2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Decentralized Navigation of Multiple Agents Based on ORCA and Model Predictive Control*

Hui Cheng[1], Qiyuan Zhu[1], Zhongchang Liu[1], Tianye Xu[1], and Liang Lin[1,2]

Abstract— **This paper presents a decentralized strategy for collision-free navigation of multiple agents. This strategy combines the Optimal Reciprocal Collision Avoidance (ORCA) algorithm and Model Predictive Control (MPC). Concretely, each agent applies the decentralized ORCA algorithm to compute the collision-avoiding velocities with respect to its neighbors. The derived velocities serve as constraints of a MPC problem whose solution provides the optimal control input that can ensure optimal motion of the agent. The states predicted from the agents' dynamic models are used in the ORCA algorithm to compute the ORCA velocity regions in future steps. This ORCA-MPC combined approach doesn't need a priori the preferred velocity of each agent in comparison to the traditional ORCA algorithm and its existing variants. Simulation results illustrate the effectiveness of the proposed method, and show that this new algorithm can reduce velocity vibrations in the traditional ORCA algorithm.**

I. INTRODUCTION

Navigation for a multi-agent system aims to generate collision-free trajectories for all agents so that they can move from their initial positions to target positions safely. This problem usually needs to handle constraints such as obstacles [1] and model dynamics of the agents [2], [3]. The study of these problems can be applied not only for guiding motions of robots, but also can be used for localization [4] and simulation of human behaviors in the virtual world [5].

Multi-agent navigation algorithms can be divided into centralized algorithms and decentralized algorithms. Centralized algorithms regard all agents as a single system, and a global planner collects information from all agents and computes their trajectories together [6], [7]. This method scales poorly and it has to compute in very high dimensional spaces as the number of agents increases. Decentralized planners tend to decompose a group navigation problem into a set of correlated individual agent navigation problems, where each agent computes a collision-free trajectory independently. Inter-agent correlations may be built on either communication systems or sensing devices. Some commonly used distributed approaches include probabilistic road-maps [8], dynamic window [9] or reciprocal velocity obstacle (RVO) [10] based methods.

The Optimal Reciprocal Collision Avoidance (ORCA) [11] algorithm, reformulated from the RVO algorithm in [10], is an effective decentralized navigation method for a large number of agents. It assumes that all agents are willing to cooperative with each other in the way that each agent takes half the responsibility of avoiding pair-wise collision. Under this reciprocal criterion, each agent first generates a set of permitted velocities with respect to all its neighbors, from which the velocity that is as close to the predefined preferred velocity as possible is chosen as the new velocity for the next motion. The permitted velocities form a linear set which enables efficient computation of the optimal new velocity by solving a low-dimensional linear programming problem.

The above procedure of the ORCA algorithm has two main issues to be taken care of. The first one is to determine the preferred velocity of every single agent. One simple way is to use a vector pointing towards the direction of its target position [12] while more complicated methods use global motion planning algorithms [13], [14]. The other issue is the assumption of instantaneous jump of the agents' velocities, which may require unlimited control forces in practical applications. To fix this problem, one promising approach is to take the agents' dynamic models into consideration when generating the velocity constraints. For example, the paper [12] extended RVO to acceleration-velocity obstacles (AVO) to compute velocity constraints. For agents with linear differential constraints, the paper [14] proposed LQR-Obstacles, while agents with non-holonomic dynamics are investigated in [2]. However these algorithms would render the velocity solution space non-convex, and the optimal solution has to be derived from the convex subset of the solution space.

The model predictive control (MPC) method can predict future trajectories of a system subject to the system dynamics and various constraints. So, it is an attractive approach to controlling agent motions as well as avoiding obstacles. The usage of MPC for navigation of a single agent has been reported in [15]–[17] recently. Motivated by these results, this paper proposes an ORCA-MPC combined approach for multi-agent navigation. Concretely, the ORCA algorithm is used by each agent to generate independently its set of permitted velocities. These velocities serve as constraints in a MPC problem whose solution can provide the optimal control input for an agent by minimizing a cost function that penalizes its control cost and its distance to the target position. Since the cost function is irrelevant to any agent's preferred velocity, this ORCA-MPC algorithm doesn't need an additional procedure to compute the preferred velocity. Moreover, each agent's trajectory and velocity are optimized and smoothed due to the incorporation of the agents' dynam-

*This work is supported by the Fundamental Research Funds for the Central Universities (17lgpy119,15lgjc28), Guangdong Natural Science Foundation (1614050001452), NSFC-Shenzhen Robotics Projects (U1613211), National Natural Science Foundation of China (61622214) and the National Key R&D Program of China (2016YFB1001004).

