2023 European Control Conference (ECC)
June 13-16, 2023. Bucharest, Romania

MPC Based Motion Planning For Mobile Robots Using Velocity Obstacle Paradigm

Nicola Piccinelli, Federico Vesentini, Riccardo Muradore

Abstract— **Model Predictive Control (MPC) has been increasingly adopted in robotics in recent years for several tasks including the real-time control of mobile robots. The main advantage of using the MPC is the possibility of adopting an optimal control policy under a set of constraints which can be used to enforce safety, like collision-free manoeuvres. In literature a well-known approach to ensure real-time collision avoidance for multi-agent systems is the Velocity Obstacle (VO) paradigm which considers the robot as a point-mass, neglecting its dynamic and kinematic constraints. In this paper, we propose an MPC-based motion planning solution that directly considers the dynamics and the kinematics of the mobile robots, exploiting the VO as a constraint on the configuration space of the controlled system. We evaluated the proposed solution in simulation using holonomic and non-holonomic kinematic and dynamic models.**

I. Introduction

Motion planning and path planning for multi-agent systems are crucial aspects in scenarios such as automated warehouses, where autonomous agents need to move safely while carrying out their tasks, i.e. by avoiding collisions with expected and unexpected obstacles. The generation of collision-free trajectories can be addressed via centralised or decentralised approaches. In centralised solutions the agents are treated as single entities and, as in [1], a global path planner computes their safe trajectories. Unfortunately, such kinds of solutions scale poorly with a large number of agents and they are also not robust in case of network failures. On the other hand, in decentralised approaches, the planning is computed by every single agent where the information about the environment and the other agents is acquired via communication systems and/or sensing devices like cameras, laser scanners etc.

VO paradigm [2] and its enhanced formulations [3], [4], [5] represent one of the most famous decentralised methods for the collision avoidance of autonomous ground [6], [7], [8] and aerial robots [9], [10]. VO has recently been adapted also for the safe navigation of planar manipulators with two revolute degrees of freedom [11], where usually other approaches are adopted, such as Artificial Potential Fields (APFs) [12], [13] and Dynamic Movement Primitives (DMPs) [14]. VO is a geometric approach based on the construction of cones in the velocity space, induced by the dynamic environment, containing all the velocity vectors that may cause collisions between a robot and other moving obstacles. In general, VO-based approaches assume that autonomous robots do not communicate with each other, but only perceive the surrounding dynamic environment, i.e. they can acquire position, velocity and radius of moving obstacles or other agents.

Authors in [15] used an MPC technique for formation control and collision avoidance of Unmanned Aerial Vehicles (UAVs) forced to fly at a certain fixed altitude. The UAVs are modelled as two-dimensional points of mass whose dynamics are described by a linear state-space discrete model. Researchers in [16] proposed an obstacle avoidance algorithm for Unmanned Ground Vehicles (UGVs) where safe trajectories are generated by adopting a non-linear MPC approach, to better approximate the dynamics of each vehicle. In a similar contribution [17], an MPC scheme based on convex quadratic programming provides safe trajectories for autonomous vehicles whose dynamics are obtained by performing a linear approximation of a non-linear time-varying model.

In [18] a decentralized supervisory controller for multiple robotic agents is proposed. The controller is based on Control Barrier Functions (CBF) and ensures obstacle avoidance considering general nonlinear robot dynamics with limited actuation capability. The algorithm does not require agent-to-agent communication, and a simple broadcasting scheme can improve its performance.

Finally, an attempt to integrate the MPC with the VO paradigm, in particular with the Optimal Reciprocal Collision Avoidance (ORCA) [5], is proposed in [19] where the derived velocities from ORCA serve as constraints of an MPC problem whose solution provides the optimal control input that can ensure the optimal motion of the agent.

In this paper, we propose a novel distributed non-linear MPC-based motion planning for planar mobile robots. Each robot, as in the VO paradigm, senses the surrounding environment and solves an optimization problem to compute the collision-free trajectory to the desired position. The proposed methodology can be adopted both for holonomic and non-holonomic robots and can be integrated into a torque control loop. The trajectories are computed by restricting the configuration space of the controlled agent using the collision cones induced by the other agents and obstacles and thus constraining the optimization problem along the prediction horizon. Differently from [19], we rely on the original VO paradigm to reduce conservativeness.

The main contributions of the paper are:

- the design of a non-linear MPC motion planning to deal with kinematic constraints, and

This work was partly supported by the project MIUR “Dipartimenti di Eccellenza” 2018-2022.

Nicola Piccinelli, Federico Vesentini and Riccardo Muradore are with the Department of Computer Science, University of Verona, Italy (email: {nicola.piccinelli, federico.vesentini, riccardo.muradore}@univr.it

- the integration of Velocity Obstacle as an optimization constraint to provide collision-free trajectories by accounting for the dynamics of the robot.

The paper is organized as follows, in Section II we will review the VO approach. In Section III we will present the methodology and describe in detail the integration of the VO into the optimization problem. In Section IV we will present some preliminary simulation results with different kinematic structures. Finally in Section V we will draw a few conclusions and present the future works.

