This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2025.3564209

Singularity-free trajectory tracking for steerable wheeled mobile robots

Michele Cipriano, Giuseppe Oriolo, Andrea Cherubini

***Abstract*—Omnidirectional robots can be realized using Mecanum wheels or using a suitable arrangement of conventional steerable wheels. The latter group, known as omnidirectional steerable wheeled mobile robots (SWMRs), are known to have a lower cost with respect to the former, and to be more robust thanks to the presence of conventional wheels. Nevertheless, their modeling and control is complex, due to the presence of singularities in their representation. This paper proposes a framework for trajectory tracking of SWMRs using Nonlinear Model Predictive Control (NMPC) based on a real-time iteration scheme. The NMPC generates feasible motions for the robot, avoiding model singularities of the mobile base, together with bounds on driving and steering velocities on the wheels. Our NMPC works alongside a finite state machine, responsible for singularity avoidance during starting and stopping motion, and a state trajectory generation scheme based on dynamic feedback linearization, which makes our framework capable of tracking any trajectory. Our approach is validated on a Neobotix MPO-700 on various trajectories.**

***Index Terms*—Motion Control; Wheeled Robots**

I. Introduction

OMNIDIRECTIONAL SWMRs (Fig. 1) have greater maneuverability than other wheeled mobile robots [1], can transport higher payloads with respect to omnidirectional robots equipped with Mecanum wheels or with omni wheels [2], and can operate also on inclined surfaces. Nevertheless, modeling and controlling these robots is not trivial, due to the presence of model singularities [3], which need to be handled with particular care, to avoid affecting the robot functionalities.

While many different approaches for modeling and control of omnidirectional SWMRs exist in the literature, none of them fully exploits the potentialities of these platforms. The main property of this kind of robots, indeed, is that their Instantaneous Center of Rotation (ICR) can be located anywhere on the plane [4]. This causes a parametrization, based on two-dimensional Cartesian [5] or on polar coordinates [6], leading to singularities, which can make it difficult to develop a control scheme. Sorour et al. [3] developed an ICR-based controller which handles singularities of the steering axes. That work was further improved in [7], where the singularity of the ICR at infinity is taken into account through a complementary route strategy. Although these approaches consider all parameterization singularities, the velocity and acceleration bounds are only considered at the level of the ICR, often resulting in undesired motions with high velocity and high acceleration of the steerable wheels. A singularity-free representation is presented in [8] and [9], and used in [10], where a singularity-free motion controller is developed. Here, time scaling is performed, to satisfy velocity and acceleration constraints on the wheels, resulting in non-optimal motion execution.

Fig. 1: Neobotix MPO-700 steerable wheeled mobile robot.

In this paper, we consider the problem of trajectory tracking for an omnidirectional SWMR, equipped with two or more actuated caster wheels. The robot is required to follow a user-defined reference pose trajectory in an environment free of obstacles, without violating the driving and steering velocity constraints of each wheel. Trajectory tracking in robotics is solved traditionally with a variety of methods, see [11], [12]. Here, we propose a framework which makes use of NMPC [13]. While many existing works use MPC on differential drive robots [14], on autonomous vehicles such as cars [15] or tractor-trailers [16], and on wheeled-legged robots [17], the application of MPC to SWMRs has yet to be explored.

Our NMPC is supported by:

- a *finite state machine*, responsible for starting and stopping the motion of the robot, while guaranteeing that it never encounters model singularities,
- a *state trajectory generation scheme* based on dynamic feedback linearization [18], which generates reference configurations and control inputs for the NMPC itself, given the reference pose trajectory.

Manuscript received: November 20, 2024; Revised: January 24, 2025; Accepted: March 24, 2025.

This paper was recommended for publication by Editor Lucia Pallottino upon evaluation of the Associate Editor and Reviewers' comments.

Michele Cipriano and Giuseppe Oriolo are with the Dipartimento di Ingegneria Informatica, Automatica e Gestionale, Sapienza Università di Roma, Italy. E-mail: *lastname*@diag.uniroma1.it.

Andrea Cherubini is with École Centrale de Nantes, France. E-mail: andrea.cherubini@ec-nantes.fr.

Digital Object Identifier (DOI): see top of this page.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 06,2025 at 09:24:04 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2025.3564209

The NMPC is formulated as a nonlinear programming problem, and solved using the real-time iteration scheme [19].

The contributions of our paper, with respect to the reviewed literature, are the following:

- we propose a framework for trajectory tracking of SWMR, which generates motions that satisfy driving and steering velocity bounds on all wheels;
- our NMPC is, to the best of our knowledge, the first model predictive controller to be applied to SWMRs.

Furthermore, considering the taxonomy of singularities presented in [3], our framework:

- cannot encounter the singularity of ICR on one of the steerable axes, thanks to the NMPC constraints;
- cannot encounter singularities due to the mobile base's zero velocity, thanks to the finite state machine (which, as will be explained in Sect. III-A, makes use of singularity-free kinematic models);
- does not present a singularity when the ICR is at infinity, thanks to our parametrization.

The paper is structured as follows. Section II presents the kinematic model of the mobile base, and discusses in detail its singularities. Section III introduces the proposed framework, describing the finite state machine, the state trajectory generation scheme, and the NMPC. Section IV and V validates the proposed framework on multiple simulations and experiments, which have been performed using a Neobotix MPO-700. Section VI concludes the paper, and discusses future works.

II. Kinematic Model

In this section, we will develop the kinematic model of a SWMR, following the analysis presented in [1]. Note that, while our mobile base is equipped with off-centered steerable wheels, its kinematic model is identical to the one described in [1], which considers centered steerable wheels. In general, when using off-centered wheels one can expect greater maneuverability and higher payload.

Consider an SWMR equipped with $n \geq 2$ steerable wheels driven by independent motors. With reference to Fig. 2, we will denote with $\boldsymbol{\xi} = (x, y, \theta) \in SE(2)$ the pose of the mobile base, with (x, y) the position of a representative point, and θ its orientation. Denote by S_i the point where the axis of the i-th joint intersects the plane of motion, and by W_i the projection on the same plane of the i wheel center. Let $\boldsymbol{p}_i^S$ and $\boldsymbol{p}_i^W$ respectively be their positions in the world frame $\mathcal{F}$, and β_i be the steering angle of the i-th wheel measured with respect to the mobile frame $\mathcal{F}'$ (see Fig. 2). Each wheel is also controlled by two independent velocities, the driving velocity v_i^W and the steering velocity ω_i, which are taken as control inputs. We define the whole robot configuration via $\boldsymbol{q} = (\boldsymbol{\xi}, \boldsymbol{\beta})$, where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_n)$.

