Non-Prehensile Object Transportation via Model Predictive Non-Sliding Manipulation Control

Mario Selvaggio, Akash Garg, Fabio Ruggiero, *Senior Member, IEEE*, Giuseppe Oriolo, *Fellow, IEEE*, and Bruno Siciliano, *Fellow, IEEE*

***Abstract*— This article proposes a model predictive non-sliding manipulation (MPNSM) control approach to safely transport an object on a tray-like end-effector of a robotic manipulator. For the considered non-prehensile transportation task to succeed, both non-sliding manipulation and the robotic system constraints must always be satisfied. To tackle this problem, we devise a model predictive controller enforcing sticking contacts, i.e., preventing sliding between the object and the tray, and assuring that physical limits such as extreme joint positions, velocities, and input torques are never exceeded. The combined dynamic model of the physical system, comprising the manipulator and the object in contact, is derived in a compact form. The associated non-sliding manipulation constraint is formulated such that the parametrized contact forces belong to a conservatively approximated friction cone space. This constraint is enforced by the proposed MPNSM controller, formulated as an optimal control problem that optimizes the objective of tracking the desired trajectory while always satisfying both manipulation and robotic system constraints. We validate our approach by showing extensive dynamic simulations using a torque-controlled 7-degree-of-freedom (DoF) KUKA LBR IIWA robotic manipulator. Finally, demonstrative results from real experiments conducted on a 21-DoF humanoid robotic platform are shown.**

***Index Terms*— Manipulators, robot control, robotics and automation, robots, service robots.**

I. Introduction

SERVICE robots are developed to assist human beings in performing tasks that are typically dull, dangerous, or repetitive. To date, they have been realized in different forms and structures and employed in various applications ranging from household and personal assistance to industrial collaboration [1], [2]. These robots usually operate semi-autonomously in human-centered environments and must satisfy multiple requirements. One of them consists in exhibiting compliant human-like manipulation skills. However, most robotic systems nowadays are still missing this essential feature and are equipped with simple, prehensile grippers, which are used to pick, and only limitedly manipulate, a relatively narrow variety of objects. The main problem with this solution is assuring that the grasp holds all the time, which requires it to resist all the forces that could reasonably act on the object during the manipulation tasks, without causing too high internal stresses [3], [4]. When this cannot be guaranteed, e.g., due to external interactions or highly dynamic movements, the object is likely to slip and fall from the fingers.

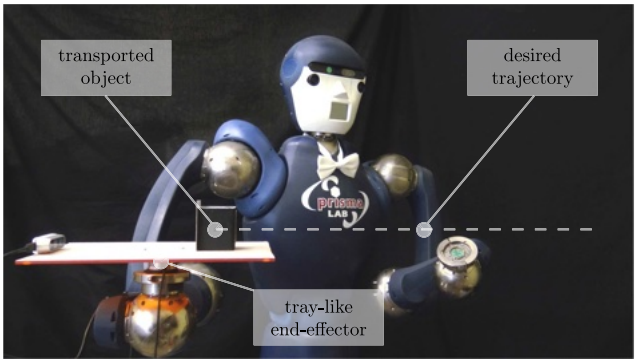

Fig. 1. Illustration of the main problem addressed in this article: a robotic manipulator (blue) has to transport an object (black cube) along a desired trajectory (dashed gray) on a tray-like end-effector (orange/white) while guaranteeing a sticking behavior, i.e., satisfying both non-prehensile manipulation (i.e., friction cones) and robotic system constraints such as joint limits, maximum torques, etc.

To extend the set of manipulative actions, manipulate objects of different size and shape, exhibit bigger operative workspace, and enhance the dexterity in dynamic tasks, non-prehensile robotic manipulation has recently emerged as a valid alternative to grasping manipulation [5]. Non-prehensile manipulation circumvent the problem of grasping and retaining an object by realizing manipulative actions that jointly exploiting frictional, gravity, and inertial forces such as pushing, throwing, and striking, which humans commonly employ to carry out everyday manipulation tasks. The success of non-prehensile manipulation has already been demonstrated along with several applications, mainly in industrial-like settings, for instance to re-orient parts on a planar surface [6]. The main advantage resides in using simpler robotic end-effectors, purposely designed to robustly perform manipulation tasks with a wider variety of objects. However, this comes at the expense of developing and endowing robots with control methods that rely on accurate and reliable models of the

Manuscript received 30 June 2022; revised 14 February 2023; accepted 13 April 2023. Date of publication 30 June 2023; date of current version 18 August 2023. This work was supported by the European Union's Horizon 2020 Research and Innovation Program under Grant 101017008 (Harmony). Recommended by Associate Editor M. Cannon. *(Corresponding author: Mario Selvaggio.)*

Mario Selvaggio, Fabio Ruggiero, and Bruno Siciliano are with the PRISMA Laboratory, Department of Electrical Engineering and Information Technology, University of Naples Federico II, 80125 Naples, Italy (e-mail: mario.selvaggio@unina.it; fabio.ruggiero@unina.it; bruno.siciliano@unina.it).

Akash Garg and Giuseppe Oriolo are with the Department of Information, Automation and Management Engineering (DIAG), Sapienza University of Rome, 00185 Rome, Italy (e-mail: garg.1892171@studenti.uniroma1.it; oriolo@diag.uniroma1.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCST.2023.3277224.

Digital Object Identifier 10.1109/TCST.2023.3277224

mechanics involved. For this reason, to date few robots have been shown capable of non-prehensile manipulation skills which, in principle, would enable them to perform a broader range of dexterous manipulation tasks, the simplest one being transporting an object along a trajectory.

In this article, we consider the problem of non-prehensile transportation of an object along a desired trajectory using a tray-like end-effector. In the considered setting, it is in general not possible to prevent any induced motion of the object relative to the hand. The goal is to satisfy both manipulation and robotic system constraints along the performed trajectory, i.e., preventing the object to slide and fall. Solving this problem would enable service robots to perform dexterous, waiter-like object transportation tasks, which can be exploited to, e.g., serve lunch on a tray to patients in a hospital. A robotic manipulation system composed of a serial robotic arm endowed with a tray-like end-effector is used to carry out the considered task (see Fig. 1). Such a system exhibits several robotic system constraints (such as limited range of joint motion, maximum feasible torques, etc.) that must always be satisfied together with non-sliding manipulation constraints (i.e., friction cones) that prevent sliding of the object and, thus, task failures. From a technical viewpoint, carrying a payload modifies the robot's dynamics which, in turn, must not only counteract but also opportunely regulate its motion to prevent sliding and avoid violating robotic system constraints.

Thus, we propose a model predictive non-sliding manipulation (MPNSM) control architecture for a robot arm transporting an object on a tray-like end-effector in a non-prehensile configuration. Our MPNSM controller allows the realization of the considered task encoded as a desired trajectory which must be tracked while taking into account both manipulation and inherent robot constraints in a unified and principled way. This is mainly achieved by realizing contact forces between the hand and the object that satisfy friction cone constraints, thus enforcing a non-sliding behavior. In addition, to guarantee a continuous execution of the task, we consider the rate-of-change of the joint torques as output of our controller (in literature, this is referred to as jerk control [7]) which, once integrated over time, returns the overall input of our robotic system. This procedure makes the obtained joint torque profile (and thus the system accelerations) continuous and allows integrating force signals stemming from the interaction as feedback into the controller [8]. We derive and incorporate the combined system (manipulator and object) dynamical model and its related constraints into a nonlinear model predictive control (NMPC) problem [9]. Its solution returns the optimal control inputs relying on the most recent measurement/estimation of the system's state and accounting for its evolution over a future time horizon. This is the first work addressing the tray-based non-sliding manipulation problem from this perspective to the authors' knowledge.

II. Related Works

When a robot does not firmly hold an object, there exist motions induced by inertial or external forces that can not always be inhibited [10]. In that case, known as non-prehensile grasp, the object can still be manipulate typically employing a sequence of non-prehensile manipulation primitives [5], [11]. These include throwing [12], catching [13], batting [14], pushing [15], rolling [16], among others. We restrict the overview of related works to the so-called non-sliding non-prehensile manipulation primitive (sometimes also denoted as *dynamic grasping* [17]) which aims to immobilize the object to the end-effector (as it was firmly grasped) by exploiting the combined action of inertial, gravity, and frictional forces. This is complementary to the sliding non-prehensile manipulation primitive, which aims to realize controlled object motions by exploiting the same system of forces [18], [19]. In the past, several methodologies were devised for robots equipped with flat palm end-effectors to carry out non-prehensile manipulation tasks [20], [21]. In the following, a few examples are reported.

A motion planning framework that explicitly considers reaction and friction forces as kino-dynamic constraints for the non-prehensile transportation of a bottle is proposed in [22]. A task priority control scheme featuring sliding mode and admittance control for human-robot collaborative transportation of an object on a tray is designed in [23]: a human operator guides a 7-degree-of-freedom (DoF) robot arm through a force sensor located at the robot tool. A method to change the inclination of the tray when the object is rotating around an edge based on the computation of the zero-moment-point of the object, modeled as an inverted pendulum, is proposed in [24] for a waiter humanoid robot that transports objects on a tray. A framework that offline evaluates and eventually re-plans a trajectory based on the occurrence of slippage due to inertial forces that occur on a grasped object during highly dynamic robot tasks is presented in [25]. A shared-control teleoperation architecture exploiting similar concepts has been designed to safely transport an object placed on a tray-like end-effector preventing its sliding in [26].

Several recent papers have focused instead on the problem of executing robotic manipulation and, in general, interaction tasks using model predictive control (MPC). To date, the common trend is to integrate contact forces tracking as an objective and feedback their measure in the controller [8]. For instance, a whole-body MPC for dynamically stabilizing a mobile manipulator while executing end-effector pose tracking tasks while skillfully planning for end-effector contact forces is devised in [27]. The controller directly computes the actuation torques and the forces exerted on the environment. However, since the authors do not model the environment, the interaction task is accomplished as a force regulation problem, and it is successful only under the circumstance that the environment impedance is underestimated. A MPC strategy that is aware of its environment and can plan whole-body motion for a mobile manipulator while avoiding collisions is proposed in [28]. Through task-space admittance control, it can track any desired interaction forces and torques. System identification and an adaptive control method extend the MPC formulation presented in [27] to deal with mobile manipulation tasks in unknown environments in [29]. The employed modeling strategy was derived under the assumption that the environment can be described by a linear mass-spring-damper system rigidly attached to the robot. In the context of

non-prehensile manipulation, MPC has been used to perform pushing tasks requiring different levels of accuracy [30].

However, none of the above works has jointly considered the robotic system and the manipulation constraints in a unified model predictive non-sliding control approach to perform non-prehensile object transportation tasks.

A. Contributions

To fill the above-mentioned gap in the literature, this article introduces an MPNSM control approach to perform non-prehensile object transportation tasks while jointly satisfying manipulation and robotic system constraints. In the following, the main technical contributions of this work with respect to the previous literature are summarized.

