**IEEE Robotics and Automation Letters (RAL) paper presented at the
2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)**

A Disturbance-Aware Trajectory Planning Scheme based on Model Predictive Control*

Luca Paparusso[1], Navvab Kashiri[2], and Nikos G. Tsagarakis[2]

Abstract— **Despite the development of numerous trajectory planners based on computationally fast algorithms targeting accurate motion of robots, the nowadays robotic applications requiring compliance for interaction with environment demand more comprehensive schemes to cope with unforeseen situations. This paper discusses the problem of online Cartesian trajectory planning, targeting a final state in a desired time interval, in such a way that the generated trajectories comply with the tracking abnormalities due to considerable motion disturbances. We propose a planning scheme based on Model Predictive Control. It utilises a novel strategy to monitor the tracking performance via state feedback and consequently update the trajectory. Also, it ensures the continuity of the generated reference while accounting for realistic implementation constraints, particularly due to computational capacity limits. To validate the efficacy of the proposed scheme, we examine a practical robotic manipulation scenario in which a given task is executed via a Cartesian impedance controller, while an external interaction interrupts the motion. The performance of the proposed strategy as compared to that of a state-of-the-art study is demonstrated in simulation. Finally, a set of experiments verified the effectiveness of the proposed scheme in practice.**

I. Introduction

From industrial manipulators performing pick-and-place actions to humanoids interacting with environment, it is fundamental to feed the robot controllers with affordable and smooth reference trajectories, respecting the limitations imposed by the operative context. The problem of finding the best possible trajectory can be formulated as an optimisation problem, aiming at minimising a cost function subject to constraints [1]. Trajectory generation algorithms can be divided into two main categories: the first aims to enforce that the end-effector exactly follows a desired geometric path (see e.g. [2]–[4]); the second, point-to-point (PTP) planning, is geared to reach a final state regardless of the path, which is discussed in this paper.

Point-to-point trajectory planning is often based on optimally reaching the target state while respecting a set of constraints [5]. Although such methods do not include explicit path constraints resulting in a parameterisation of the equations of motion, it is still possible to define a map of via-points to be respected [6]. For applications characterised by repetitive actions in a pre-defined fixed environment, e.g. industrial non-collaborative manipulators, the trajectory planning problem can be solved offline, so that the stabilisation around the obtained reference trajectory is left to the controller [7]. On the contrary, it is a common requirement in collaborative and/or legged robots to generate reference trajectories online using the robot state, to adapt to unforeseen and/or changing scenarios. In [8], a sensor fusion algorithm is used to merge data from sensors to generate and modify reference trajectories in real time in a Human-Robot Collaboration (HRC) scenario, maximising the productivity while respecting safety constraints.

To solve the optimal trajectory planning problem online at a suitable rate, the optimisation algorithm needs to respect computational limits. Motion planners based on the dynamic model of the system allow to simultaneously find the optimum path and trajectory, and deliver higher performance. However, the computational capacity of robotic platforms, especially mobile robots, is often limited, and requires the optimisation problem to be divided into two independent sub-optimisations: one for path generation and another for association of time instants to the points along the path (speed profile).

In the last twenty years, Model Predictive Control (MPC) [9] has been representing one of the most suitable techniques for multivariable planning and control, due to its capability of providing online optimal solutions while respecting constraints. Although this approach was initially utilised only in the chemical and petrochemical process industry characterised by a slow dynamics, due to the lower computational power of early processors, nowadays MPC is exploited in a wide range of applications, from autonomous vehicles to robotics and motors control [10]–[14]. Despite notable technological improvement in the computer industry, the computational burden still embodies the main drawback of this approach, thus requiring the above-said problem division (into path optimisation and speed profile optimisation) when the system model is fairly complex e.g. robotic manipulators.

This work introduces a PTP trajectory planning scheme acting in the Cartesian space, using the robot state feedback to adapt online to motion disturbances resulting in large tracking errors of the controller. While it is essential in a majority of robotic applications to carry out a given PTP motion in a given time, a large number of MPC-based solutions adopt a fixed-length receding horizon, thereby preventing from setting a target time. The approach presented in [15] addresses this shortfall by using linear time-invariant MPC with fixed terminal time to obtain a shrinking horizon covering the whole motion time and sequentially refining

*This work was supported by the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 871237 (SOPHIA).

[1]L. Paparusso is with the Department of Mechanical Engineering, Politecnico di Milano, 20156 Milano, Italy `luca.paparusso@polimi.it`

[2]N. Kashiri and N. G. Tsagarakis are with the Humanoids and Human Centered Mechatronics lab, Istituto Italiano di Tecnologia, 16163 Genova, Italy `navvab.kashiri@iit.it`

the planned trajectory. Nevertheless, it relies on feedforward simulation for the system state, rather than the actual robot state. It therefore does not allow the planner to adapt to unforeseen interactions and interruptions that may lead to dangerous behaviours.

To tackle this problem, we propound a planning scheme that supervises the task execution and guarantees compliance with motion and safety constraints. In particular, the output reference trajectory - which may not be precisely tracked due to a desired/required compliant behaviour rendered by the controller - and the eventual extension of the trajectory duration are determined and updated online, according to a feedback-based decision-making process. Moreover, in contrast to the majority of the state-of-the-art studies e.g. [15] requiring the implementation of the entire algorithm to generate a real-time responsive performance, the proposed scheme splits into real-time and non-real-time parts. This is to account for realistic implementation constraints, particularly due to computational capacity limits. The computational time of the non-real-time part is then estimated and compensated to improve performance. To foster a homogeneous transition between the generated trajectories and to allow for running MPC at a lower rate than the controller, we propose a novel target selector that runs in real time and selects a suitable reference at every time instant, while monitoring the controller performance.