The position of the i-th steering joint S_i is defined as

$$\boldsymbol{p}_i^S = \begin{pmatrix} x \\ y \end{pmatrix} + \boldsymbol{R}(\theta) \begin{pmatrix} b_i \\ a_i \end{pmatrix},$$

and the position of the i-th wheel W_i is defined as

$$\boldsymbol{p}_i^W = \boldsymbol{p}_i^S + \boldsymbol{R}(\theta + \beta_i) \begin{pmatrix} 0 \\ -d \end{pmatrix}.$$

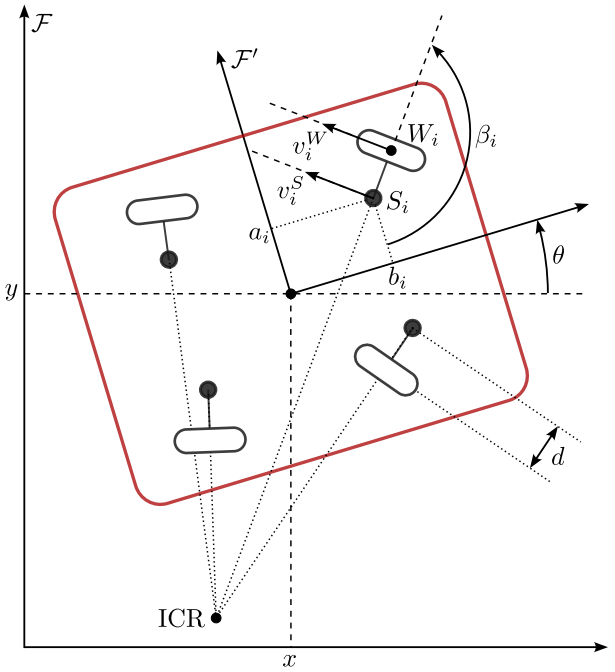


Fig. 2: Schematic model of an SWMR. Note that, even if the figure represents a robot equipped with four wheels, our approach is generic and works with an arbitrary number of wheels.

where $\boldsymbol{R} \in SO(2)$ is a rotation matrix, and with b_i, a_i, d defined in Fig. 2.

Under to the assumption of no slipping (i.e., wheel contact point velocity orthogonal to wheel zero motion line), each wheel is subject to the Pfaffian constraint:

$$\begin{pmatrix} -\sin(\theta + \beta_i) & \cos(\theta + \beta_i) \end{pmatrix} \dot{\boldsymbol{p}}_i^W = 0. \tag{1}$$

By combining the above equations, it is possible to rearrange the n constraints in matrix form

$$\underbrace{\begin{pmatrix} -\sin(\theta + \beta_1) & \cos(\theta + \beta_1) & \Delta_1 & 0 \dots 0 \\ -\sin(\theta + \beta_2) & \cos(\theta + \beta_2) & \Delta_2 & 0 \dots 0 \\ \vdots & \vdots & \vdots & \vdots \\ -\sin(\theta + \beta_n) & \cos(\theta + \beta_n) & \Delta_n & 0 \dots 0 \end{pmatrix}}_{\boldsymbol{A}^T(\boldsymbol{q})} \dot{\boldsymbol{q}} = 0, \tag{2}$$

with $\Delta_i = b_i \cos\beta_i + a_i \sin\beta_i$.

For the mobile base to perform a motion without slipping, all wheel axles must instantaneously intersect at the same point, the ICR. The existence of an ICR can also be seen as a geometric constraint (ICR constraint), which requires all wheel orientations to be coordinated. In the following, we will study how the ICR constraint affects the robot mobility.

A. ICR constraint satisfied

Whenever the robot configuration is such that there exists an ICR, it is possible to derive a kinematic model through the use of *coordinating functions* for β_i [1], with $i \geq 2$. The idea is to let the ICR be defined by the trajectory of $\boldsymbol{\xi}$. In the following, we will assume that $\boldsymbol{\xi}$ is twice differentiable and persistent

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 06,2025 at 09:24:04 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2025.3564209

(i.e., the platform is always in motion). Considering the i-th constraint in (2) and solving for β_i yields the coordinating function[1]

$$\beta_i(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) = \arctan\frac{-\dot{x}\sin\theta + \dot{y}\cos\theta + b_i\dot{\theta}}{\dot{x}\cos\theta + \dot{y}\sin\theta - a_i\dot{\theta}},\ i = 2, \dots, n. \tag{3}$$

The last $n-1$ constraints of (2) are automatically satisfied as a result, and it is hence possible to consider only the first constraint and the first four coordinates:

$$\begin{pmatrix} -\sin(\theta+\beta_1) & \cos(\theta+\beta_1) & \Delta_1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ \theta \\ \beta_1 \end{pmatrix} = 0, \tag{4}$$

Assuming the robot is controlled at the acceleration level, it is easy to obtain the reduced kinematic model

$$\begin{aligned} \dot{x} &= v_1^S\cos(\theta+\beta_1) + \omega(b_1\sin\theta + a_1\cos\theta) \\ \dot{y} &= v_1^S\sin(\theta+\beta_1) + \omega(-b_1\cos\theta + a_1\sin\theta) \\ \dot{\theta} &= \omega \\ \dot{\beta}_1 &= \omega_1 \\ \dot{v}_1^S &= a \\ \dot{\omega} &= a_\omega, \end{aligned} \tag{5}$$

with $\boldsymbol{\chi} = (x, y, \theta, \beta_1, v_1^S, \omega)$ denoting the robot state, and $\boldsymbol{u} = (a, a_\omega, \omega_1)$ denoting the control inputs. In the following, we will denote model (5) as $\dot{\boldsymbol{\chi}} = \boldsymbol{f}(\boldsymbol{\chi}, \boldsymbol{u})$.

Note that the angular velocity ω does not represent a direct input, but it is realized by imposing to β_i the orientations specified by

$$\beta_i(v_1^S, \omega, \beta_1) = \arctan\frac{v_1^S\sin\beta_1 + \omega(b_i - b_1)}{v_1^S\cos\beta_1 + \omega(a_1 - a_i)}, \tag{6}$$

which can be obtained from (3).

Coordinating functions present a singularity whenever S_i is stationary (i.e., $\dot{\boldsymbol{p}}_i^S = \boldsymbol{0}$). This needs to be considered when designing a controller. Note that this occurs only in two cases: when the platform is stationary ($v_1^S = 0, \omega = 0$) or when the ICR position coincides with that of one of the S_i's ($v_1^S = (b_1 - b_i)/\sin\beta_1 = (a_i - a_1)/\cos\beta_1$).

If the position of the ICR does not change through time (which happens when all $\dot{\beta}_i = 0$), and if the ICR does not coincide with any of the joint positions S_i, all coordinating functions are free of singularity, as long as the platform is moving.