II. Background

The velocity obstacle paradigm [2] is a geometric construction used to determine the velocity vectors that cause collisions between an autonomous agent and the moving obstacles[1]. The dynamic environment consists of a robot A and many moving obstacles B_j, $j = 1, ..., m$, assumed to be holonomic and disk-shaped. A is characterised by position vector $\boldsymbol{x}_A$, radius r_A and velocity vector $\boldsymbol{v}_A$. Moreover, it is able to acquire the position $\boldsymbol{x}_{B_j}$, radius r_{B_j} and velocity vector $\boldsymbol{v}_{B_j}$ of every obstacle B_j. VO represents the dynamic environment into the velocity space that, for objects moving in $\mathbb{R}^2$ or $\mathbb{R}^3$, is given by the tangent bundle $T\mathbb{R}^n$ of $\mathbb{R}^n$, $n = 2, 3$. A motion in the velocity space is given by the pair

$$(\boldsymbol{x}_A(t), \boldsymbol{v}_A(t)) \in T\mathbb{R}^n \simeq \mathbb{R}^n \times \mathbb{R}^n. \tag{1}$$

For every pair (A, B_i) let

$$\boldsymbol{v}_{AB_j} = \boldsymbol{v}_A - \boldsymbol{v}_{B_j}, \quad j = 1, \ldots, m \tag{2}$$

define the relative velocity of A with respect to B_j and let

$$r_{AB_j} = r_A + r_{B_j}, \quad j = 1, \ldots, m \tag{3}$$

define the enlarged obstacle radius r_{B_j}. The corresponding relative trajectory in the velocity space is defined as

$$trj_{AB_j} = \{(\boldsymbol{x}, \dot{\boldsymbol{x}}) : \boldsymbol{x}(t_0) = \boldsymbol{x}_A, \, \dot{\boldsymbol{x}}(t_0) = \boldsymbol{v}_{AB_j}\}. \tag{4}$$

and

$$CC_{AB_j} \triangleq \{\boldsymbol{v}_{AB_j} | trj_{AB_j} \cap \hat{B}_j \neq 0\} \tag{5}$$

called *Relative Collision Cone* of A induced by B_j. The *Absolute Collision Cone* of A with respect to B_j is defined as follows

$$CC_{A_j} \triangleq \{\boldsymbol{v}_A | \boldsymbol{v}_{AB_j} \in CC_{AB_j}\} \tag{6}$$

and it is obtained by translating the relative collision cone CC_{AB_j} by $\boldsymbol{v}_A$ via Minkowski sum

$$CC_{A_j} = CC_{AB_j} \oplus \boldsymbol{v}_{B_j}. \tag{7}$$

A does not collide with B_j as long as its velocity vector remains outside CC_{A_j}. The absolute collision cone can be thought as the Velocity Obstacle, VO_A, for A induced by B_j; in particular, A must select a new velocity vector $\boldsymbol{v}_A^{new}$ such that

$$\boldsymbol{v}_A^{new} = \underset{\boldsymbol{v} \in \mathrm{V} \setminus MVO_A}{\operatorname{argmin}} ||\boldsymbol{v}_A^{pref} - \boldsymbol{v}||_2, \tag{8}$$

[1]In a multi-agent scenario, due to the distributed approach all the uncontrolled agents are treated as moving obstacles.

where V is the set of admissible velocities for A and MVO_A is the *Multiple Velocity Obstacle* for A induced by multiple obstacles

$$MVO_A \triangleq \bigcup_{j=1}^{m} CC_{A_j}. \tag{9}$$

Modern formulations of Velocity Obstacles are based on truncated velocity obstacle $VO_{A_j}^{\tau}$. If A selects a velocity vector outside $VO_{A_j}^{\tau}$ means that no collisions with B occur until time $1/\tau$. A planner based on truncated cones is called Finite-Time Velocity Obstacle (FVO) [20]. The FVO converges to VO for $\tau \to \infty$, since the cut-off circle collapses to a single point, $VO_{A_j}^{\infty} \to VO_{A_j}$.

III. Methodology

The control architecture we propose, shown in Figure 1, can be subdivided into two main parts: a global planner which provides a list of waypoints to accomplish a desired task and a local planner which provides the motion planning and performs the collision avoidance maneuvers with respect to dynamic obstacles.

The local planner is based on the VO paradigm, but instead of considering the controlled robot as a holonomic point mass, we take into account its kinematics and dynamics. The optimal control action is computed by considering a set of inequality constraints based on the Signed Minimum Distance (SMD) between the optimal velocity and the obstacle set B. Moreover, such an approach does not need any modification to the VO formulation and thus calculates less conservative solutions.