[1] School of Data and Computer Science, Sun Yat-sen University, Guangzhou, P. R. China. [2] SenseTime Group (Limited), Shenzhen, P. R. China. Correspondence to: Zhongchang Liu (Email: zcliu@foxmail.com).

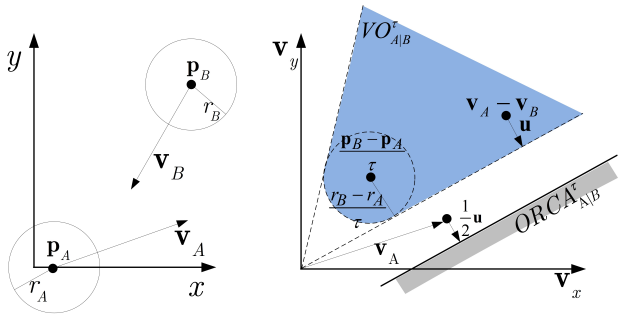


Fig. 1. Left: position configurations of two agents. Right: $VO^{\tau}_{A|B}$ (light blue shadow), and $ORCA^{\tau}_{A|B}$ (grey shadow) for agent A when $(\mathbf{v}_A - \mathbf{v}_B)$ lies in $VO^{\tau}_{A|B}$.

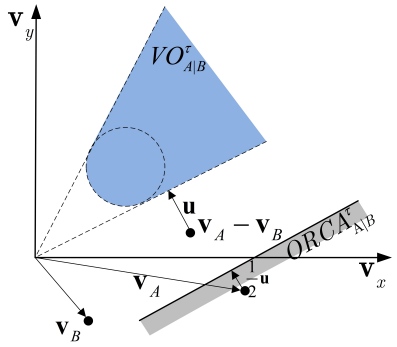


Fig. 2. $ORCA^{\tau}_{A|B}$ when $(\mathbf{v}_A - \mathbf{v}_B)$ is out of $VO^{\tau}_{A|B}$.

ic models. Another advantage of this new approach is that the permitted velocity set generated by the ORCA algorithm is still convex regardless of the dynamic models of the agents.

The rest of this paper is organized as follows. In section II, the ORCA algorithm is reviewed. The ORCA-MPC algorithm is designed in Section III. Section IV presents some simulation examples and Section V concludes this paper.

II. INTRODUCTION OF THE ORCA ALGORITHM

The optimal reciprocal collision avoidance (ORCA) algorithm is a distributed collision avoidance algorithm that allows each agent to compute independently the optimal moving velocities in each step [11]. Briefly speaking, this algorithm assumes that each agent can obtain the relative distance and velocity with respect to every neighboring agent. Based on these information, the agent computes the set of velocities, namely the velocity obstacles, that will cause collision with any agent in a finite time horizon. Then, a velocity, which is outside of the velocity obstacle and is the closest to the agent's preferred velocity, is chosen as the new velocity in the next step of movement. The details of this ORCA algorithm can be found in [11], and the key definitions that are useful in this paper are presented in the following.

An open ball of radius r centered at $\mathbf{p}$ is defined as

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

The *velocity obstacle* $VO^{\tau}_{A|B}$ of agent A with respect to agent B in the time horizon τ is defined as

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

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. The illustration of these definitions in 2-D space can be found on the left-hand side of Fig. 1. Clearly, $VO^{\tau}_{A|B}$ represents the set of relative velocities that the two agents will collide with each other within time τ. Note that for agent B, its velocity obstacle is $VO^{\tau}_{B|A} = -VO^{\tau}_{A|B}$.

Let $\mathbf{v}_A$ and $\mathbf{v}_B$ be the two agents' current velocities, respectively. Denote by $\mathbf{u}$ the vector with minimum length that points from $(\mathbf{v}_A - \mathbf{v}_B)$ to the boundary of $VO^{\tau}_{A|B}$:

$$\begin{aligned} \mathbf{u} = &(\arg\min_{\mathbf{v} \in \partial VO^{\tau}_{A|B}} \|\mathbf{v} - (\mathbf{v}_A - \mathbf{v}_B)\|) \\ &- (\mathbf{v}_A - \mathbf{v}_B). \end{aligned} \tag{3}$$

In other words, $\mathbf{u}$ is the smallest velocity change required so that the relevant velocity $\mathbf{v}_A - \mathbf{v}_B$ can "escape" from the velocity obstacle $VO^{\tau}_{A|B}$.