1) In our previous work on the topic (see [26]), an optimization-based architecture for transporting an object on a tray was preliminarily introduced. However, only manipulation constraints were formulated and included in the main optimization problem. The controller may thus require inputs that actuators cannot provide and/or the robotic system may exceed its extreme physical limits. In this work, we address this problem, extending our previous architecture by integrating robotic system constraints and receding horizon control capabilities.
2) The work [7] uses the time derivative of input torques to control a humanoid robot at the jerk level. Building upon this approach, our work proposes and extensively validates the use of jerk control within a MPC framework for non-prehensile object manipulation tasks.
3) We derived analytical models for the computation of (parametrized) contact forces and their time evolution under mild assumptions. These are used within the system dynamics constraint of the devised MPNSM control approach. Besides this, we mathematically proved the absence of internal force terms within non-prehensile manipulation setup, such as the one considered in this article.
4) Finally, we release the simulation code used to demonstrate the performance of the devised controller to the community for a possible future benchmark.

III. System Modeling

In this section, the system and the contact models, which will be employed to implement the devised controller in Section V, are introduced. The combined manipulator-object dynamic model is derived in a compact form in Section III-A while contact models and the parametrization of the contact forces are dealt with in Section III-B.

A. Combined Manipulator-Object Dynamics

Let us consider a serial robot manipulator whose state can be uniquely described through the pair $(q, \dot{q})$ with $q \in \mathbb{R}^n$ being the vector of generalized coordinates (n denotes the joints number) and $\dot{q} \in \mathbb{R}^n$ its time derivative (joint velocities). The dynamics of the manipulator can be expressed by the following equation of motion:

$$M_m(q)\ddot{q} + C_m(q, \dot{q})\dot{q} + n_m(q) = \tau - \tau_{\text{ext}} \tag{1}$$

where $M_m(q) \in \mathbb{R}^{n\times n}$ is the symmetric positive-definite robot joint-space inertia matrix, $C_m(q, \dot{q}) \in \mathbb{R}^{n\times n}$ is the matrix of centrifugal/Coriolis terms, $n_m(q) \in \mathbb{R}^n$ is the gravity vector, $\tau \in \mathbb{R}^n$ is the vector of joint torques (representing the overall control input of the robotic system), and $\tau_{\text{ext}} \in \mathbb{R}^n$ is the joint torque vector corresponding to an external load. In the envisioned scenario, the external load is attributable to the presence of an object to be transported, which is in contact with the robot's end-effector. The object is assumed to be a rigid body whose dynamics can be expressed as

$$M_o(x_o)\dot{\mathcal{V}} + C_o(x_o, \mathcal{V})\mathcal{V} + n_o(x_o) = \mathcal{F}_o \tag{2}$$

where $x_o = (p, \phi)$ is the object pose composed by the position $p \in \mathbb{R}^3$ and parametrized orientation $\phi \in \mathbb{R}^3$ or $\mathbb{R}^4$ (e.g., ZYX Euler angles, unit quaternion, etc.), $M_o(x_o) \in \mathbb{R}^{6\times6}$ is the object positive-definite mass/inertia matrix, $C_o(q, \dot{q}) \in \mathbb{R}^{6\times6}$ is the matrix of centrifugal/Coriolis terms, $n_o(q) \in \mathbb{R}^6$ is the gravity vector, $\mathcal{V} = (v, \omega) \in \mathbb{R}^6$ is the object twist, with $v \in \mathbb{R}^3$, $\omega \in \mathbb{R}^3$ expressing its linear and angular velocities, respectively; and $\mathcal{F}_o = (f_o, \tau_o) \in \mathbb{R}^6$ is the object wrench, with $f_o, \tau_o \in \mathbb{R}^3$ force and torque vectors, respectively, all specified with respect to the body reference frame $\{\mathcal{O}\}$ whose origin is placed at the object's center of mass.

As long as the contact between the object and the tray is maintained,[1] the body wrench $\mathcal{F}_o$ can be transformed into the corresponding manipulator torques τ_{ext} through the equation

$$\tau_{\text{ext}} = J_o^T(q)\mathcal{F}_o \tag{3}$$

where $J_o \in \mathbb{R}^{6\times n}$ is the object geometric Jacobian matrix [31]. When the same condition holds, the object parametrized pose can be retrieved from joint values using the forward kinematic function, i.e., $x_o = \kappa(q)$. Substituting (2) and (3) into (1), and using the following differential kinematics equations:

$$\begin{aligned} \mathcal{V} &= J_o(q)\dot{q} \\ \dot{\mathcal{V}} &= J_o(q)\ddot{q} + \dot{J}_o(q, \dot{q})\dot{q} \end{aligned} \tag{4}$$

leads to the following combined manipulator-object dynamic model:

$$\tilde{M}(q)\ddot{q} + \tilde{C}(q, \dot{q})\dot{q} + \tilde{n}(q, \dot{q}) = \tau \tag{5}$$

where $\tilde{M} \in \mathbb{R}^{6\times6}$ is the symmetric and positive-definite mass/inertia matrix, $\tilde{C} \in \mathbb{R}^{6\times6}$ the Coriolis-centrifugal matrix and $\tilde{n} \in \mathbb{R}^6$ the gravitational force of the combined system that can be written as follows (dropping their arguments):

$$\begin{aligned} \tilde{M} &= M_m + J_o^{\mathrm{T}} M_o J_o \\ \tilde{C} &= C_m + J_o^{\mathrm{T}}(C_o J_o + M_o \dot{J}_o) \\ \tilde{n} &= n_m + J_o^{\mathrm{T}} n_o. \end{aligned} \tag{6}$$

As stated above, τ on the right-hand side of (5) represents the vector of joint torques that are the overall control input

[1] It is worth noting that this is not an assumption: it is instead a condition that will later be formulated as a constraint and enforced by the devised optimization-based model predictive controller.

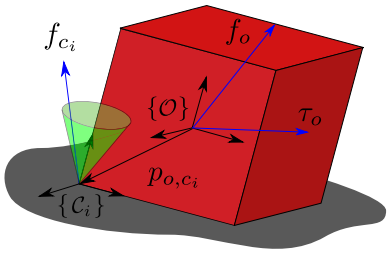

Fig. 2. Drawing of the object body wrench $\mathcal{F}_o = (f_o, \tau_o)$, i-th contact force f_{c_i} and friction cone (shaded green). p_{o,c_i} is the vector defining the i-th contact position in the body frame $\{\mathcal{O}\}$.

of the robotic system manipulating the object. However, as τ contributes to the realization of the object body wrench $\mathcal{F}_o$ through contact forces transmitted to the object (as shown later), its choice must respect manipulation constraints that prevent the object from sliding. Moreover, τ must also be chosen to satisfy the inherent robotic system constraints, i.e., it must lay within a specific range and generate the evolution of the manipulator states that is compatible with the allowable joint range of motion/velocity. Thus, the primary objective of this work is to find τ that satisfies both robotic system and non-sliding manipulation constraints, which are better described in Section III-B.

It is worth noting that, inverting (3), the combined dynamic model (5) can be alternatively derived in the object coordinates, where manipulation constraints (see later) are more easily handled. However, the approach followed here allows to more conveniently treat the robotic system's physical limits as state constraints and directly calculate the overall system's control inputs, i.e., the manipulator torques.

B. Contact Model and Contact Forces Parametrization

Let us consider the model of the object dynamics in (2). The object body wrench $\mathcal{F}_o$ can be realized by opportunely generating contact forces F_c between the object and its manipulandum. Before explicitly defining F_c, we must introduce some modeling assumptions. We assume that the object's shape and dynamical properties are known and coincident with a cuboid of known material. Moreover, we assume that the overall contact surface between the cuboid object and the tray can be approximated by discretizing it, i.e., using a finite number of n_c contact points located in the vertexes of the object in contact (thus $n_c = 4$ in the considered case). The i-th contact point is thus identified by a contact frame $\{\mathcal{C}_i\}$ whose pose is known and expressed in $\{\mathcal{O}\}$ by $q_{o,c_i} = (p_{o,c_i}, R_{o,c_i}) \in \mathrm{SE}(3)$.

The tray/object interaction behavior can be described introducing a suitable contact model. In general, the set of wrenches that can be transmitted across the i-th contact is described by a wrench basis $B_{c,i} \in \mathbb{R}^{6\times m_i}$, where m_i denotes the dimension of the generalized forces at the contact. $B_{c,i}$ maps the components of the contact forces, which are transmissible through the contact point, into the 6-D wrench space. Assuming a *point contact with friction* model [32], only the linear forces $f_{c_i} \in \mathbb{R}^3$ can be transmitted through the i-th contact, thus $m_i = 3$.

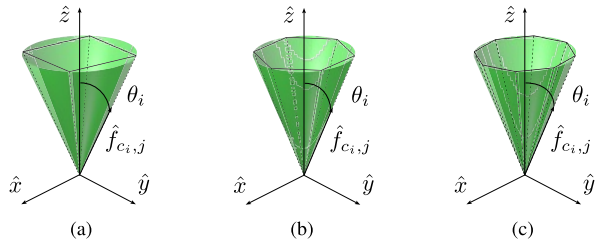

Fig. 3. Approximation of circular friction cones (in shaded green) with polyhedral cones (black lines) with different number of edges k. (a) $k = 4$. (b) $k = 6$. (c) $k = 8$.

The body wrench $\mathcal{F}_o$ can thus be expressed as

$$\mathcal{F}_o = \mathrm{G}F_c, \quad G = \left[\mathrm{Ad}^T_{q^{-1}_{o,c_1}} B_{c,1}, \ldots, \mathrm{Ad}^T_{q^{-1}_{o,c_{n_c}}} B_{c,n_c}\right] \tag{7}$$

where $G \in \mathbb{R}^{6\times 3n_c}$, usually referred to as *grasp matrix* in the robotic grasping literature, maps the stacked vector of contact forces $F_c = [f^T_{c_1}, \ldots, f^T_{c_{n_c}}]^T \in \mathbb{R}^{3n_c}$ to the body wrench $\mathcal{F}_o$ exerted at the object's center of mass (see Fig. 2). The matrices involved in the calculation of G in (7) can be expressed as ([33])

$$\mathrm{Ad}^T_{q^{-1}_{o,c_i}} = \begin{bmatrix} R_{o,c_i} & 0 \\ \hat{p}_{o,c_i} R_{c_i,o} & R_{o,c_i} \end{bmatrix}, \quad B_{c,i} = \begin{bmatrix} I_{3\times 3} \\ 0_{3\times 3} \end{bmatrix} \tag{8}$$

where $\hat{p}_{o,c_i} \in \mathrm{so}(3)$ denotes the skew-symmetric matrix associated with the vector $p_{o,c_i} \in \mathbb{R}^3$ (i.e., the position of the i-th contact point expressed in $\{\mathcal{O}\}$).

The non-sliding manipulation constraint coincides with the contact forces belonging to the *friction cone* space. This means that, in the i-th contact, the three components $f_{c_i,x}$, $f_{c_i,y}$, $f_{c_i,z}$ of the contact force $f_{c_i} \in \mathbb{R}^3$ must satisfy the constraint

$$\sqrt{f^2_{c_i,x} + f^2_{c_i,y}} \le \mu f_{c_i,z}, \quad f_{c_i,z} \ge 0 \tag{9}$$

where $\mu > 0$ is the static friction cone coefficient, assumed to be known and uniform for all the contacts. Whenever the constraint (9) is satisfied for all the contacts,[2] the object can be manipulated, that is, it is transported along the desired trajectory while preventing it from sliding on the tray.