A. Signed Minimum Distance

As shown in Figure 2, let $\boldsymbol{q}_{AB_j}$ and $\boldsymbol{w}_{AB_j}$ be the two tangent points to the obstacle B_j induced by the collision cone CC_{AB_j}. Let α_{AB_j} be the angle between the relative velocity vector $\boldsymbol{v}_{AB_j}$ and $\boldsymbol{h}_{AB_j}$

$$\alpha_{AB_j} = \arccos\left(\frac{\boldsymbol{h}_{AB_j} \cdot \boldsymbol{v}_{AB_j}}{||\boldsymbol{h}_{AB_j}||_2 ||\boldsymbol{v}_{AB_j}||_2}\right) \tag{10}$$

where $\boldsymbol{h}_{AB_j} = \boldsymbol{x}_{B_j} - \boldsymbol{x}_A$ is the vector from the agent position $\boldsymbol{x}_A$ to the obstacle position $\boldsymbol{x}_{B_j}$. Similarly, let θ_{AB_j} be the angle between the cone boundary versors $\boldsymbol{q}_{AB_j}$, $\boldsymbol{w}_{AB_j}$ and $\boldsymbol{h}_{AB_j}$. The relative velocity $\boldsymbol{v}_{AB_j}$ is collision causing if and only if

$$|\alpha_{AB_j}| \leq \theta_{AB_j} \tag{11}$$

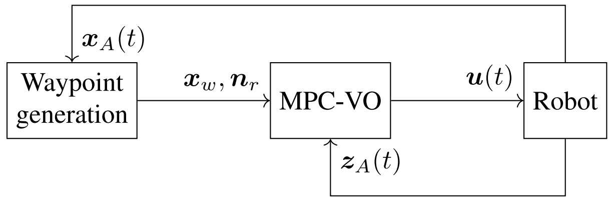


Fig. 1. The proposed control architecture. The waypoint generation block is part of the global planner and provides the next waypoint. The MPC acts as a local planner and plans the collision-free trajectory and collision avoidance.

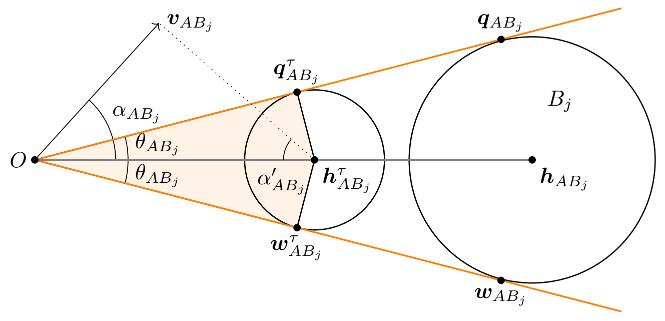


Fig. 2. A geometric representation of the truncated collision cone for robot A induced by the obstacle B_j. The two orange lines represent the boundaries of the original collision cone.

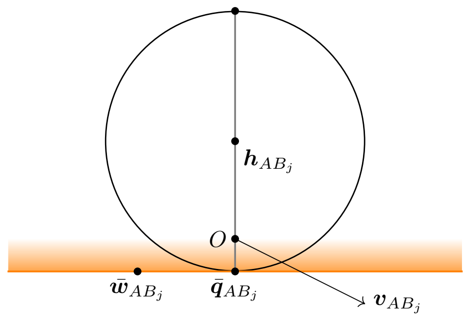


Fig. 3. A representation of the half-plane construction in case of collision. The collision-causing half-plane is highlighted as the orange shaded area with the boundary represented by the horizontal orange line. The SMD $\bar{s}_{AB_j}$ is computed as the distance between v_{AB_j} and that line.

and its minimum distance to the cone's boundary d_{AB_j} can be computed as

$$d_{AB_j} = \begin{cases} d^m_{AB_j}, & \text{if } |\alpha_{AB_j}| \leq \frac{\pi}{2} \\ \|\boldsymbol{v}_{AB_j}\|_2, & \text{otherwise} \end{cases} \tag{12}$$

where

$$d^m_{AB_j} = \min(d(\boldsymbol{v}_{AB_j}, \boldsymbol{q}_{AB_j}), d(\boldsymbol{v}_{AB_j}, \boldsymbol{w}_{AB_j})) \tag{13}$$

with

$$d(\boldsymbol{v}, \boldsymbol{n}) = \|\boldsymbol{v} - (\boldsymbol{v} \cdot \boldsymbol{n})\boldsymbol{n}\|_2 \tag{14}$$

be the point-line shortest distance between the point $\boldsymbol{v}$ and the line defined by the direction versor $\boldsymbol{n}$. Finally, let the SMD s_{AB_j} be

$$s_{AB_j} = \begin{cases} d_{AB_j}, & \text{if } |\alpha_{AB_j}| \leq \theta_{AB_j} \\ -d_{AB_j}, & \text{otherwise} \end{cases} \tag{15}$$

where d_{AB_j} is positive if $\boldsymbol{v}_{AB_j}$ is inside the CC_{AB_j}, negative if outside and zero on the boundary. The signed minimum distance s_{AB_j} will be used later on to define the collision-free constraints in the optimization problem. We now extend the definition of s_{AB_j} also in the case of truncated collision cones $CC^\tau_{AB_j}$.