We validate the effectiveness of the proposed scheme by applying it to a practical robotic manipulation scenario, in which a PTP task is given to the planner and the corresponding trajectories are executed by a Cartesian impedance controller, while an external interaction prevents the end-effector from following the reference trajectory. The performance of the proposed strategy is compared to that of a state-of-the-art study in simulation, to underline its ability to account for unforeseen interactions/interruptions, as the main contribution of this work. Finally, a set of experiments on a real robot were carried out not only to demonstrate the effectiveness of the scheme in handling unforeseen interactions, but also to validate its online applicability and suitability for relaxation of the real-time layer, by leaving out of this layer the computationally burdensome part of the algorithm.

The rest of this paper is organised as follows. An overview of the proposed planning logic is presented in Sec. II, introducing the key features of each element of the scheme. The MPC structure developed in [15], used as a starting point to our work, is presented in Sec. III. In Sec. IV, a detailed mathematical formulation of the proposed planning scheme is provided, analysing the different scenarios influencing the algorithm. A set of simulations and experiments on a humanoid robot arm is reported in Sec. V to validate the approach, followed by comments on the obtained results. Finally, the conclusions of this work and potential future studies are presented in Sec. VI.

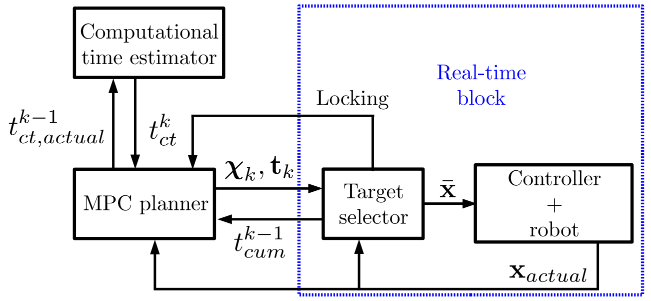

Fig. 1. Planning scheme.

II. Proposed Planning Logic Description

In contrast to approaches using the problem division (into path and speed profile optimisation) to reduce the computational burden at the expense of obtaining sub-optimal solutions, we choose the MPC formulation introduced in [15] as the base of the proposed planning scheme, to render optimal solutions to the PTP fixed-time trajectory planning problem. It should be noted that, the base planner can be substituted with any other MPC planner provided that it permits to enforce the generated trajectories, i.e. solutions to the predictive horizons, to reach the target state at a given target time. This versatility is guaranteed by the proposed planning architecture, whose design focuses on the online adaptation of the generated trajectories to unforeseen interruptions and interactions with environment.

The proposed trajectory planning scheme is depicted in Fig. 1. It relies upon two main entities which are separated and run at different rates, namely an MPC planner and an innovative target selector with performance monitoring capabilities. The scope of the MPC planner is to generate optimal trajectories allowing the robot end-effector to reach a target state in a target time. The target selector runs in real time and receives the generated trajectories as inputs. It monitors the tracking performance, and chooses the reference to be fed to the controller at each time instant. In addition, a computational time estimator is employed to predict the time delay introduced at each MPC execution so that the planner can account for it, as described in Sec. IV.

In receding horizon MPC [9], only the first control action of each predictive horizon is applied, and the problem is shifted forward for a new execution, i.e. optimisation. In this work, we define the problem in a way that the predicted states of a predictive horizon define a trajectory to move the end-effector from the current state to the desired target state in a target time. To enable a closed-loop behaviour, we utilise the current state of the end-effector for an online adaptation of the trajectory and motion time. The MPC planner solves the optimisation problem at a rate f_{MPC} and, as soon as a new predictive horizon is available, it is taken as planned reference trajectory by the target selector, while the old one is discarded. We denote with f_{ts} the rate at which the target selector computes the current reference from the planned reference trajectory and feeds it to the controller.

The closed-loop performance and corresponding predictive horizon updates can be subject to challenges/disadvantages that should be carefully addressed to assure the effectiveness of the approach. First, it may not always be possible to find a solution to the MPC optimisation problem, due to discretisation and approximations. Moreover, the switch between the old and new planned trajectories needs to be homogenised for the controller to properly track the reference. As deepened in Sec. IV, the target selector plays a fundamental role in generating continuous references adapting to the robot states, thereby acting as a performance estimator and a filter between the independent trajectories generated by the MPC planner and the controller.

III. MPC Planner - Preliminaries

In this section, the guidelines shown in [15] are recalled to derive an MPC planning problem, with fixed terminal time and final state constraints. To provide a computationally fast solution for the optimal trajectory generation problem, that allows to simultaneously optimise the path and trajectory without splitting the problem into multiple sub-optimisations, the formulation relies on a decoupled linear kinematic model for describing the end-effector motion in the Cartesian space. This model is extracted assuming that the system dynamics can be cumulatively respected by properly limiting the end-effector velocity, acceleration and jerk. A set of modifications in the notation and formulation are introduced to make it suited for our problem. The states described in the present work are the end-effector Cartesian coordinates X, Y and Z with respect to a chosen orthonormal reference frame. Defining the state and input vectors as

$$\mathbf{x} = \begin{bmatrix} X & \dot{X} & \ddot{X} & Y & \dot{Y} & \ddot{Y} & Z & \dot{Z} & \ddot{Z} \end{bmatrix}^T,$$
$$\mathbf{u} = \begin{bmatrix} \dddot{X} & \dddot{Y} & \dddot{Z} \end{bmatrix}^T,$$

respectively, it is possible to derive the discrete state-space equations of the model, using the predictive first-order hold method [16] with sampling period h, as