In this paper, the definition of the *optimal reciprocal collision avoidance region, $ORCA^{\tau}_{A|B}$, for agent A with respect to agent B within time τ* consists of two parts. The first part is the same as that in [11] and is defined for the case that $\mathbf{v}_A - \mathbf{v}_B$ belongs to $VO^{\tau}_{A|B}$ as shown in Fig. 1,

$$ORCA^{\tau}_{A|B} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}_A + \frac{1}{2}\mathbf{u}))\mathbf{n} \geq 0\}, \tag{4}$$

where $\mathbf{n}$ is the normal vector of $\mathbf{u}$. The increment $\frac{1}{2}\mathbf{u}$ 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.

The second part is defined for the case that $(\mathbf{v}_A - \mathbf{v}_B)$ is outside of $VO^{\tau}_{A|B}$. Since agents are collision free within time τ, the set $ORCA^{\tau}_{A|B}$ is the whole velocity space, and agent A's new velocity $\mathbf{v}^{new}_A$ will be chosen as the preferred velocity $\mathbf{v}^{pref}_A$ [11]. However, the relative preferred velocity $\mathbf{v}^{pref}_A - \mathbf{v}^{pref}_B$ (and thus the new relative velocity $\mathbf{v}^{new}_A - \mathbf{v}^{new}_B$) may lie in $VO^{\tau}_{A|B}$. If so, the agent has to abandon this new velocity at the next step. This could cause sharp vibrations for the new velocities between two steps. To avoid this adverse situation, we define $ORCA^{\tau}_{A|B}$ as follows:

$$ORCA^{\tau}_{A|B} = \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}^{opt}_A + \frac{1}{2}\mathbf{u}))\mathbf{n} \leq 0\}. \tag{5}$$

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

The set of optimal reciprocal collision avoiding velocities for agent A within time τ is the intersection of the half-planes $ORCA^{\tau}_{A|B}$ for all neighbors B of A, that is,

$$ORCA^{\tau}_{A} = D(0, \mathbf{v}^{max}_A) \cap (\bigcap_{B \neq A} ORCA^{\tau}_{A|B}) \tag{6}$$

where $\mathbf{v}_A^{max}$ is the maximum allowed velocity of agent A. At last, the new velocity $\mathbf{v}_A^{new}$ is chosen from $ORCA_A^\tau$ and is supposed to be as close to the preferred velocity $\mathbf{v}_A^{pref}$ as possible, i.e.

$$\mathbf{v}_A^{new} = \underset{\mathbf{v}\in ORCA_A^\tau}{\arg\min} \left\| \mathbf{v} - \mathbf{v}_A^{pref} \right\|. \tag{7}$$

Agent B would take a similar procedure to generate its new velocity.

III. ORCA-MPC COMBINED COLLISION AVOIDANCE

The primary task of this paper is to navigate multiple agents moving towards their target locations as fast as possible while avoiding collisions. This problem is formulated as a combination of distributed MPC control problems for all agents, where each agent solves a single MPC problem to generate the optimal control input for itself. The formulation of each agent's MPC problem takes the ORCA region presented in Section II as the agent's permitted velocity set.

In the remaining part of this section, the formulation of the MPC problem for a single agent is presented first. Then, how the ORCA algorithm provides velocity constraints for the MPC problem formulation is discussed. The last subsection presents how to cope with obstacles in the workspace.

A. Model Predictive Control Problem

In this paper, the dynamic model of each agent is described by the following second order difference equation:

$$\mathbf{p}_t = \mathbf{p}_{t-1} + \mathbf{v}_{t-1}\tau_s + \frac{1}{2}\mathbf{a}_{t-1}\tau_s^2, \tag{8}$$

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \mathbf{a}_{t-1}\tau_s, \tag{9}$$

where $t = 1, 2\ldots$, and $\mathbf{p}_t$ and $\mathbf{v}_t$ are the position and velocity of an agent at time t, respectively. $\mathbf{a}_t$ is the acceleration and is treated as the control input. τ_s is the sampling time period which is smaller than τ in (2).