To enforce this constraint in the controller, it is convenient to parametrize the contact forces and make the constraint linear in the chosen parameters. This can be achieved by conservatively approximating the i-th friction cone as a polyhedron generated by a finite number $k \in \mathbb{N}$ of unit vectors $\hat{f}_{c_i,1}, \ldots, \hat{f}_{c_i,k} \in \mathbb{R}^3$. When $k = 4$, the circular friction cone is conservatively approximated by an inscribed pyramid ([33]). In any case, to relax the conservativeness of this approach, tighter approximations of the circular cone can be obtained using more edges, i.e., increasing k (see Fig. 3).

The procedure for the calculation of the j-th friction cone edge for a generic k is given by

$$\hat{f}_{c_i,j} = R_z(2\pi j/k) R_y(\theta) \hat{z} \tag{10}$$

where $\theta = \arctan \mu$ and it denotes the pyramid (or cone) semi-aperture angle. However, it is worth noting that the size, and thus the computational burden, of the optimal controller presented later will increase with k. With this choice, F_c can be conveniently parametrized, i.e., it can be written as

$$F_c = \hat{F}_c \Lambda, \quad \Lambda = \left(\lambda_{c_1,1}, \ldots, \lambda_{c_{n_c},k}\right) \in \mathbb{R}^{kn_c} \tag{11}$$

[2]Note that this is a conservative condition we use throughout this article.

where

$$\hat{F}_c = \text{diag}\big(\hat{F}_{c,1}, \ldots, \hat{F}_{c,n_c}\big), \quad \hat{F}_{c,i} = \big[\hat{f}_{c_i,1}, \ldots, \hat{f}_{c_i,k}\big] \quad (12)$$

is the matrix describing the friction cone space geometry, as its columns contain the vectors of the friction cone edges as calculated in (10), while Λ denotes the vector of contact force parameters, that constitute the components along the friction cone borders. At this point, for F_c to belong to the approximated friction cone space, it is sufficient to choose

$$\Lambda_i \geq 0 \quad \forall\, i = 1, \ldots, kn_c \quad (13)$$

meaning that F_c must be a non-negative linear combination of the friction cone boundaries through the vector of the coefficients Λ. This constraint will be enforced by the controller described in Section V.

IV. Contact Forces and Parameters Computation

The quantity that more effectively describes the interaction state is the contact force. However, measuring forces at the contact is generally complex, especially for extended contact geometries. Since our goal is to enforce the friction cone constraints mentioned earlier, in this section, we introduce the contact force (and the related parameters) calculation procedure used throughout this work.

A. Contact Forces Computation

Considering (7), for a given $\mathcal{F}_o$ (usually directly or indirectly measurable), the considered system of equations must be solved for F_c to retrieve contact forces. The expression in (7) denotes a system of six equations in $m_i \times n_c$ unknowns. In the considered case ($m_i = 3$ $n_c = 4$), the solution in terms of F_c is thus indeterminate. Anyway, it is possible to solve and derive an expression for F_c, solving the following optimization problem:

$$\min_{F_c} \|F_c\|^2 \quad (14a)$$

$$\text{s.t. } \text{GF}_c = \mathcal{F}_o. \quad (14b)$$

Solving (14) allows finding, among many, the solution vector having the minimum two-norm. It is worth to note that the constraint (13), which accounts for the contact forces feasibility, is not considered in (14). When the constraint (13) is not binding, it is possible to derive a closed-form solution for F_c applying the method of Lagrange multipliers. This leads to

$$F_c = G^{\dagger}\mathcal{F}_o \quad (15)$$

where $\dagger$ denotes the Moore–Penrose inverse operator, that constitutes the minimum two-norm result of the contact forces vector F_c. Of course, as previously stated, this calculation procedure generates valid results as long as the object is stationary to the tray, i.e., F_c belongs to the friction cone space that, in turn, means (13) is satisfied. Indeed, contact forces that do not belong to the friction cone space, i.e., lead to at least one $\Lambda_i < 0$, are impossible to be physically realized.

However, from the mathematical point of view, (15) is not the unique solution of the original system of equations in (7): in principle, other solutions can potentially be considered. For instance, the solution

$$F_c = G^{\dagger}\mathcal{F}_o + PF_{c,0} \quad (16)$$

where $P = P^T \in \mathbb{R}^{3n_c \times 3n_c} \geq 0$ is the projector onto the null-space of G, i.e., $GP = 0$, and $F_{c,0}$ is a generic vector, also satisfies (7). It can easily be shown that this solution can be obtained by modifying the cost function in the previously-introduced optimization problem (14).

In the following, we show that contact forces components lying in the null space of the matrix G must not be considered since they are representative of internal forces that feasible contact points displacements can not generate in non-prehensile manipulation settings. Indeed, for contact forces to be realized, contact points displacements should be generated through the manipulator's actuation system according to the following relation:

$$\delta F_c = KJ(q)\delta q \quad (17)$$

where the contact points Jacobian matrix $J(q) \in \mathbb{R}^{3n_c \times n}$ relates the contact point velocities to actuators' velocities, i.e., $v_c = J(q)\dot{q}$, and $K \in \mathbb{R}^{3n_c \times 3n_c}$ is a diagonal positive-definite matrix, that plays the role of stiffness and relates contact point displacements to contact forces variation, i.e., $\delta F_c = K\delta p_c$ (see [34] for further details). This shows that all the contact forces that can be generated/controlled by the contact points displacements are in the range space of the $J(q)$ matrix. Neglecting for the moment the presence of the manipulator, i.e., considering that our tray can be actuated by linear and rotational rigid body displacements δq in the Cartesian space, we can construct $J(q) = J$ geometrically as the following constant matrix (see Fig. 2):

$$J(q) = J = \begin{bmatrix} \vdots & \vdots \\ R_{o,c_i} & \hat{p}_{o,c_i} R_{o,c_i} \\ \vdots & \vdots \end{bmatrix}.$$

The fact that J is constant is a consequence that the tray can only undergo rigid body motions. Consequently, the set of contact points can only rigidly translate and rotate in space but not "deform" (i.e., they cannot modify their relative distances). Considering now the product between columns of the null-space projector P and J, that appears substituting (17) in the second term on the right-hand side of (16), and observing that $J = G^T$, we get

$$P^T J = P^T G^T = O$$

where O is the null matrix of appropriate dimensions. This result shows that no contact forces lying in the null space of the G matrix can be generated/actuated by the tray/contact points displacements. Alternatively, one may observe that columns of the null space projector P are linearly independent of the columns of J. Contrarily, the term $G^{\dagger}\mathcal{F}_o$ is the component of the solution that is in the range space of J, i.e., such that $\text{rank}([J|G^{\dagger}\mathcal{F}_o]) = \text{rank}(J) = 6$, denoting that it is the only one that can be theoretically generated. It is worth remarking that the practical realization of contact forces depends on the

satisfaction of the frictional constraints, i.e., (13) needs to be satisfied.

B. Contact Forces Parameters Computation

Substituting (11) into (14) allows solving for Λ directly and enforcing the non-sliding condition (13). In the considered case, the expression $\mathcal{F}_o = G\hat{F}_c\Lambda$ constitutes a system of six equations in $k \times n_c$ unknowns, and the solution in terms of Λ is again indeterminate in the considered case ($k = 4$ and $n_c = 4$). Following similar arguments, it is possible to solve and derive an expression for Λ, solving a two-norm minimization problem analogously to what done for F_c in Section IV-A

$$\min_{\Lambda} \|\Lambda\|^2 \tag{18a}$$

$$\text{s.t. } \mathrm{G}\hat{\mathrm{F}}_c\Lambda = \mathcal{F}_o \tag{18b}$$

$$\Lambda_i \geq 0 \quad \forall\, i = 1, \ldots, kn_c \tag{18c}$$

where (13) has been introduced in (18c). When the constraint (18c) is not binding, a closed-form solution can be derived applying the method of Lagrange multipliers and, after exploiting (15), it can be written as follows:

$$\Lambda = \left(\mathrm{G}\hat{\mathrm{F}}_c\right)^{\dagger}\mathcal{F}_o \tag{19}$$

where $\dagger$ denotes the Moore–Penrose inverse operator and is used to minimize the two-norm of the contact forces coefficients. It is worth remarking that our goal is to enforce the constraint in (13), while acting on the manipulator torque control inputs, and (19) is essential to derive the relation between these two quantities. The expression (19), and in particular its time derivative, tells us how the contact forces parameters vary when the object wrench varies, and it is included in the dynamics constraint of our MPNSM controller, which is introduced in Section V.

As mentioned above, $\mathcal{F}_o$ can usually be directly or indirectly measured by a force/torque (F/T) sensor. However, in torque-controlled manipulators, under the assumption of fast motor dynamics, a more practical way to obtain $\mathcal{F}_o$, to be used in (19), is to use the commanded torques τ to obtain $\dot{\mathcal{V}}$ using (4), where $\ddot{q}$ can be obtained through (1) using the measured τ_{ext}. Modern manipulators are indeed equipped with external torque sensing. In both ways, body force (reconstructed) measurements and the contact forces computed from them are feedback to the controller that uses these to compute the subsequent optimal control inputs.

V. Model Predictive Non-Sliding Manipulation Control

In this section, we derive the optimization-based MPNSM controller leveraging the NMPC approach. In general, this approach aims to find the optimal sequence of control inputs and the corresponding state trajectory over the finite-length prediction horizon, subject to constraints on the state trajectory and the control inputs. Model predictive controllers account for the model of the system to be controlled and find a solution starting from the current state on each iteration. The first timestep of the computed control trajectory is applied before the controller runs again, and new control inputs are computed. Denoting with x the system state and u the control input, the underlying discrete-time optimal control problem (OCP) we aim to solve has the following form:

$$\min_{x,u} \|x_e^* - x_e\|_{Q_e}^2 + \sum_{i=0}^{N-1} \|x_{i+1}^* - x_{i+1}\|_{Q_i}^2 + \|u_i\|_{R_i}^2 \tag{20a}$$

$$\text{s.t. } x_0 = \bar{x}(0) \tag{20b}$$

$$x_{i+1} = f_k(x_i, u_i) \tag{20c}$$

$$\underline{x} \leq x_i \leq \bar{x} \tag{20d}$$

$$\underline{u} \leq u_i \leq \bar{u} \tag{20e}$$

with $N \in \mathbb{R}_{>0}$ indicating the steps of the prediction horizon and where the function (20a) denotes a least square cost function to be minimized, composed of two weighted two-norms: the state difference from the desired values x^* and the input u. Here, $\|v\|_A = v^T A v$ is the quadratic form with a suitable weighting matrix, while $Q_i, Q_e \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_x}$ and $R_i \in \mathbb{R}^{n_u} \times \mathbb{R}^{n_u}$ denote the diagonal entries of the corresponding positive semi-definite weight matrices.

To account for the contact state of the object/tray interaction, we opted for defining an *extended system state* including, besides the proper state q and $\dot{q}$, the manipulator input torques τ and the contact force coefficients Λ. In this way, we can more easily feedback both the manipulator torques and the contact forces into the controller via (20b). Our extended state vector thus reads as $x = (\tau^T, q^T, \dot{q}^T, \Lambda^T)^T$. With this choice, we increase the relative degree of our system dynamics and choose as a control input $u = \dot{\tau}$. This is convenient since it gives rise in any case to a continuous torque profile, which constitutes the real input to our robotic system. The matrix Q in (20) can thus be partitioned into Q_τ, Q_q, $Q_{\dot{q}}$, Q_Λ of opportune dimensions, where the generic Q_χ block corresponding to the related χ quantity.