Let $\boldsymbol{h}^\tau_{AB_j} = \boldsymbol{h}_{AB_j}/\tau$ and $r^\tau_{AB_j} = r_{AB_j}/\tau$ be the vector pointing from the agent A to the cut-off circle of $CC^\tau_{AB_j}$ with radius r_{AB_j} and let $\boldsymbol{q}^\tau_{AB_j}$ and $\boldsymbol{w}^\tau_{AB_j}$ be the tangent points to the obstacle B_j induced by the collision cone. In this case, a relative velocity $\boldsymbol{v}_{AB_j}$ is collision causing if and only if

$$\begin{cases} s_{AB_j} > 0 \wedge \|\boldsymbol{v}_{AB_j} - \boldsymbol{h}^\tau_{AB_j}\|_2 \leq r^\tau_{AB_j}, & \text{if } |\alpha'_{AB_j}| \leq \theta'_{AB_j} \\ s_{AB_j} > 0, & \text{otherwise} \end{cases} \tag{16}$$

where $\theta'_{AB_j} = \frac{\pi}{2} - \theta_{AB_j}$ and

$$\alpha'_{AB_j} = \arccos\left(\frac{-\boldsymbol{h}_{AB_j} \cdot (\boldsymbol{v}_{AB_j} - \boldsymbol{h}^\tau_{AB_j})}{\|\boldsymbol{h}_{AB_j}\|_2 \|\boldsymbol{v}_{AB_j} - \boldsymbol{h}^\tau_{AB_j}\|_2}\right). \tag{17}$$

The minimum distance to the boundary $d^\tau_{AB_j}$ is

$$d^\tau_{AB_j} = \begin{cases} \left|\|\boldsymbol{v}_{AB_j} - \boldsymbol{h}^\tau_{AB_j}\|_2 - r^\tau_{AB_j}\right|, & \text{if } |\alpha'_{AB_j}| \leq \theta'_{AB_j} \\ d^m_{AB_j}, & \text{otherwise} \end{cases} \tag{18}$$

and the resulting truncated SMD $s^\tau_{AB_j}$ is defined as

$$s^\tau_{AB_j} = \begin{cases} d^\tau_{AB_j}, & \text{if (16)} \\ -d^\tau_{AB_j}, & \text{otherwise} \end{cases}. \tag{19}$$

To address the possibility of a colliding situation and correctly incorporate the VO paradigm into the optimization problem, a further extension of the definition of the SMD function is necessary. This is important because when computing the optimal policy, it is not always guaranteed that the problem will remain feasible throughout the process. Therefore, by including a definition of the SMD function that considers the possibility of collisions, we can ensure that the optimization problem remains valid. In this case, we have $\|\boldsymbol{h}_{AB_j}\|_2 \leq r_{AB_j}$ which means that the robot A is colliding with the obstacle B_j and thus the computation of a collision cone fails.

As shown in Figure 3, we propose to build a half-plane, that allows us to define a collision-free subspace and provides a distance towards the line which defines the boundary. So in this case, a relative velocity $\boldsymbol{v}_{AB_j}$ is collision causing if and only if

$$\text{sign}((\boldsymbol{v}_{AB_j} - \bar{\boldsymbol{q}}_{AB_j})^T(-\bar{\boldsymbol{q}}_{AB_j})) \geq 0 \tag{20}$$

and its minimum distance is defined as

$$\bar{d}_{AB_j} = d(\boldsymbol{v}_{AB_j}, \bar{\boldsymbol{q}}_{AB_j} - \bar{\boldsymbol{w}}_{AB_j}) \tag{21}$$

where $\bar{\boldsymbol{q}}_{AB_j}$ is the closest intersection point between the obstacle circle B_j and the vector $\boldsymbol{x}_{B_j}$ and $\bar{\boldsymbol{w}}_{AB_j}$ is the perpendicular direction to $\bar{\boldsymbol{q}}_{AB_j}$. In this colliding situation, the SMD $\bar{s}_{AB_j}$ is defined as

$$\bar{s}_{AB_j} = \begin{cases} \bar{d}_{AB_j}, & \text{if (20)} \\ -\bar{d}_{AB_j}, & \text{otherwise} \end{cases}. \tag{22}$$

B. MPC formulation

As mentioned before, since we aim at providing a collision avoidance strategy suitable for mobile robots with different kinematics and dynamics, let the dynamic model of a generic mobile platform defined by the Ordinary Differential Equation (ODE)

$$\dot{\boldsymbol{z}}_A = f_A(\boldsymbol{z}_A, \boldsymbol{u}_A) \tag{23}$$

where $\boldsymbol{z}_A$ is the state vector, $\boldsymbol{u}_A$ is the input vector and f_A is the ODE modelling the dynamics. For the sake of simplicity, we assume $\boldsymbol{z}_A$ to always contain the position $\boldsymbol{x}_A$ and the velocity $\boldsymbol{v}_A$ of the robot A and this part of the states is observable from the measurements.