$$\boldsymbol{\xi}(i+1) = \Phi\boldsymbol{\xi}(i) + (\Gamma + \frac{1}{h}(\Phi - I)\Gamma_1)\mathbf{u}(i), \tag{1}$$

$$\mathbf{y}(i) = C\boldsymbol{\xi}(i) + \frac{1}{h}C\Gamma_1\mathbf{u}(i), \tag{2}$$

with

$$\tilde{\Phi} = \begin{bmatrix} 1 & h & \frac{h^2}{2} \\ 0 & 1 & h \\ 0 & 0 & 1 \end{bmatrix}, \quad \frac{1}{h}\tilde{\Gamma}_1 \begin{bmatrix} \frac{h^3}{24} \\ \frac{h^2}{6} \\ \frac{h}{2} \end{bmatrix}, \quad \tilde{\Gamma} \begin{bmatrix} \frac{h^3}{6} \\ \frac{h^2}{2} \\ h \end{bmatrix},$$

$C = \text{blkdiag}([I_3, I_3, I_3])$, $\Phi = \text{blkdiag}([\tilde{\Phi}, \tilde{\Phi}, \tilde{\Phi}])$, $\Gamma = \text{blkdiag}([\tilde{\Gamma}, \tilde{\Gamma}, \tilde{\Gamma}])$ and $\Gamma_1 = \text{blkdiag}([\tilde{\Gamma}_1, \tilde{\Gamma}_1, \tilde{\Gamma}_1])$. The new state vector $\boldsymbol{\xi}$ is then defined as

$$\boldsymbol{\xi}(i) := \mathbf{x}(i) + \frac{1}{h}\Gamma_1\mathbf{u}(i), \tag{3}$$

and the vector of controlled variables $\mathbf{z}(i)$ is chosen as

$$\mathbf{z}(i) := \mathbf{x}(i). \tag{4}$$

Let us define the number of equally-spaced discretisation intervals of each predictive horizon with $N \in \mathbb{N}$, $N > 1$. To clarify the notation of the model equations, adapting them to a scenario in which multiple MPC optimisations are performed, from now on $\phi_{k,i}$ represents the generic variable $\phi(i)$ in discrete time for the predictive horizon k, when $k \in \mathbb{N}$, $k \geq 0$ and $i = 0, 1, ..., N$. The sampling time h of the predictive horizon k is also shown by h_k.

The objective of the MPC planner, under the assumption of proper tracking from the controller, is to generate planned trajectories allowing the end-effector to move from an initial state $\mathbf{r}_{in}$ at time t_{in} to a final state $\mathbf{r}_f$ at time t_f. To perform this task, the k-th MPC optimisation problem is set to respect the final time and terminal constraint conditions described by

$$t_{k,N} = t_f, \tag{5}$$

$$\mathbf{z}_{k,N} = \mathbf{r}_f. \tag{6}$$

It is necessary to notice that (5) will be modified in Sec. IV-B, to manage further features and different scenarios.

It is also of interest to limit the values of Cartesian position, velocity, acceleration and jerk along each predictive horizon, writing linear constraints as follows

$$F_k\boldsymbol{\mathcal{U}}_k \leq \mathbf{f}_k, \tag{7}$$

$$G_k\boldsymbol{\mathcal{Z}}_k \leq \mathbf{g}_k, \tag{8}$$

where F_k, G_k, $\mathbf{f}_k$ and $\mathbf{g}_k$ are appropriately designed matrices and vectors, with $\boldsymbol{\mathcal{Z}}_k^T = \begin{bmatrix}\mathbf{z}_{k,1}^T, \mathbf{z}_{k,2}^T, ..., \mathbf{z}_{k,N}^T\end{bmatrix}$ and $\boldsymbol{\mathcal{U}}_k^T = \begin{bmatrix}\mathbf{u}_{k,0}^T, \mathbf{u}_{k,1}^T, ..., \mathbf{u}_{k,N}^T\end{bmatrix}$. The constraint matrices and vectors are built to satisfy the expressions

$$\begin{aligned} &|\mathbf{u}_{k,i}| \leq [u_{max}^X,\, u_{max}^Y,\, u_{max}^Z]^T, \\ &|\mathbf{z}_{k,i}| \leq [(\mathbf{w}_{max}^X)^T,\, (\mathbf{w}_{max}^Y)^T,\, (\mathbf{w}_{max}^Z)^T]^T \\ &\quad 0 \leq i \leq N-1, \end{aligned}$$

defining $\mathbf{w}_{max}^{(\cdot)} := [s_{max}^{(\cdot)},\, v_{max}^{(\cdot)},\, a_{max}^{(\cdot)}]^T$. The values of the limits $\mathbf{w}_{max}^{(\cdot)}$ and $u_{max}^{(\cdot)}$ will be provided in Sec. V.

The cost function to be minimized for the predictive horizon k is written as

$$V_k(\boldsymbol{\mathcal{U}}_k) = \sum_{i=1}^{N-1} ||\mathbf{z}_{k,i}||_{Q_{k,i}}^2 + \sum_{i=0}^{N} ||\mathbf{u}_{k,i}||_{R_{k,i}}^2, \tag{9}$$

where $||\cdot||$ is the L2-norm, and $Q_{k,i}$ and $R_{k,i}$ are the weighting matrices. The first summation goes from $i = 1$ to $i = N-1$ as the initial and final values of the controlled variables vector, i.e. state variables (4), do not depend on the optimisation.

Finally, the optimisation of the predictive horizon k can be expressed as

$$\begin{aligned} \underset{\boldsymbol{\mathcal{U}}_k}{\text{minimize}} \quad & V_k(\boldsymbol{\mathcal{U}}_k), \\ \text{subject to} \quad & (1)\text{-}(4), (6)\text{-}(8), \end{aligned} \tag{10}$$

given the initial condition $\mathbf{x}_{k,0}$ [17]. The strategy for setting the initial condition is developed in Sec. IV-B.