Define a cost function $c(\mathbf{z}_k, \mathbf{a}_k)$ as follows:

$$c(\mathbf{z}_k, \mathbf{a}_k) = w_{obj}\|\mathbf{p}_k - \mathbf{p}^*\|^2 + w_a\|\mathbf{a}_k\|^2. \tag{10}$$

Here, $w_{obj} > 0$ and $w_a > 0$ are weighting factors. $\|\mathbf{p}_k - \mathbf{p}^*\|$ is the distance between the position $\mathbf{p}_k$ at step k and the target position $\mathbf{p}^*$. The minimization of this term represents the demand for fast target reaching. The term $\|\mathbf{a}_k\|^2$ represents control cost.

Using the above cost function, the MPC problem at each time t takes the following form:

$$\underset{\mathbf{z}_k, \mathbf{a}_k}{\arg\min} \sum_{k=1}^{N} c(\mathbf{z}_k, \mathbf{a}_k) \tag{11}$$

$$\begin{aligned} \text{s.t. } & \forall k = 1, 2, \ldots, N, \\ & \mathbf{z_0} = \mathbf{z}_t \\ & \mathbf{z}_k = f(\mathbf{z}_{k-1}; \mathbf{a}_{k-1}) \\ & g(\mathbf{z}_k, \{\mathbf{z}_{neigh,k}\}) \le 0 \\ & \underline{\mathbf{a}} \le \mathbf{a}_k \le \overline{\mathbf{a}} \\ & \underline{\mathbf{z}} \le \mathbf{z}_k \le \overline{\mathbf{z}}. \end{aligned} \tag{12}$$

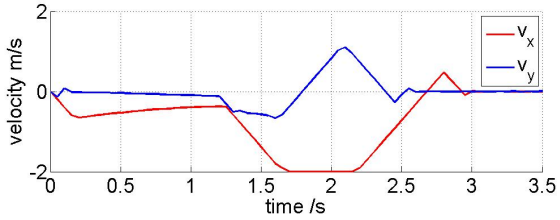
where N is the number of predicted steps, and $N \cdot \tau_s$ is supposed to be no larger than the time horizon τ of the ORCA algorithm; $\mathbf{z}_k = (\mathbf{p}_k, \mathbf{v}_k)$ represents the agent's state at the kth prediction step, and its initial value $\mathbf{z}_0$ at each prediction period $[t+1, t+N]$ is set as the agent's current state $\mathbf{z}_t$. $f(\mathbf{z}_{k-1}; \mathbf{a}_{k-1})$ is the state equation of the agent as defined in (8) and (9). The function $g(\mathbf{z}_k, \{\mathbf{z}_{neigh,k}\}) \le 0$ are the linear velocity constraints generated by the ORCA algorithm in (4) and (5) with respect to all neighboring agents' states $\{\mathbf{z}_{neigh,k}\}$. The states $\mathbf{z}_k$ and control inputs $\mathbf{a}_k$ in all prediction steps $k = 1, 2, \ldots, N$ are confined by corresponding lower and upper bounds. Note in (12) that the symbol "$\le$" for vectors is defined for element-wise comparison.

At each execution time, each agent solves independently a corresponding MPC problem to predict the optimal states and the optimal control inputs in the future N steps. Then the predicted optimal control input of the first step is implemented. The resulting trajectories of all agents will be collision free as guaranteed by the velocity constraints provided by the ORCA algorithm.

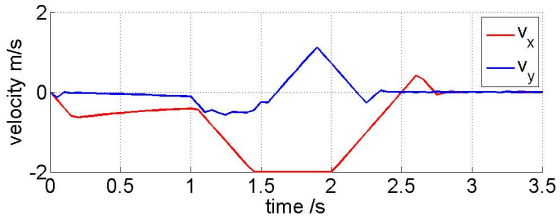
B. Velocity Constraints for the MPC Problem

