Linear Model Predictive Control in SE(3) for online trajectory planning in dynamic workspaces

Nicolas Torres Alberto, Antun Skuric, Lucas Joseph, Vincent Padois, David Daney

▶ **To cite this version:**

Nicolas Torres Alberto, Antun Skuric, Lucas Joseph, Vincent Padois, David Daney. Linear Model Predictive Control in SE(3) for online trajectory planning in dynamic workspaces. 2022. hal-03790059

HAL Id: hal-03790059
https://hal.science/hal-03790059v1

Preprint submitted on 28 Sep 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Linear Model Predictive Control in SE(3) for online trajectory planning in dynamic workspaces

Nicolas Torres Alberto[1,2], Antun Skuric[2], Lucas Joseph[2], Vincent Padois[2] and David Daney[2]

***Abstract*— Efficient workspace sharing of collaborative robots and human operators remains an unsolved problem in the industry. This problem goes beyond the use of a priori or a posteriori safety measures and has to be tackled at the control level.**

To address the need of adaptation to human presence as well as to endow the robot with the ability to adapt interactively to new Cartesian targets, a linear Model Predictive Controller is proposed in this paper. This controller computes acceleration-bounded optimal Cartesian trajectories in SE(3) over a receding horizon.

The pertinence of the proposed control architecture is demonstrated using experiments with the Franka Emika robots in different scenarios implying both adaptation of the maximum allowed velocity to comply with human presence and on-the-fly update of a Cartesian goal pose.

I. Introduction

Safety in robotics has always been a concern, but modern industrial trends have specially exacerbated this in search for better integration of robots and humans in working environments. Robots are now required to optimally use their capacities while safely sharing their workspace with the operators around them.

Safety for collaborative robots is addressed through norms or technical specifications. In this domain, one of the most recent standards is the ISO 15066 Technical Specification [1]. Although innovative, considering safety from an energetic level [2] remains a rather coarse approach that often leads to unpractical settings and/or hard to certify installations. Overall, collaborative robots in the industry are mostly used as standard robots, behind cages or costly and inefficient immaterial safety zones.

Addressing safety in an efficient way when considering shared workspaces is a complex problem that requires an online evaluation of multiple moving bodies. Fig. 1 aims to depict this situation: a dynamically safe area for the robot to evolve can be computed based on the state of both the operator and the robot as well as on their near future motions. Nevertheless, computing this potential area and taking online control decisions based on this computation in an accurate enough fashion to allow for close encounters such as the ones in Fig. 1 remains an opened research problem [3], [4], [5].

[1]Stellantis, Centre Technique Vélizy, France
nicolas.torres@stellantis.fr
[2]Inria, AUCTUS Team, Talence, France
antun.skuric@inria.fr, lucas.joseph@inria.fr, vincent.padois@inria.fr, david.daney@inria.fr

This work is funded by the ANRT and Stellantis Group and performed within the framework of the OpenLab AI.

This work is supported by BPI France through the LICHIE project in collaboration with Airbus.

Fig. 1. A safe workspaces in a collaborative environment cannot be defined with static zones as it depends on the robot-operator states and tasks.

In its most general form, it boils down to solving a global optimal control problem online that determines at each time a control trajectory that satisfactorily combines safety and efficiency. Despite some research effort in this direction [6], this very general problem is far from being solved in a generic way and goes far beyond the scope of this paper. Yet, a clear required feature emerges from the search for optimality: the need for prediction of the effect of control actions over a time horizon.

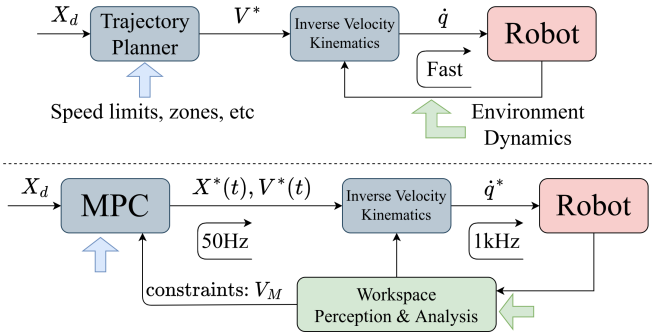

Fig. 2. Classical vs proposed control architecture.

Beyond safety, a close dynamic interaction between a human operator and a robot may also imply on-the-fly re-definitions of the task to be achieved by the robot. This cannot be solved by "classical control architectures" (such as the one depicted in the top of Fig. 2) which generally consider offline planning solutions, incapable of reacting to dynamic events implied in the considered collaborative scenarios. Instead one needs to consider online re-planning at the task level.

Model Predictive Control is the ideal control approach to combine both online re-planning and prediction of the control action over a horizon. It offers an ideal framework to optimally and interactively exploit robot capacities while respecting safety limits.

As a consequence, this article proposes a hierarchical cascade of closed-loop controllers as suggested in [7] combining both online trajectory re-planning over a receding horizon and task space control in two stages: a high-level safety-aware operational space model predictive controller and a high-frequency lower-level task-to-joint-space controller.

