Collision avoidance with differential drive robots using MPC-ORCA

Glauber R. Leite [*] Arthur da C. Vangasse [*] Ícaro B. Q. Araújo [*] Heitor J. Savino [*]

[*] *Instituto de Computação, Universidade Federal de Alagoas, AL, (e-mail: {grl, vangasse, icaro, heitor.savino}@ic.ufal.br).*

Abstract: This work describes a distributed solution using Model Predictive Control (MPC), including the Optimal Reciprocal Collision Avoidance (ORCA) algorithm applied to mobile robots following individual trajectories. Differential drive robots are used, defined by their position on the plane and controlled by velocity commands. Based on an explicit system model and velocity constraints designated by the collision avoidance algorithm, the MPC-ORCA computes optimal control actions to minimize a cost function over a prediction horizon. The methodology can efficiently handle multivariable control systems using state-space model representation and convex quadratic programming (QP). Simulation results show that the combined strategy MPC-ORCA provided smooth and collision-free trajectories in a changing environment. It also requires no trajectory replanning neither direct communication between agents.

Keywords: Model Predictive Control; Autonomous Mobile Robots; Distributed Control; Collision Avoidance; Cooperative Robots

1. INTRODUCTION

The growing employment of robots in diverse industry domains, due to increasing demands in productivity and flexibility, leads to the densification of robots in the factory place (Graetz and Michaels, 2018). Thus, autonomy and decentralization have become essential in the paradigm of Industry 4.0, mainly when it comes to scalability (Lu, 2017).

Distributed robotic systems have inherent advantages such as parallelism to achieve global objectives faster and simplifying problems due to decomposition and task allocation. Also, there are situations where multiple simple robots can perform particular functionalities better than a unique complex robot (Arkin and Balch, 1998). Conversely, coordinating distributed robots is a challenging problem since group performance is not programmed directly and the behavior of each agent limits the development of the solution (Ota, 2006; Sycara, 1998). In the context of collision avoidance, providing collision-free trajectories requires eventual or continuous replanning through algorithms such as A*, D*, or Potential Fields (Stentz, 1995; Mouad et al., 2012).

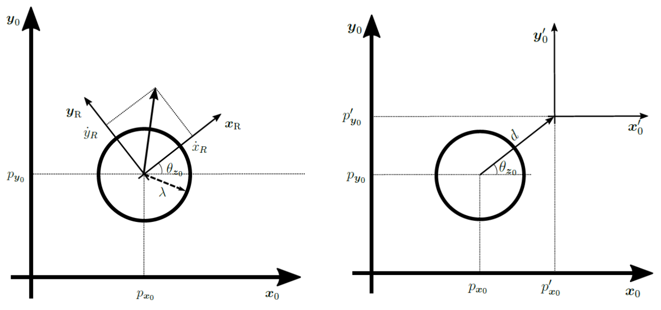
An approach for this problem relies on local control actions for online reaction to robots and obstacles in route of collision, ensuring collision-free trajectories while keeping the smallest deviation from predefined routes. These algorithms compute regions in velocity space that would put the agent into collision risk in some future moment, then choose a velocity command close to the one computed offline by the global planner outside those collision regions.

Velocity Obstacles (VO), proposed by Fiorini and Shiller (1993), is one of these first algorithms to handle moving obstacles. However, some scenarios present unrealistic oscillations when those obstacles are other agents. For that, Van den Berg et al. (2008) adapted the VO algorithm to consider reciprocity between agents, resulting in the Reciprocal Collision Obstacles (RVO) approach. Later, RVO was reformulated as the Optimal Reciprocal Collision Avoidance (ORCA) by Van Den Berg et al. (2011), generating optimal halfspaces solvable with convex optimization.

Model predictive control (MPC) is an approach to design optimal control actions through predictions of the output of a system aware of its model and constraints. Convex optimization can properly implement MPC. Thus, Cheng et al. (2017) assigned ORCA halfspaces as constraints on an MPC problem so that each robot generates its collision-free trajectories. Cheng et al. (2017) focused on holonomic mobile robots and considered only the desired target position to generate trajectories online. Mao et al. (2020) follows a similar direction with non-holonomic mobile robots, still relying on online generated trajectories.

The contribution of this paper is to extend the MPC-ORCA approach from Cheng et al. (2017) considering two aspects: i) differential drive robots controlled by linear (forward) and rotational velocities; ii) and an offline trajectory planner for each robot, that produces single-agent trajectories unaware of neighbors, which provides greater planning freedom for distinct tasks strategies, such as maximum coverage tasks. The second contribution differentiates this work from Mao et al. (2020).

The state-space model is used to write the linearized robot's kinematics, written as a MPC problem with ORCA



(a) Robot model. (b) Mobile robot linearization.

Figure 1. Model of a mobile robot and assumption of an operation point for linearization.

constraints related to its neighbors. The problem is converted into a convex optimization problem and solved using quadratic programming, which provides unique global minimum solutions.