Note in the MPC problem formulation that the velocity constraints $g(\cdot) \le 0$ in the future N steps are supposed to be provided by the ORCA algorithm which can generate collision-free velocities in future $\tau \ge N \cdot \tau_s$ time period. However, the previously introduced ORCA algorithm assumes that the relative positions and velocities of the agents remain unchanged in τ since the agents' dynamic models are ignored. This could render the solutions far from optimal. To derive the optimal solutions, one should run the ORCA algorithm at each time $t+k$ for $1 \le k \le N$ by using the actual relative positions and velocities at $t+k$. Apparently, this information is not yet available at each current execution time t. A natural alternative is to use the predicted states based on the dynamic model in (8) and (9). We will discuss three potential choices of the predicted states. For simplicity of presentation, we consider a pair of agents, A and B, only. In each prediction step $k = 1, 2, \ldots N$, an agent, say A,

- case (i): uses the states $\mathbf{z}_{A,t+k}^-$, $\mathbf{z}_{B,t+k}^-$ predicted at the last time $t-1$ to compute the collision avoidance region $ORCA_{A,t+k}^\tau$. So, agent A needs agent B's predicted state $\mathbf{z}_{B,t+k}^-$ for computation, which requires the agents to be able to communicate with each other about this information. In addition, the old predictions are not updated to count in the agents' current states which are likely to render the resulted $ORCA_{A,t+k}^\tau$ not a collision avoidance region any more.
- case (ii): uses the states $\mathbf{z}_{A,t+k}^{ca}$, $\mathbf{z}_{B,t+k}^{ca}$ predicted at the current time t under the assumption that the *accelerations* (or control inputs) of both agents remain constant in the future N steps. This is a reasonable setup based on existing known information. However, the agents need to communicate with each other about their control inputs.



(a) The current states, $\mathbf{z}_{A,t}$, $\mathbf{z}_{B,t}$, are used in the ORCA algorithm to generate velocity constraints for MPC in all N future steps.



(b) The predicted states, $\mathbf{z}^{cv}_{A,t+k}$, $\mathbf{z}^{cv}_{B,t+k}$, $k = 1, \ldots, N$, are used in the ORCA algorithm to generate velocity constraints for MPC.

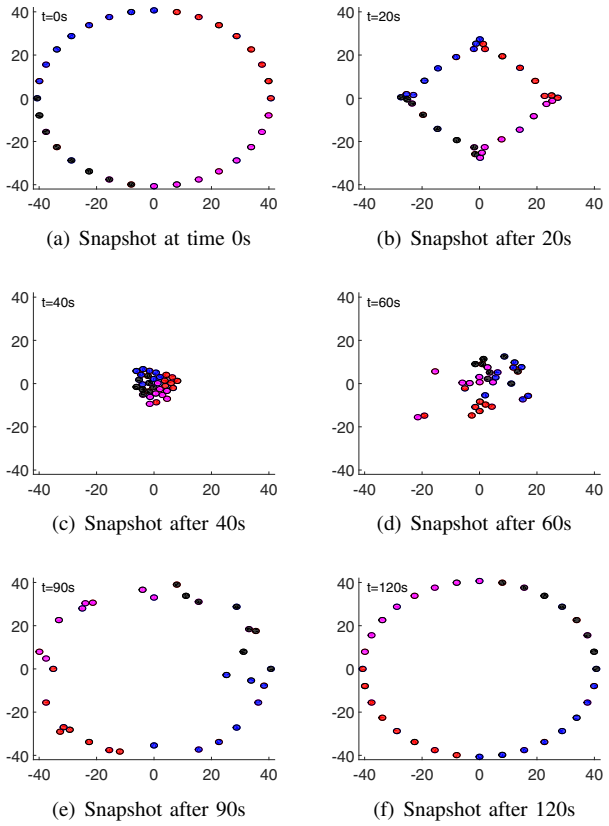
Fig. 3. To avoid collision with agent B, the velocity of agent A reacts earlier using the predicted states (b) than using in all N future steps the current states (a).

- case (iii): uses the states $\mathbf{z}^{cv}_{A,t+k}$, $\mathbf{z}^{cv}_{B,t+k}$ predicted at the current time t under the assumption that the *velocities* of both agents remain constant in the future N steps. This method doesn't require inter-agent communication since agent A can infer $\mathbf{z}^{cv}_{B,t+k}$ by observing agent B's current state. Using predicted states, the agents can react earlier than using the current states, $\mathbf{z}_{A,t}$ and $\mathbf{z}_{B,t}$, in all N future steps (see the velocity changes in Fig. 3).