It is worth mentioning that our ultimate goal is to transport the object to the target pose following the desired trajectory (see Fig. 1), i.e., we aim to realize $x_o = x_o^*(t)$ and $\dot{x}_o = \dot{x}_o^*(t)$, where $x_o^*(t), \dot{x}_o^*(t)$ are the desired object states while satisfying both non-sliding manipulation and robotic system constraints. From this, we can calculate the reference values for the extended state x^*, in particular $q_o^*, \dot{q}_o^*$ using a standard inverse kinematics routine, under the assumption that the object is rigidly attached to the manipulator. For this, we used the closed-loop inverse kinematic (CLIK) algorithm, i.e., a Jacobian pseudo-inverse-based Newton–Raphson method [31].

With the state/input choice made, the continuous-time dynamic evolution of the extended state, which is ultimately used in (20c), can be thus written as

$$\dot{x} = f(x,u) = \begin{cases} \dot{\tau} = u \\ \dot{q} = \dot{q} \\ \ddot{q} = \tilde{M}^{-1}(x)(\tau - \tilde{C}(x)\dot{q} - \tilde{n}(x)) \\ \dot{\Lambda} = \left(G\hat{F}_c\right)^{\dagger}\left(A\dot{\tau} + B\tau + C\right) \end{cases} \tag{21}$$

where the matrices A, B, and C write as follows:

$$A = M_o J \tilde{M}^{-1}, \quad B = M_o \dot{J} \tilde{M}^{-1}, \quad C = -2M_o \dot{J} \tilde{M}^{-1}\tilde{n}. \tag{22}$$

Equation (21) describes the dynamic evolution of the combined manipulator-object system and of parametrized contact forces between them. It is worth mentioning that the

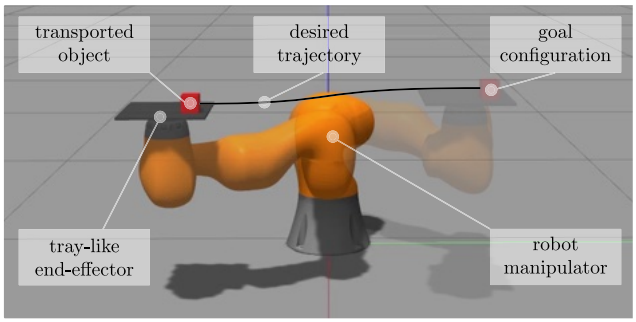

Fig. 4. Dynamic simulation scenario: a torque-controlled robotic manipulator (orange) has to transport an object (red cube) along a desired trajectory (black) on a tray-like end-effector (dark gray) while guaranteeing sticking behavior, i.e., satisfying both (non-sliding) non-prehensile manipulation and robotic system constraints such as joint limits, maximum torques, etc.

TABLE I

SIMULATION MPNSM CONTROL PARAMETERS

Parameter	Value	Parameter	Value
Q_τ	$1e^{-2}$	Q_e	$1e^{-3}$Q
Q_q	$1e^{7}$	R	$1e^{-4}$
$Q_{\dot{q}}$	$1e^{5}$	m_o	0.5
Q_Λ	1	I	1e−4

TABLE II

SIMULATED ROBOTIC SYSTEM JOINT LIMITS

Joint #	1	2	3	4	5	6	7
q [deg]	±170	±120	±170	±120	±170	±120	±170
$\dot{q}$ [deg/s]	±98	±98	±100	±98	±140	±180	±180
τ [Nm]	±176	±176	±110	±110	±110	±40	±40

expression for $\dot{\Lambda}$ has been retrieved by differentiating (19), substituting (2) and (5), under the assumptions that all the matrices entering the dynamic model hold constant over the time horizon.

As for the constraints, (20b) denotes the feedback term read from the system at each time step. Note that while the measure of $q, \dot{q}, \tau$ is readily available in torque-controlled manipulators, the measure of Λ or F_c must be retrieved indirectly from the measure of $\mathcal{F}_o$ (which can be conveniently measured through a F/T sensor installed at the end-effector of the manipulator) and using (19). The constraint on the system dynamics (20c) is constructed discretizing (21). We denote with $\underline{q} = (\underline{q}_1, \ldots, \underline{q}_n)$, $\underline{\dot{q}} = (\underline{\dot{q}}_1, \ldots, \underline{\dot{q}}_n)$, $\underline{\tau} = (\underline{\tau}_1, \ldots, \underline{\tau}_n)$, $\underline{\Lambda} = (\underline{\Lambda}_1, \ldots, \underline{\Lambda}_{n_c,k})$ the lower bounds on joint positions, joint velocities, joint torques, and contact force coefficients, respectively. Similarly $\bar{q} = (\bar{q}_1, \ldots, \bar{q}_n)$, $\bar{\dot{q}} = (\bar{\dot{q}}_1, \ldots, \bar{\dot{q}}_n)$, $\bar{\tau} = (\bar{\tau}_1, \ldots, \bar{\tau}_n)$, and $\bar{\Lambda} = (\bar{\lambda}_1, \ldots, \bar{\lambda}_{n_c,k})$ denote the upper bounds on the respective quantities. Thus, in (20d), $\underline{x} = (\underline{q}, \underline{\dot{q}}, \underline{\tau}, \underline{\Lambda})^T$ and $\bar{x} = (\bar{q}, \bar{\dot{q}}, \bar{\tau}, \bar{\Lambda})^T$ represent the lower and the upper bounds on the extended system states. These include both the physical bounds of the robotic system and the non-sliding manipulation constraints. Similarly, the control inputs to the system are bounded with lower bound, $\underline{u} = (\underline{u}_1, \ldots, \underline{u}_n)^T$, and upper bound, $\bar{u} = (\bar{u}_1, \ldots, \bar{u}_n)^T$, which are free to be chosen.

VI. DYNAMIC SIMULATION SETUP

The proposed controller is implemented in C++ using the ROS middleware. The simulations are performed using the physics-based Gazebo dynamic simulator. We considered a robotic system that consists of a 7-DoF KUKA LBR IIWA manipulator equipped with a purposely designed tray-like end-effector and a cube-shaped object (see Fig. 4). The object of dimension 40 × 40 × 40 mm is placed in contact with the top of the tray attached to the manipulator flange. In order to keep the simulation setup as realistic as possible, we assign the dynamic properties of the KUKA LBR IIWA manipulator within the simulation environment, corresponding to those identified for the real robotic manipulator [35]. The robotic system joint limits are given in Table II. The properties associated with the object have been chosen as follows: mass $m_o = 0.5$ kg and diagonal inertial matrix, $I_o = \text{diag}(1e^{-4})$ kgm^2. However, our controller is conceived for the non-prehensile transportation of a cuboid object with general dynamic parameters using a robotic manipulator. The Coulomb static friction coefficient between the tray and the object was chosen as $\mu = 0.5$. During the simulations, we placed the object at $x = -0.04$ m, $y = 0.05$ m from the tray center.

The desired trajectory for the state of the object, $x_o^*(t), \dot{x}_o^*(t)$, is obtained using a quintic polynomial with rest-to-rest object velocity and acceleration. The beginning of the desired trajectory coincides with the initial position of the object's geometric center, that is $p_o(0) = \begin{bmatrix}0.59 & -0.31 & 0.52\end{bmatrix}^T$ m in the robot base frame. Unless otherwise stated, the desired trajectory completion time is set $T = 1.5$ s and the final desired position of the object is $p_o(T) = \begin{bmatrix}0.59 & 0.31 & 0.52\end{bmatrix}^T$ m. The object is, thus, required to cover a distance of 0.62 m along the $\hat{y}$-axis of the robot base frame. During the simulation, the manipulator is controlled at the joint torque level to track the desired trajectory. The solution of the problem in (20) (i.e., the rate of change of the joint torques) is used to obtain the required updated torque at each simulation step i according to the following update rule:

$$\tau_{i+1} = \tau_i + \dot{\tau}_i \Delta t \tag{23}$$

with Δt being the control loop cycle time. In the MPNSM controller, we close the feedback loop of the joint values, joint velocities, joint torques and contact forces by reading the entire extended system state from the simulation environment. It is worth noting that the contact forces cannot be directly measured in the real case instead. In Section VII, we will retrieve them by plugging the reading of the body wrench measurements into (19).

The controller has been implemented using `acados` [36] which is a high-performance software package for nonlinear optimization problems. The package inherently uses `CasADi` [37] to formulate nonlinear functions as a front end. The controller was first implemented in `MATLAB`, which was later used to generate the C/C++ code library of our OCP. The generated library was then included within our software framework written in C++. We used the sequential quadratic programming (SQP) approach, where the quadratic programs (QPs) resulting from the approximation of the NMPC problem (20) are solved sequentially to obtain Newton directions leading to the solution starting from the provided guess. To obtain faster convergence of the NMPC problem, we relied on the real-time iteration (RTI) scheme of the SQP method [38]. The RTI scheme exploits the fact that the OCPs

obtained at two consecutive time instants are closely related, and thus it performs only one complete Newton step per sampling time. In order to provide a reasonable guess at time instant i, we employ a warm-started SQP that utilizes the shifted solution obtained at time instant $i-1$. We further use the condensing approach to solve the resulting QPs by reducing the variable space of a QP and forwarding simulation of the system dynamics. The resulting QP sub-problem can be solved much faster using dense general-purpose QP solvers. We perform *partial condensing* of the QP before calling the solver [39]. This exploits the structure of the QP resulting from the reformulation of an NMPC problem. After the implementation of the controller using `acados`, with an NMPC horizon length $N = 10$, the solver takes 5.4 ms on an average (1.3 ms standard deviation) to solve one step of the problem on an Intel[3] Core[4] i7-9750H CPU @ 2.60 GHz. The cost function and the object's inertial parameters are given in Table I. The simulation cycle time is then set to 5 ms, using the previous time step solution if the computation takes longer. The simulation code can be downloaded from the following link: https://github.com/prisma-lab/nonprehensile-object-transp

VII. Dynamic Simulation Results

In this section, we present the dynamic simulation results obtained using our MPNSM control approach for non-prehensile object transportation tasks.

A. Validation of the Contact Forces Calculation Procedure

In this subsection, we present the validation of contact force calculation procedure introduced in Section IV. We compared the forces calculated from the readings of the body wrench $\mathcal{F}_o$ using (15) and the one directly extracted from the Gazebo dynamic simulation environment. This is possible since, in the simulation environment (as also considered in our model), the contact surface is discretized with $n_c = 4$ contact points, and the friction cone is approximated with a pyramid. According to the simulator documentation, the default open dynamic engine (ODE) internal solver calculates the contact forces using the quick step method that relies upon an iterative projected Gauss-Seidel procedure whose accuracy depends on the number of iterations set. The friction parameter between the object and the tray is set to $\mu = 0.5$. For this simulation, we used the quintic-polynomial desired trajectory explained in Section VI, and employed the proposed MPNSM control method to track the desired trajectory. The comparison between the calculated (continuous line) and the measured (dashed line) contact force components for each contact point is shown in Fig. 5 for a $L = 0.62$ m long linear trajectory performed in 1.5 s. It is possible to note that the computed contact forces closely follow the measured ones with a maximum discrepancy of less than 0.1 N, which can be attributed to the different calculation procedures. This result validates our contact force calculation routine, which can now be effectively used in a practical case where direct contact force measurement is not possible. In Sections VII-B–VII-E, we show that our approach

[3]Registered trademark.
[4]Trademarked.

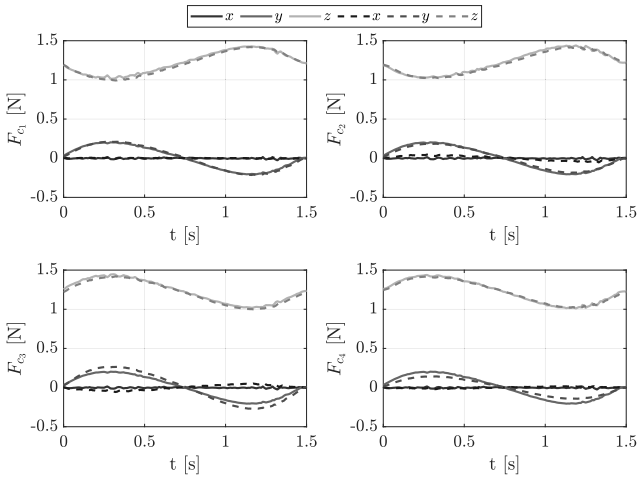

Fig. 5. Validation of the contact forces calculation procedure: comparison between calculated (continuous line) and measured (dashed line) contact force components for each contact point (c_i) along a $L = 0.62$ m long linear trajectory performed in $T = 1.5$ s.

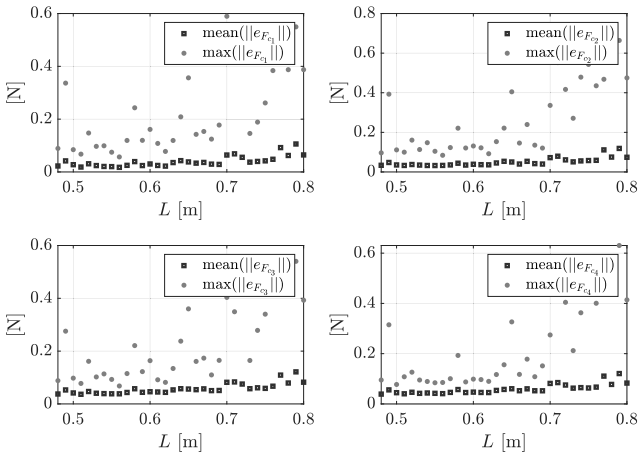

Fig. 6. Evaluation of the contact forces calculation procedure tracking performance: trend of the mean (black squares) and max (gray dots) norm of the instantaneous error vector for each contact point (c_i) along a linear trajectory with variable length L performed in $T = 1.5$ s.

works robustly in simulated/real setups using body wrench measurements despite the simplified model considered here.

To additionally evaluate the tracking performance of measured versus calculated forces, we performed a set of dynamic simulation experiments with varying trajectory length L involving the same setup. In more detail, we considered the trajectory length L varying from a minimum value of 0.48 m to a maximum value of 0.8 m with increments of 0.01 m (33 simulations). The trajectory duration has been kept 1.5 s, such that longer trajectories involve higher accelerations, which in turn changes the required contact forces. Fig. 6 shows for each contact point (c_i) the mean (black squares) and max (gray dots) norm of the error vector, which is constructed as follows:

$$e_{F_{c_i}} = \begin{bmatrix} \left(F_{c_i}^{\text{meas}}(0) - F_{c_i}(0)\right)^T \\ \vdots \\ \left(F_{c_i}^{\text{meas}}(T) - F_{c_i}(T)\right)^T \end{bmatrix}$$

where $F_{c_i}^{\text{meas}}(t) \in \mathbb{R}^3$ denotes the contact force measured at the instant t in the i-th contact, T denotes the trajectory duration and the norm operator is intended to be applied row-wise.

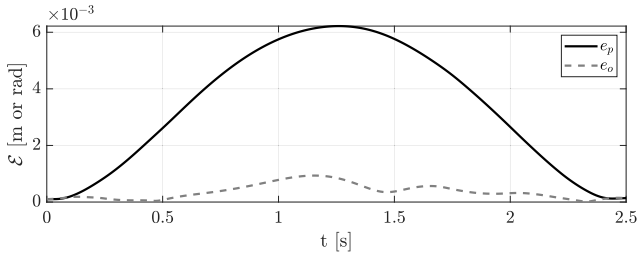

Fig. 7. Validation of the tracking performance along a slow ($T = 2.5$ s) linear trajectory. The norm of the error terms e_p and e_o is contained as no manipulation and robotic system constraints are met.

As noted from the figure, both the mean and the maximum error norms are low for shorter (thus slower) trajectories while they increase for longer (faster) trajectories. The explanation is readily available: the proposed calculation procedure does not account for the physical friction cone constraints becoming binding along trajectories requiring high accelerations.

B. Tracking Performance—No Constraints

In this subsection, the devised MPNSM control approach's tracking performance is showcased when no manipulation and robotic system constraints are met, i.e., they are satisfied with a strict inequality sign. This case is reproduced by slowing down the trajectory execution, i.e., by setting the desired trajectory completion time to $T = 2.5$ s. In Fig. 7 we show the tracking error $\mathcal{E} = (e_p, e_o)$, with $e_p(t) = ||p_o^*(t) - p_o(t)||$ and $e_o(t) = ||\phi_e(t)||$, where ϕ_e is the vector of Euler angles extracted from the rotation matrix error, i.e., $R_o^{*\mathrm{T}} R_o$, where R_o^* and R_o are is the desired and the current rotation matrix, respectively. In the considered case, the motion executed by the object closely follows the desired one with a maximum position error norm of $e_p^{\max} = 6 \times 10^{-3}$ m and a maximum orientation error norm of $e_o^{\max} = 1.5 \times 10^{-3}$ m. It is worth noting that the errors decrease to zero before the trajectory ends thanks to the predictive capabilities of the controller. Moreover, the shape of the error can be attributed to the particular choice of the cost function and its weights in the OCP (20) and may be further reduced by refined tuning. The result shown here can be used as a baseline to compare how the tracking performance results are affected when constraints are met or when external disturbance torques are applied, as considered in Sections VII-C and VII-D.

C. Tracking Performance—Non-Sliding and Robot Constraint

In this subsection, the tracking performance of the devised MPNSM control approach is showcased when both input and manipulation/robotic system constraints are met. This case is reproduced by re-scaling down to $T = 1.5$ s the completion time of the desired linear trajectory considered in Section VII-B. This makes the non-sliding manipulation constraints binding, i.e., the inequality constraint $\Lambda \geq 0$ holds with equality for some Λ_i at the optimal point in some portions of the trajectory as shown in Figs. 8 and 9. As it is possible to note, this is in general associated with higher tracking errors when compared to the baseline case (i.e., no constraints) previously shown in Section VII-B.

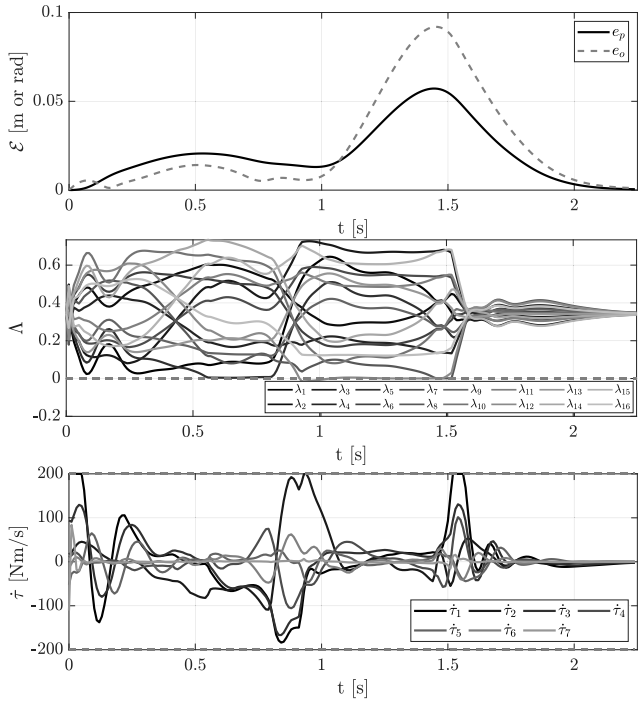

Fig. 8. Validation of the tracking performance along a fast ($T = 1.5$ s) trajectory. The norm of the error terms (e_p and e_o—top graph) is higher as manipulation (Λ—middle graph) and input constraints ($\dot{\tau}$—bottom graph) are met.

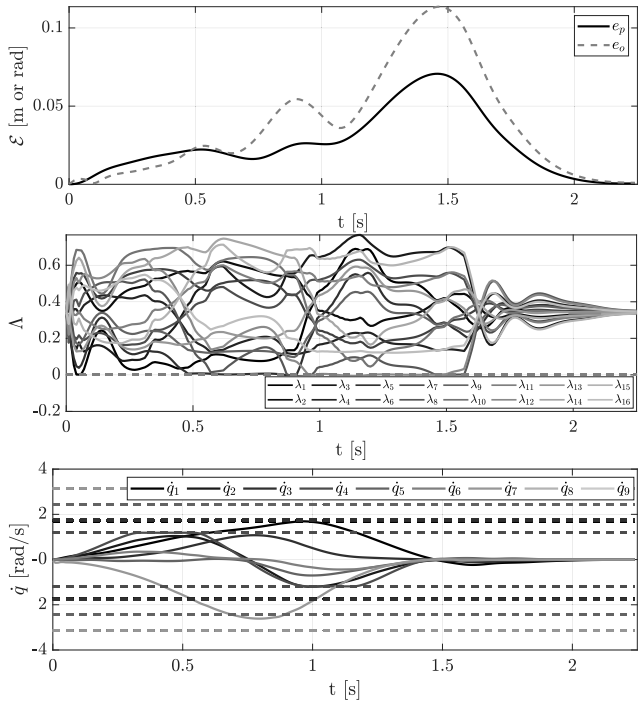

Fig. 9. Validation of the tracking performance along a fast ($T = 1.5$ s) trajectory. The norm of the error terms (e_p and e_o—top graph) is higher as manipulation (Λ—middle graph) and robotic system velocity constraints ($\dot{q}$—bottom graph) are met.

In addition, to show the influence of input and state constraints on the tracking performance, we restrict the upper/lower bounds on $\dot{\tau}$ to ± 200 and relax those on joint positions, velocities, and torques in the case considered in Fig. 8, while we re-apply the original robotic system bounds and relax those on the input in the case considered in Fig. 9.

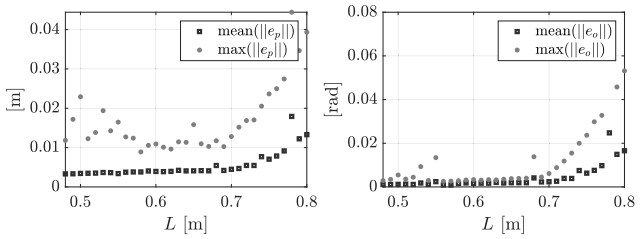

Fig. 10. Evaluation of the tracking performance: the trend of the mean (black squares) and max (gray dots) norm of the error along a variable length L linear trajectory performed in $T = 1.5$ s.