More specifically, the focus of this paper lies in continuously finding the optimal pose and twist trajectory that will drive a system from an initial pose to some target pose, subject to velocity and acceleration constraints that account for safety and the real robot capacities. While this is not a fully novel problem, predictive algorithms in the literature:

- often use non-linear solvers [6] [8] which may not be appropriate for frequent re-planning;
- are formulated at the joint-space level [9] [10] which is not ideal when considering task-space specification as a central feature;
- mostly disregard orientation.

By leveraging the relation between the pose space (the Lie Group $\mathbb{SE}(3)$) and its tangents space (the Lie algebras $\mathfrak{se}(3)$) [11] [12] [13], this work proposes a Linear Model Predictive Control (MPC) capable of estimating the evolution position and orientation with faster linear MPC solvers.

First, Section II presents some groundwork concepts and notation conventions employed in this work. Section III follows with the relation between the Lie Group and its Lie algebras and how it can be exploited for optimization problems, then describes the proposed control architecture. Afterwards, Section IV goes into details on the experimental setup and the results of implementing it on a real robot. Finally, a discussion and conclusions are presented in Section V.

II. Pose prediction in $\mathbb{SE}(3)$ vs linear MPC

A. Pose trajectory prediction

Formulating the control problem over a receding horizon at the Cartesian level requires the prediction of the end-effector pose given a horizon of control input, here considered at the velocity level.

Given some system with initial pose $\boldsymbol{\mathcal{X}}_i \in \mathbb{SE}(3)$ subject to a body twist $\boldsymbol{\nu}(t) \in \mathfrak{se}(3)$, its discretized pose trajectory $\boldsymbol{\mathcal{X}}_k = \boldsymbol{\mathcal{X}}(t)\big|_{t=k\Delta t} \in \mathbb{SE}(3)$ can be described as:

$$\begin{aligned}\boldsymbol{\mathcal{X}}_k &= \boldsymbol{\mathcal{X}}_i e^{\boldsymbol{\nu}_0\Delta t} e^{\boldsymbol{\nu}_1\Delta t} \dots e^{\boldsymbol{\nu}_{k-1}\Delta t} \\ &= \boldsymbol{\mathcal{X}}_i \prod_{i=0}^{k-1} e^{\boldsymbol{\nu}_i\Delta t}\end{aligned} \tag{1}$$

Given initial and target poses $\boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{X}}_d \in \mathbb{SE}(3)$, this relation can be used incrementally to compute desired twists at each time instant given some tracking error [14]. Nevertheless, the pose horizon is a non-linear function of the twist trajectory. Yet, to be solved at a decent control rate for reactivity, the model predictive controller should be expressed in a linear form. The next subsection recalls the basics of such a formulation.

B. Linear systems in a receding horizon & MPC

A given dynamic system can be described at a time instant t_n with a state vector $\boldsymbol{x}_n$, representing its current position and orientation; it is actuated with an input vector $\boldsymbol{u} \in \mathfrak{se}(3)$ representing its body twist.

The resulting relation between the input and state of a discretized linear system is formulated as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{2}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ represent its state and input matrices.

Discretizing a receding time window in h time steps of duration Δt_h (total duration $T = h\Delta t_h$), one obtains $t_k = (n+k)\Delta t_h$ where $k = 0, 1, \dots, h$, which yields state and input vectors:

$$\boldsymbol{X}_{0\dots h} = \begin{bmatrix} \boldsymbol{x}_{k|n} \\ \boldsymbol{x}_{k+1|n} \\ \vdots \\ \boldsymbol{x}_{k+h|n} \end{bmatrix} \quad \boldsymbol{U}_{0\dots h-1} = \begin{bmatrix} \boldsymbol{u}_{k|n} \\ \boldsymbol{u}_{k+1|n} \\ \vdots \\ \boldsymbol{u}_{k+h-1|n} \end{bmatrix} \tag{3}$$

$$\overline{\boldsymbol{X}} = \boldsymbol{X}_{0\dots h} \quad \overline{\boldsymbol{X}} = \boldsymbol{X}_{1\dots h} \quad \underline{\boldsymbol{X}} = \boldsymbol{X}_{0\dots h-1} \quad \underline{\boldsymbol{U}} = \boldsymbol{U}_{0\dots h-1}$$

that allows extending (2) for a horizon:

$$\overline{\boldsymbol{X}} = \boldsymbol{A}'\underline{\boldsymbol{X}} + \boldsymbol{B}'\underline{\boldsymbol{U}} \tag{4}$$

where $\boldsymbol{A}', \boldsymbol{B}'$ are appropriately size diagonal matrices based on $\boldsymbol{A}, \boldsymbol{B}$ from (2) while $\overline{\boldsymbol{X}}$ and $\underline{\boldsymbol{U}}$ respectively correspond to the future states and control inputs of the system.

