Linear Time-Varying MPC for Nonprehensile Object Manipulation with a Nonholonomic Mobile Robot

Filippo Bertoncelli[1], Fabio Ruggiero[2], Lorenzo Sabattini[1]

***Abstract*— This paper proposes a technique to manipulate an object with a nonholonomic mobile robot by pushing, which is a nonprehensile manipulation motion primitive. Such a primitive involves unilateral constraints associated with the friction between the robot and the manipulated object. Violating this constraint produces the slippage of the object during the manipulation, preventing the correct achievement of the task. A linear time-varying model predictive control is designed to include the unilateral constraint within the control action properly. The approach is verified in a dynamic simulation environment through a Pioneer 3-DX wheeled robot executing the pushing manipulation of a package.**

I. Introduction

In a robotic nonprehensile manipulation task, the object is subject only to unilateral constraints imposed by both the robot manipulating it and the environment. A complicated manipulation task can be split into many simpler subtasks, usually called *manipulation primitives* [1]. Among these primitives, the *pushing* operation is a simple solution, also adopted by humans, in those situations where the size of the manipulated object prevents an easy grasp by a gripper, or it is too heavy to be dexterously handled. Manipulation by pushing is intuitively simple, but it sets interesting control problems originated by the presence of the friction forces, which are complex to accurately model and causing an unpredictability of the object's motion [2].

This paper tackles the problem of a nonholonomic mobile robot pushing an object in the environment. The use of a mobile robot for pushing manipulation is justified to overcome the physical limits imposed by a static manipulator's workspace, or when the object is too large and/or too heavy to be grasped by a mobile manipulator with a gripper. Practical applications can be primarily found in warehouses and industries for handling of goods [3]. The pursued approach is the design of a linear time-varying (LTV) model predictive control (MPC) [4] which explicitly includes the pushing constraints. Violating these constraints means that the forces exerted by the robot on the object do not belong to the friction cones at the contact points. This induces the slipping of the object, reduces the precision of the manipulation task, and worsens the overall performance.

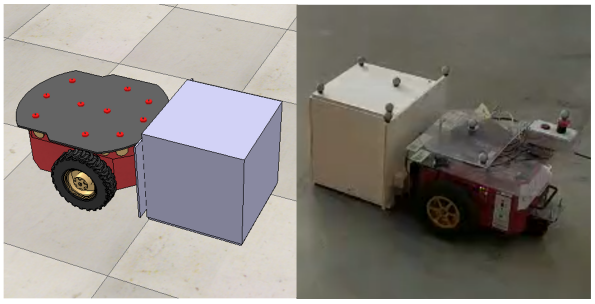

Fig. 1. Example of a pushing manipulation through a mobile robot. On the left, the simulation environment. On the right, the robot in action during the carried out experiments.

Building upon [5], we propose the use of a classic mobile robot to perform a pushing operation (see Fig. 1), by considering the robot's nonholonomic constraints in the controller formulation explicitly, and introducing a motion constraint that considers the pushing dynamics to maintain a stiff contact between the robot and the pushed object.

II. Related work

A thorough literature review reveals that object manipulation with a mobile robot is typically achieved with proper tools [6], like grippers, or with multi-robot systems caging the object [7]. This latter approach takes inspiration from the natural world. Small animals, like ants, collaborate to transport heavy and large loads: several works try to mimic the behavior of ants to achieve collaborative transportation for groups of mobile robots [8], [9]. However, this problem is often solved considering approaches in which force closure is achieved [10], planning the motion of the robots in such a way that they are opportunely displaced around the object [7]. As an example, robots can be controlled to create a formation around the objects, in such a way that, locally exchanging information, they can transport it as designed in [11]. Along the same lines, the presence of a large number of robots, that can be attached to the object in such a way that they can exchange a force with the object itself, is considered in [12]. The center of mass of the object is approximated as the centroid of the positions of the robots. Geometrical properties of the object are also estimated in [13] based on the robots' relative positions.

The approaches mentioned above resemble the most common solution exploited in robotics for solving the problem of moving an object: the pick-and-place method, where the

The research leading to these results has been partially supported by the WELDON project, in the frame of Programme STAR, financially supported by UniNA and Compagnia di San Paolo. The authors are solely responsible for its content.

Authors would like to thank Dr. L. Fontanili, Prof. M. Milani and the Hydraulic System Design Research Group at the University of Modena and Reggio Emilia, for their support in the development of the experiments.

[1] Authors are with the Department of Sciences and Methods for Engineering (DISMI), University of Modena and Reggio Emilia, Italy.

[2] Author is with the PRISMA Lab, Department of Electrical Engineering and Information Technology, University of Naples, Via Claudio 21, 80125, Naples, Italy.