IV. Proposed Planner Formulation

This section details the formulation description and the features of the planning logic introduced in Sec. II. The target selector formulation is developed in Sec. IV-A, while its connection with the MPC planner, together with the relative exchange of information, are examined in Sec. IV-B.

A. *Target Selector formulation*

The target selector receives an optimal trajectory from the MPC planner, expressed in discrete form. Referring to the notation defined in Sec. III, the optimal trajectories can be described as

$$\mathbf{t}_k = \begin{bmatrix} t_{k,0} & t_{k,1} & t_{k,2} & \dots & t_{k,N} \end{bmatrix}^T,$$
$$\mathcal{X}_k = \begin{bmatrix} \mathbf{x}_{k,0}^T & \mathbf{x}_{k,1}^T & \mathbf{x}_{k,2}^T & \dots & \mathbf{x}_{k,N}^T \end{bmatrix}^T.$$

The increasing time sequence $t_{k,i}$ of the predictive horizon k is collected in vector $\mathbf{t}_k$, while the corresponding state values $\mathbf{x}_{k,i}$, describing the trajectory in terms of position, velocity and acceleration along the three Cartesian coordinates, are collected in vector $\mathcal{X}_k$.

Given the current time $\bar{t} \in \mathbb{R}$, and the k-th predictive horizon being the latest one available, the reference $\bar{\mathbf{x}}$ to be given to the controller at the time $\bar{t}$ is obtained with a piecewise linear interpolation, as

$$\bar{\mathbf{x}} = \frac{\mathbf{x}_{k,j+1} - \mathbf{x}_{k,j}}{t_{k,j+1} - t_{k,j}}(\bar{t}_m - t_{k,j}), \quad t_{k,j} \leq \bar{t}_m < t_{k,j+1}, \quad j \in \{0, 1, ..., N-1\}, \tag{11}$$

defining

$$\bar{t}_m := \bar{t}. \tag{12}$$

In case $\bar{t}_m < t_{k,0}$ or $\bar{t}_m \geq t_{k,N}$, we will assign $\bar{\mathbf{x}} = \mathbf{x}_{k,0}$ and $\bar{\mathbf{x}} = \mathbf{x}_{k,N}$, respectively. The piecewise linear interpolation formula for the reference extraction (11) justifies the adoption of the predictive first-order hold method for the discretisation of the MPC model.

At this point in our study, we would like to provide the target selector with a strategy to induce an adaptive behaviour for the output reference with respect to the actual end-effector state. Even between one MPC execution and the following, an external disturbance, e.g. an operator stopping/colliding with the robot, could cause the actual position of the end-effector to be very far from the reference one. When exploiting a Cartesian impedance controller, this leads to a significant accumulation of potential energy, which is released when the disturbance stops acting, e.g. the operator releases the robot. To address such a risky behaviour, we define a maximum allowed spatial distance d_{max} and set the reference to stop and wait for the end-effector whenever

$$||\bar{\mathbf{x}}^e - \mathbf{x}^e_{actual}|| \geq d_{max}, \tag{13}$$

where $\bar{\mathbf{x}}^e$ is a reduced version of vector $\bar{\mathbf{x}}$ containing only the position values, and $\mathbf{x}^e_{actual} = [X_{actual}, Y_{actual}, Z_{actual}]^T$ is a vector containing the actual Cartesian coordinates of the end-effector.

Supposing that the predictive horizon k is the last one available, let us denote with $t^{k,\bar{t}}_{cum}$ the cumulative time for which the locking condition (13) has been satisfied, i.e. accumulation of time since the predictive horizon k is available until the current time instant $\bar{t}$. This quantity is used to account for the lost time in which the end-effector cannot properly track the reference between one MPC execution and the following. Hence, the adaptive reference is obtained by modifying (12) into

$$\bar{t}_m := \bar{t} - t^{k,\bar{t}}_{cum}. \tag{14}$$

As soon as the new predictive horizon $k+1$ is available, the value $t^{k+1,\bar{t}}_{cum}$ will be used, ignoring the previous cumulative $t^{k,\bar{t}}_{cum}$, since the MPC planner has already accounted for $t^{k,\bar{t}}_{cum}$ in the new optimisation problem, as discussed in Sec. IV-B.

B. *Planning Strategy*

A disturbance in the task execution, caused by an interruption and/or interaction with environment, reduces the remaining time to complete motion without getting closer to the target, so that a solution to the optimisation problem might not be found. It is therefore essential to develop a strategy aimed at increasing the final time. As discussed above, the target selector measures the time $t^{k,\bar{t}}_{cum}$ in which the reference is stopped due to sufficiently large tracking errors i.e. (13), between two consecutive predictive horizons. In this section, we detail the proposed planning scheme relying upon the exchange of information between the MPC planner and the target selector introduced in the previous subsection, as a key contribution of this work. We shall formalise a strategy for the MPC planner to elaborate the responses coming from the target selector.

We define as *locking* the condition in which (13) is verified, i.e. the reference provided by the target selector is in the interruption phase, waiting for the end-effector to get closer. When a new MPC execution is run, two different scenarios can be observed:

1) when the locking condition is inactive, the distance between the end-effector and the reference is lower than the threshold value d_{max}, thus it is not critical to use the actual end-effector state as initial state, which can instead be initialised from the previous predictive horizon via interpolation. Nevertheless, since the time at which the predictive horizon is available does not coincide with the time at which it is computed, to guarantee a uniform transition between two predictive horizons and a good initial state estimation, it is necessary to account for the time required to compute the new horizon. As the computational time of the current MPC execution $t^k_{ct,actual}$ is unknown, we estimate it using a moving average of the computational times of previous executions, to obtain t^k_{ct}. The initial state is then estimated through a piecewise linear interpolation based on the previous predictive horizon. Therefore, indicating with $\bar{t}_k$ the time at which the k-th predictive horizon is calculated,

we can set its initial condition and time as

$$t_{k,0} = \bar{t}_k + t_{ct}^k, \tag{15}$$

$$\mathbf{x}_{k,0} = \frac{\mathbf{x}_{k-1,j+1} - \mathbf{x}_{k-1,j}}{t_{k-1,j+1} - t_{k,j}} \left((t_{k,0} - t_{cum}^{k-1,\bar{t}}) - t_{k-1,j} \right), \tag{16}$$

$$t_{k-1,j} \leq t_{k,0} - t_{cum}^{k-1,\bar{t}} < t_{k-1,j+1}, \quad j \in \{0, 1, ..., N-1\},$$

in which the piecewise linear interpolation for the calculation of $\mathbf{x}_{k,0}$ accounts also for the lost time $t_{cum}^{k-1,\bar{t}}$, i.e. the time interval in which the reference has not proceeded. It is then crucial to substitute (5) with the final time of the k-th predictive horizon obtained from

$$t_{k,N} = t_{k,N-1} + t_{cum}^{k-1,\bar{t}}, \tag{17}$$

being

$$t_{0,N} = t_f. \tag{18}$$

Finally, to provide the algorithm with robustness to occasional unfeasible predictive horizons, the last available predictive horizon is kept until a new one can be calculated. A representation of the inactive locking condition at the k-th predictive horizon activation for the X coordinate is shown in Fig. 2;

2) when the locking condition is active at a new MPC execution, i.e. the distance between the end-effector and the reference is greater than the threshold value d_{max}, it is necessary to re-generate the trajectory by considering the actual end-effector state $\mathbf{x}_{actual}(\bar{t})$ as the predictive horizon initial condition, namely

$$\mathbf{x}_{k,0} = \mathbf{x}_{actual}(\bar{t}). \tag{19}$$

The initial time $t_{k,0}$ is calculated from (15), and the new target time $t_{k,N}$, instead of (5), is obtained as

$$t_{k,N} = \max\left(\frac{t_f - t_{in}}{||\mathbf{r}_f - \mathbf{x}_{0,0}||}, \frac{\eta}{v_{max}} \right) ||\mathbf{r}_f - \mathbf{x}_{k,0}|| + t_{k,0}. \tag{20}$$

It implies the remaining time for moving the end-effector is proportional to the remaining gap between the actual end-effector position and the desired final one. The proportionality constant is then set to the maximum of the inverse of the initial average velocity and the inverse of the maximum allowed velocity v_{max}, adjusted by a correction factor η to account for the system dynamics. Lastly, in case a solution to the optimisation problem is not found, the reference position is set to the current position of the end-effector while waiting for new targets, since it is not possible to execute the previous predictive horizon without provoking risky motions of the robot. A representation of the active locking condition at the k-th predictive horizon activation for the X coordinate is shown in Fig. 3.

It should be noted that, while in this work the trigger for active/inactive locking was set based on monitoring of positional errors (with respect to the threshold d_{max}), it may also be set using collision detection approaches e.g.

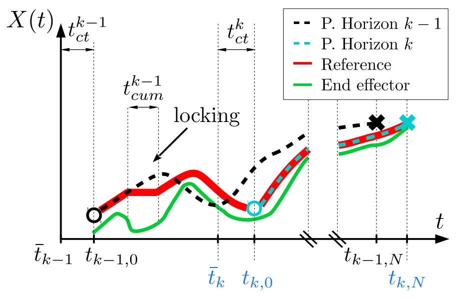

Fig. 2. Inactive locking at the MPC activation.

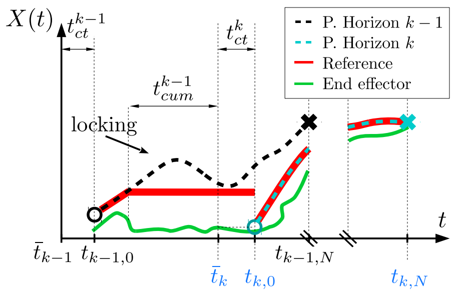

Fig. 3. Active locking at the MPC activation.

[18], [19]. This choice (positional error monitoring) allows for less sensitivity to model parameters accuracy, thanks to the robustness of the compliance regulation controller when joint torque sensing is available. However, when using other control methods (rather than an impedance controller) that do not permit replication of a given compliance, it is essential to exploit collision detection schemes. The fine performance of such methods relies on the accuracy of the model parameters and/or integration with parameter estimation/identification methods, which may be computationally burdensome for robots with large number of degrees of freedom e.g. humanoids and quadrupeds.

C. Comments on the choice of d_{max}

We now present a guideline to select the value of d_{max} with regard to disturbance rejection. As the force applied by the robot is associated to its impedance and its deviation from the reference pose, when dynamic motions are not concerned, the margin d_{max} needs to be selected as a sufficiently small value to prevent the robot from the exertion of large forces compromising the safety of the operation. At the same time, d_{max} must be large enough to allow for admissible interactions and to avoid repetitive and unnecessary initialisation of the predictive horizon with the actual state feedback. Given the dominance of compliance set by the impedance controller in quasi-static interactions, as the common condition in HRC, the absolute value of the force generated by the robot can be written as

$$|F| = \sqrt{K_X^2 e_X^2 + K_Y^2 e_Y^2 + K_Z^2 e_Z^2}, \tag{21}$$

where $K_{(\cdot)}$ denotes the stiffness of the impedance controller along the three Cartesian directions $(\cdot)$, and $e_{(\cdot)}$ represents the corresponding positional error of the end-effector with

Fig. 4. The CENTAURO robot employed for the experiments [20].

respect to the reference. Considering an identical error e' along the three directions, as a conservative assumption, we estimate an order of magnitude for the maximum disturbance rejection capability of the planner, in terms of deviation from the nominal trajectory, as

$$e_{rej} = \sqrt{e_X^2 + e_Y^2 + e_Z^2} = \sqrt{3}e' = \frac{\sqrt{3}F_{max}}{\sqrt{K_X^2 + K_Y^2 + K_Z^2}}, \tag{22}$$

where F_{max} is the maximum admissible force that guarantees safety. We therefore set d_{max} equal to e_{rej}, to allow for disturbances and interactions up to F_{max}.

V. Simulation and Experiments

For the validation of the proposed scheme, we develop a set of simulations and experiments. For experimental evaluations, we employ the left arm of the CENTAURO robot [20] shown in Fig. 4, while utilising six degrees of freedom. The MPC planner runs at 5 Hz in a *Non-Real-Time* (NRT) *node* on the *Robot Operating System* (ROS) [21] framework, implemented on a PC with an AMD Ryzen 7 1700X CPU and 32 GB RAM, while the software architecture *XBotCore* [22] is adopted to guarantee the *Real-Time* (RT) execution of the target selector and the Cartesian impedance controller, which run at 320 Hz. The exclusion of the MPC planner from the RT scheme relaxes the RT layer, and allows each predictive horizon trajectory to reach the target and remain available until the new one is computed.

The number of discretisation intervals of each predictive horizon is chosen as $N_k = 11$. A scaling of the differential equations is used to contrast the effects of the predictive horizon sampling time reduction while approaching the target. The end-effector, starting at rest from a homing configuration, targets moving 0.2 m, -0.2 m and 0.05 m along X, Y and Z, respectively, concluding the trajectory with zero velocity and acceleration. The desired time to reach the target is set to 10 s, and the position, velocity, acceleration and jerk constraints are equally set for all the Cartesian coordinates to $v_{max}^{(\cdot)} = 0.2$ m/s, $a_{max}^{(\cdot)} = 5$ m/s^2 and $u_{max}^{(\cdot)} = 100$ m/s^3, respectively. The MPC cost function weighting matrices are selected as $Q_{k,i} = \text{blkdiag}([\bar{Q}_{k,i}, \bar{Q}_{k,i}, \bar{Q}_{k,i}])$ and $R_{k,i} = \text{blkdiag}([\bar{R}_{k,i}, \bar{R}_{k,i}, \bar{R}_{k,i}])$ with

TABLE I

Selected values of the diagonal Cartesian stiffness matrix

X $\left[\frac{N}{m}\right]$	Y $\left[\frac{N}{m}\right]$	Z $\left[\frac{N}{m}\right]$	Roll $\left[\frac{Nm}{rad}\right]$	Pitch $\left[\frac{Nm}{rad}\right]$	Yaw $\left[\frac{Nm}{rad}\right]$
900	900	900	150	150	150

$\bar{Q}_{k,i} = \text{blkdiag}([0, 1, 1])$ and $\bar{R}_{k,i} = 0.01$. The threshold value and the safety factor are chosen as $d_{max} = 0.05\ m$ and $\eta = 1.3$.

The Cartesian impedance controller used to validate the approach is based on [23], with the diagonal Cartesian stiffness matrix values selected as in Tab. I. The orientation reference for the end-effector is set constant and equal to the initial one, and the double diagonalisation method is adopted to calculate the damping matrix, choosing the damping factors as $\xi_X = \xi_Y = \xi_Z = \xi_{Roll} = \xi_{Pitch} = \xi_{Yaw} = 0.7$.

It should be noted that, while the proposed scheme accounts for all different active/inactive locking conditions, the situation in which the locking is active and a solution to the predictive horizon cannot be found did not occur within the experiments executed, thanks to the appropriateness of the target time update policy.

A. Simulation

To evaluate the performance of the proposed scheme as compared to a baseline, this section presents a comparative simulation between our approach and the algorithm proposed in [15]. Particularly, the scenario in which the robot is stopped for a prolonged time interval by an external intervention, e.g. a human operator, is examined.

The reference and actual robot trajectories associated with the three Cartesian coordinates are shown in Fig. 5. The end-effector is left free to track the reference from the beginning of the motion to a time instant equal to 2 s, when we simulate a physical stop caused by an external intervention. As for the reference generated by the proposed scheme, as soon as the reference moves away from the end-effector by a distance larger than the threshold d_{max}, the locking condition is activated and the new predictive horizon is calculated using the actual state of the end-effector as initial condition. As discussed, the target time to complete the motion is shifted ahead to contrast the time loss caused by the interaction. After 8.5 s from the beginning of the interaction, we simulate the release of the end-effector so that it can complete the motion and reach the final state. Instead, using the algorithm shown in [15], the planner fails around the time 9.6 s, where the MPC optimization problem becomes unfeasible. This happens since the time loss caused by the prolonged interruption is not accounted for, and the remaining time is not sufficiently long for completing the motion while respecting the constraints. It should be also noticed that the repetitive closed-loop initialisation generates a highly discontinuous reference and could increase the tracking error of the impedance controller. This comparison showed that, while the approach introduced in [15] renders feasible trajectories

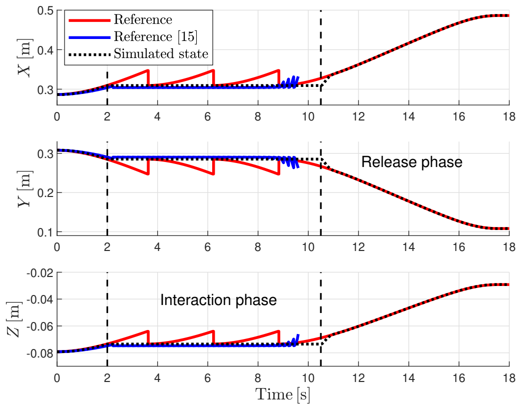

Fig. 5. Comparative simulation between the references generated by this work algorithm and [15].