Remark 1: Note from the above design that the ORCA-MPC combined approach is still a decentralized algorithm. There are three other major advantages of our ORCA-MPC combined approach over existing ORCA based navigation methods. i) The trajectories of the agents using MPC will be much smoother than those generated by the pure ORCA algorithm that assumes instantaneous velocity changes for the agents [11]. ii) Incorporating the dynamic models of the agents, some variants of the ORCA algorithm may ruin the convexity of the velocity constraints and render the solutions suboptimal [12], [14]. In contrast, our ORCA-MPC algorithm retains the convex velocity constraints provided by the ORCA algorithm, and can generate optimal control inputs. iii) The ORCA algorithm needs the preferred velocity $\mathbf{v}^{pref}$ preplanned so as to compute the optimal new velocity in (7). In contrast, our ORCA-MPC approach does not need this preplanning procedure as long as the target position of the agent is provided. This feature is extremely beneficial in real-time applications, such as avoiding obstacles to be discussed in the sequel.

C. Obstacles

As in original ORCA algorithms, small-sized obstacles can be considered as passive agents so that they can be avoided by the active agents. For large-sized obstacles, the ORCA algorithm may become inapplicable when the obstacles are too large to be treated as disc-agents. A solution is to model the observed obstacles as constraints for the agents' positions in the MPC problem. The main difficulty is the non-convexity of these constraints rendered by obstacles. To solve this problem, one approach is to use the maximum convex region of feasible positions [18], and another approach is to divide the non-convex region into a series of convex sub-regions [15], [19]. The latter approach is adopted in this paper while the implementation details are omitted.



(a) Snapshot at time 0s (b) Snapshot after 20s (c) Snapshot after 40s (d) Snapshot after 60s (e) Snapshot after 90s (f) Snapshot after 120s

Fig. 4. Snapshots of collision-free motions of 32 agents exchanging positions with the one at the antipodal point of the circle. (The units of the X-Y axes are meters in this paper unless otherwise specified.)

IV. SIMULATION RESULTS

In this section, we conduct computer simulations to evaluate the proposed ORCA-MPC combined multi-agent navigation algorithm. The simulations were carried out using Matlab R2014a on a laptop with Intel Dual Core i7 CPU and 8GB RAM. The operation system of the computer is Windows 7 Professional 64-bit. The sampling time period is $\tau_s = 50$ms and the time window of the ORCA algorithm is $\tau = 3$ seconds. In the MPC problem formulation, the predicted states in case (iii) are used to generate velocity constraints. The MPC problem is solved by FORCES using efficient interior point method [20], [21].

A. 32 Agents Swapping Positions

In this simulation, 32 agents initially located uniformly on a circle as shown in the first figure of Fig. 4. Every pair

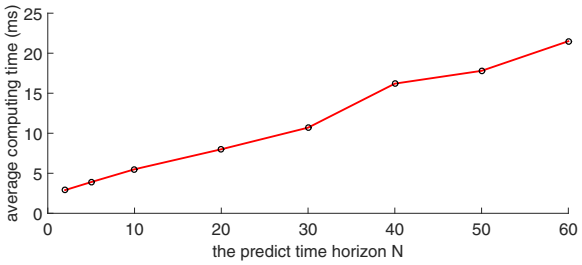


Fig. 5. Average computing time per agent per execution vs predict time horizon N (The number of agents is $n = 32$).

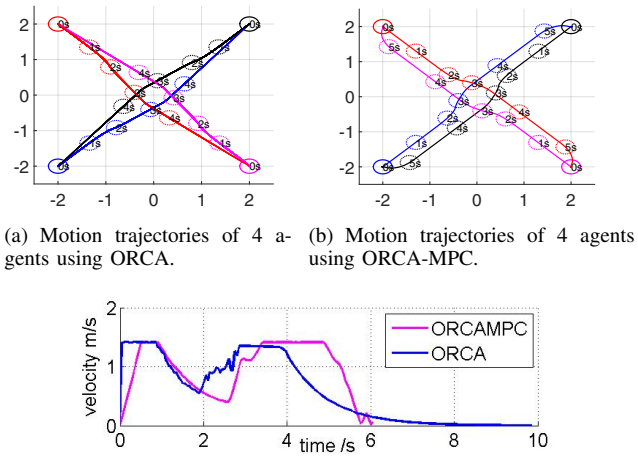
of agents at antipodal positions are expected to exchange positions with each other. So, the circle center may become the bottleneck of all agents' paths. The number of predict steps of the MPC algorithm is set as $N = 20$. Simulation results in Fig. 4 show snapshots of the agents in six different time steps. It is seen that all agents are successfully navigated to their target positions without causing collision. By varying the prediction steps N, we see from Fig.5 approximately linear increase of the average computing time for an agent. This shows acceptable computational performance of the algorithm since even when $N = 60$, the predicted time horizon $N * \tau_s = 3$s reaching its maximum allowed value τ, the average computing time (about 23ms) is still far smaller than the sampling time period $\tau_s = 50$ms.