This paper is organized as follows: Section 2 presents the model and linearization for the differential drive robot. Section 3 shows the construction of the MPC problem and its formulation as quadratic programming optimization problem. Section 4 explores the ORCA algorithm, which is integrated with the MPC in Section 5. Section 6 presents a simulation scenario, while results are given in Section 7. Concluding remarks and future directions are summarized at the end.

2. ROBOT MODEL

Figure 1a shows the representation of each robot. The world frame is defined by $\mathcal{F}_0 = \{O_0, x_0, y_0, z_0\}$, fixed at O_0 and with coordinate system with the unitary orthogonal axes $\{x_0, y_0, z_0\}$. We assume each agent bounded by a circle of radius $\lambda \in \mathbb{R}$. The coordinate system of the robot is defined by the frame $\mathcal{F}_R = \{O_R, x_R, y_R\}$, with origin at the coordinate $O_R = (p_{x_0}, p_{y_0})$ at the center of the circle in global coordinates, with $p_{x_0}, p_{y_0} \in \mathbb{R}$, and $\{x_R, y_R, z_R\}$ unitary orthogonal axes. The rotation of the robot's reference system $\mathcal{F}_R$ relative to the global $\mathcal{F}_0$ is given by $\theta_{z_0} \in \mathbb{S}^1$. Thus, the configuration of a robot can be written as

$$\mathbf{q} = [p_{x_0} \;\; p_{y_0} \;\; \theta_{z_0}]^T \in \mathbb{Q},$$

with $\mathbb{Q} \subset \mathbb{R}^2 \times \mathbb{S}^1$ the set of possible configurations given in the global reference system. At the local frame, the velocity is given by

$$\boldsymbol{v}^R = [v_{x_R} \;\; v_{y_R} \;\; \omega_{z_R}]^T \in \mathbb{R}^3,$$

where $v_{x_R} = \dot{p}_{x_R}$, $v_{y_R} = \dot{p}_{y_R}$, and $\omega_{z_R} = \dot{\theta}_{z_R}$ are the robot linear velocities in x_R, and y_R, and angular velocity around z_R, respectively.

Differential drive robots are subjected to non-holonomic constraints reducing the admissible velocity space. This model considers that (Ahmad Abu Hatab, 2013) the robot cannot move sideways, i.e., $v_{y_R} = 0$, and no sideslip over axes x_R or y_R. Thus, the control action for each robot actuated by accelerations is

$$\boldsymbol{u}^R = \begin{bmatrix} \dot{v}_{x_R} \\ \dot{\omega}_{z_R} \end{bmatrix}.$$

The presented non-holonomic constraints restrain the construction of a linear time-invariant model. To circumvent this problem, define a maneuvering frame as shown in Figure 1b within a small distance d from the center of the robot to perform the linearization as explored by Ren and Atkins (2007). The frame is defined in the same orientation as the global frame with origin (p'_{x_0}, p'_{y_0}) as

$$\begin{bmatrix} p'_{x_0} \\ p'_{y_0} \end{bmatrix} = \begin{bmatrix} p_{x_0} \\ p_{y_0} \end{bmatrix} + d \begin{bmatrix} \cos\theta_{z_0} \\ \sin\theta_{z_0} \end{bmatrix}. \qquad (1)$$

Taking a double derivative and considering the transformation from the robot to the global frame, one can obtain

$$\begin{bmatrix} \dot{v}_{x_R} \\ \dot{\omega}_{z_R} \end{bmatrix} = \begin{bmatrix} \cos\theta_{z_0} & -d\sin\theta_{z_0} \\ \sin\theta_{z_0} & d\cos\theta_{z_0} \end{bmatrix}^{-1} \begin{bmatrix} \ddot{p}_{x_0} + v_{x_R}\omega_{z_R}\sin\theta_{z_0} + d\omega_{z_R}^2\cos\theta_{z_0} \\ \ddot{p}_{y_0} - v_{x_R}\omega_{z_R}\cos\theta_{z_0} + d\omega_{z_R}^2\sin\theta_{z_0} \end{bmatrix} \qquad (2)$$

Through feedback linearization, the control actions

$$u^0 = \begin{bmatrix} \ddot{p}_{x_0} \\ \ddot{p}_{y_0} \end{bmatrix} = \begin{bmatrix} a_{x_0} \\ a_{y_0} \end{bmatrix}$$

applied at the operation point dismiss the kinematic constraints, presenting the behavior of a holonomic robot. Discretizing the model, with step-time T_s, the second-order model of the robot is

$$\begin{bmatrix} p_{x_0}(k+1) \\ p_{y_0}(k+1) \\ v_{x_0}(k+1) \\ v_{y_0}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{x_0}(k) \\ p_{y_0}(k) \\ v_{x_0}(k) \\ v_{y_0}(k) \end{bmatrix} + \begin{bmatrix} 0.5T_s^2 & 0 \\ 0 & 0.5T_s^2 \\ T_s^2 & 0 \\ 0 & T_s^2 \end{bmatrix} \begin{bmatrix} a_{x_0}(k) \\ a_{y_0}(k) \end{bmatrix}. \qquad (3)$$