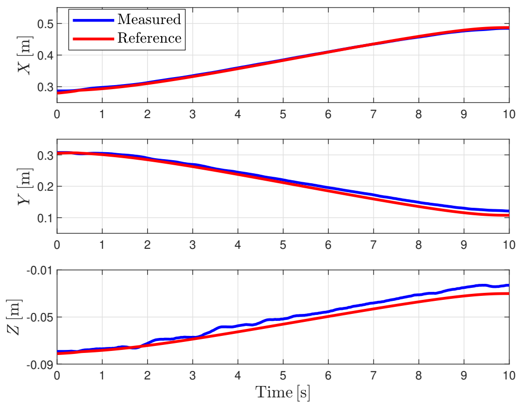

Fig. 6. Reference and measured trajectories in the free motion experiment.

in free motions, it fails to handle non-negligible disturbances, e.g. resulting from an external intervention.

B. Free Motion experiment

The first experiment consists in the end-effector tracking the calculated reference trajectories without any external interference. The online-computed reference trajectory and the actual one are shown in Fig. 6 for the three Cartesian coordinates. The lower accuracy of the controller along the Y and Z directions is due to the model errors affecting the dynamics compensation of the controller when operating at a relatively low impedance. Despite the presence of these errors, which are exploited to show the robustness of our planning scheme, the selected threshold d_{max} allows the planner to distinguish this case of low tracking performance from the one in which the end-effector is stopped by external causes. The measured state of the end-effector is therefore never used by MPC during the motion and the initial state of each predictive horizon is estimated using (16). This test evidences the smoothness of the trajectory generated by the target selector, thanks to the suitable estimation of the predictive horizon initial state, and the prediction of the computational time t_{ct}^{k} at each MPC execution.

The computational time of MPC and the relative estimation performed at each execution are shown in Fig. 7.

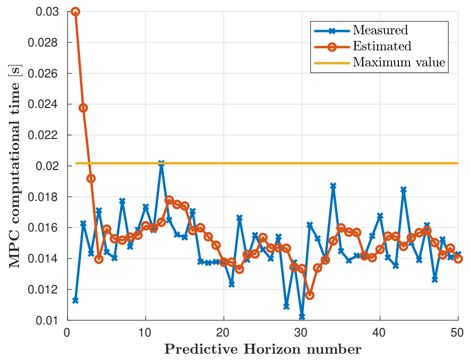

Fig. 7. Measured and estimated MPC computational time.

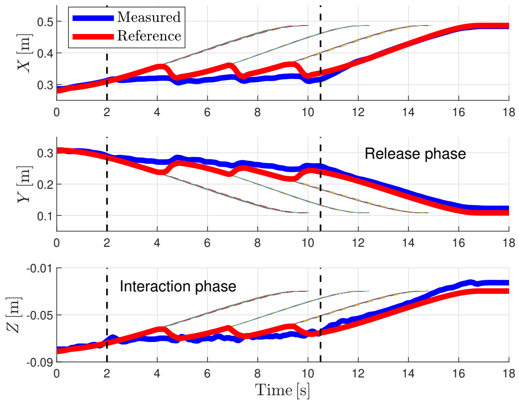

Fig. 8. Reference and measured trajectories (thick lines), and predictive horizons (fine lines) in the interrupted motion experiment.

The moving average estimator is set to an initial value of 0.03 s at the first execution (as a conservative guess). As shown in the figure, the estimated value progressively adapts to the measured trend. The maximum MPC computational time recorded during the test is about 0.02 s, which is fairly smaller than the MPC activation rate.

C. Interrupted Motion experiment

The second experiment, which shows more clearly the effectiveness of the proposed scheme, replicates the scenario presented in Sec. V-A. The physical interaction of the operator with the robot creates an interruption of approximately 8.5 s that prevents the end-effector from following the reference trajectory (the contact starts around the time 2 s, and ends at 10.5 s approximately). The reference and measured trajectories associated with the three Cartesian coordinates, together with the predictive horizons, are shown in Fig. 8. When locking mode is inactive, the predictive horizons are superimposed, since the initial state is estimated from the previous solution and the computational time of MPC is compensated. Instead, whenever the locking mode activates, the actual state is used as the initial condition.

D. Quicker Motion experiment

In the last experiment, we repeat the test described in Sec. V-C while the desired target time is set to 5 s, to produce a quicker motion and verify the robustness of the algorithm. The results are shown in Fig. 9.

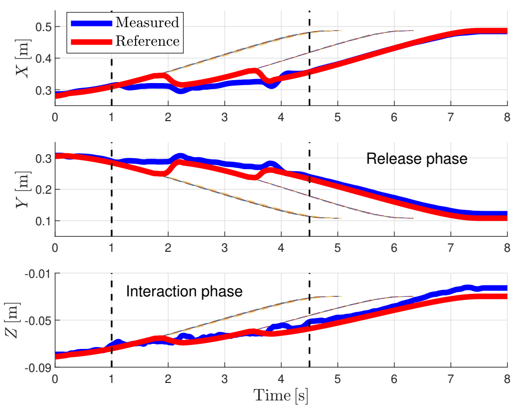

Fig. 9. Reference and measured trajectories (thick lines), and predictive horizons (fine lines) in the quicker motion experiment.

VI. Conclusions

This paper presented a Cartesian trajectory planning scheme, based on Model Predictive Control, allowing to manage unforeseen disturbances such as interactions with external objects/agents, interrupting the task execution. Differently from previous works, which employ the trajectory planner as a standalone unit, we proposed an MPC-based scheme to generate optimal trajectories while exchanging information with a novel target selector, responsible for defining the higher level decision-making process and sequentially updating the reference to the controller. This decoupling enables the separation of the computationally burdensome part of the planner from the real-time layer, and confers a high flexibility to the MPC formulation, so that its initial conditions and final targets can be externally defined through communication with the target selector. Moreover, to provide a smooth reference when switching between consecutive predictive horizons, the computational time of MPC is estimated and compensated during executions. The efficacy of the proposed approach has been experimentally validated on a robot arm, analysing the resulting motion with and without external interaction.

Future works shall focus on the integration with an orientation planner e.g. [24], and providing the planner with time optimality to automatically define the target time at each iteration, depending on the imposed kinematic constraints.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, May 2006.

[2] J. Bobrow, S. Dubowsky, and J. Gibson, "Time-Optimal Control of Robotic Manipulators Along Specified Paths," *The International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, Sept. 1985.

[3] D. Constantinescu and E. A. Croft, "Smooth and time-optimal trajectory planning for industrial manipulators along specified paths," *Journal of Robotic Systems*, vol. 17, no. 5, pp. 233–249, 2000.

[4] N. van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers, "Path-following NMPC for serial-link robot manipulators using a path-parametric system reformulation," in *2016 European Control Conference (ECC)*, June 2016, pp. 477–482.

[5] S. Yue, D. Henrich, W. L. Xu, and S. K. Tso, "Point-to-Point trajectory planning of flexible redundant robot manipulators using genetic algorithms," *Robotica*, vol. 20, no. 3, pp. 269–280, May 2002.

[6] H. Liu, X. Lai, and W. Wu, "Time-optimal and jerk-continuous trajectory planning for robot manipulators with kinematic constraints," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 2, pp. 309–317, Apr. 2013.

[7] S. F. P. Saramago and V. Steffen, "Trajectory Modeling of Robot Manipulators in the Presence of Obstacles," *Journal of Optimization Theory and Applications*, vol. 110, no. 1, pp. 17–34, July 2001.

[8] M. Ragaglia, A. M. Zanchettin, and P. Rocco, "Trajectory generation algorithm for safe human-robot collaboration based on multiple depth sensor measurements," *Mechatronics*, vol. 55, pp. 267–281, Nov. 2018.

[9] E. F. Camacho and C. B. Alba, *Model Predictive Control*. Springer Science & Business Media, Jan. 2013.

[10] L. Singh and J. Fuller, "Trajectory generation for a UAV in urban terrain, using nonlinear MPC," in *Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148)*, vol. 3, June 2001, pp. 2301–2308 vol.3.

[11] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path Planning and Tracking for Vehicle Collision Avoidance Based on Model Predictive Control With Multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, Feb. 2017.

[12] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2015, pp. 3346–3351.

[13] M. Svenstrup, T. Bak, and H. J. Andersen, "Trajectory planning for robots in dynamic human environments," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 4293–4298.

[14] Kay-Soon Low and H. Zhuang, "Robust model predictive control and observer for direct drive applications," *IEEE Transactions on Power Electronics*, vol. 15, no. 6, pp. 1018–1028, Nov. 2000.

[15] M. M. G. Ardakani, B. Olofsson, A. Robertsson, and R. Johansson, "Real-time trajectory generation using model predictive control," in *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2015, pp. 942–948.

[16] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design, Third Edition*. Englewood Cliffs, NJ: Prentice Hall, June 1997.

[17] J. M. Maciejowski, *Predictive Control: With Constraints*. Harlow, England: Prentice Hall, 2002.

[18] A. De Luca and R. Mattone, "Sensorless robot collision detection and hybrid force/motion control," in *IEEE international conference on robotics and automation*, 2005, pp. 999–1004.

[19] M. Guo, H. Zhang, C. Feng, M. Liu, and J. Huo, "Manipulator residual estimation and its application in collision detection," *Industrial Robot: An International Journal*, vol. 45, pp. 354–362, June 2018.

[20] N. Kashiri, L. Baccelliere, L. Muratore, A. Laurenzi, Z. Ren, E. M. Hoffman, M. Kamedula, G. F. Rigano, J. Malzahn, S. Cordasco, *et al.*, "Centauro: A hybrid locomotion and high power resilient manipulation platform," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1595–1602, 2019.

[21] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: An open-source Robot Operating System," *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, pp. 5–10, May 2009.

[22] L. Muratore, A. Laurenzi, E. M. Hoffman, A. Rocchi, D. G. Caldwell, and N. G. Tsagarakis, "XBotCore: A Real-Time Cross-Robot Software Platform," in *2017 First IEEE International Conference on Robotic Computing*, Apr. 2017, pp. 77–80.

[23] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger, "Cartesian impedance control of redundant robots: Recent results with the DLR-light-weight-arms," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 3, Sept. 2003, pp. 3704–3709 vol.3.

[24] M. Shahbazi, N. Kashiri, D. Caldwell, and N. Tsagarakis, "On the orientation planning with constrained angular velocity and acceleration at endpoints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 7033–7038.