B. Comparison Between ORCA-MPC and ORCA

In this section, we compare the performance of the ORCA algorithm with the ORCA-MPC algorithm. The number of predict steps of the ORCA-MPC algorithm is set as $N = 10$.

1) Trajectories: For clarity, a small number of 4 agents are considered for the scenario in section IV-A. The agents' trajectories are presented in Fig. 6(a) and 6(b) where both algorithms can navigate the agents to their target positions without causing collision. The velocities of one of the four agents are shown in Fig. 6(c) where the fluctuations around time period 1s to 3s indicate that the agent was changing its speed to avoid collision with others. As seen in this period, the velocity curve using the ORCA-MPC algorithm is much smoother than that using the ORCA algorithm.

2) Computing Time: To evaluate the computing time of the ORCA-MPC algorithm as the number of agents increases, navigation tasks similar to that in subsection IV-A are performed for different number n of agents. For each n, 200 simulations are conducted using the ORCA algorithm and the ORCA-MPC algorithm, respectively, and the computing time of each algorithm is averaged over all agents, all executions and all simulations. From Fig.7, we can see linear increases of the averaged computing time for both algorithms as the number of agents increases, owing to the distributed nature of both algorithms. However, the average computing time of the ORCA-MPC algorithm increases much faster than that of the ORCA algorithm, and it will exceed the sampling period $\tau_s = 50$ms when the number of agents is larger than 190 approximately. So, the new algorithm may be suitable when the number of agents is not too large.



(a) Motion trajectories of 4 agents using ORCA.

(b) Motion trajectories of 4 agents using ORCA-MPC.

(c) Velocities of one of the four agents using two algorithms

Fig. 6. Comparison between ORCA and ORCA-MPC. In (a) and (b), colored circles represent different agents, and dotted lines are their trajectories.

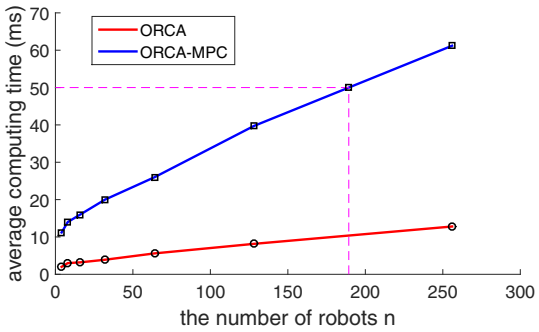
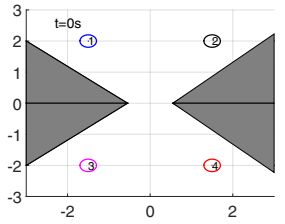


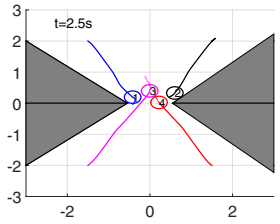
Fig. 7. Average computing time per agent per execution vs the number of agents (The predict time horizon is $N = 10$).

C. Navigation with Static Obstacles

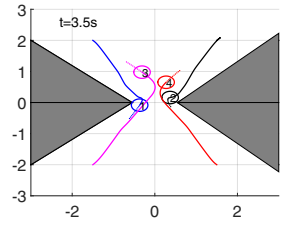
In this simulation, we present the scenario where the workspace contains four agents and two large static obstacles. The initial configurations are shown in Fig. 8(a) where agents in the same column are supposed to exchange their positions. The grey areas are the obstacles and the narrowest part between them allows at most two agents to pass through simultaneously. Fig. 8(b) to Fig. 8(d) show snapshots of the simulation result where the four agents traveled along the edges of the obstacles, encountered with each other at the narrowest passage, and finally reached their target positions, respectively. The velocities of the four agents are drawn in Fig. 8(e) where one can see clearly the process of decelerations and accelerations near the narrowest passage around time 2.5s to 4s. From the velocities in the y direction one sees that the agents 1 and 2 reduced their vertical speeds to nearly zero so that the other two agents 3 and 4 can pass through first. Velocities after time 5s show that agents 3 and 4 arrived at their target positions earlier than the agents 1 and 2. This result illustrates that the ORCA-MPC algorithm can solve the multi-agent navigation problem in environments with static obstacles, and no preplanning for the preferred