Corresponding author's email filippo.bertoncelli@unimore.it

object is grasped stiffly and is then moved to the desired location. While pick-and-place is a common and effective solution in several cases, it cannot always be applied. This is particularly true when the size of the object is too large, when its shape is unknown a priori, when it is excessively heavy, when a firm grasp can damage its surface, or when the external environment places some constraints on the use of a multi-robot system. Nonprehensile manipulation approaches can thus be exploited in these cases. Specifically, these approaches include methods in which the robot imposes the object's motion through unilateral constraints only, such as in the case of pushing. The advantage is the possibility of using only one robot for the operation and the possibility of breaking a contact and create new ones during the same task [1], [2]. Nevertheless, nonprehensile manipulation requires taking into account the robot, the object, and the environmental dynamics, which is often a nontrivial task in the presence of the friction, as for pushing.

Recent implementations of nonprehensile manipulation with robots saw the use of flexible elements like ropes and cables. A robot equipped with a flexible cable is shown in [14], where a planning method is proposed to exploit the cable for moving the object. Objects with general shape are instead addressed in [15], where two mobile robots are connected through a cable, and they cooperatively pull a heavy object. However, such physical interconnection between the two robots may significantly limit the freedom of motion.

To avoid these issues, mobile robots can directly perform nonprehensile manipulation by directly pushing the object [16]. However, it is necessary to guarantee the possibility for the mobile robot to change the pushing direction. This means that the robot must freely move in the environment, without hitting obstacles, to change its relative position to the object. Uncertainties in control and motion execution are addressed in [17] employing an appropriate motion planning strategy, that considers an increased size of the pushed object, to include repositioning maneuvers of the pushing robot. A reinforcement learning framework is proposed in [18] to define the motion pattern for two robots pushing a box. However, a very simplistic scenario is considered where dynamics are neglected. By measuring the instantaneous direction, a robot or a group of robots is guided by an artificial potential field in [19] to push an object. Also in this case, dynamic effects, such as friction, are not considered, making the proposed method unsuitable for complex situations, such as in the presence of non-uniform friction. A fuzzy controller is instead designed in [20] to control two robots pushing an object with known geometrical properties. Slipping of the mobile robot's wheels during a pushing operation is avoided in [21] through a nonlinear MPC design. However, the friction between the manipulated object and the robot is neglected, causing possible slippage of the object during the manipulation task. Finally, the complexity of determining the optimal sequence of actions to manipulate an object by pushing is solved offline in [5] through machine learning. A convex hybrid MPC program is then solved online to achieve planar manipulation.

Differently from [5], the proposed design explicitly includes the nonholonomic nature of the most common mobile robots available in the market with the dynamic model formulation. Besides, a motion constraint is designed to avoid the slippage of the contact with pushed object during the manipulation, made necessary by the robot nonholonomy.

III. Problem statement

Consider a wheeled mobile robot moving in a bi-dimensional environment, where a polygonal planar object has to be manipulated. Let Σ_w and Σ_r be the global reference frame and the body frame attached to the center of the robot's axle, respectively. Let the pose of the mobile robot at time t be represented by the vector $\chi_r(t) = \begin{bmatrix} x_r(t) & y_r(t) & \theta_r(t) \end{bmatrix}^T \in \mathbb{R}^3$, where $x_r(t), y_r(t) \in \mathbb{R}$ represent the position of Σ_r in Σ_w, and $\theta_r \in \mathbb{R}$ is the rotation of Σ_r with respect to Σ_w. A visualization is provided in Fig. 2. In a similar way, let $\chi_o(t) = \begin{bmatrix} x_o(t) & y_o(t) & \theta_o(t) \end{bmatrix}^T \in \mathbb{R}^3$ represent the pose of the object, as shown in Fig. 3.

We provide a solution to the following problem.

Problem *Control the motion of the mobile robot to manipulate the object through pushing maneuvers, in such a way that the trajectory $\chi_o(t)$ tracks the desired one $\chi_d(t)$ with the desired accuracy, starting from the initial pose $\chi_o(0)$.*

In the following, we will assume the mobile robot to behave as a unicycle. The choice is motivated by the simplicity of notation introduced by such a model, and by the fact that several real-world mobile robots (as differential-drive robots) can be represented according to this formulation [22]. We assume that all the considered contacts are rigid, that the robot wheels do not slip, and that the forces exchanged in the interaction follow Coulomb's model of friction. Moreover, we decompose each contact force in two components, aligned with the edges of the friction cone [23]. The angle between each component and the contact normal is

$$\theta_\mu = \tan^{-1} \mu, \tag{1}$$

where $\mu > 0$ is the friction coefficient associated with the interacting surfaces. A visualization of the used decomposition is provided in Figure 3. The motion of the controlled system is assumed quasistatic (i.e, it is slow enough that inertial forces are negligible). Moreover, we assume the mobile robot to be equipped with a planar end-effector (i.e., a planar contact surface), such that the surface used to interact with the object is consistent and homogeneous. During the interaction, the end-effector is supposed to be parallel to one of the sides of the polygonal object. This type of interaction is typically referred to as *line contact*, modeled as if the only contact points were the extreme points of the line [24].