In the first case, $\dot{\tau}_1$ is saturating to the upper limit in some portions of the performed trajectory. In the second case, $\dot{q}_4$ is saturating to the upper/lower limits in some portions of the performed trajectory. In both cases, the tracking error is larger compared to the baseline case (no constraints) reaching a maximum of $e_p^{\max} = 0.06$ m, $e_o^{\max} = 0.09$ rad and $e_p^{\max} = 0.07$ m, $e_o^{\max} = 0.12$ rad, respectively, along the performed trajectory.

Finally, we evaluated the tracking performance when the trajectory length is varied as done in Section VII-A. Fig. 10 shows the trends of the mean (black squares) and max (gray dots) norm of the position (e_p) and orientation (e_o) errors. These are constructed as follows:

$$e_p = \begin{bmatrix} \left(p_o^*(0) - p_o(0)\right)^T \\ \vdots \\ \left(p_o^*(T) - p_o(T)\right)^T \end{bmatrix}, \quad e_o = \begin{bmatrix} (\phi_e(0))^T \\ \vdots \\ (\phi_e(T))^T \end{bmatrix}$$

where p_o^* denotes the desired position extracted from the desired path, ϕ_e is the vector of Euler angles extracted from the rotation matrix error, and T is the trajectory duration. The norm operator is intended to be applied row-wise. As noted from the figure, both the mean and the maximum error norm are low for shorter (thus slower) trajectories, while they increase for longer (faster) trajectories. The explanation is readily available: the tracking performance is penalized by the robotic and manipulation constraints becoming binding along trajectories requiring high accelerations.

The variation of the object dynamic properties, initial position, and dimension is not causing significantly different results in terms of tracking performance, as also shown in [40].

D. Tracking Performance—External Disturb Applied

Torque-controlled robots have the advantage of ensuring compliant behavior to external environment interactions. This is an advantage compared to position-control robots since compliance avoids giving rise to high contact forces when unintended interactions occur. To demonstrate the ability of our torque-based MPNSM control approach to handling this situation, we applied a disturbance torque in simulation, emulating an unwanted collision of the end-effector along the $\hat{x}$-direction (orthogonal to the performed motion). We again consider a $T = 2.5$ s trajectory duration to show the effect of the disturbance isolated from those of the potential constraint violation. The considered disturbance torque τ_d is computed as follows:

$$\tau_d = J_e^T \left[A\left(\sin\left(\frac{t_d}{D}\pi\right)\right), 0, 0, 0, 0, 0 \right]^T, \quad 0 < t_d < D \quad (24)$$

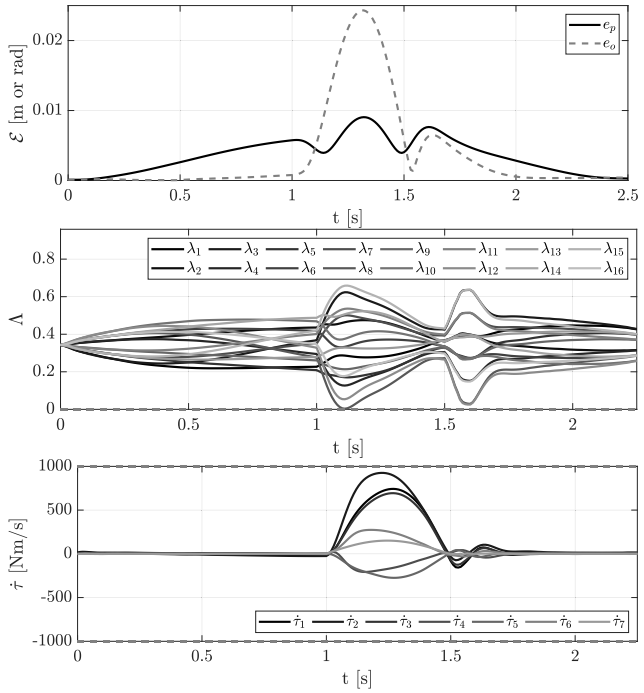

Fig. 11. Validation of the tracking performance along a slow ($T = 2.5$ s) trajectory. The norm of the error terms (e_p and e_o—top graph) is higher as the torque disturbance is applied between $t = 1$ s and $t = 1.5$ s. Manipulation force coefficients (Λ—middle graph) and input ($\dot{\tau}$—bottom graph) trends are shown.

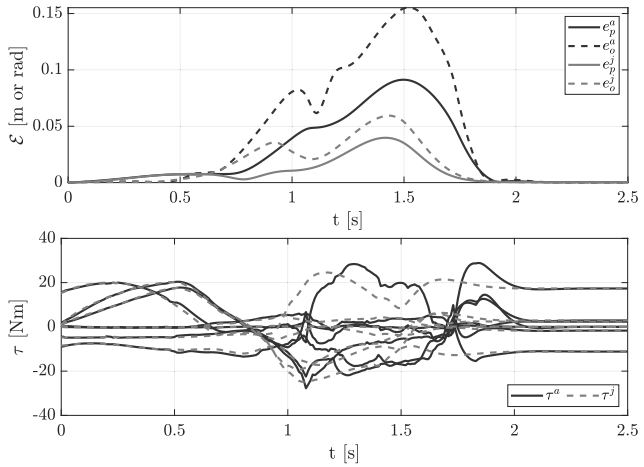

Fig. 12. Comparison of the tracking performance and control torques between acceleration- and jerk-based MPNSM controllers along a $T = 1.5$ s linear trajectory. The norm of the error terms (e_p and e_o—top graph) is comparable in the two cases while smoother torques (τ—bottom graph) can be obtained by the jerk-based one.

where $t_d = t - t_{\min}$ and $D = 0.5$ s is the duration of the disturbing torque signal and $A = 20$ N denotes its amplitude. During the time frame $t_{\min} = 1$ s and $t_{\min} + D = 1.5$ s, the robotic system control input is computed as $\tau(t) = \tau_{\text{mpc}}(t) + \tau_d(t)$, with τ_{mpc} being the controller output calculated as in (23). Results are shown in Fig. 11. As it is possible to note, the system is perturbed by the applied disturbance torque, and the tracking performance is significantly deteriorated when compared to the baseline experiment (shown in Section VII-B). Moreover, when disturbance torque is applied, contact force parameters touch their lower bound. However, our controller can readily compensate for and recover from the perturbed state using

TABLE III
REAL ROBOTIC SYSTEM JOINT LIMITS

Joint #	1	2	3	4	5	6	7	8	9
q [deg]	± 170	± 120	± 120	± 170	± 120	± 170	± 170	± 170	± 170
$\dot{q}$ [deg/s]	± 57	± 57	± 57	± 57	± 57	± 57	± 57	± 57	± 57
τ [Nm]	± 176	± 176	± 110	± 110	± 110	± 40	± 40	± 40	± 40

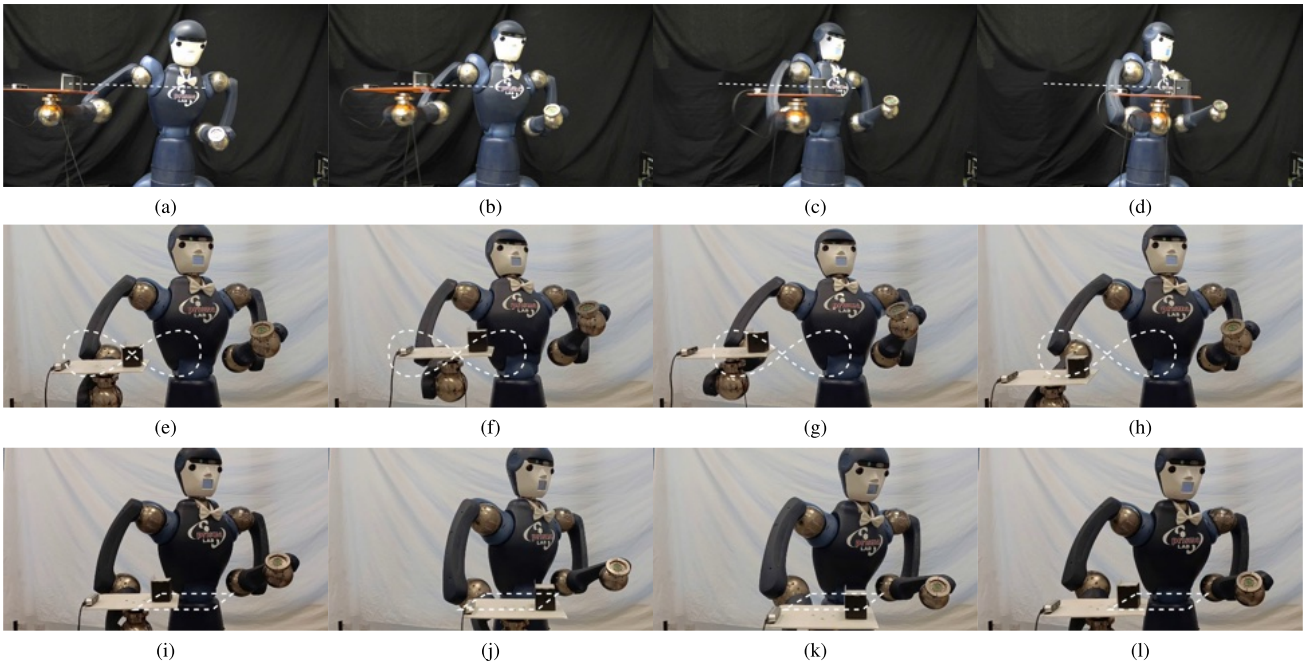

Fig. 13. Experiments using the proposed model-predictive non-sliding manipulation control approach. The desired trajectory is shown as a white dashed line. (a)–(d) Key frames of RoDyMan robot tracking a 1.5 s linear trajectory. (e)–(h) Key frames of RoDyMan robot tracking a 5.5 s Lemniscate-like trajectory. (i)–(l) Key frames of RoDyMan robot tracking a 4.5 s rectangular trajectory featuring three via points.

TABLE IV
REAL MPNSM CONTROL PARAMETERS

Parameter	Value	Parameter	Value
Q_τ	$1e^{-4}$	Q_e	$5e^{-4}$Q
Q_q	$1e^{7}$	R	$1e^{-4}$
$Q_{\dot{q}}$	$1e^{5}$	m_o	0.236
Q_Λ	$13-3$	I	$4.54e{-}5$

the optimal control input shown in the bottom graph. Further simulation results can be seen in the video accompanying this article.

E. Comparison With Acceleration-Based Controller

In this subsection, we compared the performance of our jerk-based MPNSM controller with an acceleration-based one. This is derived by reducing the system state to $x = (q^T, \dot{q}^T)^T$, and considering τ as an output of the controller. Robotic system constraints are thus imposed as state/input bounds while non-sliding manipulation constraints are easily formulated as an inequality relation on a nonlinear function of the two.

Results of the comparison are shown in Fig. 12 where we report the tracking error and the torque control input for the two controllers along the same $T = 1.5$ s linear trajectory where only manipulation constraints are becoming binding. The superscript a or j denotes the terms related to the acceleration and the jerk case, respectively. Despite the two controllers showing comparable tracking performance (e_p and e_o—top graph) with our jerk-based MPNSM controller we obtained smoother control torques (τ—bottom graph) that is sometimes beneficial when vibration and actuators' stress reduction is of interest. This is due to the integration rule shown in (23) and to the additional constraints on control torque time derivative that can be imposed in our jerk-based controller.

It is worth mentioning that the acceleration-based MPNSM controller is slightly more efficient exhibiting an average computation time of 0.048 ms along the considered trajectory. However, comparing acceleration- and jerk-based control approaches is beyond the scope of this article.

VIII. REAL EXPERIMENTS SHOWCASE

To demonstrate the validity of our approach, we conducted additional simulations and real experiments employing the RoDyMan humanoid robot. It is a 21-DoF robot made of a custom-built mobile base, a two-DoFs torso, two one-Dof shoulders, and two six-DoFs Shunk Powerball arms. Additional construction details can be found in [41]. For our experiments, we employed only the kinematic chain starting at the torso and ending at the tip of the robot's right arm (nine DoFs). A plastic tray-like end-effector was attached to it through a 3-D printed support, which embedded a Shunk 6-Axis F/T sensor. A calibrated Intel RealSense Depth Camera D415 was mounted on the tray with the purpose of tracking and recording the object displacement thanks to a QR-code pattern and the VISP auto tracker module [42]. The object is a steel hollowed cuboid of dimensions $60 \times 60 \times 70$ mm,

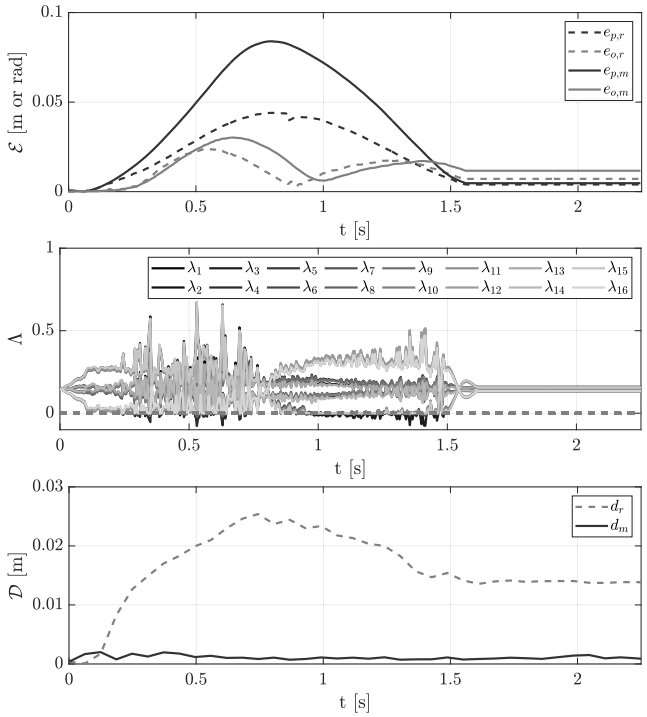

Fig. 14. Comparison of the trajectory tracking performance along a 1.5 s reference trajectory using RoDyMan robot. The norm of the error terms (e_p and e_o—top graph) is higher when the MPNSM control is used. Evolution of the non-sliding manipulation coefficients (Λ—middle graph) and norm of the object displacement (d—bottom graph) in the two cases.

whose inertial properties are: mass $m_o = 0.236$ kg and diagonal inertial matrix, $I_o = \text{diag}(4.5375 \times 10^{-5})$ kgm^2. The friction coefficient between the object and the tray has been experimentally identified in $\mu = 0.2$. The robot was position-controlled, and its set point was extracted from the output trajectory solution of the MPNSM controller given in (20). The robot control cycle time is set to 8 ms. In-text citation for Table III contains the real system control parameters while the robot physical limits are given in Table IV. A picture of the experimental setup is given in Fig. 1.

We compared the system's performance while tracking the reference trajectory using the robot-embedded position control and the MPNSM controller. The first object trajectory chosen for the real experiment is a quintic polynomial rest-to-rest linear Cartesian trajectory with initial point $p_i = [0.7, -0.5, 1.34]$ m, and final point $p_e = [0.7, 0.15, 1.34]$ m expressed in the robot base frame, whose duration is $T = 1.5$ s. As in the simulation, the desired orientation is kept constant and corresponding to the tray facing the upward direction. The corresponding joint trajectory was computed using the Jacobian pseudo-inverse CLIK routine. A timed sequence of key frames taken during the performed experiments is shown in Fig. 13(a)–(d). At the same time, quantitative data are plotted in Fig. 14 in terms of tracking error $\mathcal{E}$, manipulation constraint Λ and object displacement $\mathcal{D}$ for the two cases. The subscript r and m denote the tracking of the reference and of the NPNSM trajectory, respectively. The MPNSM controller performance trends in the real experiments are close to what was achieved and widely discussed in Section VII. In summary, when manipulation (or robotic system) constraints become binding (one or multiple $\Lambda_i = 0$ in the middle graph), tracking performance ($e_{p,m}$ and $e_{o,m}$ in the top graph) are penalized with respect to the reference case ($e_{p,r}$ and $e_{o,r}$ in the same graph) in favor of safety, i.e., the manipulator does not exceed its limits and the object does not slide and fall from the tray. This can be seen from the norm of the object displacement $d(t) = ||p_o(0) - p_o(t)||$ in the bottom graph shown for the two cases. Better insights can be captured from the complete video of the experiments.[5]

To additionally prove the robustness of our controller we consider two additional trajectories: 1) a rectangular path in the horizontal plane, shown at the top of Fig. 15, featuring three via points obtained imposing trapezoidal velocity profiles with acceleration time equal to 0.2 s at the transitions between segments and 2) a Lemniscate-like path in the vertical plane, shown at the top of Fig. 16, obtained employing a piecewise cubic B-spline curve enclosed by its control points. The validation of the performance using the proposed MPNSM control onto the Rodyman robot is shown in the graphs of Figs. 15 and 16, respectively. A timed sequence of keyframes taken during the performed experiments is shown in Fig. 13(e)–(l), where the desired trajectory is shown in the overlay. In both cases, it can be noted that when the robotic system constraints become binding ($\dot{q}$—bottom graph), the tracking performance is penalized ($\mathcal{E}$—top graph), while the contact force coefficients (Λ—middle graph) are still kept greater than zero.

IX. Discussion

The proposed MPNSM controller relies on the knowledge of the manipulated object dynamic model and its friction coefficient. While in industrial settings, it is reasonable to assume knowledge about the manipulated objects, this is generally not the case in unknown environments. One interesting future research direction can be exploring online dynamic parameter estimation procedures for non-prehensile manipulation setups, such as the one considered in this article. However, to solve this problem, one must account for the dichotomy between moving safely (i.e., avoid sliding) and exciting the object dynamics (required to identify its parameters). To this end, the naturally smooth object transportation task considered in this work can easily become hybrid in view of the stick and slip phenomenon: uncertainties in the transported object model can indeed trigger the transition between static and viscous friction. In this case, the advantages of using jerk-based MPC to obtain smoother torque profiles for manipulation will be even more evident.

Although our controller achieved encouraging results overall, several points need to be better discussed. Regarding the mathematical formulation of our controller, it is worth mentioning that in the extended state, we could have considered adding only τ. Indeed, the object wrench, the contact forces, and their coefficients (on which we formulate the non-sliding constraint) can all be expressed as a function of the proper manipulator state plus τ. However, including Λ in the extended state allowed to more easily formulate its constraint as a state constraint and more straightforwardly feedback the contact force measurement [8]. This redundancy, however, increased

[5]https://github.com/prisma-lab/nonprehensile-object-transp

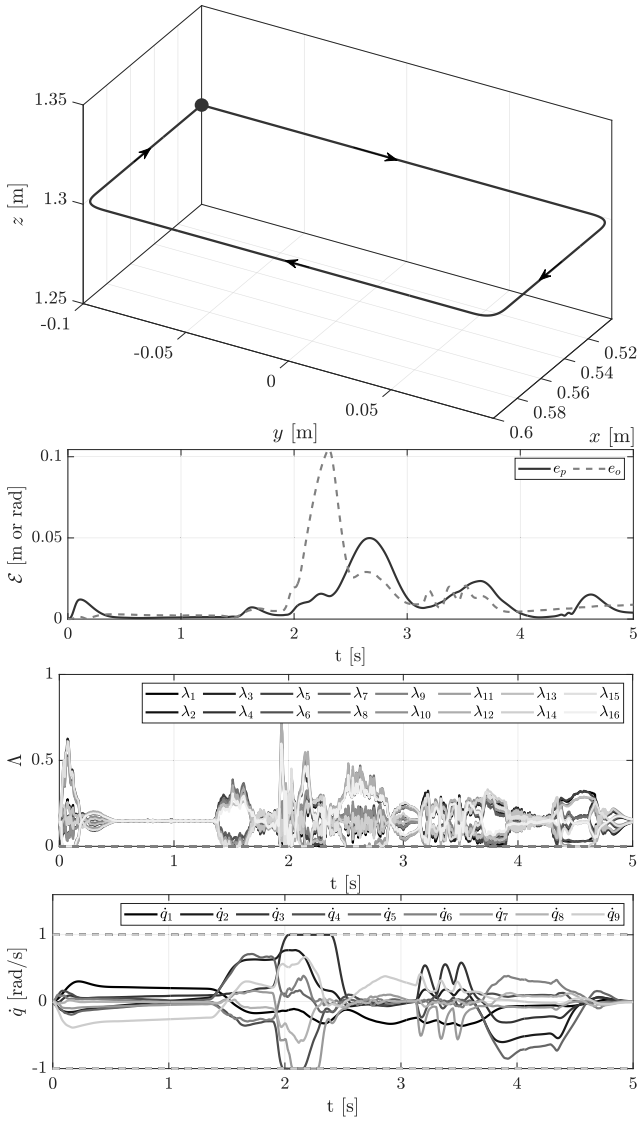

Fig. 15. Validation of the tracking performance along the rectangular, 4.5 s duration trajectory. The black dot denotes the start/endpoint, the arrows indicate the direction. The norm of the error terms (e_p and e_o—top graph) is higher as manipulation (Λ—middle graph) and input constraints ($\dot{\tau}$—bottom graph) are met.