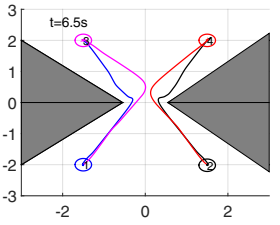
(a) Initial configuration of the agents and obstacles.



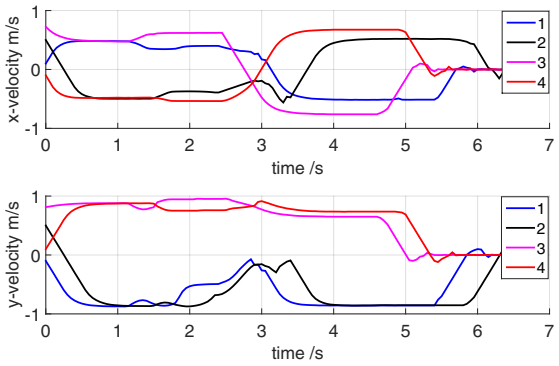
(b) Agents encountered at the passage.



(c) Agents passed through the passage.



(d) Every agent arrived at its target position.



(e) Velocities of the four agents

Fig. 8. Navigation of four agents using the ORCA-MPC algorithm in a workspace containing static obstacles.

velocities is needed for the agents.

V. CONCLUSIONS

In this paper, a distributed multi-agent navigation method is proposed by combing the optimal reciprocal collision avoidance criterion and model predictive control (ORCA-MPC). In the proposed algorithm, each agent generates velocity constraints in N future steps using the ORCA algorithm and the predicted states from the agent's dynamic model. Under these constraints, the local optimal control input for each agent is derived by solving a MPC problem that minimizes the agent's control cost and the distance between the agent's current position and the target position. The proposed approach can avoid collisions with agents and obstacles without predefining the preferred velocities for the agents. Simulation results show that the velocity trajectories using the new method are much smoother than those using the traditional ORCA algorithm that allows instantaneous velocity changes. In the future, the performances of the proposed approach will be evaluated in experiments. In addition, navigating agents with non-holonomic dynamics will be a direction to extend the ORCA-MPC combined method.

REFERENCES

[1] I. Karamouzas and S. J. Guy, "Prioritized group navigation with formation velocity obstacles," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 5983–5989.

[2] J. Alonso-Mora, A. Breitenmoser, M. Rufli, et al, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed Autonomous Robotic Systems*, 2013, pp. 203–216.

[3] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "Smooth and collision-free navigation for multiple robots under differential-drive constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 4584–4589.

[4] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[5] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, 2009.

[6] G. Sanchez and J.-C. Latombe, "Using a PRM planner to compare centralized and decoupled planning for multi-robot systems," in *IEEE International Conference on Robotics and Automation*, 2002, pp. 2112–2119.

[7] P. Švestka and M. H. Overmars, "Coordinated path planning for multiple robots," *Robotics and Autonomous Systems*, vol. 23, no. 3, pp. 125–152, 1998.

[8] L. Jaillet and T. Siméon, "A prm-based motion planner for dynamically changing environments," in *IEEE International Conference on Intelligent Robots and Systems*, 2004, pp. 1606–1611.

[9] 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.

[10] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[11] J. Van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Springer Tracts in Advanced Robotics*, vol. 70, pp. 3–19, 2011.

[12] J. Van den Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3475–3482.

[13] J. Van den Berg, D. Wilkie, S. J. Guy, et al, "LQG-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *IEEE International Conference on Robotics and Automation*, 2012, pp. 346–353.

[14] D. Bareiss and J. Van den Berg, "Reciprocal collision avoidance for robots with linear dynamics using LQR-obstacles," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 3847–3853.

[15] 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.

[16] O. Andersson, W. Mariusz, R. Piotr, and P. Doherty, "Model-predictive control with stochastic collision avoidance using bayesian policy optimization," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 4597–4604.

[17] M. Odelga, P. