research fields is to use three-dimensional occupancy maps (Hagmanns et al., 2022; Macenski et al., 2020) to segment the environment. In particular, Han et al. (2018) proposes a method to adapt Octomaps to dynamic obstacle avoidance of a robotic manipulator, but they cannot achieve high-frequency tracking of dynamic objects. We chose to perform PCL filtering and sampling to obtain a minimal mesh representation of the environment. This allows estimating the distance between the robot and objects more efficiently, according to Schulman et al. (2013). We first sample the static environment without moving obstacles to obtain a mesh representation of it. Novel and unseen dynamic obstacles can be represented as clusters of points used to fit a convex mesh shell. We use a capsule approximation to fit both static and moving obstacles, as it can be efficiently parametrized when performing distance computations in Cartesian space, as shown in Figure 3, resulting in low computation times.

4.2 Simulations

Our NMPC controller is first tested in a simulated Gazebo environment. Tracking dynamic bodies and estimating their speed is a complex task that is still the subject of extensive research outside the scope of our work. As such, we use Aruco markers to represent dynamic obstacles in real and simulated experiments (Figure 4). We ran over

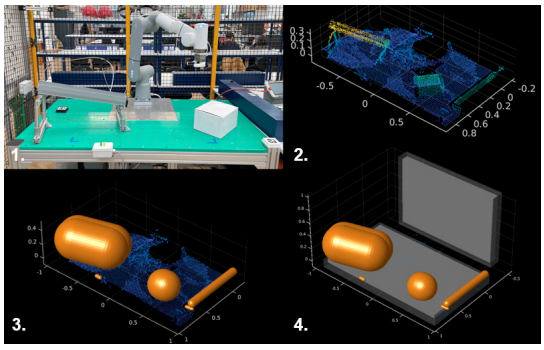

Fig. 3. Pointcloud (2) generated from RGB-D images of the real environment (1). Static obstacles $\mathcal{O}_s$ are sampled from the pointcloud (3) and used in the minimal environmental representation (4).

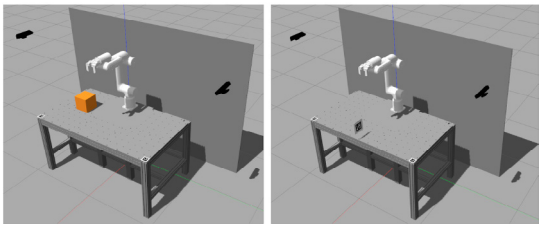

Fig. 4. Base Gazebo environment used to test the proposed pipeline in static and dynamic scenarios.

fifty simulation scenarios with different parameter combinations to characterize the computational effort required to solve the NMPC problem in a simulation environment. The results are shown in Table 1, where IQR is the Interquartile Range around the median of t_c, the time necessary to solve the optimization problem at every t_s. The total optimization horizon $T = N \cdot t_s$ is selected equal to $1s$, with $N = 2$ optimization steps and $t_s = 0.5s$, chosen as a trade-off between NMPC performance and computational time t_c. A mandatory constraint for our framework to be applicable is that $t_c << t_s$. A lower t_c also means reduced control delay when applying our framework to the real manipulator. From these requirements, the proposed combination of parameters for our implementation results in $t_c \in [0.1s, 0.28s]$. We also select $S_{min} = 40$ mm, $\Delta_{static} = 40$ mm and $\Delta_{dynamic} = 80$ mm as NMPC parameters for the real world experiments.

Table 1. Statistical characterization of t_c.

N	t_s [s]	Median t_c [s]	IQR t_c [s]	Percentage $t_c > t_s$
3	0.25	0.38	[0.32, 0.42]	**89%**
3	0.50	0.45	[0.40, 0.50]	**5.8%**
3	0.75	0.45	[0.41, 0.54]	**0%**
2	0.25	0.18	[0.13, 0.22]	**1.5%**
2	0.50	0.22	[0.15, 0.24]	**0%**
2	0.75	0.22	[0.20, 0.24]	**0%**

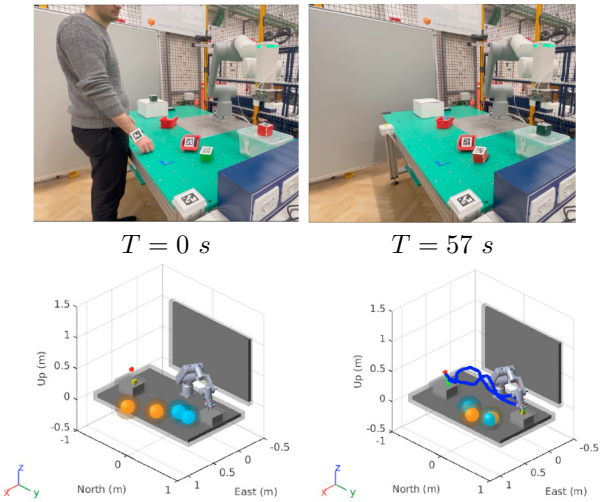

Fig. 5. In the trajectory visualizer at the bottom, the operator's hands appear as orange safety areas, while dynamic obstacles are blue. The translucent region represents the safety margin. Our NMPC computes a collision-free trajectory for the manipulator to move the block between boxes.

4.3 Real world experiments

Our NMPC algorithm is deployed on an ABB GoFa collaborative robot and tested in a collaborative environment, shown in Figure 5. The grasping subroutines are implemented using MoveIt, while the NMPC handles the broader movements in the workspace. A capsule approximation of the robot collision mesh is employed to further improve computational time, following (Han et al., 2018). The characterization of the algorithm performance showed that up to 55% of the total computation time was employed in the distance computation between robot links and obstacles. This simplification reduced such amount of time by 60%. To synchronize with perception data and compensate for the delay introduced by t_c, not negligible in real-life applications, we adopt the strategy first proposed by Grüne et al. (2017), computing the solution to the control problem in advance, with a pre-defined time offset greater than t_c, such that a control action is always available when it has to be applied.

The task devised to test our algorithm is a collaborative pick and place operation, shown in Figure 5, where our NMPC framework is used to plan a trajectory for the robot to navigate the environment, pick up a designated object and place it on the other side of the table. Concurrently, the operator removes the obstruction from the placement position. Figure 6 shows the resulting joint positions and velocities for the first three joints that are applied to reach the approach pose for the goal grasp (the top blue segment highlighted in Fig.5). For the sake of clarity, only the data for three out of the six robot joints are presented, as they are those responsible for the broader movements of the robot arm. The MPC problem, on average, is solved in 110 ms and is able to generate smooth, collision-free trajectories, allowing the manipulator to avoid the dynamic elements in the workspace. The minimum separation distance between the robot and the environment during trajectory execution is shown in Figure 7. A zero value corresponds to the robot exceeding the safety limit distance from the closest obstacle. This task is repeated five times, always obtaining consistent results.

Two other experiments are evaluated [1]. In the first, the operator intentionally enforces a violation of the safety margin Δ. The manipulator correctly stops, avoiding collision. In the third task, the operator obstructs the target pose P_F, causing the manipulator to slow down to avoid collision. Once P_F becomes unobstructed, the robot safely resumes its motion and reaches the goal via the replanned trajectory.

5. CONCLUSIONS

This work presented a novel framework that leverages a Non-linear Model Predictive Control (NMPC) algorithm to enable robotic manipulators to re-plan their motion in response to dynamic evolutions in their workspace. The proposed NMPC has been employed as a high-level motion planner to generate collision-free joint-space trajectories which were then tracked in real time by the robot low-level controllers. Safety was enforced as a hard constraint by employing a suitable collision avoidance that can enforce collaborative safety requirements. Future works will

[1] Additional video material at `https://youtu.be/GvkWKI71Ox0`

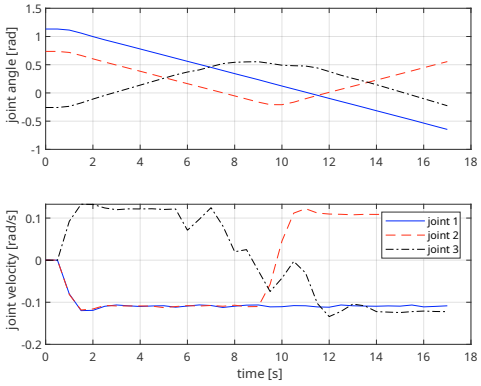

Fig. 6. From top to bottom: joint states and velocities of the first three joints of the robot resulting from the NMPC controller's input.

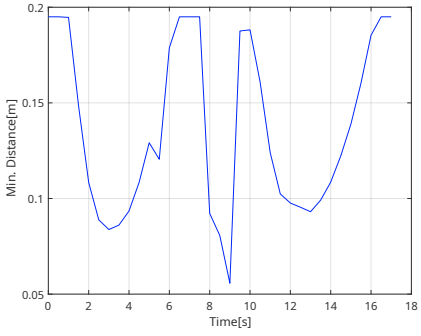

Fig. 7. Minimum distance separating the robot from the closest obstacle in $\mathcal{O}$ during trajectory execution in a collaborative pick and place experiment.

focus on exploiting nonlinearity in order to reshape the proposed collision avoidance constraints and consider real-time velocity estimation of dynamic obstacles. We would also like to explore how to directly integrate the robot's dynamics in the model and cost function to remove the need for a low-level tracking controller.

REFERENCES

Avanzini, G.B., Zanchettin, A.M., and Rocco, P. (2018). Constrained model predictive control for mobile robotic manipulators. *Robotica*, 36(1), 19–38.

Bemporad, A. (2006). Model predictive control design: New trends and tools. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 6678–6683. IEEE.

Grüne, L., Pannek, J., Grüne, L., and Pannek, J. (2017). *Nonlinear model predictive control*. Springer.

Hagmanns, R., Emter, T., Grosse-Besselmann, M., and Beyerer, J. (2022). Efficient global occupancy mapping for mobile robots using openvdb. *arXiv preprint arXiv:2211.04067*.

Han, D., Nie, H., Chen, J., and Chen, M. (2018). Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection. *Robotics and computer-integrated manufacturing*, 49, 98–104.

Kalakrishnan, M., Chitta, S., Theodorou, E., Pastor, P., and Schaal, S. (2011). Stomp: Stochastic trajectory optimization for motion planning. In *2011 IEEE international conference on robotics and automation*, 4569–4574. IEEE.

Krämer, M., Rösmann, C., Hoffmann, F., and Bertram, T. (2020). Model predictive control of a collaborative manipulator considering dynamic obstacles. *Optimal Control Applications and Methods*, 41(4), 1211–1232.

Kuwata, Y., Teo, J., Fiore, G., Karaman, S., Frazzoli, E., and How, J.P. (2009). Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on control systems technology*, 17(5), 1105–1118.

Lai, T. and Ramos, F. (2022). Ltr*: Rapid replanning in executing consecutive tasks with lazy experience graph. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8784–8790. IEEE.

Macenski, S., Tsai, D., and Feinberg, M. (2020). Spatio-temporal voxel layer: A view on robot perception for the dynamic world. *International Journal of Advanced Robotic Systems*, 17(2), 1729881420910530.

Mavrommati, A., Osorio, C., Valenti, R.G., Rajhans, A., and Mosterman, P.J. (2021). An application of model predictive control to reactive motion planning of robot manipulators. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, 915–920. IEEE.

Ragaglia, M., Zanchettin, A.M., and Rocco, P. (2018). Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements. *Mechatronics*, 55, 267–281.

Schulman, J., Ho, J., Lee, A.X., Awwal, I., Bradlow, H., and Abbeel, P. (2013). Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, 1–10. Berlin, Germany.

Song, P. and Kumar, V. (2002). A potential field based approach to multi-robot manipulation. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, 1217–1222. IEEE.

Tamizi, Yaghoubi, N. (2023). A review of recent trend in motion planning of industrial robots. *Int J Intell Robot Appl*, 7, 253–274.

Tika, A. and Bajcinca, N. (2021). Model predictive control based cooperative robot manipulation and collision avoidance in shared workspaces. In *2021 European Control Conference (ECC)*, 702–709. IEEE.

Wei, K. and Ren, B. (2018). A method on dynamic path planning for robotic manipulator autonomous obstacle avoidance based on an improved rrt algorithm. *Sensors*, 18(2), 571.

Yoshida, E. and Kanehiro, F. (2011). Reactive robot motion using path replanning and deformation. In *2011 IEEE International Conference on Robotics and Automation*, 5456–5462. IEEE.

Zanchettin, A.M., Ceriani, N.M., Rocco, P., Ding, H., and Matthias, B. (2015). Safety in human-robot collaborative manufacturing environments: Metrics and control. *IEEE Transactions on Automation Science and Engineering*, 13(2), 882–893.

Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A., and Srinivasa, S.S. (2013). Chomp: Covariant hamiltonian optimization for motion planning. *The International journal of robotics research*, 32(9-10), 1164–1193.