Finally, one can formulate the linear optimal control problem in a receding horizon as a Quadratic Program (QP) with the following general cost function:

$$\min_{\boldsymbol{u},\boldsymbol{x}} \frac{1}{2}\underline{\boldsymbol{U}}^T\boldsymbol{R}\underline{\boldsymbol{U}} + \frac{1}{2}\underline{\boldsymbol{X}}^T\boldsymbol{P}\underline{\boldsymbol{X}} + \frac{1}{2}{\boldsymbol{x}_h}^T\boldsymbol{P}_T\boldsymbol{x}_h \tag{5}$$

subject to the equality (4) and polyhedral constraints:

$$\begin{aligned}\boldsymbol{x}_m \le \boldsymbol{C}_{\boldsymbol{x}}\boldsymbol{x}_k \le \boldsymbol{x}_M \\ \boldsymbol{u}_m \le \boldsymbol{C}_{\boldsymbol{u}}\boldsymbol{u}_k \le \boldsymbol{u}_M\end{aligned} \tag{6}$$

where $\boldsymbol{R}, \boldsymbol{P}, \boldsymbol{P}_T$ represent, respectively, weighting matrices for the input, state and the terminal state. Meanwhile, $\boldsymbol{C}_{\boldsymbol{x}}, \boldsymbol{C}_{\boldsymbol{u}}$ allow formulating linear constraints with respect to the state and input; $\boldsymbol{x}_m, \boldsymbol{x}_M$ and $\boldsymbol{u}_m, \boldsymbol{u}_M$ respectively designate the state and input bounds.

Equation (5) under constraints (4) and (6) constitutes the general linear MPC formulation as a QP.

III. Linear MPC formulation on manifolds

A. Position and orientation in the tangent space

The linear MPC problem requires a linear system form like in (2), but the pose trajectory of a system evolves as the successive Lie group transformations, acting on its state (pose) to travel through the $\mathbb{SE}(3)$ manifold at each timepoint, as shown in (1). This Section presents how to link both forms while providing a rough explanation on the conceptual steps for optimizations on manifolds employed in this work, for more details, [13] goes deeper.

As explained in [12], the $\mathbb{SE}(3)$ manifold is diffeomorphic to $\mathfrak{se}(3)$, hence there exists a differentiable bijective map such as:

$$\begin{aligned} \log &: \mathbb{SE}(3) \to \mathfrak{se}(3) \qquad \boldsymbol{\xi} \to \log(\boldsymbol{\mathcal{X}}) \\ \exp &: \mathfrak{se}(3) \to \mathbb{SE}(3) \qquad \boldsymbol{\mathcal{X}} \to \exp(\boldsymbol{\xi}) \end{aligned} \tag{7}$$

The log function[1] relates the Lie group $\mathbb{SE}(3)$ to its Lie algebra $\mathfrak{se}(3)$, its tangent space, at the origin. It behaves like an euclidean space for optimization purposes. In fact, this allows representing a pose as a 6-vector instead of an homogeneous matrix[2]: given some pose $\boldsymbol{\mathcal{X}}$, then $\boldsymbol{\xi}$ can be interpreted as the equivalent twist required to drive a system from the origin frame (where position is at the origin and orientation is aligned) to some position and orientation in space $\boldsymbol{\mathcal{X}}$ in 1s.

The reparametrization of the original space into an optimization space that offers some desirable properties is often referred to as *lifting* or *pushing* while the inverse operation is a *retract* or *pull*.

Decomposing (1) for a single step, yields the relation between an infinitesimal increment in pose (body twist) and the resulting pose:

$$\boldsymbol{\mathcal{X}}_{k+1} = \boldsymbol{\mathcal{X}}_k e^{\boldsymbol{\nu}_k \Delta t} \tag{8}$$

Equation (8) is a first-order approximation, analogous to the one presented in [13] for $\mathbb{SO}(3)$ and [15] (for more general manifolds). Reformulating it in the tangent space:

$$\boldsymbol{\xi}_{k+1} = \log(e^{\boldsymbol{\xi}_k} e^{\Delta t \boldsymbol{\nu}_k}) \approx \boldsymbol{\xi}_k + \mathrm{dlog}_{\boldsymbol{\xi}_k} \Delta t \boldsymbol{\nu}_k \tag{9}$$

where $\mathrm{dlog}_{\boldsymbol{\xi}_k}$ is the log derivative[3] at $\boldsymbol{\xi}_k$, that releates additive increments in the tangent space at the origin to the right-multiplied increments in the pose space. In Layman's terms, this relates the body twist with an increment in the logarithm of the pose. This constitutes the *push-forward* operation that enables optimization in a manifold, extending the concepts introduced in this Section.

Equation (9) shows the missing link that allows the present work to formulate a pose propagation in a linear form (as in (2)) for a receding horizon, as required for the linear MPC in (5).

B. Geodesic path and tangent space

As introduced in Section II-A, this work aims at finding the pose and input trajectory that will drive the system to some target pose $\boldsymbol{\mathcal{X}}_d$. In order to embed this in the QP cost function, a distance metric formulated in the tangent space is necessary. In fact, $\mathfrak{se}(3)$ is a Riemannian manifold equipped with the geodesic distance metric known as *Log-Euclidean* ([16], Table 1):

$$e_k = || \underbrace{\boldsymbol{\xi}_k}_{\log(\boldsymbol{\mathcal{X}}_k)} - \underbrace{\boldsymbol{\xi}_d}_{\log(\boldsymbol{\mathcal{X}}_d)} ||_2^2 \tag{10}$$

[1]Throughout this article, the log and exp operations will adopt the vectors forms. This is an abuse of notation but one can go back and forth between the matrix and vector forms through the vee map and its inverse [15] [12].

[2]The one to one correspondence is maintained under the conditions detailed in [13] and [12] (Section 2.1). Otherwise, the map is surjective.

[3]Given the adopted vector form of the tangent space in this article, the $\mathrm{dlog}_{\boldsymbol{\xi}_k}$ takes the form of a 6×6 matrix.

The last challenge to overcome is that there exist multiple paths that connect two extreme poses $\boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{X}}_d \in \mathbb{SE}(3)$. Given some $\alpha(t) \in [0, 1]$ the *shortest geodesic* can be interpolated by using the path that passes through the origin and computing its "distance" in tangent space $\boldsymbol{\delta}$:

$$\boldsymbol{\mathcal{X}}_k = \boldsymbol{\mathcal{X}}_i e^{\alpha_k \boldsymbol{\delta}} \qquad \boldsymbol{\delta} = \log(\boldsymbol{\mathcal{X}}_i^{-1} \boldsymbol{\mathcal{X}}_d) \tag{11}$$

which is depicted in Fig. 3. One way to ensure that the trajectory optimization in the lifted space corresponds to (11) is through the choice of the lift function:

$$\boldsymbol{\psi}(\boldsymbol{x}) = \log(\boldsymbol{\mathcal{X}}_l^{-1} \boldsymbol{x}) \qquad \boldsymbol{\mathcal{X}}_l = \boldsymbol{\mathcal{X}}_d \tag{12}$$

where $\boldsymbol{\mathcal{X}}_l$ constitutes the lift pose. This way, the optimization can be "directed" from $\boldsymbol{\psi}(\boldsymbol{\mathcal{X}}_i) \to \boldsymbol{\psi}(\boldsymbol{\mathcal{X}}_d)$, which yields the shortest geodesic. The choice of $\boldsymbol{\mathcal{X}}_l = \boldsymbol{\mathcal{X}}_d$ stems from the application: it allows the optimization to always converge to the origin in the tangent space $\boldsymbol{\xi}_I$ because: $\boldsymbol{\psi}(\boldsymbol{\mathcal{X}}_d) = \boldsymbol{\xi}_I$. In fact, the lift function in (12) implies that $\boldsymbol{\psi}$ is in the tangent space at $\boldsymbol{\mathcal{X}}_l^{-1}$.

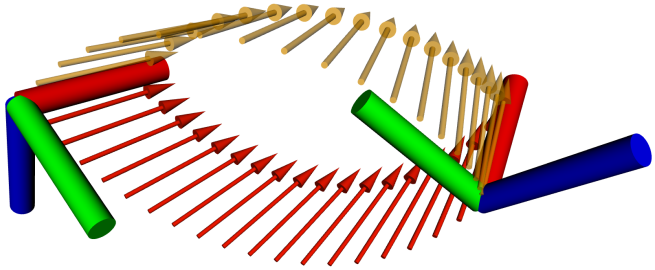

Fig. 3. There exist multiple paths that connect some initial and target poses $\boldsymbol{\mathcal{X}}_i, \boldsymbol{\mathcal{X}}_d \in \mathbb{SE}(3)$. The image shows the successive directions of the red axis of a frame (3-axis) moving through the path. It illustrates the shortest geodesic (in red) and an alternative path (in yellow). The yellow path can be interpolated in the tangent space and then transformed to poses as $\boldsymbol{\mathcal{X}}_k = e^{(1-\alpha_k)\boldsymbol{\xi}_0 + \alpha_k \boldsymbol{\xi}_d}$ for some $\alpha(t) \in [0, 1]$.

C. Operational Space MPC

Using the same notation as in (3), it is possible to extend (10) for a horizon:

$$\overline{\boldsymbol{X}}^*, \underline{\boldsymbol{U}}^* = \arg\min_{\boldsymbol{x}, \boldsymbol{u}} ||\overline{\boldsymbol{X}} - \boldsymbol{\xi}'_d||_2^2 + \boldsymbol{\gamma} ||\underline{\boldsymbol{U}}||_2^2 \tag{13}$$

$$\boldsymbol{\xi}'^T_d = [\underbrace{\boldsymbol{\psi}(\boldsymbol{\mathcal{X}}_d)^T \ldots \boldsymbol{\psi}(\boldsymbol{\mathcal{X}}_d)^T}_{h-1 \text{ times}}]^T \tag{14}$$

$$\begin{aligned} \text{s.t:} \quad \boldsymbol{x}_{k+1} &= \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k && \text{for } k = 0, \ldots, h-1 \\ \boldsymbol{u}_m(d_h) &\le \boldsymbol{u}_k \le \boldsymbol{u}_M(d_h) && \text{for } k = 0, \ldots, h-1 \\ \dot{\boldsymbol{u}}_m &\le \tfrac{\boldsymbol{u}_{k+1} - \boldsymbol{u}_k}{\Delta t} \le \dot{\boldsymbol{u}}_M && \text{for } k = 0, \ldots, h-2 \end{aligned} \tag{15}$$

where $\boldsymbol{\gamma}$ designates a weighting term for the input throughout the horizon and $\boldsymbol{\xi}'_d$ corresponds to the target pose after applying the lift function vectorized throughout the horizon.