B. ICR constraint not satisfied

Whenever the robot configuration is such that no ICR exists, the null space of $\boldsymbol{A}^T(\boldsymbol{q})$ in (2) reduces to the trivial one [1]. The kinematic model of the robot is

$$\dot{\beta}_i = \omega_i,$$

with ω_i steering velocities. In this case, the robot pose is constant, and it is only possible to control the steering joints. In this particular case, the driving velocity of the i-th wheel is given by $v_i^W = d\omega_i$.

[1]In practice, our implementation uses atan2 to produce a single result.

III. Proposed Framework

This section describes the main components of our framework (shown in Fig. 3), namely: the finite state machine, responsible for starting and stopping the robot; the state trajectory generator, which provides reference trajectories to the NMPC; and the NMPC itself, which computes control inputs, while satisfying the driving and steering velocity bounds of each wheel. In all phases of motion, model singularities are avoided.

A user-defined reference pose trajectory $\boldsymbol{\xi}^{\mathrm{ref}}(t)$ is fed to a Finite State Machine (FSM), which determines when to start/stop robot motion, hence the corresponding control strategy. The mobile base is accelerated (using open-loop commands) until all wheel driving velocities are non null. When this condition is met, the NMPC takes full control of the robot motion. In this case, a state trajectory generation scheme based on dynamic feedback linearization computes the trajectories $\boldsymbol{\chi}^{\mathrm{ref}}$ and $\boldsymbol{u}^{\mathrm{ref}}$ (using $\boldsymbol{\xi}^{\mathrm{ref}}(t)$), which are used by the NMPC to compute control inputs (a, a_ω, ω_1). Note that it is always possible to realize the given $\boldsymbol{\xi}^{\mathrm{ref}}(t)$ (whatever it is) with a motion that avoids model singularities. Therefore, using the NMPC (which enforces this avoidance) does not imply any limitation for motion generation.

A. Finite state machine

Since the ICR constraint may be not satisfied at initialization, and since the NMPC must avoid configurations in which coordinating functions (6) are singular, we designed a finite state machine (FSM) to move the robot towards a configuration free of singularity.

The FSM, shown in Fig. 4, consists of five states, defined – along with the triggering events – as follows.

- *NoICR*. The configuration of the robot is such that the ICR constraint is not satisfied. In this state, the wheels are regulated to a user-defined configuration using a proportional controller. Once the ICR constraint is satisfied, the state of the FSM becomes *Ready*.
- *Ready*. The configuration satisfies the ICR constraint, and the robot is not moving. Once a new trajectory is available, the state of the FSM becomes *Starting*.
- *Starting*. In this state, the control of the robot is given to an open loop controller, which starts its motion. Once all velocities $\dot{\boldsymbol{p}}_i^S$ become non-null, the state of the FSM becomes *Moving*.
- *Moving*. In this state, the robot moves under the action of the NMPC. When the trajectory tracking task is about to be be completed[2], the state of the FSM becomes *Stopping*.
- *Stopping*. As for *Starting*, an open loop motion makes the mobile base reduce its speed until it stops. Then, the state of the FSM becomes *Ready*.

B. Open-loop commands (starting and stopping)

In this section, we present our singularity-free strategy for handling starting and stopping motions. To this end, we

[2]We use a threshold on the pose of the robot to determine the completion of the tracking task, considering the last pose specified by $\boldsymbol{\xi}^{\mathrm{ref}}(t)$.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 06,2025 at 09:24:04 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2025.3564209

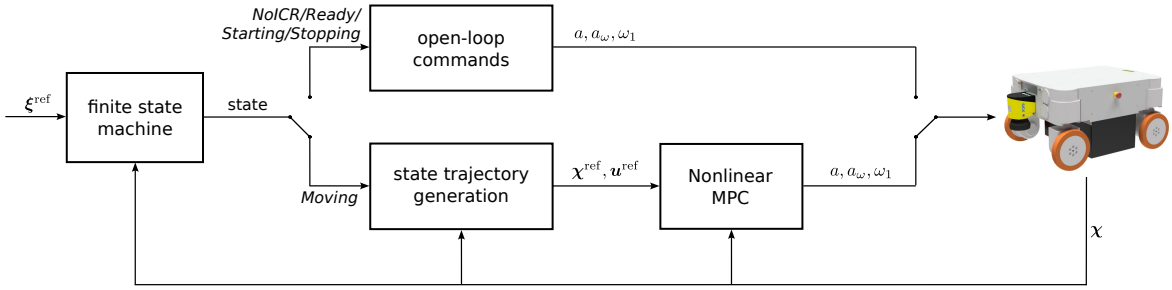


Fig. 3: Block scheme of the proposed framework.

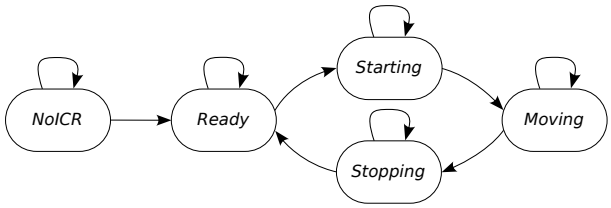


Fig. 4: Finite state machine defining the base motion.

constrain the ICR to be constant, accelerating (respectively, decelerating) the base along the arc of circle defined by initial robot position and initial ICR, until all velocities $\dot{\boldsymbol{p}}_i^S$ are non-null (respectively, null). Note that the radius of the circle is given by $R = v_1^S/\omega$. Moreover, the arc of circle degenerates in a line when the ICR is at infinity.

We choose the control inputs as follows:

- when the state of the FSM is *Starting*, we accelerate the mobile base by choosing $\boldsymbol{u} = (a^{\text{init}}, a^{\text{init}}/R, 0)$;
- when the state of the FSM is *Stopping*, we decelerate the mobile base by choosing $\boldsymbol{u} = (-K^{\text{stop}} v_1^S, -K^{\text{stop}} v_1^S/R, 0)$;

where a^{init} is a parameter, and $K^{\text{stop}} > 0$.

Note that, when the ICR is constant at infinity, the coordinating functions simplify to

$$\beta_i = \beta_1,$$

and when the ICR is constant, but not at infinity, they simplify to

$$\beta_i = \frac{R \sin\beta_1 + b_i - b_1}{R\cos\beta_1 + a - a_i}.$$

C. State trajectory generation

Once the velocities $\dot{\boldsymbol{p}}_i^S$ become non-null, the state becomes *Moving*, and the robot is controlled by the NMPC. In this section, we present the state trajectory generation scheme based on dynamic feedback linearization [18], which computes state configurations and control input trajectories for the NMPC, given a reference pose trajectory $\boldsymbol{\xi}^{\text{ref}}$ of the mobile base. Note that both state trajectory generation and NMPC are only active when the state is *Moving*.

