Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 4183427, 16 pages
https://doi.org/10.1155/2020/4183427

Research Article

A Novel Collision-Free Navigation Approach for Multiple Nonholonomic Robots Based on ORCA and Linear MPC

Run Mao,[1] **Hongli Gao**,[2] **and Liang Guo**[1]

[1]*Department of Electromechanical Measuring and Controlling, School of Mechanical Engineering, Southwest Jiaotong University, 610036 Chengdu, Sichuan, China*
[2]*Engineering Research Center of Advanced Driving Energy-Saving Technology, Ministry of Education, Southwest Jiaotong University, 610036 Chengdu, Sichuan, China*

Correspondence should be addressed to Hongli Gao; hongli_gao@swjtu.cn and Liang Guo; guoliang@swjtu.edu.cn

Received 28 March 2020; Revised 12 May 2020; Accepted 21 May 2020; Published 10 June 2020

Guest Editor: Carlos Llopis-Albert

In the study of collision-free navigation methods of multirobots, much attention has been paid to the constraints of external environment. However, most of the wheeled mobile robots are subjected to nonholonomic constraints. A collision between robots may occur if the nonholonomic constraints are neglected. This paper presents an improved approach to collision-free navigation for multi-nonholonomic robots. This approach combines the Optimal Reciprocal Collision Avoidance (ORCA) algorithm and Model Predictive Control (MPC) strategy. ORCA used a simple robot model, in which kinematics and dynamics are ignored. To cope with this problem, the MPC controller is introduced. In each ORCA step, the reference trajectory, reference control inputs, and "safe zones" are generated based on the new velocity. Consequently, the derived safe zone is transformed into the constraints of decision variables for a MPC controller. Finally, quadratic programming is used to solve the MPC problem by successive linearization of an error model of the mobile robot. Simulation results illustrate the effectiveness of the proposed method.

1. Introduction

Motion planning/collision avoidance is concerned with computing a path or trajectory between two configurations embedded in cost field, while taking into account motion constraints, static obstacles, and moving obstacles [1]. The problem has been well studied for one robot avoiding static or moving obstacles [2–4].

Many classical approaches such as artificial potential field method [5], sampling-based algorithms [6–9], and dynamic window [10] are extended to dynamic environments [11]. These approaches assume the observed obstacles to be static in a period of time [12–14] and compute an immediate action to avert collisions with the consideration of kinematics and dynamics in many cases. When considering moving obstacles, such approaches typically repeatedly plan based on continuous cycle of sensing and acting. This may work well as if the obstacles move slower than the robot, but among fast obstacles, the static assumption will not lead a collision-free navigation. The methods such as multilayer fuzzy control method [15], reactive collision avoidance [16], and rolling window method [17] are also used in the dynamic environment. However, these methods based on a simple robot model ignore the nonholonomic constraints.

The problem of collision avoidance becomes even more complicated when the obstacles are also decision-making agents instead of entities moving arbitrarily without considering their environment [18]. This problem has many applications in the areas of robotics, such as multirobots navigation and coordination in the Container Terminal Logistics System [19] and crowd simulation for computer graphics and VR [20]. In the context of collision avoidance for multiple robots, similar approaches to those for the single robot case can be applied [21–23]. However, the increase in robot density may lead to oscillations [24]. Many motion-planning methods of multiple point-mass robots have been developed [24–26]. It means that applying these methods into practice directly is difficult because the mobility of the

robot is ignored. The Net-MPC strategy is also used to navigate in multirobots environment [27–30]. However, these methods relying on the communication between robots may increase the complexity of system.

A particularly successful method for mobile robot real-time navigation is the ORCA algorithm [31]. It is an effective decentralized collision avoidance method for a large number of decision-making agents. The ORCA infers for each robot a half-plane of velocities that are allowed to be selected in order to guarantee collision avoidance without communication among the robots or central coordination. This method works well if the robots act on holonomic constraints, since the robots must move along the direction of a new selected velocity by ORCA immediately. For many nonholonomic constrained robots, such as car-like robots, the set of feasible velocities at any instant is a specific velocity in the direction of the rear wheels [32, 33]. One way to work around these constraints is to use the set of velocities that can be achieved over some time interval τ [26]. An optimal method for distributed collision avoidance among multiple nonholonomic robots is presented in [32]. Furthermore, Jeon and Lee defined the wheel velocity obstacle, which is a set of all the left and right wheel velocity pairs that induce collisions with obstacles within a given time horizon [34]. However, these methods can only guarantee collision-free with the instant of each step time. To alleviate these issues, we can select a small value for τ, but this has some implications for navigation: as τ decreases, the set of velocities that are being considered by the robot becomes smaller, and the robot can miss feasible controls.

The Model Predictive Control (MPC) method is suitable for solving the multivariable constrained control problems due to its predictive nature. It means that the MPC method can handle the environment constraints and robot constraints in a single optimization problem [35, 36]. The applications of MPC for navigation of a single robot have been reported in [36–38].

Motivated by the features of MPC and ORCA methods, in this paper, we propose a Discrete-ORCA-MPC combined approach for the problem where collisions need to be avoided among multiple nonholonomic mobile robots with decision-making ability. Concretely, the ORCA algorithm is used by each agent to generate independently its new velocity at each cycle of sensing (ORCA step time τ). This new velocity is used to generate a discrete reference control input and trajectory. Meanwhile, a series of safe zones are constructed, which serve as constraints in a MPC problem. Unlike original ORCA, the construction of safe zones needs to know the new velocities of other agents in the neighbor region at each step. Furthermore, the proof for collision-free motion of this Discrete-ORCA-MPC approach is derived. Although Nonlinear Model Predictive Control (NMPC) has been well developed, the computational effort necessary is much higher than the linear model [39, 40]. This paper uses a successive linearization approach yielding a linear, time-varying description of the system which can be solved through linear MPC. Considering the control inputs as the decision variables, it is then used to transform the optimization problem into a Quadratic Programming (QP) problem. Since this is a convex problem, QP problems can be solved by commercial numerical solvers which lead to global optimal solutions. More importantly, it is easy to a real-time implementation. In addition, we focus on differential drive mobile robots (DDMRs) in the following work, even though our approach applies more generally for the class of feedback linearizable vehicles with nonholonomic kinematics, such as car-like robots or DDMR with trailer.

The main features of the proposed Discrete-ORCA-MPC combined approach are summarized as follows:

(1) Different from the traditional methods such as artificial potential functions based on a point-mass robot model, the proposed method considers the nonholonomic constraints of robots. The point-mass robot model cannot guarantee collision-free navigation, since a car-like robot will actually need to follow an arc to achieve the selected velocity, and this arc may lead to a collision with other robot. Consequently, the proposed approach solves this problem by using a dynamic model to predict the future system.

(2) Different from the network-based method such as Distributed Model Predictive Control method, the proposed method does not rely on communications among the robots or central coordination. Therefore, it is effective for multirobots to avoid obstacles real-time without a real-time communication environment.

The remainder of this paper is organized as follows: in the next section the brief definition of the ORCA is shown. The problem statement is illustrated in Section 3. Section 4 puts out the kinematic model of DDMR and the linear MPC algorithm. Simulation results in MATLAB and C++ are shown in Section 5. Section 6 presents some conclusions.

2. Optimal Reciprocal Collision Avoidance (ORCA)

ORCA is a rigorous approach for reciprocal n-body collision avoidance that allows each agent to compute independently the optimal moving velocities in each step [31]. In this section, we review the concept of velocity obstacles and discuss its application to DDMR with nonholonomic constraints.

2.1. Definition of ORCA. ORCA provides a sufficient condition for multiple robots to avoid collisions among one another and thus can guarantee collision-free navigation [31]. Let there be a set of n disc-shaped robots (this disc-shaped assumption can easily be extended to translate polygons) moving in the plane $\mathbb{R}^2$. Each robot has a current position $\mathbf{p}$, a current velocity $\mathbf{v}$, and a radius r. It is assumed that these parameters can be observed by other robots. Furthermore, each robot has a maximum speed $\mathbf{v}_{\max}$ and a preferred velocity $\mathbf{v}_{\text{pref}}$ (for instance, a velocity directed towards the robot's goal with a magnitude equal to the robot's preferred speed). These parameters are considered as

internal state of the robot, which cannot therefore be observed by other robots. The brief definitions are presented in the following.

Let $D(\mathbf{p}, r)$ denote an open disc of radius r centered at $\mathbf{p}$:

$$D(\mathbf{p}, r) = \{\mathbf{q} \mid \|\mathbf{q} - \mathbf{p}\| < r\}. \quad (1)$$

The velocity obstacle $\mathrm{VO}^{\tau}_{A|B}$ is the set of all relative velocities of A with respect to B that will result in a collision at some moment before time τ. The geometric interpretation of velocity obstacles is shown in Figure 1. And it can be formally represented as

$$\mathrm{VO}^{\tau}_{A|B} = \{\mathbf{v} \mid \exists t \in [0, \tau] :: t\mathbf{v} \in D(\mathbf{p}_B - \mathbf{p}_A, \mathbf{r}_A + \mathbf{r}_B)\}, \quad (2)$$

where $\mathbf{p}_A$ and $\mathbf{p}_B$ are the positions of agent A and agent B, respectively. r_A (r_B) is the radius of the safe zone of agent A (agent B) and is chosen slightly larger than the radius of the agent. Note that for agent B, its velocity obstacle is $\mathrm{VO}^{\tau}_{B|A} = -\mathrm{VO}^{\tau}_{A|B}$.

Let $\mathbf{v}^{\mathrm{opt}}_A$ and $\mathbf{v}^{\mathrm{opt}}_B$ be the optimization velocities of A and B, respectively. Nominally, the optimization velocities are equal to the current velocities, such that the robots have to deviate as little as possible from their current trajectories to avoid collisions. Let $\mathbf{u}$ be the vector from $\mathbf{v}^{\mathrm{opt}}_A - \mathbf{v}^{\mathrm{opt}}_B$ to the closest point on the boundary of the velocity obstacle:

$$\mathbf{u} = \left(\underset{\mathbf{v} \in \partial \mathrm{VO}^{\tau}_{A|B}}{\mathrm{argmin}} \left\| \mathbf{v} - \left(\mathbf{v}^{\mathrm{opt}}_A - \mathbf{v}^{\mathrm{opt}}_B\right) \right\| \right) - \left(\mathbf{v}^{\mathrm{opt}}_A - \mathbf{v}^{\mathrm{opt}}_B\right), \quad (3)$$

and let $\mathbf{n}$ be the outward normal of the boundary of $\mathrm{VO}^{\tau}_{A|B}$ at point $(\mathbf{v}^{\mathrm{opt}}_A - \mathbf{v}^{\mathrm{opt}}_B) + \mathbf{u}$. Consequently, $\mathbf{u}$ is the smallest change required to the relative velocity of A and B to avert collision with τ time. To share the responsibility of avoiding collisions among the robots in a fair way, The increment $(1/2)\mathbf{u}$ means that agent A takes half the responsibility of avoiding potential collision with agent B, while the remaining half will be taken by agent B. Hence, the set $\mathrm{ORCA}^{\tau}_{A|B}$ of permitted velocities for A is the half-plane pointing in the direction of $\mathbf{n}$ starting at the point $\mathbf{v}^{\mathrm{opt}}_A + (1/2)\mathbf{u}$. More formally,

$$\mathrm{ORCA}^{\tau}_{A|B} = \left\{ \mathbf{v} \mid \left(\mathbf{v} - \left(\mathbf{v}^{\mathrm{opt}}_A + \frac{1}{2}\mathbf{u} \right) \right) \mathbf{n} \geq 0 \right\}. \quad (4)$$

This set is illustrated in Figure 2. By this definition, the chosen new relative velocities will not enter $\mathrm{VO}^{\tau}_{A|B}$, and consequently the velocities of the agent can be smoothed.