Equation (15) finally links (9) with the linear MPC formulation in (5) as:

$$\boldsymbol{A} = \boldsymbol{I}_6 \qquad \boldsymbol{B} = \mathrm{dlog}_{\boldsymbol{\xi}_i} \Delta_h t \tag{16}$$

where $\boldsymbol{A}$ is an identity matrix and $\boldsymbol{B}$ reflects the tangent space linearization at the initial robot pose, as $\boldsymbol{\xi}_i = \boldsymbol{\psi}(\boldsymbol{\mathcal{X}}_i)$, closing

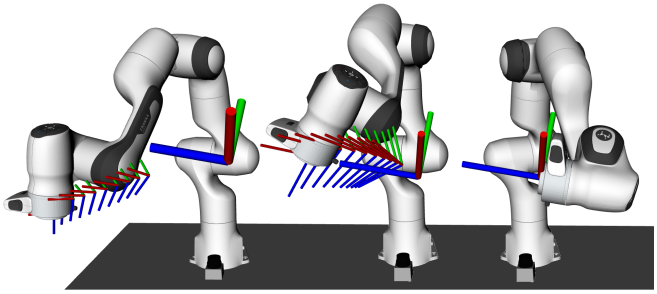

Fig. 4. The online MPC progressively "discovers" the path towards the target pose.

the feedback loop of the MPC. This constitutes the central relation that enables tangent space linear MPC.

The input constraints, where $\boldsymbol{u}_M = -\boldsymbol{u}_m$, establish the limits related to the distance between the end-effector and the human $d_h = ||\boldsymbol{p}_{ee} - \boldsymbol{p}_h||^2$. A simple piecewise function is employed for each component of $\boldsymbol{u}_M$ designated $u_{M,i}$ for $i = 1, \ldots, 6$:

$$u_{M,i}(d_h) = \begin{cases} V_m & d_h \leq d_m \\ \frac{d_h - d_m}{d_M - d_m} V_M & d_m < d_h < d_M \\ V_M & d_h \geq d_M \end{cases} \tag{17}$$

that has user defined extreme values for the minimal and maximal distances d_m, d_M and velocities V_m, V_M; all greater than zero. This function ensures a linear relation between maximum velocity and the distance inside the distance interval; outside of this range $\boldsymbol{u}_M$ is assigned either the minimal or the maximum velocity.The minimum velocity $\boldsymbol{V}_m$ is chosen to be close to zero to completely stop the robot motion when the distance is below the threshold d_m. This function is chosen for its simplicity but it showcases how to potentially integrate complex safety behaviours as velocity modulation functions, such as the robot stopping distance [17] and damage-based maximum velocities [18].

Finally, in order to recover the desired poses $\boldsymbol{\mathcal{X}}_k^{\text{des}}$ from $\overline{\boldsymbol{X}}^*$:

$$\boldsymbol{\mathcal{X}}_k^{\text{des}} = \boldsymbol{\psi}^{-1}(\boldsymbol{x}_k^*) \qquad \boldsymbol{\psi}^{-1}(x) = \boldsymbol{\mathcal{X}}_l \exp(x) \tag{18}$$

This constitutes the linear MPC formulation that allows trajectory planning in tangent space. Fig. 4 shows an example horizon computed by the MPC towards one of the target poses. It exemplifies that the MPC only plans over a limited time horizon (robot on the left). As the robot moves towards the target pose, the MPC computes the new trajectory according to the new robot state (middle robot), until it reaches the target pose (robot on the right).

An example output of the MPC control loop is shown in Fig. 5 for a simple Cartesian motion along one of the axis of the reference frame attached to the robot base (x here). The prediction horizon is 75ms composed of 15 time steps of 5ms each. The update rate of the MPC is 50Hz thus implying an interpolation of the first computed time-step as the Cartesian space state is updated at 1kHz. The resulting trajectory is compared to a time optimal trapezoidal acceleration profile computed once using Ruckig [19]. Unlike the globally planned trajectory which includes jerk bounds, the current formulation of this MPC does not embed jerk limitations, yielding some large acceleration variations. Yet, the resulting profile is very similar and this demonstrates the ability of MPC to obtain quasi-time-optimal trajectories while conferring online adaptation capabilities to the overall architecture. Moreover, while Ruckig is in practice a decoupled multi-dofs planner, the proposed approach truly accounts for 3D motions in both position and orientation.

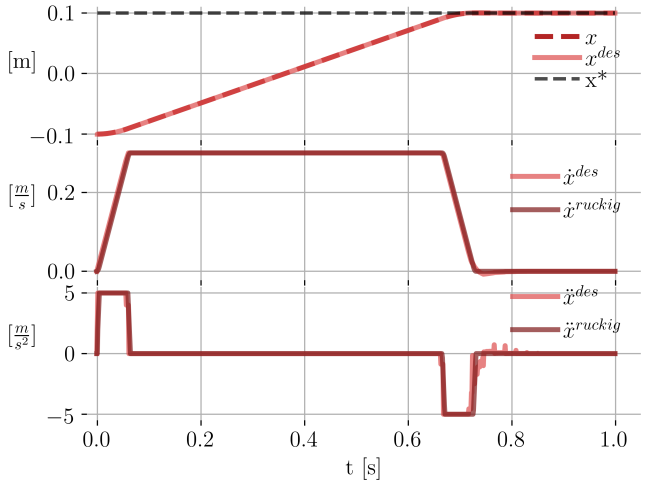

Fig. 5. This example show the result of employing the MPC to replanify at 50Hz and apply the resulting twists for a simple Cartesian motion along one of the axis of the reference frame attached to the robot base (x here). To compare optimality, the output of the jerk-bound algorithm in [19].

D. Inverse velocity kinematic solver

As depicted in Fig. 2, this work uses a fast rate control loop solving for inverse velocity kinematics. The inverse kinematics solver is similar to the one in [17]. It is formulated as a linearly constrained QP that finds the optimal joint velocity; the main difference being that a safety-aware constraint is added to limit the twist space. The resulting QP formulation is :

$$\dot{\boldsymbol{q}}^* = \arg\min_{\dot{\boldsymbol{q}}} ||\boldsymbol{v}^* - \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}||_2^2 + w_{\text{reg}}||\dot{\boldsymbol{q}}_{\text{reg}} - \dot{\boldsymbol{q}}||_2^2 \tag{19}$$

$$\text{s.t.} \quad \dot{\boldsymbol{q}}_m \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}_M \tag{20}$$

$$\boldsymbol{q}_m \leq \boldsymbol{q}(\dot{\boldsymbol{q}}) \leq \boldsymbol{q}_M \tag{21}$$

$$\boldsymbol{u}_m \leq \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \leq \boldsymbol{u}_M \tag{22}$$

where (22) is the so called safety-aware constraint, $\boldsymbol{J}(\boldsymbol{q})$ represents the robot's joint Jacobian; $\boldsymbol{u}_M$ is defined in (17) and $\boldsymbol{u}_m = -\boldsymbol{u}_M$; $\boldsymbol{v}^*$ is the output body twist from the MPC. The joint bound limit in (21) is expressed using a first order Taylor expansion on the robot joint configuration. Furthermore, $w_{\text{reg}}||\dot{\boldsymbol{q}}_{\text{reg}} - \dot{\boldsymbol{q}}||_2^2$ corresponds to a regularisation task that uses the redundant degrees of freedom of the robot to keep a desired configuration. The regularization weight, w_{reg} is chosen small enough to minimally affect the main trajectory tracking task. Such regularization implementation is detailed in [17]. In this paper the desired configuration corresponds to the robot mean joint position relative to its bounds.

IV. Results

A. Experimental setup

The objective of the following experiment is to show the controller ability to

1) find a trajectory towards arbitrary target poses while respecting the constraints previously described;
2) adapt to changing constraints online.

To do so, a 7 dof Panda robot from Franka Emika is requested to move to four different targets. These targets are not known in advance by the MPC but given on the fly. The robot Cartesian velocity is constrained using (17). The distance between the operator and the robot is determined by a Hokuyo Laser range finder placed at the robot base, as shown in Figures 1 (in the real setup) and 4 (in the simulation). The following values are used to modulate the robot velocity according to (17):

$$\begin{array}{lll} V_{m,\text{lin}} = 0.01m.s^{-1} & V_{m,\text{ang}} = 0.01s^{-1} & d_m = 0.2m \\ V_{M,\text{lin}} = 1m.s^{-1} & V_{M,\text{ang}} = 1.5s^{-1} & d_M = 1m \end{array} \quad (23)$$

In an industrial scenario, these values would be determined by the type of human-robot collaboration and technical specification used, such as the ISO TS 15066.

For this experiment, a cascade controller as shown in Fig. 2 is implemented. This schema also shows the working frequency of each controller and the feedback flow. The MPC implemented in this paper uses OSQP [20] as a QP solver while the joint velocity controller[4] used for this experiments is available online and uses qpOASES [21]. The robot model is computed using the Pinocchio library [22]. The robot control architecture is implemented using the Robot Operating System (ROS) framework and run in real-time at a frequency of 1kHz using the franka_ros library.

When solving the MPC problem for a new target pose, computation times vary a lot depending on the amount of steps and constraints in the horizon. Moreover, the first solving step (cold start) always takes more time. For a typical horizon of $h = 10, \Delta th = 50ms$ OSQP solves the problem in under 10ms. The solution of the previous QP is used as a first guess for the next one (hot start). Assuming that between two resolutions the optimal solution is close to the previous one, this helps speeding up the resolution. This leads to computation times under 1ms with a warm start.

B. Analysis

This section shows the results from periodically requesting the robot to move to a different pose. Fig. 6 presents the experiments performed on the Panda robot, including

1) the error between the current robot pose and the target pose;
2) the body twist used during the trajectory, as well as the maximum available velocities.

Result 1) shows the error between the current position $\boldsymbol{p}_k \in \mathbb{R}^3$ and orientation of the robot $\boldsymbol{R}_k \in \mathbb{SO}(3)$ and the target ones. There is a spike every time a new pose is requested. It is important to highlight that this is not the resulting tracking error of following the trajectory planned with the MPC but rather the "distance" between the current pose and the objective.