Finally, Equation (3) is a discrete second-order state space model (Ogata, 1995), that describes the dynamic behavior of a state vector $\boldsymbol{x}(k)$, in this case the position and velocity of the vehicle, in the presence of a input $\boldsymbol{u}(k)$, i.e. acceleration of the vehicle. Therefore, Equation (3) can be represented by the generalized discrete second order model, presented in Equation (4).

$$\boldsymbol{x}(k+1) = \boldsymbol{F}\boldsymbol{x}(k) + \boldsymbol{G}\boldsymbol{u}(k) \qquad (4)$$

3. MODEL PREDICTIVE CONTROL

Model Predictive Control is a methodology of control systems design that employs prediction strategies and optimization based on an explicit model (Camacho and Bordons, 2007), considering the constraints that the system's variables must obey. Usually, the optimization problem considers a limited prediction horizon of states and control actions, being employed to the actual plant the first control action only, leaving the rest. The updated problem is again resolved at the next iteration, with newly acquired measurements, a characteristic known as receding horizon.

There are many ways to implement a MPC, with different combinations of models, prediction strategies, and optimization algorithms. An example is the DMC (Dynamic Matrix Control) and its improved version QDMC

(Quadratic Dynamic Matrix Control) in treating constraints with low computational cost (Qin and Badgwell, 2003). Some of the characteristics of QDMC are:

- a linear model for the system;
- a quadratic objective function in a finite prediction horizon;
- the future behavior of the variables of interest must follow the reference and is subject to a term that punishes abrupt changes on the input (move suppression);
- the solution of the quadratic problem provides the optimal inputs.

Considering that the system's model is linear as expressed in Equation (3), a QDMC implementation was chosen. Thus, it is possible to design a quadratic objective function, which provides efficient computation and unique global minimum solutions (Boyd et al., 2004). The MPC problem with a QDMC implementation is formulated as

$$\begin{aligned} \text{minimize } & \sum_{k=0}^{N} (\boldsymbol{x}(k) - \boldsymbol{r}(k))^T \boldsymbol{Q}(k)(\boldsymbol{x}(k) - \boldsymbol{r}(k)) \\ & + \sum_{k=0}^{N-1} \boldsymbol{u}(k)^T \boldsymbol{R}(k) \boldsymbol{u}(k), \\ \text{subject to } & \boldsymbol{x}(k+1) = \boldsymbol{F}\boldsymbol{x}(k) + \boldsymbol{G}\boldsymbol{u}(k), \\ & \boldsymbol{x}_{min} \le \boldsymbol{x}(k) \le \boldsymbol{x}_{max}, \\ & \boldsymbol{u}_{min} \le \boldsymbol{u}(k) \le \boldsymbol{u}_{max}, \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0. \end{aligned} \tag{5}$$

As all state variables are observable, the state vector may be considered the very output of the system. The expression to be minimized is composed of weighted sums of quadratic components, where $\boldsymbol{x}(k) - \boldsymbol{r}(k)$ defines a mobile robot's trajectory error along the prediction horizon N. Also, the $\boldsymbol{Q}$ and $\boldsymbol{R}$ matrices are positive semidefinite and can vary along the prediction horizon. $\boldsymbol{Q}$ and $\boldsymbol{R}$ are essential tuning parameters of the controller, associating weights to the reference distance criteria and the control action, respectively. The problem constraints bound the optimization region to respect the system's dynamics, the robot's physical limitations related to position and velocity, in $\boldsymbol{x}_{min}$, $\boldsymbol{x}_{max}$, and acceleration in $\boldsymbol{u}_{min}$ and $\boldsymbol{u}_{max}$. They also impose the initial condition $\boldsymbol{x}_0$ given by the current position and velocity readings.

The MPC problem was written as a Quadratic Programming (QP) problem. A QP problem is defined as

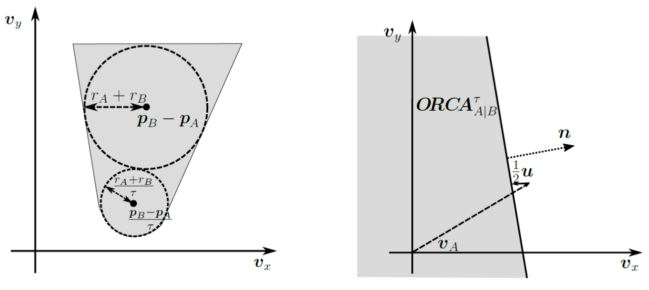
$$\begin{aligned} \text{minimize } & J(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T \boldsymbol{P} \boldsymbol{z} + \boldsymbol{c}^T \boldsymbol{z}, \\ \text{subject to } & \boldsymbol{z}_{min} \le \boldsymbol{A}\boldsymbol{z} \le \boldsymbol{z}_{max}, \end{aligned} \tag{6}$$

where the positive semidefinite matrix $\boldsymbol{P}$ and $c \in \mathbb{R}^n$ associate weights to the quadratic and linear components of the objective function. In the constraints, z_{min} and z_{max} determine the maximum and minimum limits of the $\boldsymbol{A}\boldsymbol{z}$ operation.