Each robot A performs a continuous cycle of sensing and acting with time step Δt. In each iteration, the robot acquires the radius, the current position, and the current optimization velocity of the other robots. Based on this information, the robot infers the permitted half-plane of velocities $\mathrm{ORCA}^{\tau}_{A|B}$ with respect to each other robot B. The set of ORCA velocities for agent A with respect to all robots is the intersection of the half-planes of $\mathrm{ORCA}^{\tau}_{A|B}$ induced by each other robot B:

$$\mathrm{ORCA}^{\tau}_{A} = D(\mathbf{0}, v^{\max}_A) \cap \bigcup_{B \neq A} \mathrm{ORCA}^{\tau}_{A|B}. \quad (5)$$

Note that $D(\mathbf{0}, v^{\max}_A)$ include the maximum speed constraint on the robot A.

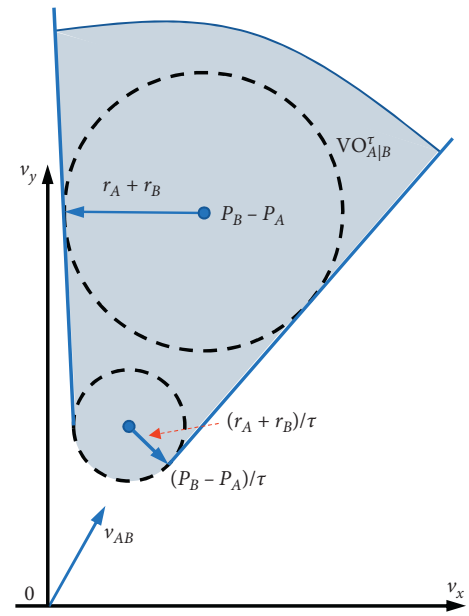

Figure 1: The velocity obstacle for a short time horizon $\mathrm{VO}^{\tau}_{A|B}$.

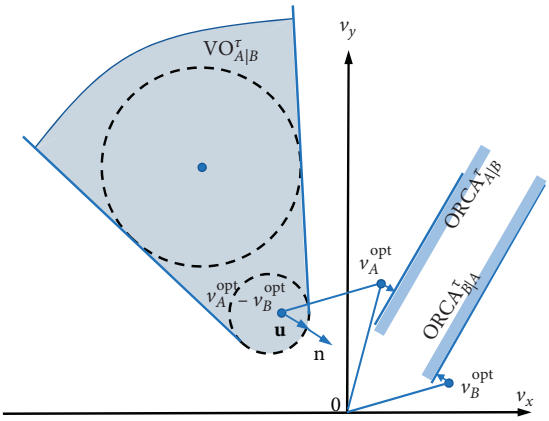

Figure 2: $\mathrm{VO}^{\tau}_{A|B}$ and $\mathrm{ORCA}^{\tau}_{A|B}$ for agent A when $(\mathbf{v}_A - \mathbf{v}_B)$ lies in $\mathrm{VO}^{\tau}_{A|B}$.

At last, the new velocity $\mathbf{v}^{\mathrm{new}}_A$ is selected from ORCA^{τ}_{A}, which is the closest to its preferred velocity $\mathbf{v}^{\mathrm{pref}}_A$ amongst all velocities inside the region of permitted velocities:

$$\mathbf{v}^{\mathrm{new}}_A = \underset{\mathbf{v} \in \mathrm{ORCA}^{\tau}_A}{\mathrm{argmin}} \left\| \mathbf{v} - \mathbf{v}^{\mathrm{pref}}_A \right\|. \quad (6)$$

The ORCA algorithm in [31] used a simple robot model, in which kinematics are ignored. Consequently, the robot reaches its new position:

$$\mathbf{p}^{\mathrm{new}}_A = \mathbf{p}_A + \mathbf{v}^{\mathrm{new}}_A \Delta t. \quad (7)$$

In many of these applications, the robot of interest is a car-like robot and subject to nonholonomic kinematic constraints [33]. Hence, the constant velocity assumption rarely holds [41].

2.2. Nonholonomic Constraints. ORCA is a velocity obstacle based algorithm, which is a first-order method since it does not integrate velocities to yield positions as functions of time [26]. It means that the control input to the robot is velocity. By considering nonholonomic kinematic constraints, the navigation problem of a differential drive robot generally followed these two steps: first, the velocity $\mathbf{v}_H$ is generated by the ORCA based on the assumption that the robot is holonomic; second, the robot tracks this velocity by using the controller with nonholonomic constraints [34]. However, these will make a tracking error δ between the holonomic trajectory and the real trajectory inevitable, as shown in Figure 3.

The real nonholonomic trajectory is defined by two sections: an arc of circumference AB covered with constant linear speed $v(t)$ and decreasing angular velocity $\omega(t)$, followed by a straight line path BC with constant speed $v(t)$. In this case, a mobile robot will actually need to follow an arc to achieve the selected velocity $\mathbf{v}_H$, and the set of ORCA_A^τ provide no guarantee of collision-free.

3. Problem Statement

In this section, we define a new concept called the safe zone. This is an extended generalization of the ORCA concept and seeks to address the difficulty of using ORCA with nonholonomic constrained agents. The proof of collision-free navigation of safe zone is also derived. Note that, the actions of ORCA are computed for each robot independently, without communication among the robots or central coordination. Unlike original ORCA algorithm, the construction of safe zone needs to know the new velocities of other agents in neighbor region at each step. This is not a severe limitation since the multiagent system has been studied extensively.

3.1. Safe Zone. We consider multiple mobile robots sharing a common workspace where the robots have the same equations of motion and state spaces. Let the state space of robot be $\mathcal{X} \subset \mathbb{R}^n$. Let $\mathbb{R}^d$ be the robot's physical workspace. Let $\mathcal{O} \subset \mathbb{R}^d$ be the geometry of robot relative to its position. For simplicity, we restrict our analysis to circular robots and obstacles, which is not a severe limitation since general polygons can be represented by a number of circles [26]. We assume the position $p \in \mathbb{R}^d$ of robot can be derived from its state $\mathbf{x} \in \mathcal{X}$ by a potentially projection function $\mathbf{q} \in \mathcal{X} \longrightarrow \mathbb{R}^d$:

$$\mathbf{p}(t) = \mathbf{q}(\mathbf{x}(t)). \tag{8}$$

The control input space $\mathcal{U} \subset \mathbb{R}^m$ is assumed to be convex. Let the continuous-time equation of motion for robot be given by a potentially nonlinear function $\mathbf{f} \in \mathcal{X} \times \mathcal{U} \longrightarrow \mathbb{R}^n$:

$$\dot{x}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{9}$$

where $\mathbf{x}(t)$ is the state and $\mathbf{u}(t)$ is the control input at time t.

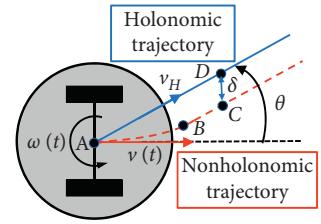

Figure 3: The holonomic trajectory is tracked by the DDMR moving along the nonholonomic trajectory with a tracking error.

Given a current state $\mathbf{x}(0)$ of robot and some constant control input $\mathbf{u}(0)$ at time 0, the state of the robot at a given time $t > 0$ is given by

$$\mathbf{x}(t) = \mathbf{g}(t, \mathbf{x}(0), \mathbf{u}(0)), \tag{10}$$

where $\mathbf{g} \in \mathbb{R} \times \mathcal{X} \times \mathcal{U} \longrightarrow \mathcal{X}$ is the solution to the differential equation of equation (9), which can be obtained numerically.

Consider two circular robots A and B in an environment. Let circle A represent the robot, and let circle B represent the obstacle. Given the position of agent A at time $t = 0$ and a control $\mathbf{u}_A$, let us denote the position agent A will be later undertaking $\mathbf{u}_A$ for time interval t by $\mathbf{p}_A(t)$. Similarly, $\mathbf{p}_B(t)$ represents the state of agent B undertaking $\mathbf{u}_B$ for time interval t. Note that the control $\mathbf{u}$ is the control inputs to the robot that results in a change in the robot's configuration. Then, the control obstacle can be defined as

$$\left\{u_A, u_B \,\middle|\, \exists t \in [0, \tau] :: \left\|p_A(t) - p_B(t)\right\| < r_A + r_B\right\}, \tag{11}$$

where $p_A(t) = q_A(g_A(t, x_A(0), u_A))$ and $p_B(t) = q_B(g_B(t, x_B(0), u_B))$ by using equations (8) and (10). As shown in Figure 4, $\mathbf{v}_A(0) \notin \mathrm{ORCA}_{A|B}^\tau$, and symmetrically, $\mathbf{v}_B(0) \notin \mathrm{ORCA}_{B|A}^\tau$. Robots A and B chose constant velocities $\mathbf{v}_{AH}$ and $\mathbf{v}_{BH}$ outside the obstacle velocity set and generate trajectory $\mathbf{p}_{AH}(t)$ and $p_{AH}(t)$, respectively. Then A and B take controls u_A and $\mathbf{u}_B$ (for instance, $u = [v_H, k(\phi - \phi_d)]$), which generate trajectory $p_A(t)$ and $\mathbf{p}_B(t)$ (red dashed line), respectively. Note that this will lead to a collision, although the minimum distance between two holonomic trajectory (blue solid line) is greater than the radii sum of two robots ($d_{\min} > r_A + r_B$).

In fact, the ORCA algorithm can be transformed using a uniform sampling scheme to deal with the nonholonomic kinematic constraints. As illustrated in Figure 5, let $\{p_{AH}(0), p_{AH}(1), \ldots, p_{AH}(N)\}$ and $\{p_{BH}(0), p_{BH}(1), \ldots, p_{BH}(N)\}$ be the uniform sampling waypoints of holonomic trajectory $\mathbf{p}_{AH}(t)$ and $\mathbf{p}_{BH}(t)$, respectively. The distance between $\mathbf{p}_A(k)$ and $p_B(k)$ is $l_{AB}(k)$ for $k \in \{0, 1, 2, \ldots, N\}$. Note that $\mathbf{p}(N)$ is the terminal point of one step of ORCA; that is, $\mathbf{p}(N) = \mathbf{p}(\tau)$. Hence, the collision avoidance set of ORCA algorithm can be reformulated as

$$\mathrm{CA}_{A|B}^\tau = \left\{u_{As}, u_{Bs} \,\middle|\, \forall k \in \{0, 1, \ldots, N\} :: l_{AB}(k) > r_A + r_B\right\}, \tag{12}$$

where

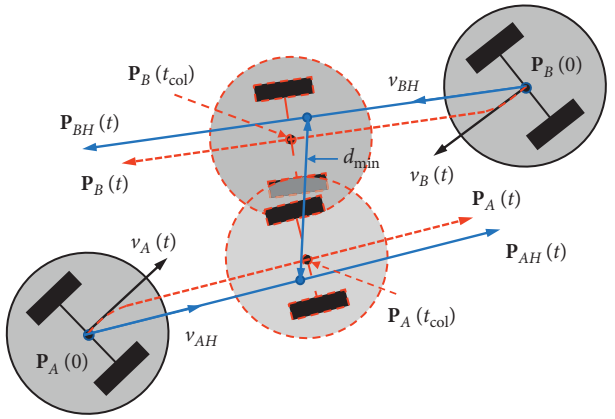

FIGURE 4: Robots A and B collide in $p_A(t_{col})$ and $p_B(t_{coal})$ caused by the deviations for the nonholonomic constraints.

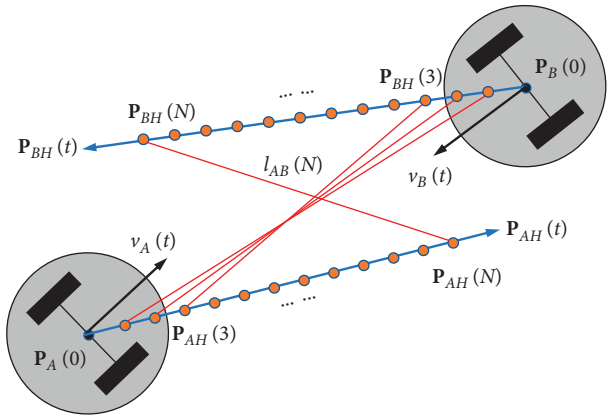

FIGURE 5: The discrete trajectories of robots A and B generated by ORCA.

$$
\begin{aligned}
l_{AB}(k) &= \left\| p_A(k) - p_B(k) \right\|, \\
u_{As} &= \left[u_A(0), u_A(1), \ldots, u_A(N-1) \right], \\
u_{Bs} &= \left[u_B(0), u_B(1), \ldots, u_B(N-1) \right].
\end{aligned} \tag{13}
$$

Let $r_s^i(k)$ be the safe zone radius of robot i at time k, and it can be defined as

$$
r_s^i(k) = \frac{1}{2} \min_{j \neq i} \left(l_{ij}(k) - r_i - r_j \right), \tag{14}
$$

where r_i and r_j are the radii of robots i and j. The constructive process of safe zone can geometrically be illustrated in Figure 6. Note that $r_s^{ij}(k) = l_{ij}(k) - r_i - r_j$ and b is side length of the maximum inscribed square. This square is the approximate safe zone for linear constraints (see next section). If robot i applies the control input $\mathbf{u}_i(k-1)$ in time $k-1$ that "walk" to the safe zone $r_s^i(k)$ in time k will lead to collision avoidance.

3.2. Proof of Collision-Free Navigation

Theorem 1. *If the safe zone of robot i at control step k is equal to the relative safe zone between robots i and j, the safe zone of robot j must have been less than or equal to the safe zone of robot i in this moment. Then, the mathematical relationship can be described by* $r_s^i(k) = r_s^{ij}(k) \Rightarrow r_s^j(k) \leq r_s^i(k)$.

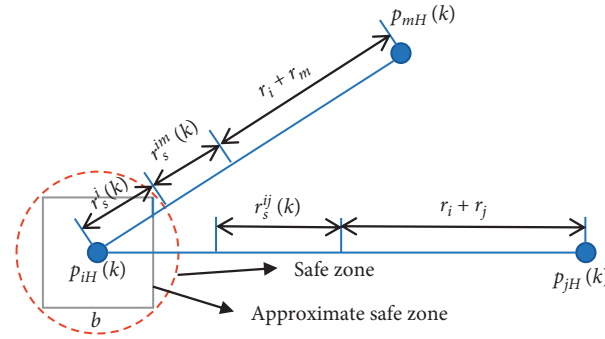

FIGURE 6: The safe zone for robot i with two neighbor robots j and k.

Proof. If $r_s^j(k) > r_s^i(k) \Rightarrow \exists m: : r_s^{jm}(k) = (1/2) \min_{m \neq j} (l_{jm}(k) - r_j - r_m) > r_s^i(k) = r_s^{ji}(k)$.

According to (14), $r_s^{ji}(k) \geq (1/2) \min_{m \neq j} (l_{jm}(k) - r_j - r_m)$.

By using reduction to absurdity, Theorem 1 is proved.

For robot i, there are n robots in its neighbor region. If the radius of safe zone at control step k is $r_s^i(k) = r_s^{ij}(k)$, it means that $r_s^{ij}(k)$ is the minimum dimension for robot i, but it does not mean that the $r_s^{ij}(k)$ is the minimum dimension for robot j. Theorem 1 gives the proof of this fact. □

Theorem 2. *In a multirobots environment, the collision-free navigation can be guaranteed when the safe zone constraints are maintained at every moment.*

Proof. Theorem $1 \Rightarrow \forall i, j \in R_s$ (robots set): $: r_i + r_j + r_s^i(k) + r_s^j(k) \leq l_{ij}(k)$.

Consider any pair of robots in one neighbor region, the sum of the radii of these two robots and their safe zones is less than the distance between these two robots. This theorem proves that the collision-free navigation for Discrete-ORCA can be guaranteed. □

4. Discrete-ORCA-MPC Combined Collision Avoidance

In this section, the brief overview of the kinematic model of a DDMR is given. Then, a successive linearization approach is introduced, which yields a linear, time-varying description of the DDMR system. The optimal control input is solved through linear MPC. Meanwhile, the safe zone and control constraints are satisfied by transforming the optimization problem in a QP problem.

4.1. Kinematics and Dynamics of DDMR. As illustrated in Figure 7, the configuration of a DDMR is given by the position of its center point $\mathbf{p} = [x, y]$ and its orientation ϕ. The configuration and high-level control input transition equations [42] are given as

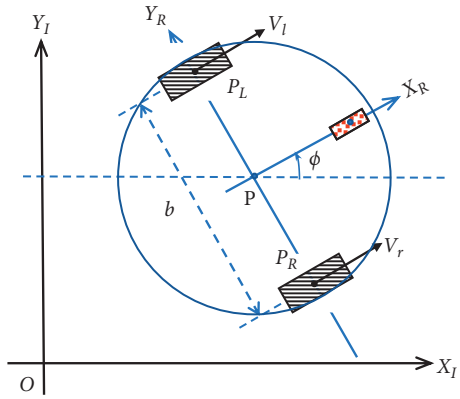

Figure 7: Structure of the differential drive mobile robot.

$$\begin{aligned}\dot{x} &= v\cos\phi,\\ \dot{y} &= v\sin\phi,\\ \dot{\phi} &= \omega,\end{aligned} \tag{15}$$

where $v = v_r + v_l/2$ is the linear velocity and $\omega = v_r - v_l/b$ is the angular velocity. b is the distance between the rear wheels of the robot, and v_r and v_l are the signed speeds of the left and right wheels, respectively. Generally, v_r and v_l are bounded such that $v_r, v_l \in [-v_{\max}, v_{\max}]$.

In general, the kinematic model can be written in the form of (9):

$$\dot{x} = f(\mathbf{x}, \mathbf{u}), \tag{16}$$

where $\mathbf{x} = [x, y, \phi]^T$ describes the configuration of point P (shown in Figure 7) and $\mathbf{u} = [v, \omega]^T$ is the control input.

Similarly, let $\mathbf{x}_r = [x_r, y_r, \phi_r]^T$ and $\mathbf{u}_r = [v_r, \omega_r]^T$ be the configuration and control input of reference system. Then, it can be described as

$$\dot{x}_r = f(\mathbf{x}_r, \mathbf{u}_r). \tag{17}$$

By expanding the right side of (16) in Taylor series around the reference point $(\mathbf{x}_r, \mathbf{u}_r)$, a linear model is obtained by discarding the high-order terms:

$$\dot{x} = f(\mathbf{x}_r, \mathbf{u}_r) + f'_{\mathbf{x}}(\mathbf{x} - \mathbf{x}_r) + f'_{\mathbf{u}}(\mathbf{u} - \mathbf{u}_r), \tag{18}$$

where $f'_{\mathbf{x}}$ and $f'_{\mathbf{u}}$ are the Jacobians of f with respect to $\mathbf{x}$ and $\mathbf{u}$, respectively, evaluated around the reference point $(\mathbf{x}_r, \mathbf{u}_r)$.

Then, the subtraction of (17) from (18) results in

$$\dot{\tilde{x}} = f'_{\mathbf{x}}\tilde{x} + f'_{\mathbf{u}}\tilde{u}, \tag{19}$$

where $\tilde{x} = \mathbf{x} - \mathbf{x}_r$ represents the error with respect to the reference robot configuration and $\tilde{u}$ is its error with respect to the reference input. Finally, the discrete-time system can be obtained:

$$\tilde{x}(k+1) = \mathbf{A}(k)\tilde{x}(k) + \mathbf{B}(k)\tilde{u}(k), \tag{20}$$

with

$$\begin{aligned}\mathbf{A}(k) &= \begin{bmatrix} 1 & 0 & -v_r(k)\sin(\phi_r)(k)T \\ 0 & 1 & v_r(k)\cos(\phi_r)(k)T \\ 0 & 0 & 1 \end{bmatrix},\\ \mathbf{B}(k) &= \begin{bmatrix} \cos(\phi_r)(k)T & 0 \\ \sin(\phi_r)(k)T & 0 \\ 0 & T \end{bmatrix},\end{aligned} \tag{21}$$

where T is the sampling period and k is the sampling time.

In this paper, let $v_i^r = \|\mathbf{v}_i^{\text{new}}\|$ and $\phi_i^r = \arctan\mathbf{v}_i^{\text{new}}$ be the reference velocity and direction of robot i. Then, the reference trajectory of robot i is obtained as (6):

$$X_i^r = \begin{bmatrix} x(k) + v_i^r\cos(\phi_i^r)T & y(k) + v_i^r\sin(\phi_i^r)T & \phi_i^r \\ x(k) + 2v_i^r\cos(\phi_i^r)T & 2y(k) + v_i^r\sin(\phi_i^r)T & \phi_i^r \\ \vdots & \vdots & \vdots \\ x(k) + N_\tau v_i^r\cos(\phi_i^r)T & N_\tau y(k) + v_i^r\sin(\phi_i^r)T & \phi_i^r \end{bmatrix}, \tag{22}$$

where $N_\tau = (\tau/T)$. It means that robot i generates N_τ waypoints in the direction of ϕ_i^r uniformly. In [40], we notice that the controllability matrix has full rank if either v_r or ω_r is nonzero. This implies the tracking of a reference trajectory being possible with linear MPC.

In the field of mobile robotics, predictive approaches to trajectory tracking seem to be very promising because the reference trajectory is known beforehand [40]. The essence of MPC approach is to optimize predictions of process behavior over a sequence of future control inputs. At each sampling time, the model predictive controller generates an optimal control sequence by solving an optimization problem [39]. In this section, the trajectory-tracking control of a differentially driven mobile robot based on MPC is presented. The main idea is to minimize the future trajectory-following errors and control errors between the robot and reference waypoints generated by Discrete-ORCA.