To compute this error:

$$\boldsymbol{e} = \boldsymbol{p}_k - \boldsymbol{p}_d \qquad \boldsymbol{\phi} = \log(\boldsymbol{R}_k^{-1}\boldsymbol{R}_d) \quad (24)$$

is used, where $\boldsymbol{e} \in \mathbb{R}^3$ reflects the position distance and $\boldsymbol{\phi} \in \mathfrak{so}(3)$ is the orientation error.

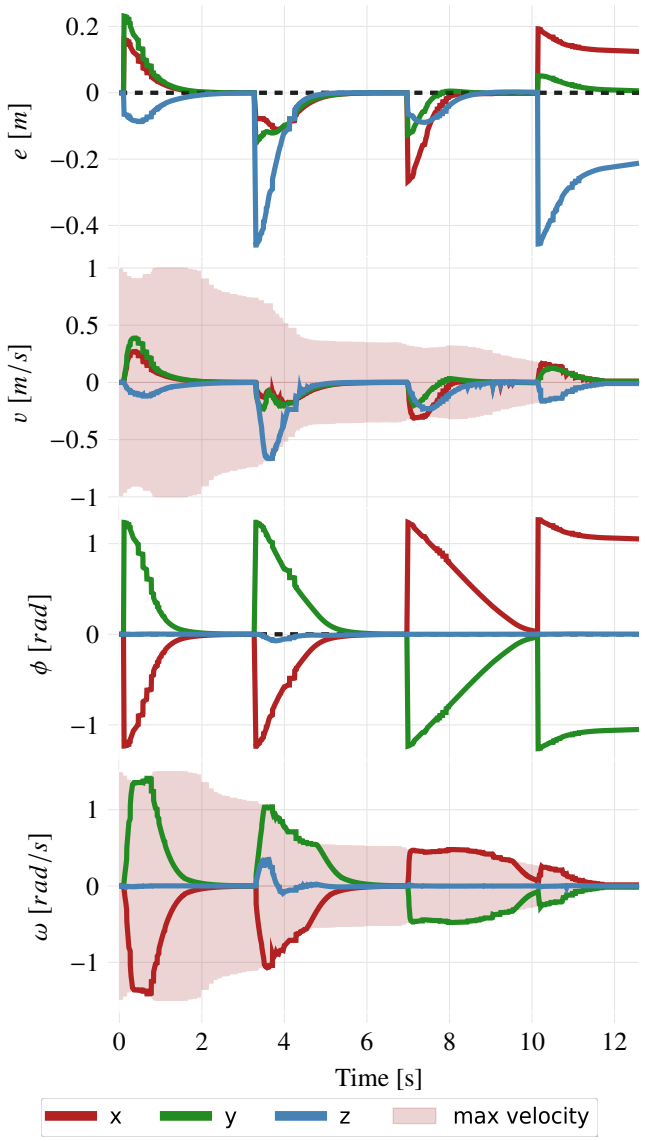

Fig. 6. Shows the result of requesting the robot to move between 4 preset poses. While the robot moves, a laser sensor is used to capture the minimal distance to a human and modulate the maximum velocity (shown in red shade). The error curves show the distance between the current position and orientation $(\boldsymbol{p}_k, \boldsymbol{R}_k)$ and the preset target poses $(\boldsymbol{p}_d, \boldsymbol{R}_d)$. The linear error is $\boldsymbol{e} = \boldsymbol{p}_d - \boldsymbol{p}_k$, while the angular error is computed as $\boldsymbol{\phi} = \log(\boldsymbol{R}_k^{-1}\boldsymbol{R}_d)$. The MPC (T=300ms, $h = 10, \Delta_h = 30ms, f_{\text{MPC}} = 50Hz$) modulates the body twist $\boldsymbol{\nu} = [\boldsymbol{v}^T, \boldsymbol{\omega}^T]^T$ respecting the safe limits imposed even when they become so low that the robot stops moving.

For result 2), the velocity curves show the body twist $\boldsymbol{\nu} = [\boldsymbol{v}^T, \boldsymbol{\omega}^T]^T$ as well as the available velocity (in a red

[4] https://gitlab.inria.fr/auctus-team/components/robots/panda/panda_qp_control

shade) that is being scaled by the distance between the robot's end-effector and the human.

The main highlight of these results is that the MPC is able to modulate velocities according to constraints changing in real-time achieving the two main objectives of this work:

1) when the human is far, the algorithm is able to saturate the velocity, maximizing the robot movement;
2) while the human approaches, the robot's movement is handicapped until it is no longer moving.

A subsidiary result of this MPC formulation is that it allows changing the robot target position on the fly, overriding the current target. The resulting robot motion keeps being smooth since it respects a continuous velocity profile as shown in Fig. 5. These results are not shown in the following paper but can be observed in the attached video.

V. Conclusion

This paper proposes an efficient linear model predictive control approach that can deal with the online planning of SE(3) motion in task space. The main features of the proposed control approach lie in its ability to generate optimal Cartesian motion which dynamically accounts both for targets updated on-the-fly and evolving motion constraints. This receding horizon approach endows the robot with the ability to interactively adapt to its environment. More particularly the safety of a human operator sharing its workspace with the robot can be accounted for through the sensor-based adaptation of maximum velocity constraints.