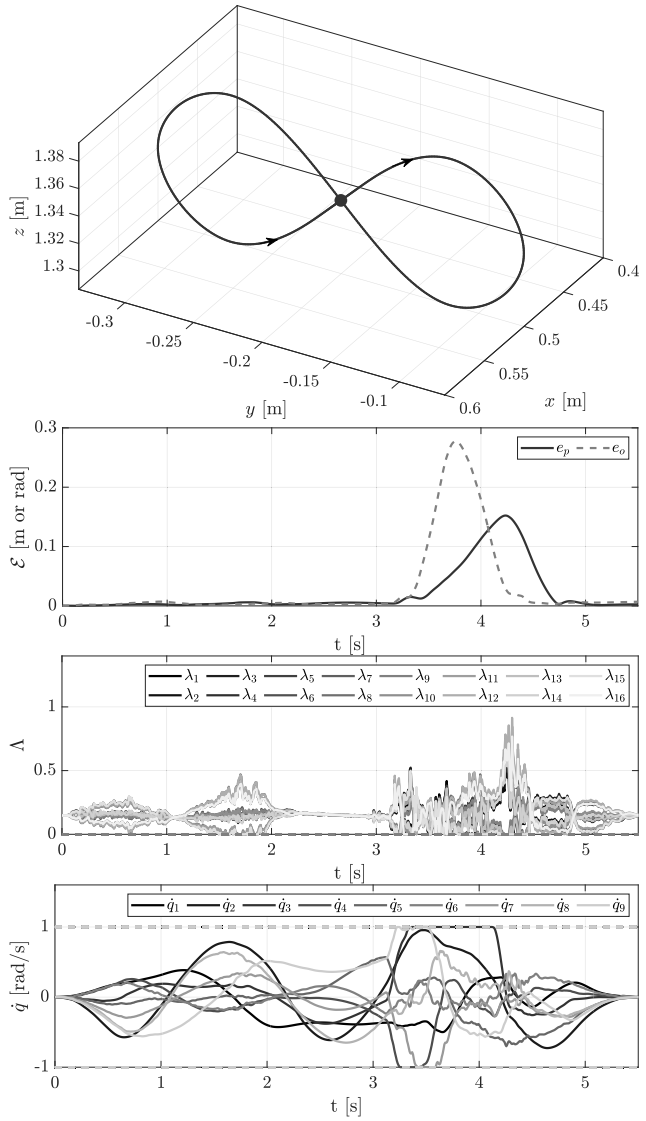

Fig. 16. Validation of the tracking performance along the Lemniscate-like, 5.5 s duration trajectory. The black dot denotes the start/endpoint, the arrows indicate the direction. The norm of the error terms (e_p and e_o—top graph) is higher as manipulation (Λ—middle graph) and input constraints ($\dot{\tau}$—bottom graph) are met.

the number of states and thus the solving time of our controller. In the future, we aim to investigate the gain in terms of accuracy and solving time by removing Λ from the extended state. Moreover, matrices appearing in the time derivative of Λ in (22) were derived under the assumption that the dynamic model holds constant over the time horizon. This might not always be true, e.g., for longer prediction horizons, and needs further attention.

From the experimental point of view, the solver convergence time is decreased when a shorter horizon length is utilized; however, the solver's trajectory tracking performance also decreases. A trade-off between the solution time, the tracking performance, and the computational complexity involved must be found depending on the task requirements. We experimentally found a good trade-off using a prediction horizon $N = 10$. This choice makes the proposed controller suitable for online implementation with a real robotic manipulator. However, the success of the solver in finding a solution at each time step heavily depends on the selected cost function and its gains.

X. Conclusion

This article proposed a model-predictive non-sliding manipulation control approach for non-prehensile object transportation using robot manipulators. We derived the combined manipulator/object dynamic model and formulated the associated non-sliding constraints that are enforced by the controller. The proposed optimization-based controller has been shown capable of safely accomplishing a trajectory tracking task with an object being transported in a non-prehensile way on a tray-like manipulator end-effector. The controller imposes that the manipulation and physical constraints of the robotic system are always respected during the executed trajectory at the expense of tracking performances. Extensive dynamic simulations and real-world experiments validated our approach and provided interesting insights for future research directions on this topic.

REFERENCES

[1] P. Fiorini and E. Prassler, "Cleaning and household robots: A technology survey," *Auton. Robots*, vol. 9, no. 3, pp. 227–235, 2000.

[2] A. Hentout, M. Aouache, A. Maoudj, and I. Akli, "Human–robot interaction in industrial collaborative robotics: A literature review of the decade 2008–2017," *Adv. Robot.*, vol. 33, nos. 15–16, pp. 764–799, Aug. 2019.

[3] C. Rosales, R. Suárez, M. Gabiccini, and A. Bicchi, "On the synthesis of feasible and prehensile robotic grasps," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 550–556.

[4] N. Mai, M. Avarello, and P. Bolsinger, "Maintenance of low isometric forces during prehensile grasping," *Neuropsychologia*, vol. 23, no. 6, pp. 805–812, Jan. 1985.

[5] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1711–1718, Jul. 2018.

[6] N. B. Zumel and M. A. Erdmann, "Nonprehensile manipulation for orienting parts in the plane," in *Proc. Int. Conf. Robot. Autom.*, vol. 3, 1997, pp. 2433–2439.

[7] A. Gazar, G. Nava, F. J. A. Chavez, and D. Pucci, "Jerk control of floating base systems with contact-stable parameterized force feedback," *IEEE Trans. Robot.*, vol. 37, no. 1, pp. 1–15, Feb. 2021.

[8] S. Kleff, E. Dantec, G. Saurel, N. Mansard, and L. Righetti, "Introducing force feedback in model predictive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2022, pp. 13379–13385.

[9] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *Int. J. Control*, vol. 93, no. 1, pp. 62–80, Jan. 2020.

[10] M. T. Mason and K. M. Lynch, "Dynamic manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, Jul. 1993, pp. 152–159.

[11] K. M. Lynch and T. D. Murphey, "Control of nonprehensile manipulation," in *Control Problems in Robotics* (Springer Tracts in Advanced Robotics), vol. 4, A. Bicchi, D. Prattichizzo, and H. Christensen, Eds. Berlin, Germany: Springer, 2003, pp. 39–57.

[12] A. C. Satici, F. Ruggiero, V. Lippiello, and B. Siciliano, "A coordinate-free framework for robotic pizza tossing and catching," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2016, pp. 3932–3939.

[13] M. M. Schill, F. Gruber, and M. Buss, "Quasi-direct nonprehensile catching with uncertain object states," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2015, pp. 2468–2474.

[14] C. Liu, Y. Hayakawa, and A. Nakashima, "Racket control and its experiments for robot playing table tennis," in *Proc. IEEE Int. Conf. Robot. Biomimetics*, Dec. 2012, pp. 241–246.

[15] F. Bertoncelli, F. Ruggiero, and L. Sabattini, "Linear time-varying MPC for nonprehensile object manipulation with a nonholonomic mobile robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2020, pp. 11032–11038.

[16] D. Serra, F. Ruggiero, A. Donaire, L. R. Buonocore, V. Lippiello, and B. Siciliano, "Control of nonprehensile planar rolling manipulation: A passivity-based approach," *IEEE Trans. Robot.*, vol. 35, no. 2, pp. 317–329, Apr. 2019.

[17] J. Z. Woodruff and K. M. Lynch, "Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2017, pp. 4066–4073.

[18] A. Gutierrez-Giles, F. Ruggiero, V. Lippiello, and B. Siciliano, "Closed-loop control of a nonprehensile manipulation system inspired by the pizza-peel mechanism," in *Proc. 18th Eur. Control Conf.*, Naples, Italy, Jun. 2019, pp. 1580–1585.

[19] M. Higashimori, R. Sakashita, and A. Shibata, "Single-actuator-based three-DoF planar manipulation via a viscoelastic and nonparallel hybrid joint mechanism," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 602–617, Jun. 2019.

[20] M. Erdmann, "An exploration of nonprehensile two-palm manipulation," *Int. J. Robot. Res.*, vol. 17, no. 5, pp. 485–503, May 1998.

[21] A. Gupta and W. Huang, "A carrying task for nonprehensile mobile manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, Oct. 2003, pp. 2896–2901.

[22] P. Lertkultanon and Q. Pham, "Dynamic non-prehensile object transportation," in *Proc. 13th Int. Conf. Control Autom. Robot. Vis.*, Dec. 2014, pp. 1392–1397.

[23] J. E. Solanes, L. Gracia, P. Muñoz-Benavent, J. V. Miro, M. G. Carmichael, and J. Tornero, "Human–robot collaboration for safe object transportation using force feedback," *Robot. Auton. Syst.*, vol. 107, pp. 196–208, Sep. 2018.

[24] J. M. Garcia-Haro, S. Martinez, and C. Balaguer, "Balance computation of objects transported on a tray by a humanoid robot based on 3D dynamic slopes," in *Proc. IEEE-RAS 18th Int. Conf. Humanoid Robots*, Nov. 2018, pp. 704–709.

[25] G. Martucci, J. Bimbo, D. Prattichizzo, and M. Malvezzi, "Maintaining stable grasps during highly dynamic robot trajectories," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2020, pp. 9198–9204.

[26] M. Selvaggio, J. Cacace, C. Pacchierotti, F. Ruggiero, and P. R. Giordano, "A shared-control teleoperation architecture for nonprehensile object transportation," *IEEE Trans. Robot.*, vol. 38, no. 1, pp. 569–583, Feb. 2022.

[27] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter, "Whole-body MPC for a dynamically stable mobile manipulator," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3687–3694, Oct. 2019.

[28] J. Pankert and M. Hutter, "Perceptive model predictive control for continuous mobile manipulation," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6177–6184, Oct. 2020.

[29] M. V. Minniti, R. Grandia, K. Fäh, F. Farshidian, and M. Hutter, "Model predictive robot-environment interaction control for mobile manipulation tasks," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2021, pp. 1651–1657.

[30] W. C. Agboh and M. R. Dogar, "Pushing fast and slow: Task-adaptive planning for non-prehensile manipulation under uncertainty," in *Algorithmic Foundations of Robotics XIII*, M. Morales, L. Tapia, G. Sánchez-Ante, and S. Hutchinson, Eds. Cham, Switzerland: Springer, 2020, pp. 160–176.

[31] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. London, U.K.: Springer, 2010.

[32] D. Prattichizzo and J. C. Trinkle, *Grasping*. Berlin, Germany: Springer, 2008, pp. 671–700.

[33] R. M. Murray, S. S. Sastry, and L. Zexiang, *A Mathematical Introduction to Robotic Manipulation*, 1st ed. Boca Raton, FL, USA: CRC Press, 1994.

[34] A. Bicchi, "Force distribution in multiple whole-limb manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 1993, pp. 196–201.

[35] C. Gaz, F. Flacco, and A. De Luca, "Extracting feasible robot parameters from dynamic coefficients using nonlinear optimization methods," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2016, pp. 2075–2081.

[36] R. Verschueren et al., "Acados: A modular open-source framework for fast embedded optimal control," 2019, *arXiv:1910.13753*.

[37] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, Mar. 2019.

[38] M. Diehl, H. G. Bock, and J. P. Schloder, "Real-time iterations for nonlinear optimal feedback control," in *Proc. 44th IEEE Conf. Decis. Control*, Dec. 2005, pp. 5871–5876.

[39] G. Frison, D. Kouzoupis, J. B. Jørgensen, and M. Diehl, "An efficient implementation of partial condensing for nonlinear model predictive control," in *Proc. IEEE 55th Conf. Decis. Control*, Dec. 2016, pp. 4457–4462.

[40] V. Morlando, M. Selvaggio, and F. Ruggiero, "Nonprehensile object transportation with a legged manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2022, pp. 6628–6634.

[41] F. Ruggiero et al., "Nonprehensile manipulation of deformable objects: Achievements and perspectives from the robotic dynamic manipulation project," *IEEE Robot. Autom. Mag.*, vol. 25, no. 3, pp. 83–92, Sep. 2018.

[42] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: A hands-on survey," *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 12, pp. 2633–2651, Dec. 2016.