The $\boldsymbol{z}$ vector of the variables of interest consists of all states and inputs of the prediction horizon N, given as

$$\boldsymbol{z} = [\boldsymbol{x}(0)^\top \ ... \ \boldsymbol{x}(N)^\top \ \boldsymbol{u}(0)^\top \ ... \ \boldsymbol{u}(N-1)^\top]^\top. \tag{7}$$

The Kronecker product is used to define the quadratic matrix component $\boldsymbol{P}$ and the linear component $\boldsymbol{c}$ from the objective function considering the prediction horizon,



(a) Velocity obstacles region. (b) ORCA halfspace.

Figure 2. ORCA halfspace for agent B relative to agent A.

using $\boldsymbol{Q}$, $\boldsymbol{R}$, and the reference trajectory r formulated in the MPC problem (5). It is also used to compute the constraint matrix $\boldsymbol{A}$ and its limits, using the robot model and its motion constraints.

4. OPTIMAL RECIPROCAL COLLISION AVOIDANCE

A local control algorithm can avoid collision when following a trajectory provided by a global trajectory planner. Hence, the agent reacts to a changing environment, handling trajectory perturbation from the movement of near agents. Besides, since first-order methods such as VO (Fiorini and Shiller, 1993), RVO (Van den Berg et al., 2008), and ORCA (Van Den Berg et al., 2011) are calculated with only position and velocity data from immediate neighbors, they do not require explicit communication between agents, becoming efficient approaches for decentralized systems.

To illustrate the ORCA algorithm as seen in (Van Den Berg et al., 2011), consider a scenario with two circular robots A and B, described by their respective positions ($\boldsymbol{p}_A$, $\boldsymbol{p}_B$), radius (r_A, r_B), and absolute velocities ($\boldsymbol{v}_A$, $\boldsymbol{v}_B$). As shown in Figure 2a, a velocity obstacle region $\boldsymbol{VO}^\tau_{A|B}(\boldsymbol{v}_B)$ for agent B relative to A is computed for a time window τ applying the Minkowski sum

$$\boldsymbol{VO}^\tau_{A|B}(\boldsymbol{v}_B) = CC^\tau_{A|B} \oplus \boldsymbol{v}_B. \tag{8}$$

between a collision cone

$$\begin{aligned} \boldsymbol{CC}^\tau_{A|B} = \{\boldsymbol{v} \mid \exists t \in [0, \tau] :: \boldsymbol{v}t \\ \in \boldsymbol{B}(\boldsymbol{p}_B - \boldsymbol{p}_A, r_A + r_B)\}, \end{aligned} \tag{9}$$

and velocity $\boldsymbol{v}_B$ from agent B, where $\boldsymbol{B}(\boldsymbol{p}, r)$ is an open ball centered in $\boldsymbol{p}$ with radius r.

Considering that $\boldsymbol{v}_A$ is inside the region $\boldsymbol{VO}^\tau_{A|B}(\boldsymbol{v}_B)$, the algorithm computes an $\boldsymbol{u}_{\text{ORCA}}$ vector that represents the smallest move that leads the agent to the border of the velocity obstacle region, depicted as $\partial\boldsymbol{VO}^\tau_{A|B}(\boldsymbol{v}_B)$ in

$$\boldsymbol{u}_{\text{ORCA}} = \left(\underset{\boldsymbol{v}^*_A \in \partial \boldsymbol{VO}^\tau_{A|B}(\boldsymbol{v}_B)}{\operatorname{argmin}} ||\boldsymbol{v}^*_A - \boldsymbol{v}_A|| \right) - \boldsymbol{v}_A \tag{10}$$

Along with $\boldsymbol{u}_{\text{ORCA}}$, it is calculated a normal vector $\boldsymbol{n}$, which is normal to the contour of $\boldsymbol{VO}^\tau_{A|B}(\boldsymbol{v}_B)$ at $\boldsymbol{v}_A + \boldsymbol{u}$ pointing inside the velocity object region.