4.2. MPC Algorithm. The ideal of MPC concept it to find the values of control variable that minimize the receding horizon quadratic cost function based on the predicted robot tracking error model (20). In general, MPC can be proven to be stable using a terminal cost and terminal constraints in its optimization problem [43]. The cost function at time k can be stated as

$$\begin{aligned}\Phi(k) = \sum_{i=1}^{h} &\tilde{x}^T(k+i\,|\,k)Q\tilde{x}(|k+i|k)\\ &+ \tilde{u}^T(|k+i-1|k)R\tilde{u}(k+i-1\,|\,k),\end{aligned} \tag{23}$$

where h is the prediction horizon and $\mathbf{Q}$ and $\mathbf{R}$ are the weighting matrices (diagonal matrix with $\mathbf{Q} \geq 0$ and $\mathbf{R} \geq 0$). Note that, the angle error $(\phi - \phi_r)$ must be wrapped in radians to $[-\pi, \pi]$. The notation $a(m\,|\,n)$ indicates the value of a at the predicted time m based on the current time n. In order to reformulate the optimization problem in a usual

quadratic programming form, the $\widehat{x}(k+1)$ and $\widehat{u}(k)$ are defined as

$$\widehat{x}(k+1) = \begin{bmatrix} \widetilde{x}(k+1\,|\,k) \\ \widetilde{x}(k+2\,|\,k) \\ \vdots \\ \widetilde{x}(k+h\,|\,k) \end{bmatrix}, \qquad \widetilde{u}(k) = \begin{bmatrix} \widetilde{u}(k\,|\,k) \\ \widetilde{u}(k+1\,|\,k)) \\ \vdots \\ \widetilde{u}(k+h-1\,|\,k) \end{bmatrix}. \tag{24}$$

Then, the original optimal problem (23) can be rewritten as

$$\Phi(k) = \widehat{x}^T(k+1)\widehat{Q}\widehat{x}(k+1) + \widehat{u}^T(k)\widehat{R}\widehat{u}(k), \tag{25}$$

where

$$\widehat{Q} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{Q} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{Q} \end{bmatrix}, \qquad \widehat{R}(k) = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{R} \end{bmatrix}. \tag{26}$$

Therefore, (20) can be recasted as

$$\widehat{x}(k+1) = \widehat{A}(k)\widehat{x}(k) + \widehat{B}(k)\widehat{u}(k), \tag{27}$$

with

$$\widehat{A}(k) = \begin{bmatrix} \mathbf{A}(k\,|\,k) \\ \mathbf{A}(k\,|\,k)\mathbf{A}(k+1\,|\,k) \\ \vdots \\ \boldsymbol{\alpha}(k,0) \end{bmatrix}, \qquad \widehat{\mathbf{B}} = \begin{bmatrix} \mathbf{B}(k\,|\,k) & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{A}(k+1\,|\,k)\mathbf{B}(k\,|\,k) & \mathbf{B}(k+1\,|\,k) & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{\alpha}(k,1)\mathbf{B}(k\,|\,k) & \boldsymbol{\alpha}(k,2)(\mathbf{B}k+1\,|\,k) & \cdots & \mathbf{B}(k+h-1\,|\,k) \end{bmatrix}, \tag{28}$$

where

$$\boldsymbol{\alpha}(k,j) = \prod_{i=j}^{h-1} (\mathbf{A}(k+i\,|\,k)). \tag{29}$$

Rearrange (25) and (27), the cost function (23) can be rewritten as

$$\Phi(k) = \frac{1}{2}\widehat{u}^T(k)\mathbf{H}(k)\widehat{u}(k) + \mathbf{f}^T(k)\widehat{u}(k) + \mathbf{d}(k), \tag{30}$$

where

$$\begin{aligned} \mathbf{H}(k) &= 2\left(\widehat{B}^T(k)\widehat{Q}\widehat{B}(k) + \widehat{\mathbf{R}}\right), \\ \mathbf{f}(k) &= 2\widehat{B}^T(k)\widehat{Q}\widehat{A}(k)\widehat{x}(k\,|\,k), \\ \mathbf{d}(k) &= \widetilde{x}^T(k\,|\,k)\widehat{A}^T(k)\widehat{Q}\widehat{A}(k)\widehat{x}(k\,|\,k). \end{aligned} \tag{31}$$

The $\mathbf{H}$ is a positive definite Hessian matrix. It is the quadratic part of the cost function while the vector $\mathbf{f}$ describes the linear part. Note that $\mathbf{f}(k)$ and $\mathbf{d}(k)$ contain $\widetilde{x}(k\,|\,k)$ which is constant vector at time k. Hence, the optimization problem can be stated as to find $\widehat{u}^*(k)$ at each time step k minimizing the cost function (30):

$$\widehat{u}^*(k) = \underset{\widehat{u}}{\operatorname{argmin}}\, \Phi(k). \tag{32}$$

The solution of (32) is a sequence of optimal control; that is $\widehat{u}^*(k) = \{\widetilde{u}^*(k\,|\,k), \widetilde{u}^*(k+1\,|\,k), \ldots, \widetilde{u}^*(k+h-1\,|\,k)\}$. The MPC scheme is implicitly given by the first control action of the sequence of optimal control, $\widetilde{u}^*(k\,|\,k)$.

4.3. Collision-Free Constraints. In real world applications, the robots are inevitably subject to physical limitations. Without consideration of these limitations while computing the optimal control inputs, the robot will suffer from unpredictable mismatch. Hence, the constraints of upper and lower bounds should add to the optimization problem (32). And the control inputs constraint can be written as

$$\mathbf{u}_{\min}(k) \le \mathbf{u}(k) \le \mathbf{u}_{\max}(k). \tag{33}$$

Note that the free variable in the optimization (32) is $\widehat{u}(k)$, so the constraint (33) must be rewritten as

$$\widehat{u}_{\min}(k) \le \widehat{u}(k) \le \widehat{u}_{\max}(k), \tag{34}$$

where

$$\widehat{u}_{\min}(k) = \begin{bmatrix} \mathbf{u}_{\min}(k \mid k) - \mathbf{u}_r(k \mid k) \\ \mathbf{u}_{\min}(k+1 \mid k) - \mathbf{u}_r(k+1 \mid k) \\ \vdots \\ \mathbf{u}_{\min}(k+h-1 \mid k) - \mathbf{u}_r(k+h-1 \mid k) \end{bmatrix},$$
$$\widehat{u}_{\max}(k) = \begin{bmatrix} \mathbf{u}_{\max}(k \mid k) - \mathbf{u}_r(k \mid k) \\ \mathbf{u}_{\max}(k+1 \mid k) - \mathbf{u}_r(k+1 \mid k) \\ \vdots \\ \mathbf{u}_{\max}(k+h-1 \mid k) - \mathbf{u}_r(k+h-1 \mid k) \end{bmatrix}. \tag{35}$$

Furthermore, the collision avoidance requirement will be constructed as the constraint with respect to control inputs. Let us rewrite the safe zone (14) as

$$r_s^i(k+1) = \frac{1}{2} \min_{j \neq i} \left(l_{ij}(k+1) - r_i - r_j \right). \tag{36}$$

It means that the control inputs to robot i in time k must reach the safe zone $r_s^i(k+1)$. To make a linear constraint, the approximate safe zone is constructed as a square which is the maximum inscribed square as illustrated in Figure 6. Hence, the side length of this square b can be denoted as

$$b(k+1) = \frac{1}{\sqrt{2}} r_s^i(k+1). \tag{37}$$

Then, the collision avoidance manoeuvre can be represented as

$$-\frac{1}{2} b(k+1) \leq x(k+1) - x_r(k+1) \leq \frac{1}{2} b(k+1),$$
$$-\frac{1}{2} b(k+1) \leq y(k+1) + y_r(k+1) \leq \frac{1}{2} b(k+1), \tag{38}$$

where $x(k+1)$ and $y(k+1)$ can be obtained from the discrete time form of the kinematic equation (15):

$$x(k+1) = x(k) + T \cos(\phi(k)) v(k),$$
$$y(k+1) = y(k) + T \sin(\phi(k)) v(k). \tag{39}$$

Note that $\phi(k+1)$ is not the consideration of the navigation. Remember that $\mathbf{u} = [v, \omega]^T$, so we have $\widetilde{v}(k) = v(k) - v_r(k)$. Then rearranging (38) to (39) can obtain the collision avoidance constraint in each step time k:

$$b_l(k) \leq \widetilde{v}(k) \leq b_u(k), \tag{40}$$

where

$$b_l(k) = \min\left(\frac{-(1/2)b(k+1) + x_r(k+1) - x(k)}{T \cos(\phi(k))}, \frac{-(1/2)b(k+1) + y_r(k+1) - y(k)}{T \sin(\phi(k))} \right),$$
$$b_u(k) = \min\left(\frac{(1/2)b(k+1) + x_r(k+1) - x(k)}{T \cos(\phi(k))}, \frac{(1/2)b(k+1) + y_r(k+1) - y(k)}{T \sin(\phi(k))} \right). \tag{41}$$

Note that $b_l(k)$ and $b_u(k)$ are constants in each time step. Recalling (34), we can obtain

$$\widetilde{b}_l(k) \leq \widetilde{v}(k) \leq \widetilde{b}_u(k), \tag{42}$$

with

$$\widetilde{b}_l(k) = \min\left(b_l(k), v_{\min}(k) - v_r(k) \right),$$
$$\widetilde{b}_u(k) = \min\left(b_u(k), v_{\max}(k) - v_r(k) \right). \tag{43}$$

Finally, the constraint of free variable $\widehat{u}(k)$ for optimization problem (32) can be constructed by refreshing $\widehat{u}_{\min}(k)$ and $\widehat{u}_{\max}(k)$ in (34) as

$$\widehat{u}_{\min}(k) = \begin{bmatrix} \mathbf{u}_{\min}(k \mid k) \\ \mathbf{u}_{\min}(k+1 \mid k) - \mathbf{u}_r(k+1 \mid k) \\ \vdots \\ \mathbf{u}_{\min}(k+h-1 \mid k) - \mathbf{u}_r(k+h-1 \mid k) \end{bmatrix},$$
$$\widehat{u}_{\max}(k) = \begin{bmatrix} \mathbf{u}_{\max}(k \mid k) \\ \mathbf{u}_{\max}(k+1 \mid k) - \mathbf{u}_r(k+1 \mid k) \\ \vdots \\ \mathbf{u}_{\max}(k+h-1 \mid k) - \mathbf{u}_r(k+h-1 \mid k) \end{bmatrix}, \tag{44}$$

where

$$\widetilde{u}_{\min}(k \mid k) = \begin{bmatrix} \widetilde{b}_l(k) \\ \omega_{\min} - \omega_r \end{bmatrix},$$
$$\widetilde{u}_{\max}(k \mid k) = \begin{bmatrix} \widetilde{b}_u(k) \\ \omega_{\max} - \omega_r \end{bmatrix}. \tag{45}$$

4.4. Algorithm. The overall approach is summarized in Algorithm 1. In each ORCA step (τ), it has three things for every robot to do. First, robot i generates the discrete reference control inputs and trajectory. The number of waypoints is τ/T, and the corresponding reference control inputs are $[\|\mathbf{v}_i^{\text{new}}\|, 0]$. Second, robot i computes the safe zone $\mathbf{r}_s^i$ with respect to the neighbor region. Finally, robot i repeatedly computes the optimal control inputs at each sample time T to close the last waypoint. Then, it will go into the next ORCA step. This process will continue until every robot reaches its goal.

5. Experiments and Analysis

In this section, we conduct computer simulations to verify the proposed Discrete-ORCA-MPC algorithm. The following simulations are specifically designed to deal with multi-nonholonomic-agent navigation. These were carried

Input: R_s: list of robots; O: list of obstacles; τ: time horizon of ORCA; h: predict horizon of MPC; T: sampling period.
Output: optimal control $\mathbf{u}^*(k)$

repeat ORCA
  for each $R_i \in R_s$ **do**
    compute $\mathbf{v}_i^{\text{new}}$ for R_i using ORCA (6)
    compute reference trajectory $\mathbf{X}_i^r$ using (22);
    compute safe zone $\mathbf{r}_s^j$ for robot i at each way point using (14)
  end for
  for $k = 1; k < N_\tau; k++$ **do**
    for each $R_i \in R_s$ **do**
      compute $\widetilde{u}_i^*(k)$ for robot i using MPC (27)
      compute control input $\mathbf{u}_i(k) = \widetilde{u}_i^*(k) + \mathbf{u}_i^r(k)$;
      compute the configuration of next step $\mathbf{x}_i(k+1)$ using (15)
    end for
  end for
until (all robots reach their goal zone)

Algorithm 1: Discrete-ORCA-MPC method for multi-nonholonomic robots collision avoidance.

out using MATLAB R2018a and Visual Studio 2017 on a laptop with Intel Quad Core i5 CPU and 16 GB RAM.

To simplify the configuration, it is assumed that all the DDMR have the same parameters. The radius of robots $r = 2$ and the control input constraints are set as $\mathbf{u}_{\min} = [0, -1]$ and $\mathbf{u}_{\max} = [2, 1]$, respectively. The preferred velocity $\mathbf{v}^{\text{pref}}$ was chosen as the velocity directed towards the robot's goal with a magnitude equal to 1. Meanwhile, the sampling time period of MPC is $T = 30$ ms and the time window of the ORCA algorithm is $\tau = 300$ ms. In the MPC controller formulation, the prediction horizon $h = 6$ and optimization problem is solved using interior point method. Furthermore, the free variable constraints are set as $\widetilde{u}_{\min} = [-0.5, -0.5]$ and $\widetilde{u}_{\max} = [0.5, 0.5]$, respectively. To keep the robot close to the ORCA trajectory, the weighting matrix $\mathbf{Q}$ should be larger than $\mathbf{R}$. In this paper, these are set as $\mathbf{Q} = 3\mathbf{I}_3$ and $\mathbf{R} = 0.1\mathbf{I}_3$.

5.1. Compare MPC and PID for 4 DDMRs. As mentioned in Section 2.2, the original ORCA algorithm is a first-order method. In other words, the control input to the robot is reference velocity but not the reference trajectory. In this simulation, a first-order method based on ORCA is formulated to compare with the proposed Discrete-ORCA-MPC approach. This method uses a proportional controller to control the angular velocity, and the linear velocity is equal to the new velocity generated by ORCA. It can be formulated as

$$v_{ik}(j) = v_{ik}^{\text{new}},$$
$$\omega_{ik}(j) = \begin{cases} \min\left(\mathrm{K}_{\text{gain}}\left(\phi_{ik}^{\text{new}} - \phi_{ik}(j)\right), \omega_{\max}\right), & \phi_{ik}^{\text{new}} \geq \phi_{ik}(j), \\ \max\left(\mathrm{K}_{\text{gain}}\left(\phi_{ik}^{\text{new}} - \phi_{ik}(j)\right), \omega_{\min}\right), & \phi_{ik}^{\text{new}} < \phi_{ik}(j), \end{cases} \tag{46}$$

where v_{ik}^{new} and ϕ_{ik}^{new} represent the 2-norm value and direction of the new velocity (equation (6)) of robot i generated by ORCA in step k; that is, $v_{ik}^{\text{new}} = \|\mathbf{v}_{ik}^{\text{new}}\|$ and $\phi_{ik}^{\text{new}} = \text{atan2}(\mathbf{v}_{ik}^{\text{new}})$. $v_{ik}(j)$ and $\omega_{ik}(j)$ are the control input to robot i in control step j of ORCA step k. The proportional controller gain is set as 3. As a consequence, the constraints of angular velocity are also added to the controller. Note that we have no intention of constructing a perfect controller to following the reference velocity $\mathbf{v}^{\text{new}}$, since the tracking error to the holonomic trajectory always exists even for a "perfect" controller (see Figure 3). Furthermore, these tracking errors may lead to a collision. By contrast, the proposed MPC controller also has tracking errors; however, these tracking errors are formulated as constraints for the decision variables that the collision will not happen (see Theorem 2 and equation (45)).

In this simulation, 4 DDMRs are initially located on the four corners of a square, that is, [−25, −25] (Robot 1), [25, 25] (Robot 2), [−25, 25] (Robot 3), and [25, −25] (Robot 4), as illustrated in Figure 8. Robots 1, 2 and Robots 3, 4 at antipodal positions are expected to exchange positions with each other. Furthermore, the initial orientation of robots 1 to 4 is $(1/4)\pi$, $-(3/4)\pi$, $-(1/4)\pi$, and $(3/4)\pi$, respectively. Note that every 18 ORCA steps plot a circle as shown in Figure 8.

Figure 8 shows the motion trajectory of 4 DDMRs on the workspace without static obstacles based on the proposed Discrete-ORCA-MPC approach and a proportional controller. Comparing the trajectories of local enlarged drawing of Figures 8(a) and 8(b), at 37.5°s, Robot 2 $(-1.718, 1.56)$ collides with Robot 3 $(-1.421, -2.082)$ in Figure 8(b) (PID) while the collision did not happen in Figure 8(b) (Discrete-ORCA-MPC). Figure 9 depicts the relative distance between robots over the time where d_{ij} represents the relative distance between robots i and j. These results demonstrate the proposed method can generate a collision-free navigation for nonholonomic mobile robot effectively.

Figure 10 shows the control inputs (v and ω) to each robot. In most of the time of the navigation, the control inputs change a little with respect to time; however, these has been a great change near 28°s and 37°s. It can be explained that the robots took a sharp turn in 28°s and 37°s

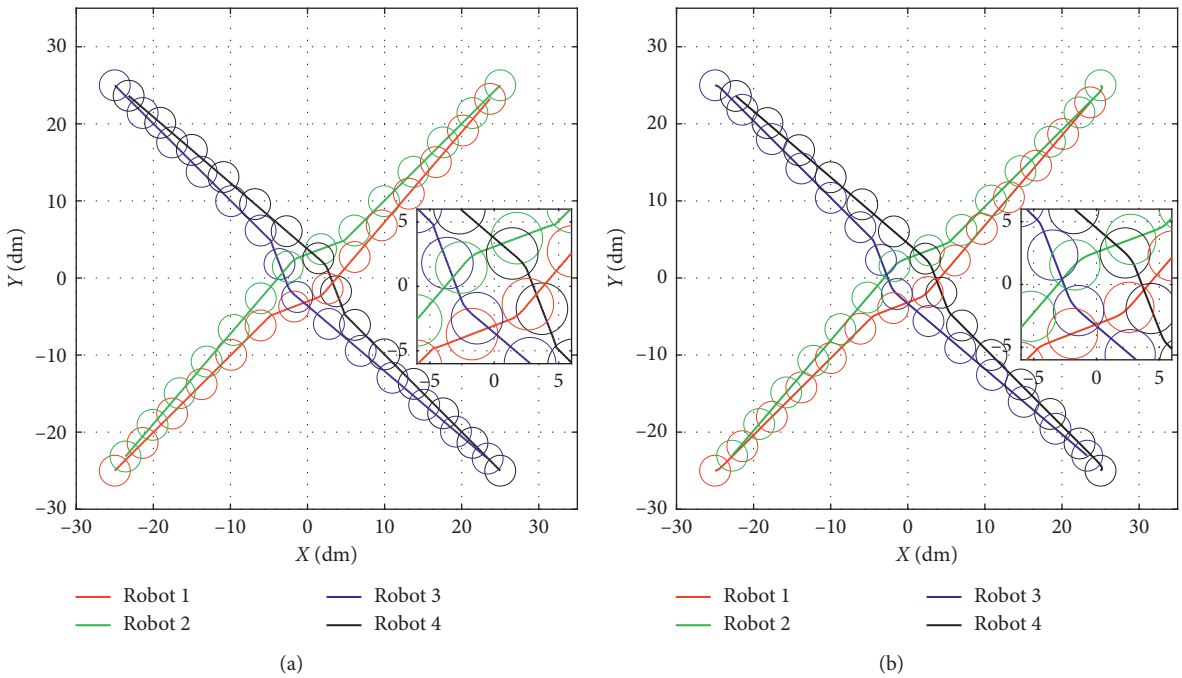

Figure 8: Motion trajectory for (a) Discrete-ORCA-MPC and (b) PID.

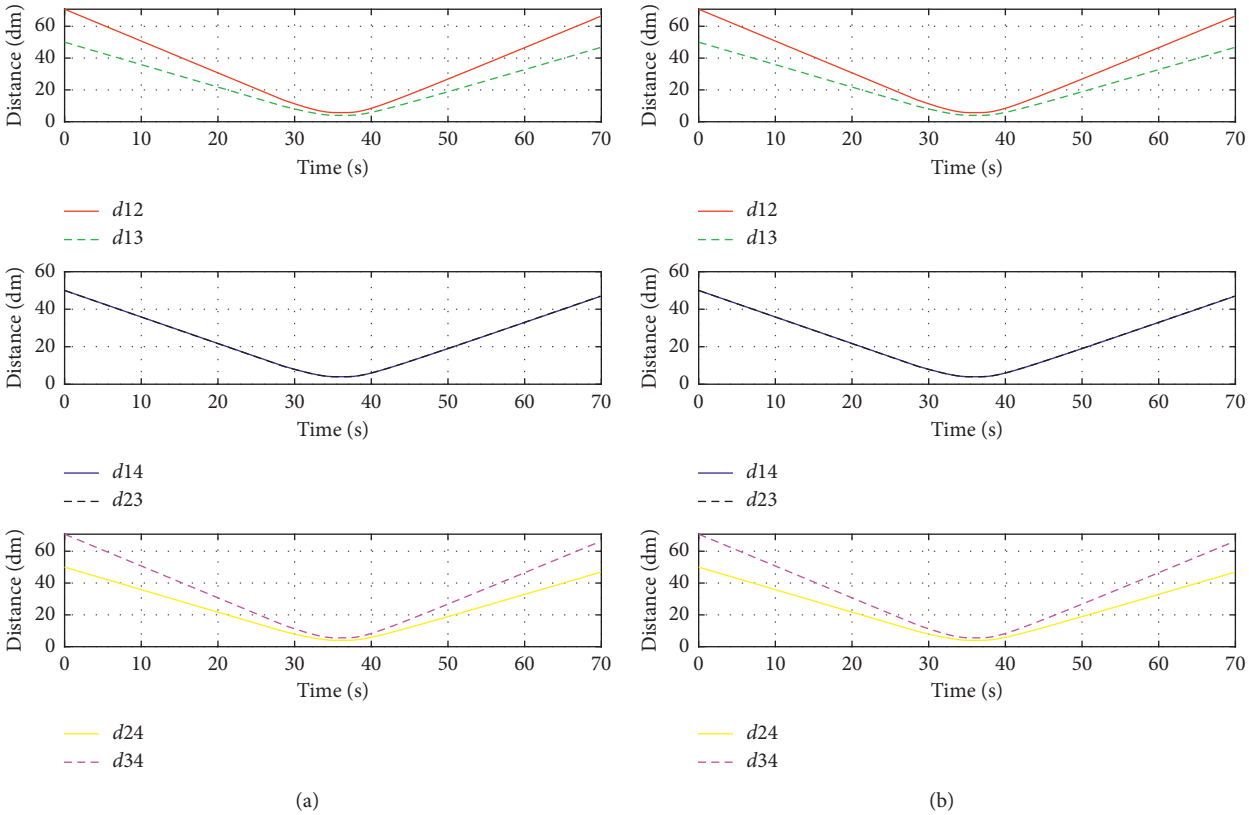

Figure 9: The relative distance between robots. (a) Discrete-ORCA-MPC. (b) PID.

as shown in Figure 8. In particular, the rapid growth of angular velocity ω is inevitably accompanied by a decline in the linear velocity ω. In other words, the robot should slow down for taking a sharp turn. It also can be seen that the control inputs are inside the limits imposed by the constraints. In Figure 11, it demonstrates that the state errors

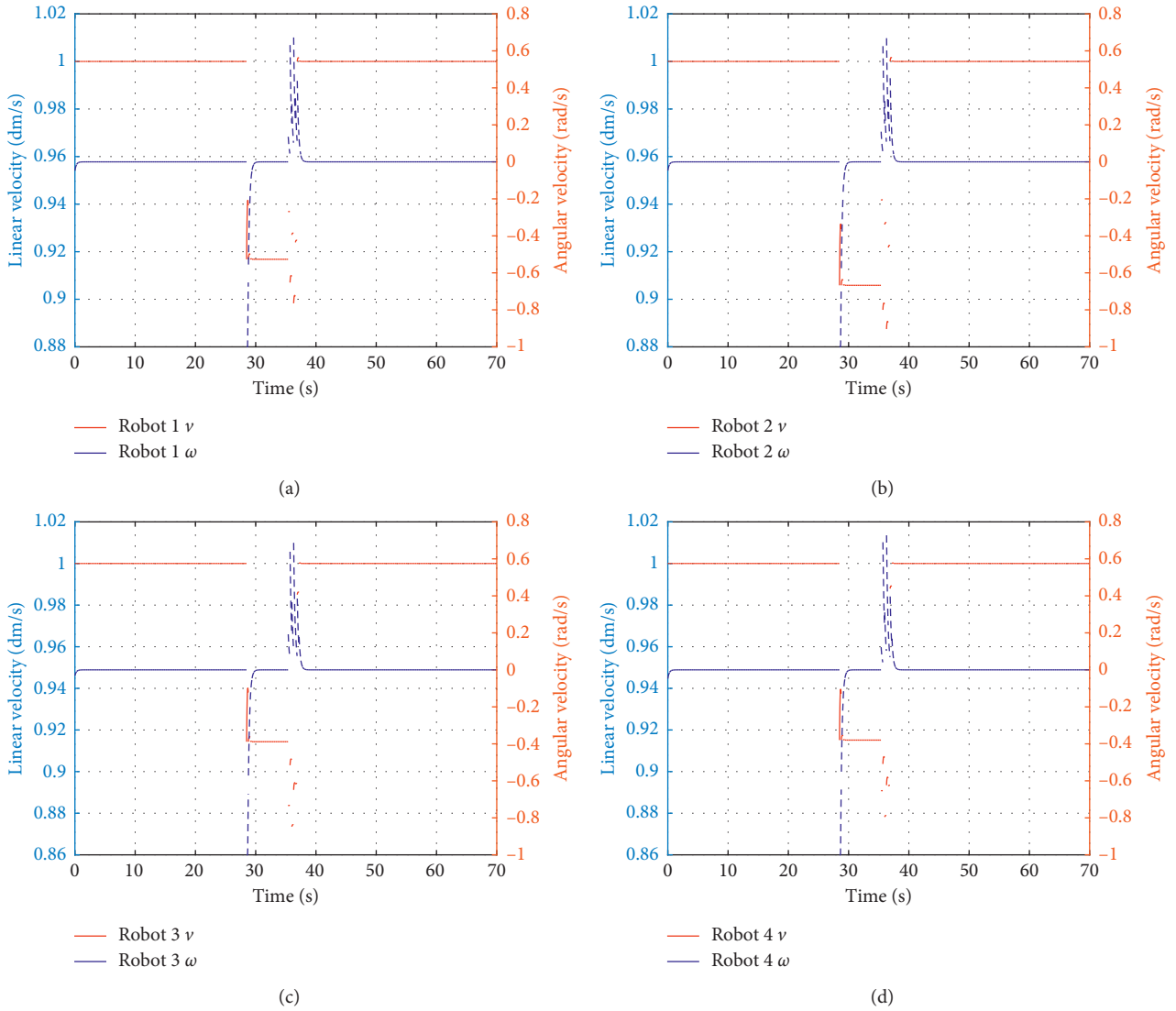

Figure 10: Control inputs of the 4 robots with respect to time. (a) Robot 1 control inputs. (b) Robot 2 control inputs. (c) Robot 3 control inputs. (d) Robot 1 control inputs.

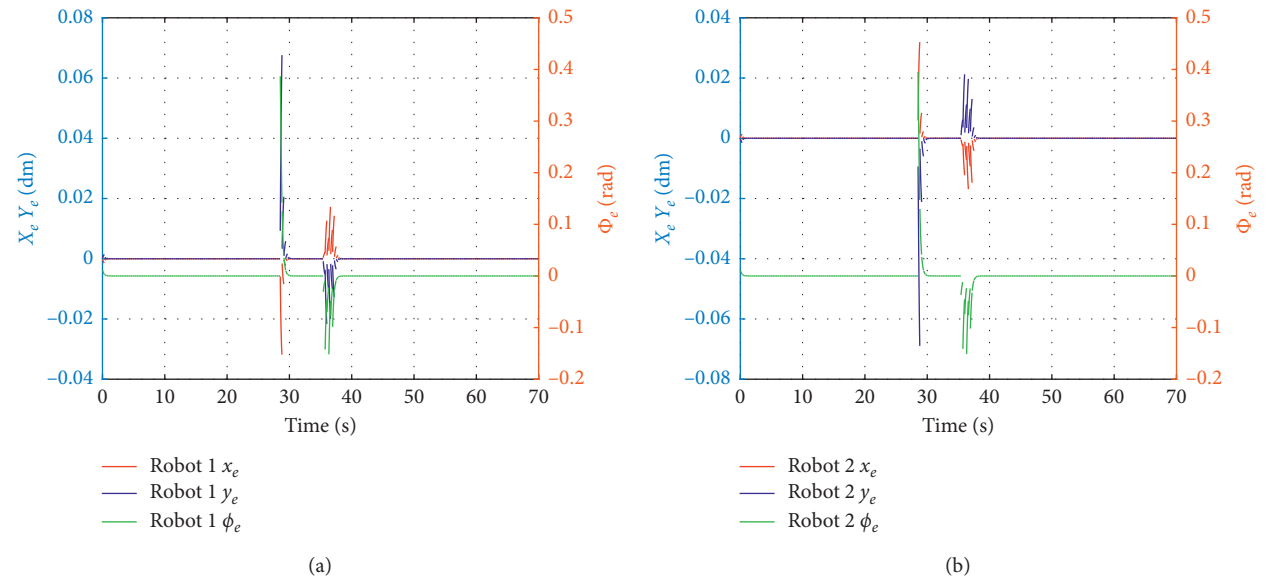

Figure 11: Continued.

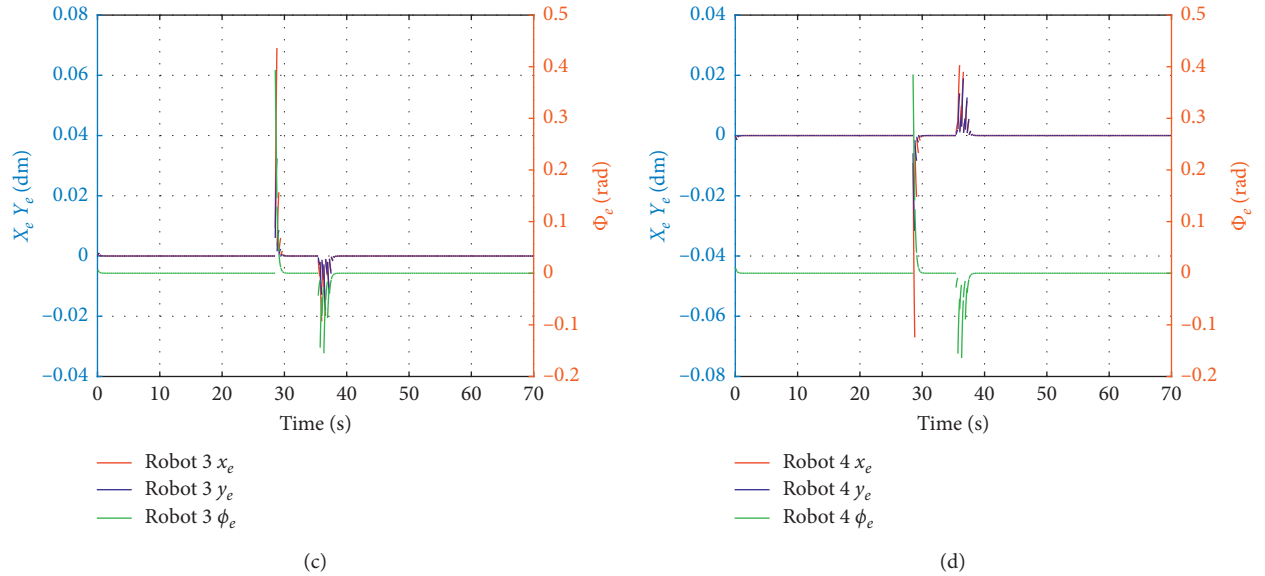

Figure 11: Robots state errors compared to the holonomic trajectory. (a) Robot 1 state inputs. (b) Robot 2 state inputs. (c) Robot 3 state inputs. (d) Robot 4 state inputs.

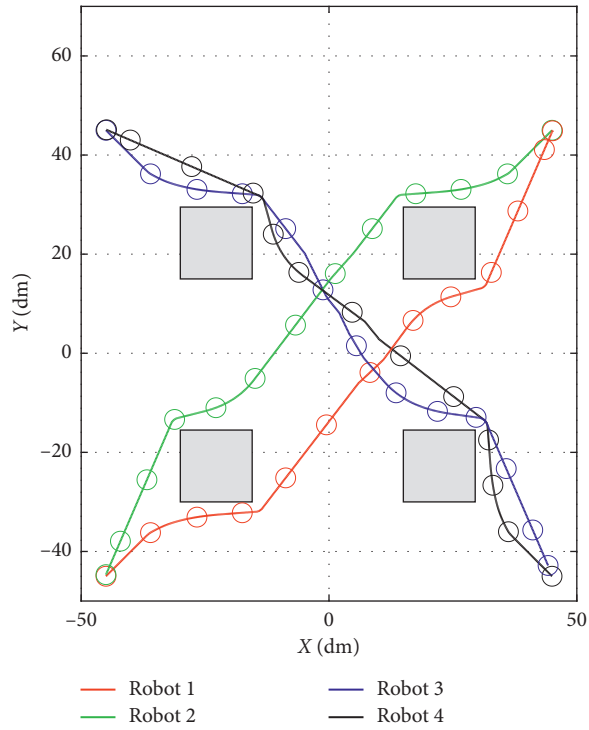

Figure 12: Motion trajectory of 4 DDMRs with static obstacles.

are compared with the trajectory generated by ORCA. It can be seen that maximum absolute errors of the position $[x_e, y_e]$ and orientation ϕ_e are 0.07 and 0.4, respectively. Obviously, position errors are much less than the orientation error ϕ_e. This is reasonable because the smaller $[x_e, y_e]$ shows a better performance of MPC controller. This simulation demonstrates that the proposed Discrete-ORCA-MPC algorithm has the ability to cope with the collision avoidance problem of multi-nonholonomic-agent.

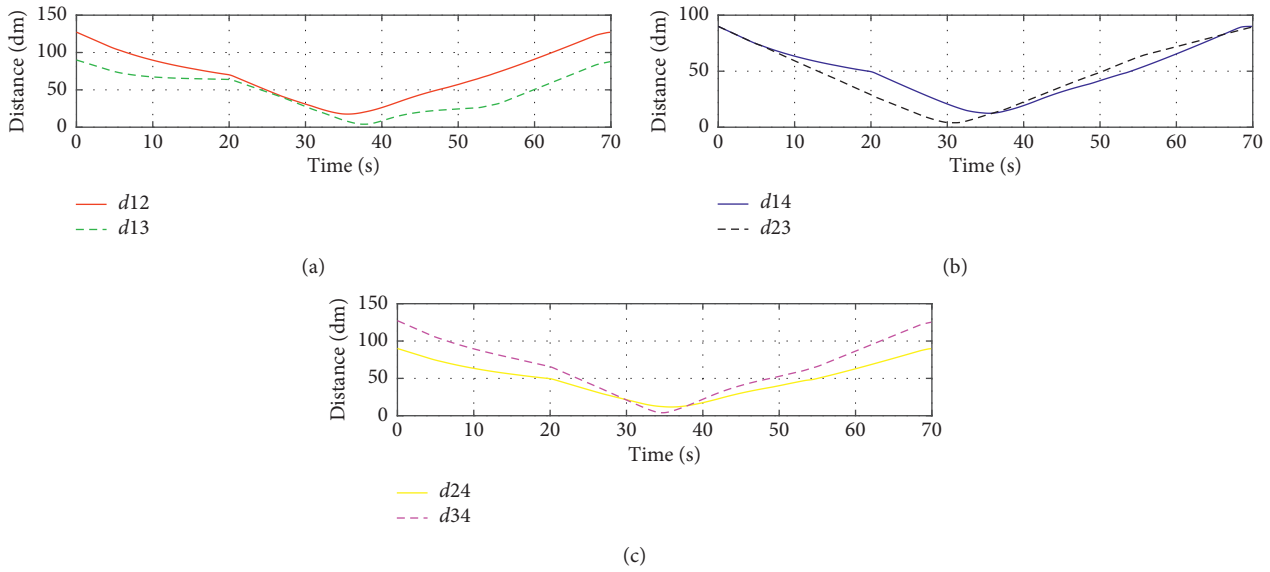

Figure 13: The relative distance between robots.

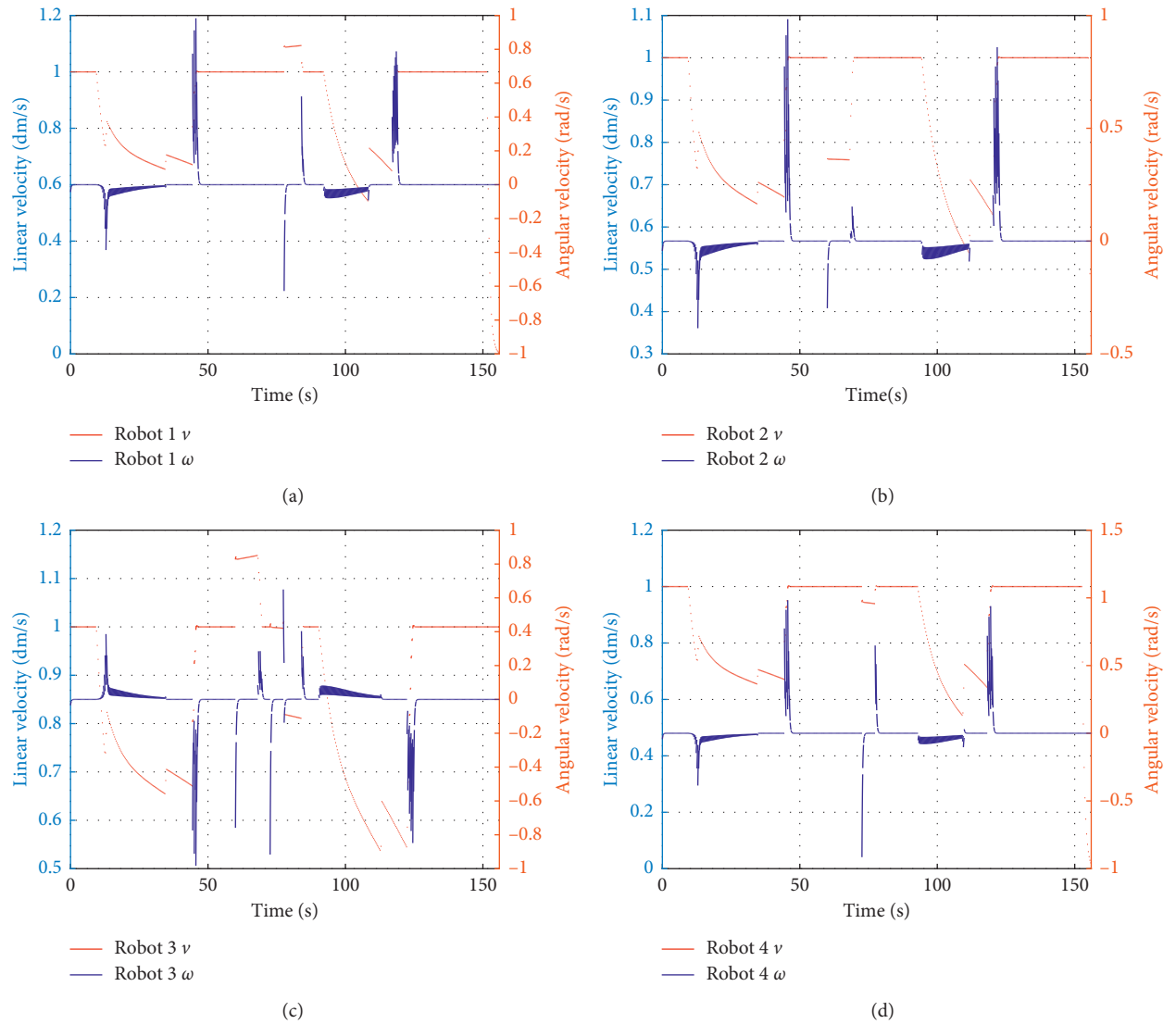

Figure 14: Control inputs of the 4 robots with respect to time (consider static obstacles). (a) Robot 1 control inputs. (b) Robot 2 control inputs. (c) Robot 3 control inputs. (d) Robot 4 control inputs.

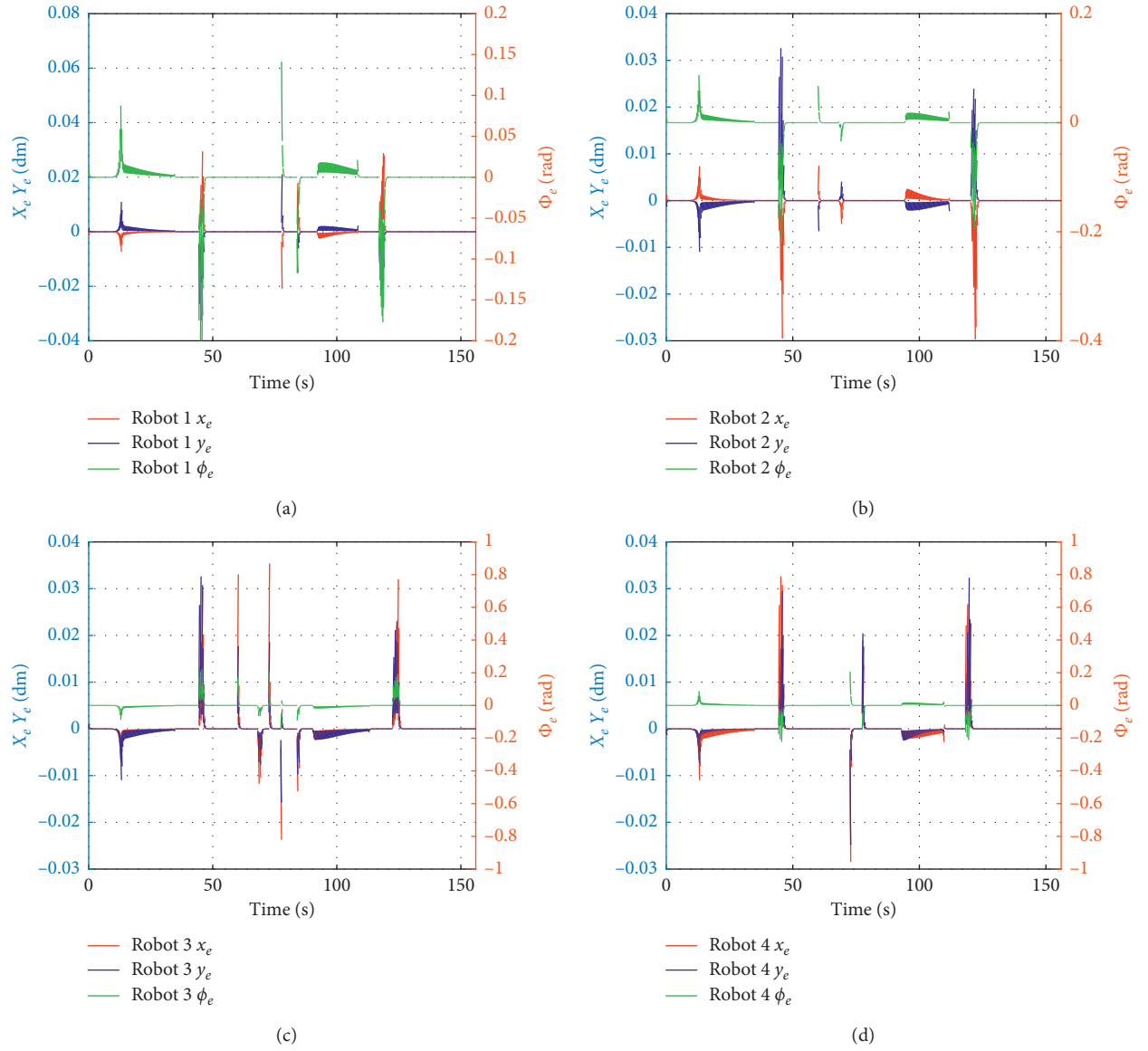

Figure 15: Robots state errors compared to the holonomic trajectory generated by ORCA (consider static obstacles). (a) Robot 1 state errors. (b) Robot 2 state errors. (c) Robot 3 state errors. (d) Robot 1 state errors.

5.2. 4 DDMRs with Static Obstacle. It is similar to the first simulation; 4 DDMRs are initially located on the four corners of a square also, that is, [−45,−45] (Robot 1), [45, 45] (Robot 2), [−45, 45] (Robot 3) and [45, −45] (Robot 4), as illustrated in Figure 12. Unlike the first one, this simulation considers static obstacle avoidance. These four static obstacles are located at [−22.5, −22.5], [22.5, 22.5], [−22.5, 22.5], and [22.5, −22.5] with sides that are 15 dm in length. Figure 13 depicts the relative distance between robots over the time. Furthermore, the initial orientations of Robots 1 to 4 are $(1/4)\pi + 0.1$, $-(3/4)\pi + 0.1$, $-(1/4)\pi + 0.1$, and $(3/4)\pi + 0.1$, respectively. This is why the trajectories of these four robots are asymmetrical. More sharp turn can be seen compared to the first simulation which leads to more sudden change in control inputs as shown in Figure 14. Meanwhile, the constraints of control inputs $[\mathbf{u}_{\min}, \mathbf{u}_{\max}]$ and the errors with respect to reference control inputs $[\widehat{u}_{\min}, \widehat{u}_{\max}]$ are satisfied. In Figure 15, it can also be seen that the position errors $[x_e, y_e]$ are much less than the orientation error ϕ_e. The maximum value of position errors is less than 0.1 which means that the performance of the obtained trajectory tracking is good enough.

This simulation demonstrates that the proposed Discrete-ORCA-MPC algorithm has the ability to cope with the navigation problem of multi-nonholonomic-agent with static obstacles.

5.3. Performance Results. In order to test the performance of Discrete-ORCA-MPC, we carried on large-scale simulations to compare the mean computation times of our method with Priority-Based Noncooperative Distributed Model Predictive Control (PB-NC-DMPC) [44]. PB-NC-DMPC is an effective NC-DMPC strategy that deals with the problem of

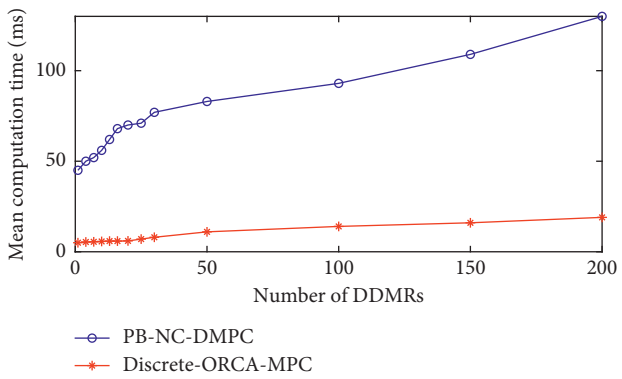

Figure 16: Comparison of computation times.

loss of the prediction consistency property while reducing the computation time as well. In the problem of collision avoidance of networked vehicles, PB-NC-DMPC showed better performance compared to NC-DMPC and Centralized MPC. The test scenario contains many robots whose goal is to move across a circle to the antipodal position. This scenario is the same as the published literature, Jur et al. [31]. In PB-NC-DMPC, we implemented the collision avoidance controller based on the same linear predictive model as (20). The controller parameters of PB-NC-DMPC are the same as Discrete-ORCA-MPC and remain unchanged for all robots in all simulations.

Figure 16 depicts the mean computation times of Discrete-ORCA-MPC and PB-NC-DMPC plotted over increasing number of DDMRs. It shows that running Discrete-ORCA-MPC is computationally cheaper than PB-NC-DMPC.

6. Conclusions

In this paper, we have described a method, namely, Discrete-ORCA-MPC, for collision avoidance among multiple nonholonomic robots in planar environments. In particular, we extended the traditional ORCA method to a Discrete-ORCA strategy which can generate a uniform distribution of safe zones in the direction of selected new velocity by ORCA. The idea was to introduce the collision-free constraints to MPC optimization problem. The presented Discrete-ORCA-MPC method allows for smooth and safe navigation and good performance was observed in simulation experimental tests.

The results of this paper show that our approach is capable of letting a homogeneous group of robots with nonlinear equations of motion safely avoid collisions at real-time computation rates. The collision-free navigation problem of nonhomogeneous group of robots will be solved through adjusting the safe zone. Many challenges for future development still remain. The future work should aim at further improving the adaptability in uncertain environment.

Data Availability

The data used to support the finds of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was carried out with the support of the Sichuan Science and Technology Program (no. 20ZDYF1891).

References

[1] J. Alonso-Mora, P. Beardsley, and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 404–420, 2018.

[2] V. Kunchev, L. Jain, V. Ivancevic, and A. Finn, "Path planning and obstacle avoidance for autonomous mobile robots: a review," in *Proceedings of the International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, Springer, Bournemouth, UK, pp. 537–544, October 2006.

[3] A. Mukhtar, L. Xia, and T. B. Tang, "Vehicle detection techniques for collision avoidance systems: a review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2318–2338, 2015.

[4] S. Campbell, W. Naeem, and G. W. Irwin, "A review on improving the autonomy of unmanned surface vehicles through intelligent collision avoidance manoeuvres," *Annual Reviews in Control*, vol. 36, no. 2, pp. 267–283, 2012.

[5] S. S. Ge and Y. J. Cui, "Dynamic motion planning for mobile robots using potential field method," *Autonomous Robots*, vol. 13, no. 3, pp. 207–222, 2002.

[6] J. P. van den Berg and M. H. Overmars, "Roadmap-based motion planning in dynamic environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 885–897, 2005.

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[8] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[9] J. P. Gonzalez and M. Likhachev, "Search-based planning with provable suboptimality bounds for continuous state spaces," in *Proceedings of the Fourth Annual Symposium on Combinatorial Search*, Barcelona, Spain, July 2011.

[10] J. Wang, S. Wu, H. Li, and J. Zou, "Path planning combining improved rapidly-exploring random trees with dynamic window approach in ROS," in *Proceedings of the 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, Wuhan, China, May 2018.

[11] X. Zhong, X. Zhong, and X. Peng, "Velocity-change-space-based dynamic motion planning for mobile robots navigation," *Neurocomputing*, vol. 143, no. SI, pp. 153–163, 2014.

[12] J. Borenstein and Y. Koren, "The vector field histogram—fast obstacle avoidance for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 7, pp. 278–288, 1991.

[13] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[14] F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond, "A local collision avoidance method for non-strictly convex polyhedra," *Proceedings of Robotics: Science and Systems*, vol. 4, 2008.

[15] H. A. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 4, pp. 524–539, 2004.

[16] G. Antonelli, F. Arrichiello, and S. Chiaverini, "Experiments of formation control with multirobot systems using the null-space-based behavioral control," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1173–1182, 2009.

[17] C. G. Zhang and Y. G. Xi, "Rolling path planning and safety analysis of mobile robot in dynamic uncertain environment," *Control Theory & Applications*, vol. 20, pp. 37–44, 2003.

[18] M. G. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics & Autonomous Systems*, vol. 100, pp. 171–185, 2017.

[19] Y. Dobrev, M. Vossiek, M. Christmann, I. Bilous, and P. Gulden, "Steady delivery: wireless local positioning systems for tracking and autonomous navigation of transport vehicles and mobile robots," *IEEE Microwave Magazine*, vol. 18, no. 6, pp. 26–37, 2017.

[20] D. Thalmann, "Crowd simulation," in *Encyclopedia of Computer Graphics and Games*, pp. 1–8, Springer, Berlin, Germany, 2016.

[21] Z. Peng, S. Yang, G. Wen, A. Rahmani, and Y. Yu, "Adaptive distributed formation control for multiple nonholonomic wheeled mobile robots," *Neurocomputing*, vol. 173, pp. 1485–1494, 2016.

[22] S. Liu, D. Sun, and C. Zhu, "A dynamic priority based path planning for cooperation of multiple mobile robots in formation forming," *Robotics and Computer-Integrated Manufacturing*, vol. 30, no. 6, pp. 589–596, 2014.

[23] Y. Zhang, Y. Wang, and Y. Yao, "Distributed adaptive sliding mode control for attitudes synchronization of multiple autonomous underwater vehicles," *Advances in Mechanical Engineering*, vol. 9, no. 11, Article ID 1687814017737262, 2017.

[24] J. V. den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pp. 1928–1935, IEEE, Pasadena, CA, USA, May 2008.

[25] A. Prasad, B. Sharma, and J. Vanualailai, "A new stabilizing solution for motion planning and control of multiple robots," *Robotica*, vol. 34, no. 5, pp. 1071–1089, 2016.

[26] 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.

[27] W. Ren, W. Randal, and Beard, *Distributed Consensus in Multi-Vechicle Cooperative Control*, Springer Science & Business Media, Berlin, Germany, 2008.

[28] G. Lou, W. Gu, Y. Xu, M. Cheng, and W. Liu, "Distributed MPC-based secondary voltage control scheme for autonomous droop-controlled microgrids," *IEEE Transactions on Sustainable Energy*, vol. 8, no. 2, pp. 792–804, 2017.

[29] R. Van Parys and G. Pipeleers, "Distributed mpc for multi-vehicle systems moving in formation," *Robotics and Autonomous Systems*, vol. 97, p. 08, 2017.

[30] S. Yoo and B. Park, "Connectivity preservation and collision avoidance in networked nonholonomic multi-robot formation systems: unified error transformation strategy," *Automatica*, vol. 103, pp. 274–281, 2019.

[31] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, pp. 3–19, Springer, Berlin, Germany, 2011.

[32] J. Alonso-Mora, A. Breitenmoser, M. Rufli, B. Paul, and S. Roland, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed Autonomous Robotic Systems*, pp. 203–216, Springer, Berlin, Germany, 2013.

[33] D. Wilkie, J. Van Den Berg, and D. Manocha, "Generalized velocity obstacles," in *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5573–5578, IEEE, St. Louis, MO, USA, October 2009.

[34] J. D. Jeon and H. B. Lee, "Wheel velocity obstacles for differential drive robot navigation," *Journal of Automation and Control Engineering*, vol. 3, no. 5, 2015.

[35] B. Alrifaee, J. Maczijewski, and D. Abel, "Sequential convex programming mpc for dynamic vehicle collision avoidance," in *Proceedings of the 2017 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 2202–2207, IEEE, Mauna Lani, HI, USA, August 2017.

[36] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Shared steering control using safe envelopes for obstacle avoidance and vehicle stability," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 441–451, 2016.

[37] O. Andersson, M. Wzorek, P. Rudol, and P. Doherty, "Model-predictive control with stochastic collision avoidance using bayesian policy optimization," in *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4597–4604, IEEE, Stockholm, Sweden, May 2016.

[38] H. Yu, J. Duan, S. Taheri, H. Cheng, and Z. Qi, "A model predictive control approach combined unscented kalman filter vehicle state estimation in intelligent vehicle trajectory tracking," *Advances in Mechanical Engineering*, vol. 7, no. 5, Article ID 1687814015578361, 2015.

[39] F. Kuhne, W. F. Lages, and J. G. da Silva Jr., "Model predictive control of a mobile robot using linearization," in *Proceedings of Mechatronics and Robotics*, Citeseer, New York, NY, USA, pp. 525–530, 2004.

[40] G. Klančar and I. Škrjanc, "Tracking-error model-based predictive control for mobile robots in real time," *Robotics and Autonomous Systems*, vol. 55, no. 6, pp. 460–469, 2007.

[41] M. Rufli, J. Alonso-Mora, and R. Siegwart, "Reciprocal collision avoidance with motion continuity constraints," *IEEE Transactions on Robotics*, vol. 29, no. 4, pp. 899–912, 2013.

[42] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, Cambridge, UK, 2006.

[43] M. Jan, *Predictive Control with Constraints*, Prentice Hall, Upper Saddle River, NJ, USA, 2002.

[44] B. Alrifaee, F.-J. Heßeler, and D. Abel, "Coordinated non-cooperative distributed model predictive control for decoupled systems using graphs," *IFAC-PapersOnLine*, vol. 49, no. 22, pp. 216–221, 2016.