Future work will focus on extending linear constraints in the MPC for the Cartesian pose state, to limit the effective position and orientation space. Furthermore, this articles shows acceleration-bound trajectory planning. A prospective result in the future is to extend this technique for the jerk, allowing for smoother trajectories.

This work also opens doors to potentially exploiting a more efficient expression of Cartesian constraints such as convex polytopes [23]. This type of constraints goes beyond classical box-like bounds and is a step towards more complex adaptation of the robot behaviour based on its motion capabilities as well as related to the motion capabilities of the human sharing the workspace.

References

[1] ISO/TS-15066, *Robots and robotic devices - Collaborative robots*, International Organization for Standardization, Geneva, Switzerland, 2016.

[2] L. Joseph, V. Padois, and G. Morel, "Experimental validation of an energy constraint for a safer collaboration with robots," in *2018 International Symposium on Experimental Robotics (ISER 2018)*, Buenos Aires, Argentina, Nov. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01883995

[3] P. Long, C. Chevallereau, D. Chablat, and A. Girin, "An industrial security system for human-robot coexistence," *Industrial Robot: An International Journal*, 2017.

[4] J. Mainprice, R. Hayne, and D. Berenson, "Goal set inverse optimal control and iterative replanning for predicting human reaching motions in shared workspaces," *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 897–908, 2016.

[5] M. Eckhoff, R. J. Kirschner, E. Kern, S. Abdolshah, and S. Haddadin, "An mpc framework for planning safe and trustworthy robot motions," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 4737–4742.

[6] S. Kleff, A. Meduri, R. Budhiraja, N. Mansard, and L. Righetti, "High-Frequency Nonlinear Model Predictive Control of a Manipulator," in *IEEE International Conference on Robotics and Automation (ICRA 2021)*, Xi'an, China, May 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02993058

[7] R. Lober, O. Sigaud, and V. Padois, "Task feasibility maximization using model-free policy search and model-based whole-body control," *Frontiers in Robotics and AI*, vol. 7, 2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2020.00061

[8] A. S. Sathya, J. Gillis, G. Pipeleers, and J. Swevers, "Real-time robot arm motion planning and control with nonlinear model predictive control using augmented lagrangian on a first-order solver," in *2020 European Control Conference (ECC)*, 2020, pp. 507–512.

[9] D. Nicolis, F. Allevi, and P. Rocco, "Operational Space Model Predictive Sliding Mode Control for Redundant Manipulators," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1348–1355, Aug. 2020, conference Name: IEEE Transactions on Robotics.

[10] G. P. Incremona, A. Ferrara, and L. Magni, "Mpc for robot manipulators with integral sliding modes generation," *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 3, pp. 1299–1307, 2017.

[11] A. Saccon, J. Hauser, and A. P. Aguiar, "Optimal control on lie groups: The projection operator approach," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2230–2245, 2013.

[12] N. Torres Alberto, L. Joseph, V. Padois, and D. Daney, "A linearization method based on Lie algebra for pose estimation in a time horizon," in *ARK 2022 - 18th International Symposium on Advances in Robot Kinematics*, Bilbao, Spain, June 2022. [Online]. Available: https://hal.archives-ouvertes.fr/hal-03621688

[13] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration theory for fast and accurate visual-inertial navigation," *CoRR*, vol. abs/1512.02363, 2015. [Online]. Available: http://arxiv.org/abs/1512.02363

[14] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017, ch. 9, Trajectory Generation.

[15] 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

[16] S. Jayasumana, R. I. Hartley, M. Salzmann, H. Li, and M. T. Harandi, "Kernel methods on the riemannian manifold of symmetric positive definite matrices," *CoRR*, vol. abs/1412.4172, 2014. [Online]. Available: http://arxiv.org/abs/1412.4172

[17] L. Joseph, J. K. Pickard, V. Padois, and D. Daney, "Online velocity constraint adaptation for safe and efficient human-robot workspace sharing," in *International Conference on Intelligent Robots and Systems*, Las Vegas, United States, Oct. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02434905

[18] N. Mansfeld, B. Djellab, J. R. Veuthey, F. Beck, C. Ott, and S. Haddadin, "Improving the performance of biomechanically safe velocity control for redundant robots through reflected mass minimization," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5390–5397.

[19] L. Berscheid and T. Kröger, "Jerk-limited real-time trajectory generation with arbitrary target states," *Robotics: Science and Systems XVII*, 2021.

[20] 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. [Online]. Available: https://doi.org/10.1007/s12532-020-00179-2

[21] J. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpoases: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, 12 2014.

[22] J. Carpentier, G. Saurel, G. Buondonno, J. Mirabel, F. Lamiraux, O. Stasse, and N. Mansard, "The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives," in *IEEE International Symposium on System Integrations (SII)*, 2019.

[23] A. Skuric, V. Padois, and D. Daney, "On-line force capability evaluation based on efficient polytope vertex search," in *ICRA 2021 - IEEE International Conference on Robotics and Automation*, Xi'an, China, May 2021. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02993408