IV. Modeling

In this section, we introduce the mathematical model of the system motion and the constraints applied within the MPC controller for planar manipulation tasks. First, we describe a second-order model of the mobile robot and its error dynamic for the desired trajectory, then the mathematical model of the pushed object motion is introduced.

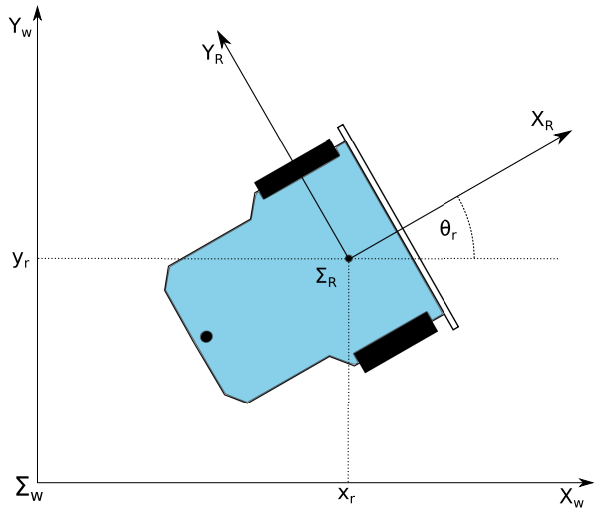

Fig. 2. Schematic representation of the differential drive mobile robot. The black rectangles are the wheels. The black circle is the caster wheel. The frontal bumper is represented by the white rectangle in front of the wheels.

A. Robot Model

Defining $v_r, \omega_r \in \mathbb{R}$ as the linear and angular velocities of the robot, respectively (see Fig. 2), the state of the robot is defined as $\xi(t) \in \mathbb{R}^5$, given by

$$\xi(t) = \begin{bmatrix} x_r(t) & y_r(t) & \theta_r(t) & v_r(t) & \omega_r(t) \end{bmatrix}^T. \quad (2)$$

For ease of notation, in the following, dependence on time will be omitted, when not strictly necessary.

Define now $a_r, \varepsilon_r \in \mathbb{R}$ as the inputs for the robot, given as the linear and angular acceleration, respectively. Hence, the model of the robot motion can be written as

$$\dot{\xi} = \begin{bmatrix} \cos\theta_r v_r \\ \sin\theta_r v_r \\ \omega_r \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ a_r \\ \varepsilon_r \end{bmatrix}. \quad (3)$$

Solution of the Problem stated in Section III passes through the generation of the desired trajectory $\xi_d(t) = \begin{bmatrix} x_{rd}(t) & y_{rd}(t) & \theta_{rd}(t) & v_{rd}(t) & \omega_{rd}(t) \end{bmatrix}^T \in \mathbb{R}^5$ for the robot to realize the pushing maneuvers[1].

Let $e(t) = \begin{bmatrix} e_{xr}(t) & e_{yr}(t) & e_{\theta r}(t) & e_{vr}(t) & e_{\omega r}(t) \end{bmatrix}^T \in \mathbb{R}^5$ be the error vector with respect to the desired reference frame centered in $(x_{rd}(t), y_{rd}(t))$, and oriented as $\theta_{rd}(t)$, that is defined as

$$e(t) = \begin{bmatrix} \cos(\theta_{rd}(t)) & \sin(\theta_{rd}(t)) & 0 & 0 & 0 \\ -\sin(\theta_{rd}(t)) & \cos(\theta_{rd}(t)) & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} (\xi - \xi_d). \quad (4)$$

[1]Several strategies exist, in the literature, to generate trajectories for mobile robots during pushing maneuvers. Due to space limitations, this problem is not addressed in this paper. However, the reader is referred to, e.g., [2] [25], for further details.

Considering the robot motion (3) and the error vector (4), we can describe the system error dynamics as follows:

$$\dot{e}(t) = f(t) = \begin{bmatrix} \cos(e_{\theta r})(e_{vr} + v_{rd}) - v_{rd} + e_{yr}\omega_{rd} \\ \sin(e_{\theta r})(e_{vr} + v_{rd}) - e_{xr}\omega_{rd} \\ e_{\omega r} \\ a_r - \dot{v}_{rd} \\ \varepsilon_r - \dot{\omega}_{rd} \end{bmatrix}. \quad (5)$$

B. Pushed Object Model

Consider, as discussed in Section III, a polygonal object pushed by the robot with line contact on one of its sides. The contact forces are modeled using the components along the friction cone as shown in Fig. 3. We denote with $f_{iR} \in \mathbb{R}$ and $f_{iL} \in \mathbb{R}$ the right and left contact force components, respectively, for each contact point $i = \{1, 2\}$ [23]. We define the vector $f_c \in \mathbb{R}^4$ as

$$f_c = \begin{bmatrix} f_{1R} & f_{1L} & f_{2R} & f_{2L} \end{bmatrix}^T. \quad (6)$$

The total external wrench $w \in \mathbb{R}^3$, expressed in Σ_W and whose torque is applied around the object's center of mass, exerted by the robot to the object can be described by

$$w = G f_c \quad (7)$$

where $G \in \mathbb{R}^{3\times4}$ is the so called grasp matrix. For a square object of side length $2s > 0$, the grasp matrix is

$$G = \begin{bmatrix} \cos(\theta_r - \theta_\mu) & \sin(\theta_r - \theta_\mu) & s(\cos\theta_\mu + \sin\theta_\mu) \\ \cos(\theta_r + \theta_\mu) & \sin(\theta_r + \theta_\mu) & s(\cos\theta_\mu - \sin\theta_\mu) \\ \cos(\theta_r - \theta_\mu) & \sin(\theta_r - \theta_\mu) & s(\sin\theta_\mu - \cos\theta_\mu) \\ \cos(\theta_r + \theta_\mu) & \sin(\theta_r + \theta_\mu) & -s(\cos\theta_\mu + \sin\theta_\mu) \end{bmatrix}^T, \quad (8)$$

where θ_μ is given in (1), and it can be obtained through a geometrical analysis of the contact forces.

Under the assumption of quasistatic interaction, the object motion can be described using the limit surface [26], a geometric representation of the relationship between the applied force on an object and its instantaneous velocity. Inspired by [5], an ellipsoidal approximation of the limit surface is used, due to its simplicity and invertibility properties. A convex quadratic formulation of the ellipsoidal limit surface is given by $S(w) = \frac{1}{2} w^T H w$, where $H \in \mathbb{R}^{3\times3}$ is the matrix representing the ellipsoidal approximation of the limit surface. A procedure for computing such an approximation, that requires the knowledge of the object's shape and mass as well as the friction coefficient of the support surface, can be found in [27]. Through the principle of maximal dissipation [26], the object instantaneous velocity is perpendicular to the limit surface for a given wrench, which implies:

$$\begin{bmatrix} \dot{x}_o & \dot{y}_o & \dot{\theta}_o \end{bmatrix}^T = H w. \quad (9)$$

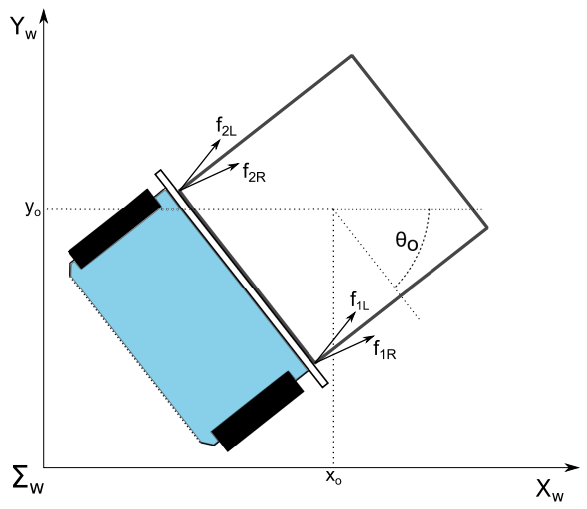

Fig. 3. Schematic representation of the pushed object.

V. MPC Controller Formulation

To correctly solve the Problem defined in Section III, a controller must be designed such as the error vector $e(t)$ in (4) is steered to zero without violating the friction constraints given by the contact between the robot and the object. This avoids the slippage of the object during the pushing manipulation. As a matter of fact, zeroing the error vector only does not imply that the object follows the desired trajectory $\chi_d(t)$. The controller makes use of a LTV MPC formulation [4] to solve the nonlinear control problem in real-time through the solution of a motion constrained optimization problem. First, a LTV approximation of the model is presented. The model considers the presence of the velocity and acceleration of the desired trajectory in the form of the measured disturbances $v(t) \in \mathbb{R}^4$, a vector of known but unmodifyable model inputs. The MPC formulation and the applied constraints are finally addressed.

A. LTV Model Approximation

As discussed in [28], the MPC formulation requires a discrete-time linear (or linearized) model to construct the optimization problem. Therefore, the model (5) is linearized and discretized. The linearization is performed around a series of predicted states $\tilde{e}(t)$ obtained through numerical integration of (5). In particular, the nonlinear error dynamics (5) can be approximated by the following LTV system

$$\dot{e}(t) = A(\tilde{e}(t), v(t))e(t) + B_u u(t) + B_v(\tilde{e}(t))v(t), \quad (10)$$

where $u(t) \in \mathbb{R}^2$ is the model input vector, $v(t) \in \mathbb{R}^4$ is the vector of measured disturbances and $\tilde{e}(t) \in \mathbb{R}^5$ is the predicted state. More specifically, we define

$$u(t) = \begin{bmatrix} a_r \\ \varepsilon_r \end{bmatrix} \quad v(t) = \begin{bmatrix} v_{rd} \\ \omega_{rd} \\ \dot{v}_{rd} \\ \dot{\omega}_{rd} \end{bmatrix} \quad \tilde{e}(t) = \begin{bmatrix} \tilde{e}_{xr}(t) \\ \tilde{e}_{yr}(t) \\ \tilde{e}_{\theta r}(t) \\ \tilde{e}_{vr}(t) \\ \tilde{e}_{\omega r}(t) \end{bmatrix}, \quad (11)$$

and

$$A(\tilde{e}(t), v(t)) = \frac{\partial f}{\partial e}\Big|_{\substack{e(t)=\tilde{e}(t) \\ v(t)}} \; B_v(\tilde{e}(t)) = \frac{\partial f}{\partial v}\Big|_{e(t)=\tilde{e}(t)}$$

$$B_u = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T, \quad (12)$$

which represents the fact that matrices are obtained performing the linearization around $(\tilde{e}(t), v(t))$, both evaluated at time t. The discrete-time equivalent model of (10), defined with sampling time $T_s > 0$, can then be obtained following the procedure given in [29]. Denoting with $k \in \mathbb{Z}$ the discrete time variable, we get

$$e[k+1] = A[k]\, e[k] + B_u[k]\, u[k] + B_v[k]\, v[k], \quad (13)$$

with

$$A[k] = e^{A(\tilde{e}[k], v[k])T_s}, \quad (14)$$

$$B_u[k] = \int_0^{T_s} e^{A(\tilde{e}[k], v[k])\tau} d\tau B_u(\tilde{e}[k]), \quad (15)$$

$$B_v[k] = \int_0^{T_s} e^{A(\tilde{e}[k], v[k])\tau} d\tau B_v(\tilde{e}[k]). \quad (16)$$

B. LTV MPC Formulation

The idea behind the MPC formulation is to optimize the future behaviour across a finite prediction horizon of p steps. At every discrete-time instant k, for a given state estimate $e[k]$, the optimal control input is computed solving the following constrained quadratic programming (QP)

$$\min_{z_k} \quad J(z_k, e[k]) \quad (17a)$$

$$\text{s.t.} \quad Mz_k < b, \quad (17b)$$

$$u_m[k+i] \leq u[k+i] \leq u_M[k+i], i = 0 \dots p-1 \quad (17c)$$

$$\Delta u_m[k+i] \leq \Delta u[k+i] \leq \Delta u_M[k+i], \quad (17d)$$

$$i = 0 \dots p-1 \quad (17e)$$

where

$$z_k = \big[u[k]^T\, f_c^T[k]\, u[k+1]^T\, f_c^T[k+1] \; \dots \; \dots \; u[k+p-1]^T\, f_c^T[k+p-1]\big]^T \quad (18)$$

is the QP decision variable containing the input vector $u[k+i]$ (that collects the linear and angular acceleration of the robots) and $f_c[k+i]$ (that collects the forces imposed on the pushed object) for $i = 0, \dots, p-1$. As will be detailed in Section V-C, such a definition of the decision variable allows us to consider the physical limitations of the robot inside the QP problem, even though the contact forces are not considered in the robot model. Besides, the cost function in (17) is defined as the following quadratic function

$$J(z_k, e[k]) = \sum_{i=0}^{p-1} \{[e[k+i]^T Q e[k+i]] + [u[k+i]^T R_u u[k+i]] + [\Delta u[k+i]^T R_{\Delta u} \Delta u[k+i]]\} + e[k+p]^T P e[k+p]. \quad (19)$$

The diagonal matrices $Q, P \in \mathbb{R}^{5\times5}$ provide the weights associated with each state variable, while $R_u, R_{\Delta u} \in \mathbb{R}^{2\times2}$ contain the weights on the amplitude of the input and the amplitude of its rate of change respectively. The terminal weight P is introduced to improve stability, as discussed in [30]. These matrices are all positive semidefinite. Inequality (17b) expresses a pushing interaction constraint, which will be described in details in Section V-C. Expression (17c) imposes upper and lower limits on the elements of the QP decision variable z_k, while (17e) sets limits on its rate of change. These constraints are imposed to guarantee the feasibility of the solution, taking into account the physical limitations of the robot actuators.

C. Pushing Constraints for Object Slippage Avoidance

Since the robot is subject to nonholonomic constraints, it cannot change its orientation instantaneously. The direction of the force applied to the pushed object is thus constrained as well. Besides, as previously discussed, the pushing force must be restricted within the friction cone to avoid object slippage during manipulation. Hence, we will now introduce a constraint for the robot motion, such that the contact between the robot and the object does not break. This allows us to guarantee that the movement of the robot produces valid pushing forces, that lie within the friction cone. As a consequence, the input for the robot does not generate any relative motion between the object and the robot itself.

The concept above is implemented imposing the following constraints

$$\begin{bmatrix}\dot{x}_r\\ \dot{y}_r\\ \dot{\theta}_r\end{bmatrix} + \omega_r \times R(\theta_r)p_{or} = \begin{bmatrix}\dot{x}_o\\ \dot{y}_o\\ \dot{\theta}_o\end{bmatrix} = HGf_c, \qquad (20)$$

where $p_{or} \in \mathbb{R}^2$ is the position of the object in the robot frame Σ_r, $R(\theta_r) \in SO(2)$ is the rotation matrix between Σ_r and Σ_w. The left-hand side of (20) represents the velocity that the object would have if the robot-object system were moving as a rigid body (i.e., no relative motion). The right-hand side expresses the motion of the object due to contact forces, as explained in Section IV-B.

In order to include (20) inside the optimization problem (17), some adjustments are required. In particular, to ensure that the contact forces lie inside the friction cone, each component of f_c is bounded to be greater than zero. The equality constraint in (20) is thus converted into a set of two double inequalities, of the form $g(z_k) \leq 0$, and linearized, at each time step k, around the point $(\tilde{e}[k], v[k], p_{or}[k])$. Matrix $M \in \mathbb{R}^{6p\times6p}$ in (17b) is finally defined as the Jacobian matrix of the left-hand side of the inequalities, computed with respect to variable z_k, while vector $b \in \mathbb{R}^{6p}$ is a zero vector.

VI. Implementation and Experiments

In this section, we discuss the implementation of the pushing system and the results of three different manipulation tests, which are representative of different operative conditions. The robot used during the simulation is a Pioneer 3-DX, a differential drive mobile robot with two actuated wheels and a castor wheel. The robot is equipped with a pushing bumper attached on the front. The robot receives velocity commands in the form v_r, ω_r through ROS [31]. At each time step k, with period $T_s = 0.1s$, the following procedure is performed. The controller first sends the velocity command to the robot, then collects the data required to predict the future behaviour and generate the linearized models. The solution of the quadratic problem discussed in section V-B is then computed and used to generate the velocity command for the future step.

Algorithm 1: Feedback procedure

send previous (v_r, ω_r)
get $e[k], v[k]$
$\tilde{e}[k] \leftarrow$predict($e[k]$,$v[k]$,z_{k-1})
$A_k, B_{u_k}, B_{v_k}, M \leftarrow$linearize($\tilde{e}[k], v[k]$)
$z_k \leftarrow$ QP($A_k, B_{u_k}, B_{v_k}, M, \tilde{e}[k], v[k]$)
$(v_r, \omega_r) \leftarrow (v_r, \omega_r) + T_s u[k]$
$z_{k-1} \leftarrow z_k$

Two case studies have been carried out in simulations, performed on a laptop with an Intel Core i7-4510U using the CoppeliaSym physics simulator. The validated controller is written in MATLAB, while ROS handles the communication with the simulator. The gains are experimentally tuned to $Q = diag([15, 20, 5, 1, 1])$, $P = 50Q$, $R_u = diag([0.1, 0.1, 0.001, 0.001, 0.001, 0.001])$, and $R_{\Delta u} = diag([0.1, 0.1, 0, 0, 0, 0])$. The results of the simulations are discussed below and are reported in the accompanying video. The video also reports the results of preliminary experiments, performed with a real robotic system in a laboratory environment.

A. Tracking along a straight line

The first case study we propose is the tracking of a straight line starting with an offset. Several simulations have been performed, with the robot starting its movement with initial error state $e(0) = \begin{bmatrix}0 & \varphi & 0 & 0 & 0\end{bmatrix}^T$, with varying $\varphi \in \{0.1, 0.2, 0.4, 0.5\}$. A representative run of the simulations, performed with $\varphi = 0.2$, is discussed hereafter. The pushed object is placed in contact with the robot in a centered position. Figure 4 depicts the planar movement of the robot and the object, measured for a representative run of the simulations. The yellow line represents the desired trajectory while the blue line and red line depicts the movement of the robot and the object, respectively. The figure clearly shows that an initial position offset can be corrected using the proposed controller and constraints. Figure 5 shows a comparison of the y components of the object position with respect to Σ_r, while being pushed, with and without the presence of the constraint in the controller. The application of the constraint significantly reduces the amplitude of the movement of the pushed object. The same conclusion can be extracted from Figure 6, that shows the positions of the object and robot controlled without the constraint (20).

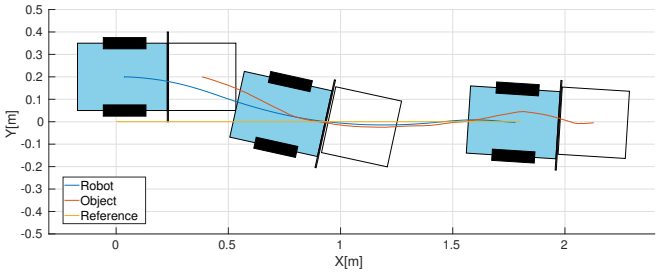

Fig. 4. Line tracking from a non-zero initial error state using the proposed controller and constraints.

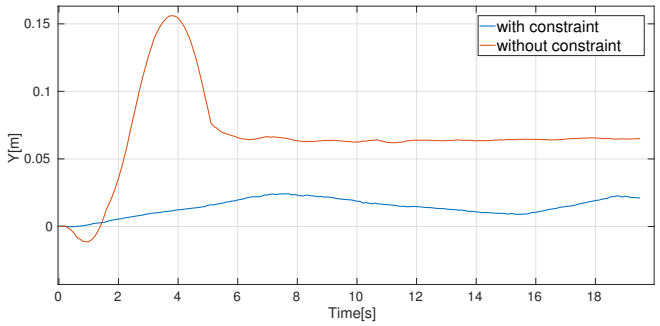

Fig. 5. Y position of the pushed object with respect to Σ_r during the manipulation.

Moreover, Figure 6 shows that, during the manipulation, the movement of the robot causes an interruption of the pushing line contact, also reflected in the spike visualized in Figure 5, that results in a loss of control over the movement of the object and ultimately in a loss of quality of the manipulation. The proposed controller and constraints maintain the line contact, with a final average object position error less than 0.01 m, while the absence of the constraint on the motion leads to an average error greater than 0.05 m.

B. Complete manipulation task

The second case study we explore is a complete manipulation. To complete the task, the robot transports the object to a desired configuration performing a series of pushing actions. Once a pushing maneuver is completed, the robot performs a repositioning maneuver to change the pushing side before starting the next action. During the maneuver the robot steps back from the object, goes around the object along a circular trajectory and then approaches slowly the object until the contact is established. Several simulations have been performed, considering different trajectories composed of straight and curve segments. The trajectory traveled by the robot and the object during a representative run of the simulations is depicted in Figure 7. In this task the robot transports the object from the initial position $\chi_o(0) = \begin{bmatrix}0.0 & 0.0 & 0.0\end{bmatrix}^T$ towards the desired configuration $\begin{bmatrix}1.65 & 0.15 & \pi\end{bmatrix}^T$ performing three pushing actions on the object. The results indicate that, with the use of the proposed controller, it is possible to track a curved line to transport the package without significant accumulation of error, that implies that the robot, when an appropriate reference trajectory is provided, can manipulate the object

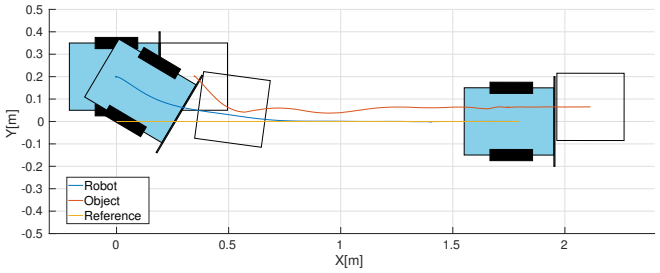

Fig. 6. Line tracking from a non-zero initial error state using the controller without the proposed constraint.

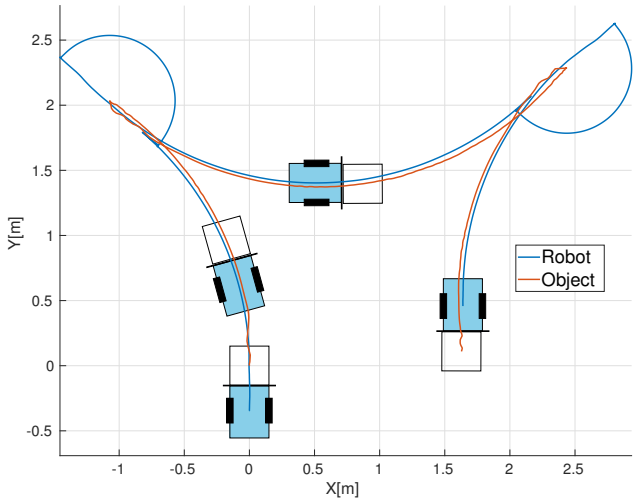

Fig. 7. Complete manipulation of the object.

across the environment. The final positon error is 0.04 m while the final orientation error is 1.8 deg.

VII. Conclusion and Future Work

In this paper, we investigated the problem of manipulating an object in a bi-dimensional environment by pushing with a nonholonomic mobile robot. In particular, we designed a predictive controller for the mobile robot with an appropriate set of constraints to ensure the correct manipulation, providing stiff contact with the object. Numerical case studies are presented, while early-stage experiments are shown in the multimedia attachment.

Future work is focused on consolidating the experimental validation of the proposed approach, taking into account possible external disturbances. We would also like to include our previous work developed in [21] within the proposed framework. Besides, we would like to extend this work to the case of a multi-robot system. Differently from what presented in Section II, the multi-robot system should not resemble a pick-and-place operation, but each agent must perform nonprehensile manipulation through pushing.

References

[1] F. Ruggiero, V. Lippiello, and B. Siciliano, "Nonprehensile dynamic manipulation: A survey," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1711–1718, 2018.

[2] K. M. Lynch and M. T. Mason, "Stable pushing: Mechanics, controllability, and planning," *The International Journal of Robotics Research*, vol. 15, no. 6, pp. 533–556, 1996.

[3] L. Sabattini, M. Aikio, P. Beinschob, M. Boehning, E. Cardarelli, V. Digani, A. Krengel, M. Magnani, S. Mandici, F. Oleari, C. Reinke, D. Ronzoni, C. Stimming, R. Varga, A. Vatavu, S. Castells Lopez, C. Fantuzzi, A. Mäyrä, S. Nedevschi, C. Secchi, and K. Fuerstenberg, "The PAN-Robots project: Advanced automated guided vehicle systems for industrial logistics," *IEEE Robotics Automation Magazine*, vol. 25, no. 1, pp. 55–64, March 2018.

[4] P. Falcone, F. Borrelli, J. Asgari, E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *Control Systems Technology, IEEE Transactions on*, vol. 15, pp. 566 – 580, 06 2007.

[5] F. R. Hogan, E. R. Grau, and A. Rodríguez, "Reactive planar manipulation with convex hybrid mpc," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 247–253, 2017.

[6] Z. Wang and M. Schwager, "Force-amplifying n-robot transport system (force-ANTS) for cooperative planar manipulation without communication," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1564–1586, 2016.

[7] A. Yamashita, T. Arai, J. Ota, and H. Asama, "Motion planning of multiple mobile robots for cooperative manipulation and transportation," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 2, pp. 223–237, 2003.

[8] H. F. McCreery and M. Breed, "Cooperative transport in ants: a review of proximate mechanisms," *Insectes Sociaux*, vol. 61, no. 2, pp. 99–110, 2014.

[9] K. Ohashi, R. Takahashi, M. Takimoto, and Y. Kambayashi, "Cooperative transportation of a rod using mobile agents," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, Dec 2014, pp. 1019–1026.

[10] D. Prattichizzo and J. Trinkle, "Grasping," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer International Publishing, 2016, pp. 955–988.

[11] J. Fink, M. A. Hsieh, and V. Kumar, "Multi-robot manipulation via caging in environments with obstacles," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 1471–1476.

[12] G. Habibi, Z. Kingston, W. Xie, M. Jellins, and J. McLurkin, "Distributed centroid estimation and motion controllers for collective transport by multi-robot systems," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1282–1288.

[13] Z. Wang and V. Kumar, "Object closure and manipulation by multiple cooperating mobile robots," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 1. IEEE, 2002, pp. 394–399.

[14] Y.-H. Kim and D. A. Shell, "Using a compliant, unactuated tail to manipulate objects," *IEEE Robotics and Automation Letters*, vol. 2, no. 1, pp. 223–230, 2017.

[15] T. Maneewarn and P. Detudom, "Mechanics of cooperative nonprehensile pulling by multiple robots," in *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*. IEEE, 2005, pp. 2004–2009.

[16] P. Kolhe, N. Dantam, and M. Stilman, "Dynamic pushing strategies for dynamically stable mobile manipulators," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 3745–3750.

[17] S. Krivic, E. Ugur, and J. Piater, "A robust pushing skill for object delivery between obstacles," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug 2016, pp. 1184–1189.

[18] K. Kovac, I. Zivkovic, and B. D. Basic, "Simulation of multi-robot reinforcement learning for box-pushing problem," in *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (IEEE Cat. No.04CH37521)*, vol. 2, May 2004, pp. 603–606 Vol.2.

[19] T. Igarashi, Y. Kamiyama, and M. Inami, "A dipole field for object delivery by pushing on a flat surface," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 5114–5119.

[20] M. A. Golkar, S. T. Namin, and H. Aminaiee, "Fuzzy controller for cooperative object pushing with variable line contact," in *Mechatronics, 2009. ICM 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–6.

[21] F. Bertoncelli, F. Ruggiero, and L. Sabattini, "Wheel slip avoidance through a nonlinear model predictive control for object pushing with a mobile robot," in *10th IFAC Symposium on Intelligent Autonomous Vehicles*, 2019.

[22] G. Oriolo, A. De Luca, and M. Vendittelli, "Wmr control via dynamic feedback linearization: design, implementation, and experimental validation," *IEEE Transactions on control systems technology*, vol. 10, no. 6, pp. 835–852, 2002.

[23] D. Prattichizzo and J. C. Trinkle, *Grasping*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 671–700. [Online]. Available: https://doi.org/10.1007/978-3-540-30301-5˙29

[24] J. Xie and N. Chakraborty, "Rigid body dynamic simulation with line and surface contact," in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, Dec 2016, pp. 9–15.

[25] J. Z. Woodruff and K. M. Lynch, "Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4066–4073.

[26] S. Goyal, A. Ruina, and J. Papadopoulos, "Planar sliding with dry friction part 1. limit surface and moment function," 1991.

[27] S.-H. Lee and M. R. Cutkosky, "Fixture planning with friction," 1991.

[28] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017.

[29] G. F. Franklin, M. L. Workman, and D. Powell, *Digital Control of Dynamic Systems*, 3rd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

[30] L. Grne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer Publishing Company, Incorporated, 2013.

[31] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.