It is defined as the $\boldsymbol{ORCA}^\tau_{A|B}$ region, the set of velocities for agent A containing the highest number of velocities closest to $\boldsymbol{v}_A$ which prevents collision to agent B. It uses the principle of reciprocal effort share between two agents. Thus, $\boldsymbol{ORCA}^{\tau}_{A|B}$ is a halfplane described by equation (11) and presented in Figure 2b.

$$\boldsymbol{ORCA}^{\tau}_{A|B} = \{\boldsymbol{v}^*_A \mid (\boldsymbol{v}^*_A - (\boldsymbol{v}_A + \frac{1}{2}\boldsymbol{u}_{\mathrm{ORCA}})) \cdot \boldsymbol{n} \leq 0\} \quad (11)$$

In a multiagent scenario, the ORCA halfplanes calculated for each neighbor are convex regions, so their intersection results in a convex polytope. Therefore, ORCA halfplanes can be used in convex optimization problems as constraints for collision-free agent movement.

5. MPC-ORCA

Using a combined strategy with MPC and ORCA offers advantages over a pure ORCA algorithm application (Cheng et al., 2017). The receding horizon from MPC provides safer operation, reacting earlier to probable collision scenarios and reducing the times that the optimization problem becomes infeasible. Also, incorporating the second-order model dynamics from equation (3) provides smoother trajectories because it does not assume sudden changes in agents' velocities.

The MPC problem (5) can be rewritten to incorporate ORCA constraints. It is defined $\boldsymbol{g}(\boldsymbol{x}(k), \{\boldsymbol{x}_{\mathrm{neigh}}(k)\}) \leq 0$ as the resulting velocity constraints from ORCA in the current agent. Besides, ORCA constraints for future states are generated applying the algorithm over the prediction horizon N, considering that agents keep their velocity constant. The combined MPC-ORCA strategy is defined as follows:

$$\begin{aligned} \text{minimize } & \sum_{k=0}^{N} (\boldsymbol{x}(k) - \boldsymbol{r}(k))^\top \boldsymbol{Q}(k)(\boldsymbol{x}(k) - \boldsymbol{r}(k)) \\ & + \sum_{k=0}^{N-1} \boldsymbol{u}(k)^\top \boldsymbol{R}(k)\boldsymbol{u}(k), \\ \text{subject to } & \boldsymbol{x}(k+1) = \boldsymbol{F}\boldsymbol{x}(k) + \boldsymbol{G}\boldsymbol{u}(k), \\ & \boldsymbol{x}_{\min} \leq \boldsymbol{x}(k) \leq \boldsymbol{x}_{\max}, \\ & \boldsymbol{u}_{\min} \leq \boldsymbol{u}(k) \leq \boldsymbol{u}_{\max}, \\ & \boldsymbol{g}(\boldsymbol{x}(k), \{\boldsymbol{x}_{\mathrm{neigh}}(k)\}) \leq 0, \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0. \end{aligned} \quad (12)$$

The ORCA halfplanes can be implemented in Equation (6), adding rows for $\boldsymbol{z}_{min}$, $\boldsymbol{z}_{max}$, and $\boldsymbol{A}$. So, the region computed for each neighbor at each step in the prediction horizon is a row defined in terms of $\boldsymbol{v}_A$, $\boldsymbol{u}_{\mathrm{ORCA}}$, and $\boldsymbol{n}$, such as

$$\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{n} \end{bmatrix}^\top \boldsymbol{x}(k) \leq (\boldsymbol{v}_A + \frac{1}{2}\boldsymbol{u}_{\mathrm{ORCA}})\boldsymbol{n}. \quad (13)$$

When in very dense scenarios with conflicting trajectories, it is possible to get infeasible optimization problems because of too many constraints. Some strategies can be employed in this case. In this work, a velocity damping with a norm close to zero is applied. It assumes that MPC predictions can anticipate infeasible situations. Besides, other agents could be moving, making the environment non-static, leading to the optimization problem out of infeasible regions.

6. SIMULATION SCENARIOS

The algorithms developed were written in Python language over the Robot Operating System (ROS) environment. ROS is a middleware intended to design robot software, counting with a library collection, tools, and simulation interfaces. In this work, we used Gazebo with the physics engine Open Dynamics Engine (ODE). Through Gazebo functionalities, a differential wheeled robot was built in the markup language Unified Robot Description Format (URDF), and forces and torques as disturbances acting over the vehicle.

Each ROS independent routine or application is defined as a node. The communication between nodes is made through message channels known as topics. The robots are controlled in simulation through a velocity command published in the topic */cmdvel* of each agent. It is worth mentioning that the controllers do not communicate directly among themselves but can make environment readings. So, robots in simulation receive commands through velocity messages as input variables, although the model in the algorithm considers acceleration too.

In this work, we used a general-purpose solver for optimization problems involving quadratic programming, the Operator Splitting Quadratic Program (OSQP), capable of high precision solutions and based on the Alternating Direction Method of Multipliers (ADMM) algorithm, that applies a first-order method to find a zero of the sum of monotone operators (Stellato et al., 2018; Boyd et al., 2010).