Consider the output function $\boldsymbol{z}(\boldsymbol{\chi}) = \boldsymbol{\xi}$. By deriving it twice, we obtain

$$\ddot{\boldsymbol{z}}(\boldsymbol{\chi}) = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{pmatrix} = \boldsymbol{m}(\boldsymbol{\chi}) + \boldsymbol{H}(\boldsymbol{\chi}) \begin{pmatrix} a \\ a_\omega \\ \omega_1 \end{pmatrix}, \tag{7}$$

with $\boldsymbol{m}(\boldsymbol{\chi}) \in \mathbb{R}^3$ and $\boldsymbol{H}(\boldsymbol{\chi}) \in \mathbb{R}^{3\times 3}$ defined as

$$\boldsymbol{m}(\boldsymbol{\chi}) = \begin{pmatrix} -\sin(\theta+\beta_1)\omega v_1^S + (b_1\cos\theta - a_1\sin\theta)\omega^2 \\ \cos(\theta+\beta_1)\omega v_1^S + (a_1\cos\theta - b_1\sin\theta)\omega^2 \\ 0 \end{pmatrix},$$

$$\boldsymbol{H}(\boldsymbol{\chi}) = \begin{pmatrix} \cos(\theta+\beta_1) & b_1\sin\theta + a_1\cos\theta & -\sin(\theta+\beta_1)v_1^S \\ \sin(\theta+\beta_1) & -b_1\cos\theta + a_1\sin\theta & \cos(\theta+\beta_1)v_1^S \\ 0 & 1 & 0 \end{pmatrix}.$$

By choosing

$$\boldsymbol{u} = \begin{pmatrix} a \\ a_\omega \\ \omega_1 \end{pmatrix} = \boldsymbol{H}(\boldsymbol{\chi})^{-1}\left(\boldsymbol{\eta} - \boldsymbol{m}(\boldsymbol{\chi})\right),$$

we can transform (7) into an equivalent chain of integrators

$$\ddot{\boldsymbol{z}} = \boldsymbol{\eta},$$

which can be easily stabilized. Indeed, exponential regulation of the trajectory tracking error $\boldsymbol{e}(t) = \boldsymbol{z}^{\text{ref}}(t) - \boldsymbol{z}(t)$, can be achieved by taking

$$\boldsymbol{\eta} = \ddot{\boldsymbol{z}}^{\text{ref}} + \boldsymbol{K}_P\boldsymbol{e} + \boldsymbol{K}_D\dot{\boldsymbol{e}}, \quad \boldsymbol{K}_P, \boldsymbol{K}_D > 0,$$

with $\boldsymbol{z}^{\text{ref}}(t)$ a twice differentiable and persistent (i.e., $v_1^S \neq 0$) reference trajectory. Note that the above decoupling matrix $\boldsymbol{H}(\boldsymbol{\chi})$ is singular at $v_1^S = 0$. Thanks to the FSM, we can assume that the reference trajectory is not stopping. However, v_1^S can still be zero during a transient. Hence, a singularity-robust inverse should be used. This singularity is structural for mobile robots [18].

At each timestep t_k, Algorithm 1 generates, given the reference trajectory $\boldsymbol{\xi}^{\text{ref}}$ and the current configuration $\boldsymbol{\chi}_k$, the state configurations $\boldsymbol{\chi}_{j|k}^{\text{ref}}$ $(j = 0, \ldots, N)$, together with the state control inputs $\boldsymbol{u}_{j|k}^{\text{ref}}$ $(j = 0, \ldots, N-1)$. These are synchronized and used by the NMPC, described in the next section, to compute control inputs $(a_{1,k}^S, a_{\omega,k}, \omega_{1,k})$ for the mobile base. In the pseudocode: function Sample discretizes a trajectory, given over interval $(t_k, t_k + N\delta)$, into $N+1$ elements, with δ timestep of the NMPC, and function $\boldsymbol{F}$ denotes the discrete-time kinematic model obtained integrating $\boldsymbol{f}$ from (5) with fourth-order Runge-Kutta over timestep δ.

D. Nonlinear Model Predictive Control

The NMPC solves, at each control cycle, a finite horizon constrained Optimal Control Problem (OCP), taking into account the kinematic model (5), wheel velocity and control

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 06,2025 at 09:24:04 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2025.3564209

Algorithm 1: StateTrajectoryGeneration

Input: $\boldsymbol{\xi}^{\text{ref}}, \boldsymbol{\chi}_k$
Output: $\boldsymbol{\chi}^{\text{ref}}_{0|k}, \dots, \boldsymbol{\chi}^{\text{ref}}_{N|k}, \boldsymbol{u}^{\text{ref}}_{0|k}, \dots, \boldsymbol{u}^{\text{ref}}_{N-1|k}$

$\boldsymbol{\xi}^{\text{ref}}_{0|k}, \dots, \boldsymbol{\xi}^{\text{ref}}_{N|k} \leftarrow \text{Sample}(\boldsymbol{\xi}^{\text{ref}})$;
$\dot{\boldsymbol{\xi}}^{\text{ref}}_{0|k}, \dots, \dot{\boldsymbol{\xi}}^{\text{ref}}_{N|k} \leftarrow \text{Sample}(\dot{\boldsymbol{\xi}}^{\text{ref}})$;
$\ddot{\boldsymbol{\xi}}^{\text{ref}}_{0|k}, \dots, \ddot{\boldsymbol{\xi}}^{\text{ref}}_{N|k} \leftarrow \text{Sample}(\ddot{\boldsymbol{\xi}}^{\text{ref}})$;
$\boldsymbol{\chi}^{\text{ref}}_{0|k} \leftarrow \boldsymbol{\chi}_k$;
for $j \leftarrow 0$ **to** $N-1$ **do**
  $\boldsymbol{\eta}_{j|k} \leftarrow \ddot{\boldsymbol{z}}^{\text{ref}}_{j|k} + \boldsymbol{K}_P(\boldsymbol{z}^{\text{ref}}_{j|k} - \boldsymbol{z}_{j|k}) + \boldsymbol{K}_D(\dot{\boldsymbol{z}}^{\text{ref}}_{j|k} - \dot{\boldsymbol{z}}_{j|k})$;
  $\boldsymbol{u}^{\text{ref}}_{j|k} \leftarrow \boldsymbol{H}(\boldsymbol{\chi}^{\text{ref}}_{j|k})^{-1}(\boldsymbol{\eta}_{j|k} - \boldsymbol{m}(\boldsymbol{\chi}^{\text{ref}}_{j|k}))$;
  $\boldsymbol{\chi}^{\text{ref}}_{j+1|k} \leftarrow \boldsymbol{F}(\boldsymbol{\chi}^{\text{ref}}_{j|k}, \boldsymbol{u}^{\text{ref}}_{j|k})$;
end
return $\boldsymbol{\chi}^{\text{ref}}_{0|k}, \dots, \boldsymbol{\chi}^{\text{ref}}_{N|k}, \boldsymbol{u}^{\text{ref}}_{0|k}, \dots, \boldsymbol{u}^{\text{ref}}_{N-1|k}$;

inputs bounds, singularities of the coordinating functions (6), and singularity of the decoupling matrix $\boldsymbol{H}(\boldsymbol{\chi})$ in the state trajectory generation scheme. In the following, we will denote as $\mathbb{I}_a^b = \{a, \dots, b\} \subset \mathbb{N}$ the subset of natural numbers containing all naturals from a to b.

As already mentioned, since the coordinating function β_i is singular when $\dot{\boldsymbol{p}}_i^S = \boldsymbol{0}$, it is important to carefully design the control scheme. A simple strategy to make the NMPC free of singularities, is to never let the position of the i-th steering joint be at rest. Since the NMPC is activated only when changing the FSM state from *Starting* to *Moving*, it is possible to constrain $\dot{\boldsymbol{p}}_i^S$ so that it is never null. Indeed, the constraint $\dot{\boldsymbol{p}}_i^S \neq \boldsymbol{0}$, with a proper change of coordinates, can be rewritten as

$$\boldsymbol{R}^T(\theta + \beta_i)\dot{\boldsymbol{p}}_i^S = \begin{pmatrix} v_i^S \\ 0 \end{pmatrix} \neq \boldsymbol{0},$$

with

$$v_i^S = \begin{pmatrix} \cos(\theta + \beta_i) \\ \sin(\theta + \beta_i) \end{pmatrix}^T \dot{\boldsymbol{p}}_i^S.$$

To satisfy the above inequality, we need to have $v_i^S \neq 0$, which is equivalent to imposing constant $\text{sgn}(v_i^S)$. Note that, because of the starting motion described in Sect. III-B, v_i^S is either positive or negative when the NMPC is activated. This implies that the constraint will simply be written as

$$\begin{cases} v_i^S > 0, & \text{if } v_i^S(t_0) > 0 \\ v_i^S < 0, & \text{otherwise} \end{cases}, \tag{8}$$

with t_0 time of activation of the NMPC. This guarantees that subsequent calls of the state trajectory generation scheme are free of singularities.

The OCP can be defined as

$$\begin{aligned} \min_{\boldsymbol{u}(\cdot)} \quad & \Phi(\boldsymbol{\chi}(t_k + T)) + \int_{t_k}^{t_k+T} \mathcal{L}(\boldsymbol{\chi}, \boldsymbol{u}) dt \\ \text{s.t. } & \dot{\boldsymbol{\chi}} = \boldsymbol{f}(\boldsymbol{\chi}, \boldsymbol{u}) \\ & v^- \leq v_i^W \leq v^+, \ \forall i \in \mathbb{I}_1^n \\ & \dot{\boldsymbol{p}}_i^S \neq \boldsymbol{0}, \ \forall i \in \mathbb{I}_1^n \\ & a^- \leq a \leq a^+ \\ & \omega^- \leq \omega_i \leq \omega^+, \ \forall i \in \mathbb{I}_1^n \\ & \boldsymbol{\chi}(t_k) = \boldsymbol{\chi}_k, \end{aligned}$$

with $T = N\delta$ the prediction horizon, and stage and terminal cost respectively defined as

$$\begin{aligned} \mathcal{L}(\boldsymbol{\chi}, \boldsymbol{u}) &= \|\boldsymbol{\chi}^{\text{ref}} - \boldsymbol{\chi}\|^2_{\boldsymbol{W}_{\boldsymbol{\chi}}} + \|\boldsymbol{u}^{\text{ref}} - \boldsymbol{u}\|^2_{\boldsymbol{W}_{\boldsymbol{u}}} \\ \Phi(\boldsymbol{\chi}) &= \|\boldsymbol{\chi}^{\text{ref}} - \boldsymbol{\chi}\|^2_{\boldsymbol{W}_{\boldsymbol{\chi}}}, \end{aligned}$$

$\boldsymbol{W}_{\boldsymbol{\chi}}, \boldsymbol{W}_{\boldsymbol{u}}$ positive semi-definite matrices, which respectively weigh the impact of tracking and control effort [15], v^- and v^+ min/max wheel driving velocity, a^- and a^+ min/max wheel driving acceleration, ω^- and ω^+ min/max wheel steering velocity and $\boldsymbol{\chi}_k$ initial configuration.

Note that the velocity constraints are linear for the coordinating wheel (since v_1^S and ω_1 are part of $\boldsymbol{\chi}$) and nonlinear for the coordinated wheels. In particular, because of the assumption of no slipping (1), the driving velocity of the coordinated wheels can be computed as

$$v_i^W = \begin{pmatrix} \cos(\theta + \beta_i) \\ \sin(\theta + \beta_i) \end{pmatrix}^T \dot{\boldsymbol{p}}_i^W.$$

Since the steering angles of the coordinated wheels are defined as $\beta_i(v_1^S, \omega, \beta_1)$, the steering velocities can simply be computed as their time derivatives.

We can transcribe the above OCP into the following nonlinear programming (NLP) problem by using multiple shooting [20]:

$$\begin{aligned} \min_{\boldsymbol{Q}_k, \boldsymbol{U}_k} \ & \Phi(\boldsymbol{\chi}_{N|k}) + \sum_{j=0}^{N-1} \mathcal{L}(\boldsymbol{\chi}_{j|k}, \boldsymbol{u}_{j|k}) \\ \text{s.t. } & \boldsymbol{\chi}_{j+1|k} = \boldsymbol{F}(\boldsymbol{\chi}_{j|k}, \boldsymbol{u}_{j|k}), \ \forall j \in \mathbb{I}_0^{N-1} \\ & v^- \leq v^W_{i,j|k}(\cdot) \leq v^+, \ \forall i \in \mathbb{I}_1^n, \forall j \in \mathbb{I}_0^{N-1} \\ & \text{sgn}(v^S_{1,j|k}) = \text{sgn}(v_1^S(t_0)), \ \forall j \in \mathbb{I}_0^N \\ & \text{sgn}(v^S_{i,j|k}(\cdot)) = \text{sgn}(v_i^S(t_0)), \ \forall i \in \mathbb{I}_2^n, \forall j \in \mathbb{I}_0^{N-1} \\ & a^- \leq a_{j|k} \leq a^+, \ \forall j \in \mathbb{I}_0^{N-1} \\ & \omega^- \leq \omega_{1,j|k} \leq \omega^+, \ \forall j \in \mathbb{I}_0^{N-1} \\ & \omega^- \leq \omega_{i,j|k}(\cdot) \leq \omega^+, \ \forall i \in \mathbb{I}_2^n, \forall j \in \mathbb{I}_0^{N-1} \\ & \boldsymbol{\chi}_{0|k} = \boldsymbol{\chi}_k, \end{aligned}$$

with vectors

$$\begin{aligned} \boldsymbol{Q}_k &= (\boldsymbol{\chi}_{0|k}, \boldsymbol{\chi}_{1|k}, \dots, \boldsymbol{\chi}_{N|k}), \\ \boldsymbol{U}_k &= (\boldsymbol{u}_{0|k}, \boldsymbol{u}_{1|k}, \dots, \boldsymbol{u}_{N-1|k}), \end{aligned}$$

collecting the decision variables of the NMPC at t_k, δ timestep of the NMPC, and the cost function evaluated using $\boldsymbol{\chi}^{\text{ref}}_{j|k}$ ($j = 0, \dots, N$) and $\boldsymbol{u}^{\text{ref}}_{j|k}$ ($j = 0, \dots, N-1$), computed by the state trajectory generation scheme. Note that, within the constraints, $(\cdot)$ denotes the use of nonlinear functions, moreover, as mentioned before, the sign constraints are a consequence of constraints (8).

Having solved the NLP problem, we extract the control sample $\boldsymbol{u}_{0|k}$ from $\boldsymbol{U}_k$ and apply it to the robot.

IV. Simulations

The proposed framework has been implemented in Python, using the acados library [21] to solve the aforementioned NLP problem with real-time iteration scheme [19]. We use the robot

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 06,2025 at 09:24:04 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2025.3564209

Neobotix MPO-700, which has $n = 4$ steerable wheels (Fig. 1). The scheme runs at 75 Hz (at the same frequency of the low-level controller) on an Intel Core i5-10210U (1.6 GHz, 8 cores) with Ubuntu 20.04 LTS. The used parameters are listed in Table I. Note that the bounds on velocities and accelerations have been chosen according to the technical specifications of the platform.

We validated our implementation in simulation on various trajectory tracking scenarios, where we highlight the avoidance of singularities, and the bounds activation in the NMPC.

In the *quick slalom with tangent orientation*, the robot is required to follow a sinusoidal trajectory while keeping its orientation tangent to it. Figure 5 shows the driving and steering velocities computed by the NMPC, together with the FSM state, and the case where the ICR is at infinity (highlighted by the blue area). In particular, at $t = 0$, the state of the FSM changes from *Ready* to *Starting*, and the robot starts its motion using open-loop commands (Sect. III-B). In this first part of the motion (highlighted by the green area on the left side of the plots), the FSM keeps the *Starting* state until the configuration of the robot is singularity-free. The state of the FSM then changes from *Starting* to *Moving*, and the NMPC is activated. Figure 6 shows the NMPC control inputs and the trajectory tracking errors. The NMPC controls the mobile base until the pose of the robot is close to the final pose, where the state of the FSM changes from *Moving* to *Stopping*. In this last part of the motion (highlighted by the green area on the right of the plots), the robot gracefully stops, and the state of the FSM finally changes to *Ready*. Note that, because the reference trajectory is too quick for the mobile base, the driving velocities of the wheels computed by the NMPC reach the bounds, and the trajectory tracking error increases. These velocities are then kept high by the NMPC, decreasing the tracking error (this is reflected by the right side plot in Fig. 6).

Note that removing the *Starting* state from the FSM would result in a complete failure of the NMPC because the use of the initial state containing $v_1^S = 0$ would make the optimization problem infeasible. If we keep *Starting* and remove the *Stopping* state, the NMPC would not be able to bring the robot to a stop. For example, if we consider the same trajectory discussed before (Fig. 7), at the end of the motion the driving and steering velocities of the wheel increase, as a consequence of the reference trajectory computed by the state trajectory generation scheme containing high values. This behavior is due to the velocity v_1^S being close to zero.

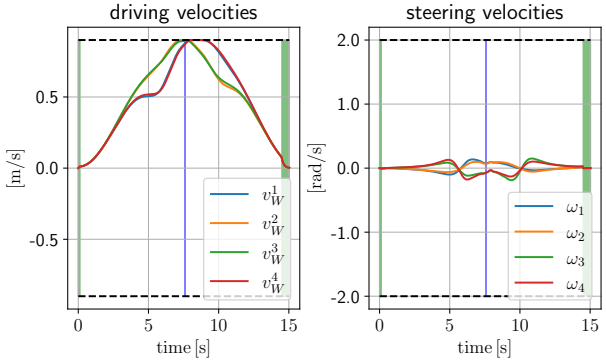


Fig. 5: Driving and steering velocities of the wheels for *quick slalom with tangent orientation*. The bounds are dashed.

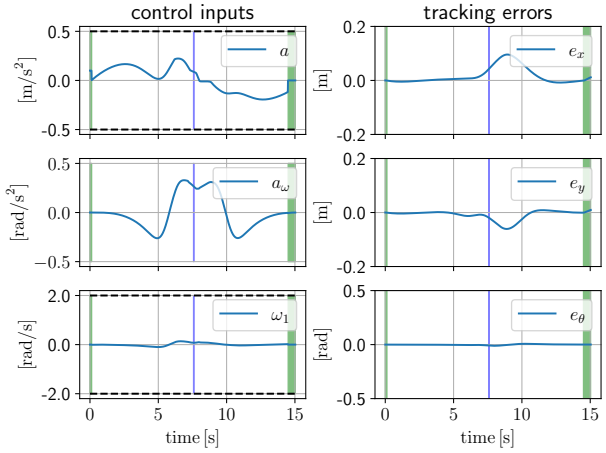


Fig. 6: NMPC control inputs and trajectory tracking error for *quick slalom with tangent orientation*.

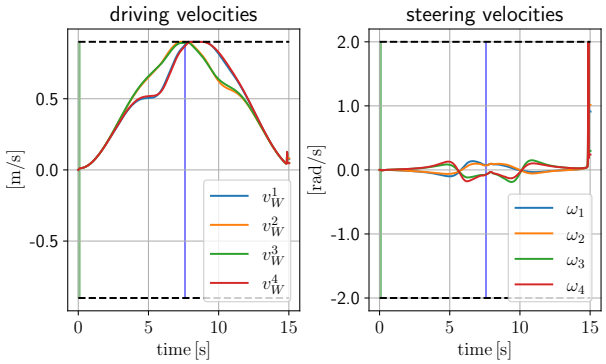


Fig. 7: Driving and steering velocities of the wheels for *quick slalom with tangent orientation* without *Stopping*.

V. Experiments

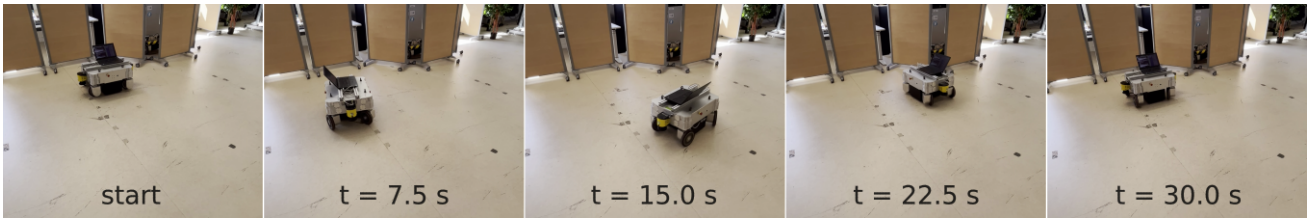
We validated our framework on the Neobotix MPO-700 platform on trajectories of different complexity. We invite the reader to watch the accompanying video[3], which shows the described experiments, as well as additional ones with different reference trajectories.

In the *circle with tangent orientation* trajectory, the robot is required to follow a circle while keeping its orientation tangent to the circle itself. Figure 8a shows a sequence of snapshots

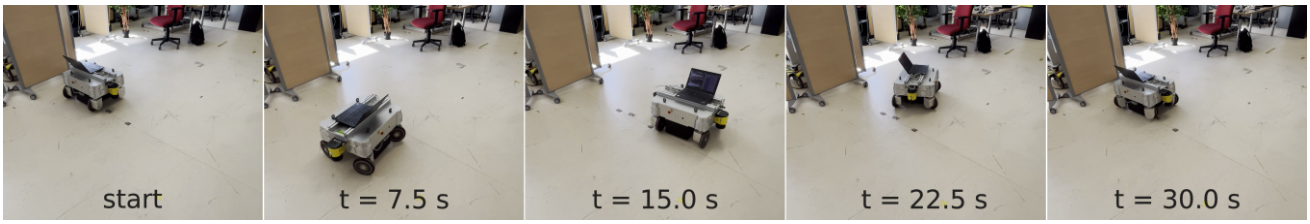
[3]https://youtu.be/0xPZSjrF9Ak

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 06,2025 at 09:24:04 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

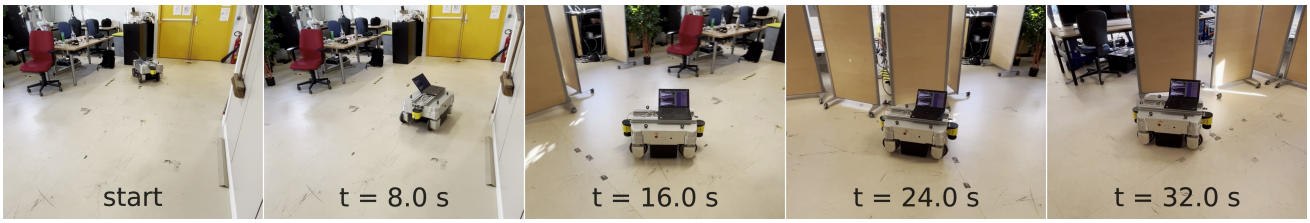
This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2025.3564209



(a) Robot tracking a *circle with tangent orientation*.



(b) Robot tracking a *circle with inward orientation*.



(c) Robot tracking a *slalom with tangent orientation*.

Fig. 8: Snapshots of the mobile base tracking a desired trajectory.

TABLE I: Parameters used in all our experiments.

Symbol	Value
$\boldsymbol{a}$	$(-0.19, 0.19, 0.19, -0.19)$ [m]
$\boldsymbol{b}$	$(0.24, 0.24, -0.24, -0.24)$ [m]
d	0.045 [m]
a^{init}	0.1 [m/s^2]
K^{stop}	1.0
$\boldsymbol{K}_P$	$\text{diag}(4.0, 4.0, 2.0)$
$\boldsymbol{K}_D$	$\text{diag}(2.0, 2.0, 1.0)$
N	5
δ	0.1 [s]
$\boldsymbol{W}_{\boldsymbol{\chi}}$	$\boldsymbol{I}_9$
$\boldsymbol{W}_{\boldsymbol{u}}$	$\boldsymbol{I}_3$
v^-	-0.9 [m/s]
v^+	0.9 [m/s]
a^-	-0.5 [m/s^2]
a^+	0.5 [m/s^2]
ω^-	-2.0 [rad/s]
ω^+	2.0 [rad/s]

TABLE II: Trajectory tracking error, with e_p position error norm and e_θ orientation error norm.

Trajectory	e_p [m]		e_θ [rad]	
	Avg	Max	Avg	Max
circle with tangent orientation	0.021	0.053	0.001	0.047
circle with inward orientation	0.014	0.048	0.001	0.031
slalom with tangent orientation	0.004	0.022	0.001	0.005

of the robot moving while tracking this trajectory. The initial configuration of the robot is such that the ICR is at infinity (because all steering angles have the same value). The robot starts its motion using an open-loop command (as described in Sect. III-B), which makes the configuration of the robot be singularity-free. At this point, the NMPC is activated, and the circular trajectory is followed. Notice how the orientation of the mobile base is always tangent to the circle during the whole motion. When the pose of the robot is close to the final configuration, the robot decelerates, successfully completing its task. This experiment highlights the capability of the NMPC to handle situations in which the ICR starts at infinity, and it is moved to the center of the circle.

Similarly to the previous experiment, in the *circle with inward orientation* trajectory, the robot is required to follow a circle while pointing its front towards the circle center. Figure 8b shows snapshots of the robot tracking this trajectory, and Fig. 9 shows the driving and the steering velocities computed by the NMPC, together with the FSM states (as before, the green area on the left side of the plots represent the *Starting* state, while the green area on the right side of the plots represent the *Stopping* state). Once the NMPC is activated, the ICR is moved to the center of the circle, and the trajectory is correctly followed. Notice how the orientation of the mobile base always points to the circle center. This highlights the omnidirectionality of the platform, which is able to follow circular trajectories with different orientations.

Finally, we consider the *slalom with tangent orientation* trajectory, where the robot is required follow a sinusoidal trajectory, while keeping its orientation tangent to it. Figure 8c shows snapshots of the mobile base moving in this scenario. Note that in this experiment (similarly to the slalom trajectory in the previous section) the ICR must necessarily go through infinity to correctly track the considered trajectory. The capability of handling such situations is due to our parametrization,

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 06,2025 at 09:24:04 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

This article has been accepted for publication in IEEE Robotics and Automation Letters. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/LRA.2025.3564209

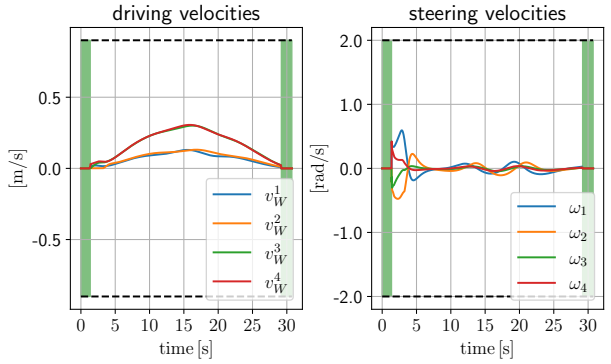


Fig. 9: Driving and steering velocities of the wheels for *circle with inward orientation*.

which does not directly used the position of the ICR in the control scheme (see [3] for more details on the taxonomy of singularities of SWMRs).

In all experiments, our NMPC is solved in less than 1 ms, making it possible to satisfy real-time constraints of the platform. Table II presents statistics on the trajectory tracking error over the three discussed trajectories. In particular, for each experiment, the average and the maximum position and orientation error norms are computed. In our experiments, we have seen that increasing N does not significantly improve tracking. Note that, while the robot is equipped with two laser scanners, the estimated pose of the robot is retrieved by an odometric localization module which use only wheel encoders. The use of additional sensors for a more precise localization will be part of future works.

VI. Conclusions

In this work, we presented a framework for trajectory tracking with omnidirectional SMWRs, which makes use of a NMPC based on real-time iteration. Our scheme is capable of tracking trajectories without violating wheels' velocity constraints, while taking into account kinematic model singularities. We have validated our approach on multiple trajectories using the Neobotix MPO-700, showing that our scheme is always able to track them. To the best of our knowledge, this is the first time NMPC has been implemented on an SWMR.

In future work, we plan to extend the framework in several ways:

1) extend the NMPC to a dual-arm mobile manipulator such as BAZAR robot [22], making it interact with the environment with its arms, while moving;
2) implement a motion planning algorithm such as kino-dynamic RRT* [23], making the robot able to navigate autonomously in an environment with obstacles;
3) further improve the performance of the framework by implementing it in C++ (while the scheme runs in real-time thanks to acados, the time consuming state trajectory generation runs in Python).

References

[1] P. R. Giordano, M. Fuchs, A. O. Albu-Schäffer, and G. Hirzinger, "On the Kinematic Modeling and Control of a Mobile Platform Equipped with Steering Wheels and Movable Legs," *2009 IEEE Int. Conf. on Robotics and Automation*, 2009.

[2] L. Tagliavini, G. Colucci, A. Botta, P. Cavallone, L. Baglieri, and G. Quaglia, "Wheeled mobile robots: State of the art overview and kinematic comparison among three omnidirectional locomotion strategies," *Journal Intelligent Robotics System*, vol. 106, nov 2022.

[3] M. Sorour, A. Cherubini, P. Fraisse, and R. Passama, "Motion Discontinuity-Robust Controller for Steerable Mobile Robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 452–459, 2017.

[4] G. Campion, G. Bastin, and B. Dandrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 1, pp. 47–62, 1996.

[5] M. Sorour, A. Cherubini, R. Passama, and P. Fraisse, "Kinematic modeling and singularity treatment of steerable wheeled mobile robots with joint acceleration limits," in *2016 IEEE Int. Conf. on Robotics and Automation*, p. 2110–2115, IEEE Press, 2016.

[6] C. P. Connette, A. Pott, M. Hagele, and A. Verl, "Control of an pseudo-omnidirectional, non-holonomic, mobile robot based on an icm representation in spherical coordinates," in *2008 47th IEEE Confe. on Decision and Control*, pp. 4976–4983, 2008.

[7] M. Sorour, A. Cherubini, A. Khelloufi, R. Passama, and P. Fraisse, "Complementary-route based icr control for steerable wheeled mobile robots," *Robotics and Autonomous Systems*, vol. 118, pp. 131–143, 2019.

[8] F. Ferland, L. Clavien, J. Frémy, D. Létourneau, F. Michaud, and M. Lauria, "Teleoperation of azimut-3, an omnidirectional non-holonomic platform with steerable wheels," in *2010 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2515–2516, 2010.

[9] L. Clavien, M. Lauria, and F. Michaud, "Estimation of the instantaneous centre of rotation with nonholonomic omnidirectional mobile robots," *Robotics and Autonomous Systems*, vol. 106, pp. 47–57, 2018.

[10] L. Clavien, M. Lauria, and F. Michaud, "Instantaneous Centre of Rotation Based Motion Control for Omnidirectional Mobile Robots with Sidewards Off-Centred Wheels," *Robotics and Autonomous Systems*, vol. 106, p. 58–68, aug 2018.

[11] E. Spyrakos-Papastavridis and J. S. Dai, "Minimally model-based trajectory tracking and variable impedance control of flexible-joint robots," *IEEE Trans. on Industrial Electronics*, vol. 68, no. 7, pp. 6031–6041, 2021.

[12] W. He, X. Tang, T. Wang, and Z. Liu, "Trajectory tracking control for a three-dimensional flexible wing," *IEEE Trans. on Control Systems Technology*, vol. 30, no. 5, pp. 2243–2250, 2022.

[13] J. Rawlings, D. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017.

[14] S. G. Tarantos and G. Oriolo, "A Dynamics-Aware NMPC Method for Robot Navigation Among Moving Obstacles," in *Intelligent Autonomous Systems 17*, Springer Nature Switzerland, 2023.

[15] M. Zanon, J. V. Frasch, M. Vukov, S. Sager, and M. Diehl, *Model Predictive Control of Autonomous Vehicles*, pp. 41–57. Springer International Publishing, 2014.

[16] M. Beglini, T. Belvedere, L. Lanari, and G. Oriolo, "An intrinsically stable mpc approach for anti-jackknifing control of tractor-trailer vehicles," *IEEE/ASME Trans. on Mechatronics*, vol. 27, no. 6, 2022.

[17] M. Bjelonic, R. Grandia, O. Harley, C. Galliard, S. Zimmermann, and M. Hutter, "Whole-body mpc and online gait sequence generation for wheeled-legged robots," in *2021 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, p. 8388–8395, IEEE Press, 2021.

[18] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR Control via Dynamic Feedback Linearization: Design, Implementation, and Experimental Validation," *IEEE Trans. on Control Systems Technology*, 2002.

[19] 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. Journal of Control*, 2020.

[20] H. Bock and K. Plitt, "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[21] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[22] A. Cherubini, R. Passama, B. Navarro, M. Sorour, A. Khelloufi, O. Mazhar, S. Tarbouriech, J. Zhu, O. Tempier, A. Crosnier, P. Fraisse, and S. Ramdani, "A collaborative robot for the factory of the future: BAZAR," *Int. Journal of Advanced Manufacturing Technology*, 2019.

[23] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE Int. Conf. on Robotics and Automation*, IEEE, 2013.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 06,2025 at 09:24:04 UTC from IEEE Xplore. Restrictions apply.
© 2025 IEEE. All rights reserved, including rights for text and data mining and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.