Since the agent is supposed to navigate through a list of N predefined waypoints $\boldsymbol{x}_w$, the MPC is defined as follows

$$\begin{aligned} \hat{\boldsymbol{u}}^{\star}_A(k+i)|_{i=0}^{k_p} = \quad & \underset{\hat{\boldsymbol{u}}_A(\cdot)}{\arg\min} \quad \mathcal{C}(\hat{\boldsymbol{z}}_A, \hat{\boldsymbol{u}}_A) \\ & \text{subject to} \quad \hat{\boldsymbol{z}}_A(k) = \boldsymbol{z}_A(k) \\ & \qquad \hat{\boldsymbol{z}}_A(k+1) = f_A(\hat{\boldsymbol{z}}_A, \hat{\boldsymbol{u}}_A) \\ & \qquad S_{AB_j}(k+i) \le 0 \end{aligned} \tag{24}$$

where

$$S_{AB_j} = \begin{cases} \bar{s}_{AB_j}, & \text{if } \|\boldsymbol{h}_{AB_j}\|_2 \le r_{AB_j} \\ s_{AB_j}, & \text{otherwise} \end{cases} \quad j = 1, \dots, m. \tag{25}$$

The number of constraints in (24) is $m\,k_p$, i.e. one constraint for each obstacle B_j for every step in the prediction horizon. It is also worth remarking that (25) must be adapted in case of truncated collision cone substituting s_{AB_j} with $s^{\tau}_{AB_j}$.

The cost function $\mathcal{C}(\hat{\boldsymbol{z}}_A, \hat{\boldsymbol{u}}_A)$ is designed to penalise the tracking error $\boldsymbol{e}(k) = \boldsymbol{x}_r(k) - \boldsymbol{x}(k)$ and the direction error $e_d(k) = (1 - \boldsymbol{n}_r^T(\boldsymbol{x}_w - \boldsymbol{x}(k))$. This second term measures how well the robot is aligned with respect to a nominal direction $\boldsymbol{n}_r$. In our case, we set $\boldsymbol{n}_r$ to be the versor pointing from the previous waypoint to the current one $\boldsymbol{x}_w$. For each control cycle, the reference trajectory driving the robot towards the current waypoint within the prediction horizon $\boldsymbol{x}_r(k)$ is defined as

$$\boldsymbol{x}_r(k+i) = \boldsymbol{x}^0_A(k) + (\boldsymbol{x}_w - \boldsymbol{x}^0_A(k))v^{des} i \Delta t \tag{26}$$

where v^{des} is the robot's desired speed, $\boldsymbol{x}^0_A(k)$ is the position of the robot A at time k and Δt is the sample-time of the controller. Additionally, we also penalize the magnitude of the computed command $\boldsymbol{u}(k)$ and its variation $\Delta\boldsymbol{u}(k)$ to control the smoothness of the control action. The resulting cost function to be minimized is

$$\begin{aligned} \mathcal{C}(\hat{\boldsymbol{z}}_A, \hat{\boldsymbol{u}}_A) = & \sum_{i=1}^{k_p} \boldsymbol{e}^T(k+i)\Lambda_p \boldsymbol{e}(k+i) + \\ & + \lambda_d e_d(k+i)^2 + \\ & + \boldsymbol{u}^T(k+i)\Lambda_u \boldsymbol{u}(k+i) + \\ & + \Delta\boldsymbol{u}^T(k+i)\Lambda_{\Delta u}\Delta\boldsymbol{u}(k+i) \end{aligned} \tag{27}$$

where Λ_p, Λ_u and $\Lambda_{\Delta u}$ are application-dependant positive definite weight matrices and λ_d is a non-negative weight scalar. This last weight allows us to decide how close the MPC controller should keep the a priori desired direction.

The current waypoint $\boldsymbol{x}_w$ is updated, with the next in the list, by an external global planner according to a minimum distance criteria between the robot position $\boldsymbol{x}_A$ and the current waypoint $\boldsymbol{x}_w$. Possible deadlock situations, e.g. when the current waypoint is unreachable due to an unaccounted obstacle, can be easily detected by the MPC controller testing the waypoint position $\boldsymbol{x}_w$ with the obstacle set B.

IV. Simulation results

We tested the proposed approach in a simulated environment using the MATLAB® 2022b Optimisation Toolbox and in C++ using libmpc++[2]. We tested two different dynamics: a holonomic and a differential-drive mobile robot. Robots are assumed to be disc-shaped with radius $r_A = 0.15\,\mathrm{m}$, total mass of $m = 1.0\,\mathrm{kg}$ and in the case of differential-drive with an inertia $I = \frac{1}{2}mr_A^2$. The two dynamics are tested on the same scenario where the mobile platform needs to travel through three waypoints: $\boldsymbol{x}^1_w = (0, 0)$, $\boldsymbol{x}^2_w = (2, 2)$, $\boldsymbol{x}^3_w = (2, 0)$ and $\boldsymbol{x}^1_w$ again. Two obstacles are placed along the nominal path obtained by connecting the waypoints by straight lines. The sample time Δt of the controller is set to $0.01\,\mathrm{s}$ $(100\,\mathrm{Hz})$ and the optimization problem is solved using Sequential Least Squares Programming (SLSQP).

The two obstacles have the same radius $r_B = 0.3\,\mathrm{m}$ (one is fixed while the other is moving in the environment) and the desired speed for the local planner is set to $v^{des} = 1.0\,\mathrm{m\,s^{-1}}$. Finally, in the simulation setup, we used the truncated collision cone with a factor $\tau = 2$, meaning that we are considering collisions happening in the next $500\,\mathrm{ms}$.

A. Holonomic Dynamic Model

In this first scenario, let $\boldsymbol{z}_A(t) \in \mathbb{R}^4$ be the state vector for a holonomic mobile robot

$$\boldsymbol{z}_A(t) = \begin{bmatrix} \boldsymbol{x}_A \\ \boldsymbol{v}_A \end{bmatrix}, \tag{28}$$

the solution to the following set of ODEs

$$\begin{cases} \dot{\boldsymbol{z}}(t) = f_A(\boldsymbol{z}_A, \boldsymbol{u}_A) = \boldsymbol{A}\boldsymbol{z}_A(t) + \boldsymbol{B}\boldsymbol{u}_A(t) \\ \boldsymbol{y}_A(t) = \boldsymbol{C}\boldsymbol{z}_A(t) \end{cases} \tag{29}$$

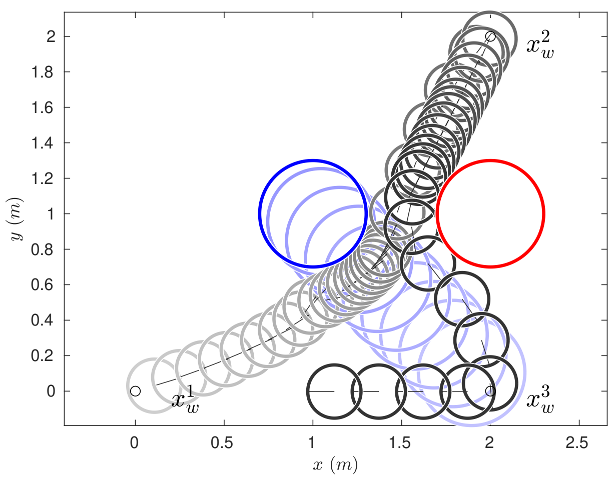
where the state space system matrices are

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{B} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m} & 0 \\ 0 & \frac{1}{m} \end{pmatrix}, \quad \boldsymbol{C} = I_4. \tag{30}$$

The prediction horizon of the MPC is set to $k_p = 10$ steps to provide smooth and collision-free trajectories as in Figure 4a. Figure 4a shows how the holonomic robot can perform a very sharp turn overcoming the waypoint $\boldsymbol{x}^2_w$ and thus performing collision avoidance on the left side of the obstacle. As shown in Figure 5b, the velocity profile is smooth and whenever possible tries to reach the nominal speed v^{des}. Is it worth mentioning that such velocities are already feasible for the kinematics of the controlled mobile robot since its model is taken into account within the MPC.

Figure 5a shows the evolution of the robot's position. It is possible to see that the robot can reach all the waypoints: $\boldsymbol{x}^2_w$ around $6\,\mathrm{s}$, $\boldsymbol{x}^3_w$ around $11\,\mathrm{s}$ and $\boldsymbol{x}^1_w$ around time $13\,\mathrm{s}$. We highlight how in the last part of the motion (i.e. between the third and the first waypoint) the robot reaches its nominal speed v^{des}: this is due to the absence of obstacles along its path. Moreover, it is also possible to notice the effect of the

[2]`https://github.com/nicolapiccinelli/libmpc`


(a) Holonomic agent

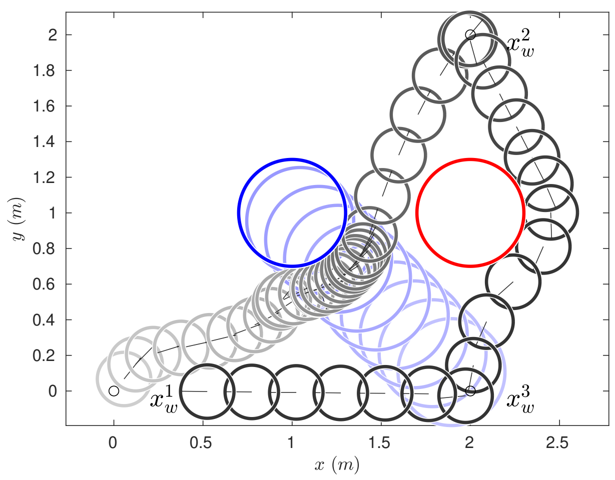
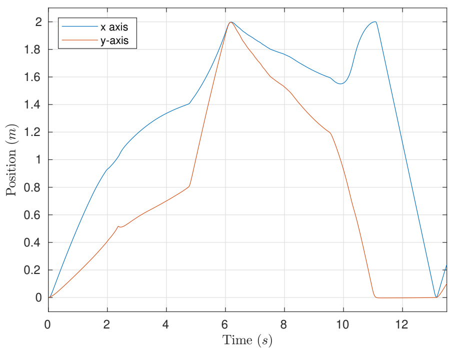
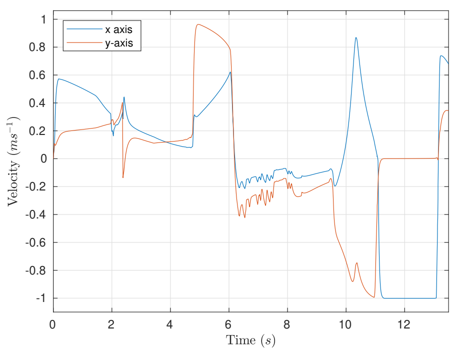

(b) Non-holonomic agent

Fig. 4. A comparison between the trajectories generated by the MPC while controlling a holonomic and non-holonomic mobile agent. The agent is represented by grey circles while the obstacles are represented by red and blue circles for static and moving obstacles, respectively. The time evolution of the trajectory is shown with an increasing level of transparency (transparency inversely proportional to time). The waypoints to be reached by the robots are $\boldsymbol{x}_w^1$, $\boldsymbol{x}_w^2$ and $\boldsymbol{x}_w^3$.


(a) Position


(b) Velocity

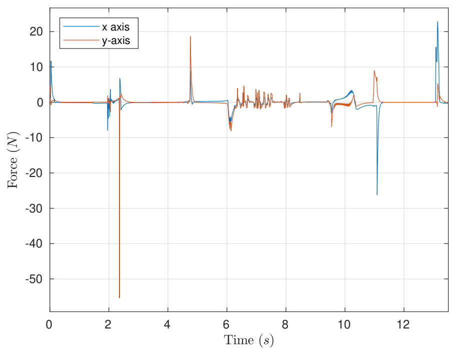

(c) Control command

Fig. 5. The behavior of a holonomic agent while moving through the predefined waypoints.

weight λ_d. In fact, while approaching $\boldsymbol{x}_w^3$ the robot does not perform the shortest path, but it steers to get closer to the nominal direction $\boldsymbol{n}_r$. Finally, Figure 5c shows the behavior of the commanded forces to the mobile robot: the peaks in the profile are due to the collision avoidance maneuvers and could be further reduced by employing box constraints on the inputs or by adjusting the weight matrices to have a smoother control action.

B. *Non-holonomic differential drive dynamic model*

Let $\boldsymbol{z}_A(t) \in \mathbb{R}^7$

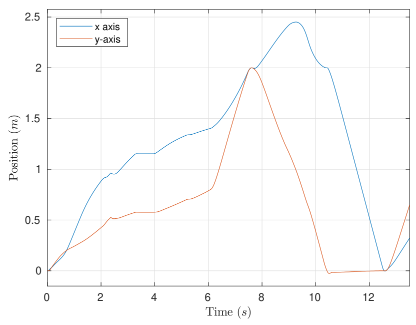
$$\boldsymbol{z}(t) = \begin{bmatrix} \boldsymbol{x}_A \\ \boldsymbol{v}_A \\ \phi \\ v \\ w \end{bmatrix} \tag{31}$$

be the state vector for a differential-drive mobile robot, where $\boldsymbol{x}_A$ and $\boldsymbol{v}_A$ are the position and velocity vectors of the robot in Cartesian space, ϕ is the robot heading, v is the magnitude of the velocity and w is the angular velocity. The robot dynamics is given by

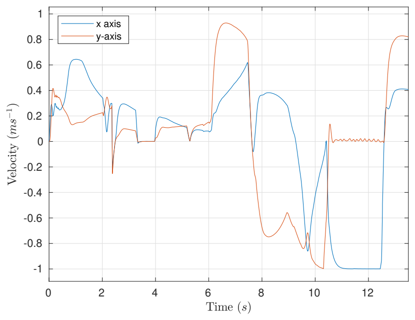
$$\dot{\boldsymbol{z}}(t) = f_A(\boldsymbol{z}_A, \boldsymbol{u}_A) = \begin{bmatrix} \boldsymbol{v}_A \\ v\cos(\phi) \\ v\sin(\phi) \\ w \\ u_1/m \\ u_2/I \end{bmatrix} \tag{32}$$

where u_1 and u_2 are the control force and the control torque computed by the MPC. We set the prediction horizon of the MPC to $k_p = 10$ steps and again, as shown in Figure 4b, the controller can guarantee a smooth and collision-free trajectory.

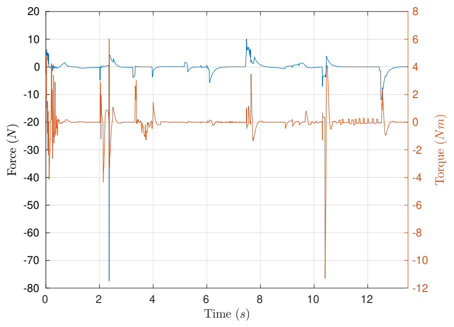
In this case, since the robot is characterized by non-holonomic dynamics, waypoint $\boldsymbol{x}_w^2$ is overcome by performing a wider turn. As shown in Figure 4b, the robot avoids the obstacle from the right side. Another different behavior can be noticed in Figure 6a, where, due to the different dynamics, the robot reaches the waypoint $\boldsymbol{x}_w^2$ 2 s later with respect to the holonomic agent. This is because, as shown in Figure 6b, the non-holonomic robot stays still while waiting for the



(a) Position



(b) Velocity



(c) Control command

Fig. 6. The behavior of a non-holonomic agent while moving through the predefined waypoints.

moving obstacle. Nonetheless, the robot reaches again all the waypoints: $\boldsymbol{x}_w^2$ around 8 s, $\boldsymbol{x}_w^3$ around 10 s and $\boldsymbol{x}_w^1$ around time 12 s. As in the previous case, it reaches the desired speed v^{des} while moving from the waypoint $\boldsymbol{x}_w^3$ towards $\boldsymbol{x}_w^1$. Also in this case, this is due to the absence of obstacles along the path. Finally, Figure 6c shows the applied linear force and torque to control the differential-drive model. Despite some peaks in the profiles due to the abrupt collision avoidance maneuver, the command is in general smooth.

Is it worth remarking that, even if the proposed control scheme aims to directly force/torque control a mobile platform, it can integrate standard velocity controllers using an admittance controller or by considering the robot's inner controller within the MPC.

V. Conclusion

In this paper, we proposed a novel formulation of an MPC-based motion planning technique providing collision-free trajectories by explicitly taking into account the kinematics and dynamics of the mobile robot. We used the Velocity Obstacle framework to constrain the optimal control problem along the prediction horizon, to find solutions that prevent the robot from entering the collision cones created by a set of obstacles. To accomplish this, we defined a signed minimum distance function for full and truncated collision cones and also in case of collision conditions. We tested our approach in a simulated scenario using both holonomic and non-holonomic dynamics and in the future, we plan to test the proposed solution in a real environment.

References

[1] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi-robot systems," in *Proceedings 2002 IEEE international conference on robotics and automation (Cat. No. 02CH37292)*, vol. 2. IEEE, 2002, pp. 2112–2119.

[2] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The international journal of robotics research*, vol. 17, no. 7, pp. 760–772, 1998.

[3] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE international conference on robotics and automation*. Ieee, 2008, pp. 1928–1935.

[4] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[5] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[6] N. Piccinelli, F. Vesentini, and R. Muradore, "Planning with real-time collision avoidance for cooperating agents under rigid body constraints," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1261–1264.

[7] N. Piccinelli and R. Muradore, "Hybrid motion planner integrating global voronoi diagrams and local velocity obstacle method," in *2018 European Control Conference (ECC)*. IEEE, 2018, pp. 26–31.

[8] J. Snape and D. Manocha, "Goal velocity obstacles for spatial navigation of multiple autonomous robots or virtual agents," in *Autonomous Robots and Multi robot Systems*, 2013.

[9] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[10] T. Battisti and R. Muradore, "A velocity obstacles approach for autonomous landing and teleoperated robots," *Autonomous Robots*, vol. 44, no. 2, pp. 217–232, 2020.

[11] F. Vesentini and R. Muradore, "Velocity obstacle-based trajectory planner for two-link planar manipulators," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 690–695.

[12] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.

[13] D. E. Koditschek, "Robot planning and control via potential functions," *The robotics review*, p. 349, 1989.

[14] M. Ginesi, D. Meli, A. Roberti, N. Sansonetto, and P. Fiorini, "Dynamic movement primitives: Volumetric obstacle avoidance using dynamic potential functions," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 4, pp. 1–20, 2021.

[15] T. Keviczky, F. Borrelli, and G. J. Balas, "A study on decentralized receding horizon control for decoupled systems," in *Proceedings of the 2004 American Control Conference*, vol. 6. IEEE, 2004, pp. 4921–4926.

[16] J. Park, D. Kim, Y. Yoon, H. Kim, and K. Yi, "Obstacle avoidance of autonomous vehicles based on model predictive control," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 223, no. 12, pp. 1499–1516, 2009.

[17] H. Jiang, Z. Wang, Q. Chen, and J. Zhu, "Obstacle avoidance of autonomous vehicles with cqp-based model predictive control," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2016, pp. 001 668–001 673.

[18] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, 2020.

[19] H. Cheng, Q. Zhu, Z. Liu, T. Xu, and L. Lin, "Decentralized navigation of multiple agents based on orca and model predictive control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3446–3451.

[20] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: highly parallel collision avoidance for multi-agent simulation," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009, pp. 177–187.