In the optimization problem, the matrices $\boldsymbol{P}$ and $\boldsymbol{A}$ are sparse, presenting a significant number of zero values, which could occupy memory space inefficiently and harm the operation of the system. Thus, these matrices are allocated in Compressed Sparse Columns (CSC) structures, compatible with OSQP, and available in the scientific computing library SciPy.

Two multiagent scenarios were designed for the validation of the combined MPC-ORCA strategy implementation. The parameters utilized in the controller were $\boldsymbol{Q}(1) = diag\,(3.0\ 3.0\ 0.0\ 0.0)$[1], $\boldsymbol{Q}(k) = diag\,(1.5\ 1.5\ 0.0\ 0.0)$, $\boldsymbol{R}(k) = diag\,(0.55\ 0.55)$, $N = 10$ and $T_s = 0.1s$. Choosing $\boldsymbol{Q}$ will penalize position error only, disregarding the velocity variables, as long as the velocity constraints are respected. $\boldsymbol{Q}(1)$ presents slightly higher values for the controller to find a solution that tries to reduce the position error in the next immediate interaction but still considers the error originated by the prediction. The assumed $\boldsymbol{R}$ values apply a small penalty to the acceleration, intending to smooth the velocity behavior, therefore no $\boldsymbol{u}_{min}$ and $\boldsymbol{u}_{max}$ constraints were needed. It is considered that the vehicles operate in a velocity range of $\pm 5m/s$, implemented as the only state constraints in $\boldsymbol{x}_{min}$ and $\boldsymbol{x}_{max}$. With a sample time of $T_s = 0.1$ seconds, a prediction horizon $N = 10$ allows predictions of 1 second. The ORCA parameter for the collision cone truncation $\tau = 5s$ was set for the algorithm computation.

[1] The function diag : $\mathbb{R}^n \to \mathbb{R}^{n\times n}$ returns a diagonal matrix with elements defined by the arguments.

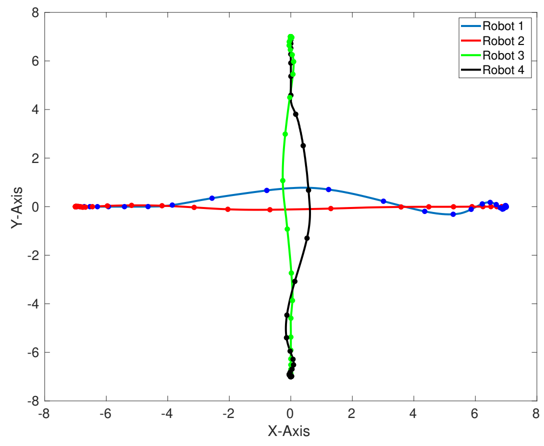


Figure 3. Resulting path for scenario 1.

Without loss of generality, both scenarios assume four differential drive robots using the linearized model aforementioned, which are initially positioned at opposed and equidistant places related to one another. Trajectories are assigned to intentionally induce a collision of all agents approximately at point $p = [0\ 0]^T$ and time $t = 10$ seconds. Table 1 inform where robots were placed and their goals in each different scenario purposed.

Table 1. Robots initial positions and goals by scenario.

Robot	**Scenario 1**		**Scenario 2**	
	$\boldsymbol{p}_{\text{start}}$	$\boldsymbol{p}_{\text{goal}}$	$\boldsymbol{p}_{\text{start}}$	$\boldsymbol{p}_{\text{goal}}$
1	$[-7\ 0]^\top$	$[0\ 7]^\top$	$[-7\ -7]^\top$	$[7\ 7]^\top$
2	$[7\ 0]^\top$	$[-7\ 0]^\top$	$[-7\ 7]^\top$	$[7\ -7]^\top$
3	$[0\ -7]^\top$	$[7\ 0]^\top$	$[7\ -7]^\top$	$[-7\ 7]^\top$
4	$[0\ 7]^\top$	$[0\ -7]^\top$	$[7\ 7]^\top$	$[-7\ -7]^\top$

Without loss of generality, the reference trajectory for the experiments was computed using a logistic function from the robot's initial position to a destination, defined with a duration of 20 seconds. It emulates smooth and realistic movements, as the position and velocity components are differentiable. The trajectory is computed offline, disregarding moving obstacles or neighbor agents.

7. RESULTS

Figure 3 presents a view for the movement of the robots during simulation. Despite the reference trajectory generating a straight path, the robots perform predictive deviations to avoid a collision [1] . The same is shown in Figure 4 from the second experiment simulation. After that, there is no collision risk, and the agents try to return to their reference trajectory [2] .

It is stated that MPC-ORCA makes it possible to react anticipated to collision, reducing occurrences of optimization infeasible situations. However, the closer the velocity maximum and minimum limits from one another, it is more difficult for the controller to return an agent to its desired trajectory after a case of infeasibility, since $\boldsymbol{v}_{max}$ and $\boldsymbol{v}_{min}$ are hard constraints in all prediction horizon. That requires softening of those constraints, i.e., increasing the admissible velocities interval.

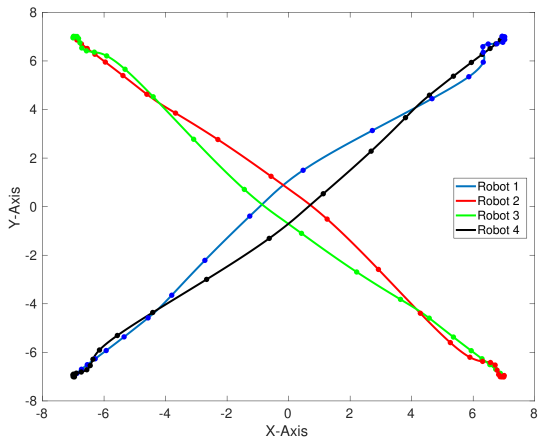


Figure 4. Resulting path for scenario 2.

Smoother collision avoidance turns are achieved when there is a penalty for acceleration, as in Equation (12). These lighter deviations are more straightforward to correct afterward than heavier ones, both from the perspective of smaller control effort and optimization algorithm computation.

To discuss the trajectory tracking behavior, Figure 5 presents the positions and velocities for a robot during simulation of the first scenario. As seen in Figure 5a, the robot was able to start following its trajectory in the x-axis and y-axis without great deviations. However, at approximately $t = 5.75$ seconds, it predicts a possible collision path, i. e. enters a velocity obstacle region in the following prediction steps. In fact, as declared before, a collision was intentionally programmed for all robots in $t = 10$ seconds. That results in the deviation that can be seen more clearly in the y-axis at time $t = 5.75$ seconds.

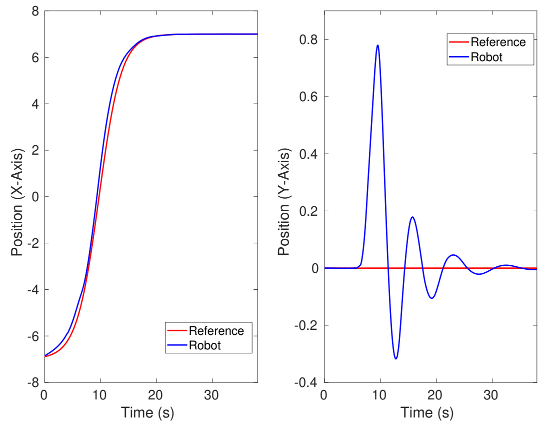
The same analysis is valid in the velocity plots, shown in Figure 5b. If there were no collision predictions in that scenario, the angular velocity would remain zero. However, it changes the same time when linear velocity changes a little its behavior and y-axis position starts to show the needed deviation from reference trajectory to avoid collision. To ease the results discussion, the y-axis position and angular velocity plots scales were zoomed in for that experiment.

8. CONCLUSION

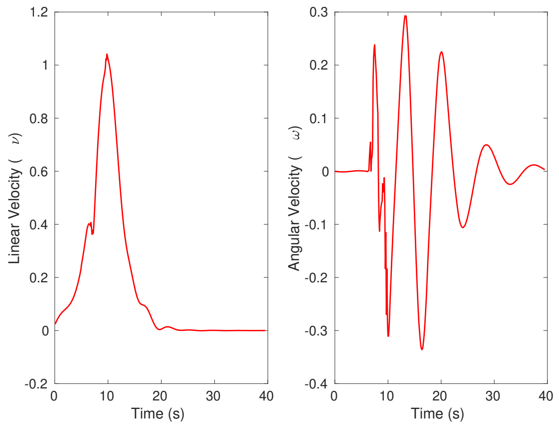
This work presented an application of model predictive control combined with a collision avoidance algorithm in a distributed system of mobile robots with individual objectives, without the need of recurrent trajectory replanning neither direct communication between agents. For each prediction horizon iteration, velocity constraints that would result in a collision were computed, then inserted in a quadratic optimization problem among other kinematic constraints, such as velocity and acceleration limits.

Differential wheeled robots were used, presenting nonholonomic constraints, demanding model linearization in

[1] Scenario 1 exhibition video in: https://youtu.be/0seApw2HIsA
[2] Scenario 2 exhibition video in: https://youtu.be/rRNrDTNMQBc



(a) Output positions for Robot 1.



(b) Input velocities for Robot 1.

Figure 5. Output and Input variables for Robot 1 in Scenario 1.

the implemented controller. However, the robots could perform smooth and collision-free trajectories, making minor deviations from reference trajectories. An agent using that approach can handle highly mutable environments where other agents can move with high velocity. Aside from supporting non-holonomic robots, an advantage of the developed solution from other studies is operating with an arbitrary preplanned trajectory as a reference instead of a single objective position.

As possible future work efforts, the proposed algorithm could be implemented in real robot scenarios, running over a navigation stack. Although this study focused on planar robots, the employed techniques could be extended to control and collision avoidance for robots operating in configuration space with higher dimensions, such as aerial vehicles or robotic manipulators. Furthermore, there could be works on implementing MPC-ORCA in a lower control level, delivering commands for wheel torque or acceleration.

ACKNOWLEDGES

We would like to thank our research group Easy Engineering and Systems that is part of the Computing Institute (IC) from Federal University of Alagoas (UFAL).

REFERENCES

Ahmad Abu Hatab, R.D. (2013). Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework. *Advances in Robotics & Automation*, 02(02). doi:10.4172/2168-9695.1000107.

Arkin, R.C. and Balch, T. (1998). *Cooperative Multiagent Robotic Systems*, 277–296. MIT Press, Cambridge, MA, USA.

Boyd, S., Boyd, S., Vandenberghe, L., and Press, C.U. (2004). *Convex Optimization*. Berichte über verteilte messysteme. Cambridge University Press. doi:10.1017/CBO9780511804441. URL `https://books.google.com.br/books?id=mYm0bLd3fcoC`.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1), 1–122. doi:10.1561/2200000016.

Camacho, E.F. and Bordons, C. (2007). *Model Predictive control*. Advanced Textbooks in Control and Signal Processing. Springer London, London, 2nd edition. doi:10.1007/978-0-85729-398-5. URL `http://link.springer.com/10.1007/978-0-85729-398-5`.

Cheng, H., Zhu, Q., Liu, Z., Xu, T., and Lin, L. (2017). Decentralized navigation of multiple agents based on ORCA and model predictive control. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe, 3446–3451N. doi:10.1109/IROS.2017.8206184.

Fiorini, P. and Shiller, Z. (1993). Motion planning in dynamic environments using the relative velocity paradigm. *Proceedings IEEE International Conference on Robotics and Automation*, 560–565. doi:10.1109/robot.1993.292038.

Graetz, G. and Michaels, G. (2018). Robots at Work. *The Review of Economics and Statistics*, 100(5), 753–768. doi:10.1162/rest_a_00754.

Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6, 1–10. doi:https://doi.org/10.1016/j.jii.2017.04.005. URL `http://www.sciencedirect.com/science/article/pii/S2452414X17300043`.

Mao, R., Gao, H., and Guo, L. (2020). A novel collision-free navigation approach for multiple nonholonomic robots based on orca and linear mpc. *Mathematical Problems in Engineering*, 2020, 1–16. doi:10.1155/2020/4183427.

Mouad, M., Adouane, L., Khadraoui, D., and Martinet, P. (2012). *Mobile robot navigation and obstacles avoidance based on Planning and Re-Planning algorithm*, volume 45. IFAC. doi:10.3182/20120905-3-HR-2030.00170. URL `http://dx.doi.org/10.3182/20120905-3-HR-2030.00170`.

Ogata, K. (1995). *Discrete-time Control Systems*, volume 2. Prentice Hall. URL `https://books.google.com.br/books?id=owQqAQAAMAAJ`.

Ota, J. (2006). Multi-agent robot systems as distributed autonomous systems. *Advanced Engineering Informatics*, 20(1), 59–70. doi:10.1016/j.aei.2005.06.002. URL `https://linkinghub.elsevier.com/retrieve/pii/S1474034605000509`.

Qin, S. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7), 733–764. doi:10.1016/S0967-0661(02)00186-7. URL `https://linkinghub.elsevier.com/retrieve/pii/S0967066102001867`.

Ren, W. and Atkins, E. (2007). Distributed multi-vehicle coordinated control via local information exchange. *International Journal of Robust and Nonlinear Control*, 17(10-11), 1002–1033. doi:10.1002/rnc.1147. URL `http://doi.wiley.com/10.1002/rnc.1147`.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2018). OSQP: An Operator Splitting Solver for Quadratic Programs. *2018 UKACC 12th International Conference on Control, CONTROL 2018*, 339. doi:10.1109/CONTROL.2018.8516834.

Stentz, A. (1995). The focussed D* algorithm for real-time replanning. *14th International Joint Conference on Artificial intelligence (IJCAI)*, 95(August), 1652–1659. doi:10.1080/03057929502503O2. URL `http://portal.acm.org/citation.cfm?id=1643031.1643113`.

Sycara, K.P. (1998). Multiagent Systems. *AI Magazine*, 19(2), 79–92. doi:10.1609/aimag.v19i2.1370.

Van Den Berg, J., Guy, S.J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In C. Pradalier, R. Siegwart, and G. Hirzinger (eds.), *Robotics Research*, 3–19. Springer Berlin Heidelberg, Berlin, Heidelberg. doi:10.1007/978-3-642-19457-3_1.

Van den Berg, J., Ming Lin, and Manocha, D. (2008). Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, volume 23, 1928–1935. IEEE. doi:10.1109/ROBOT.2008.4543489. URL `http://ieeexplore.ieee.org/document/4543489/`.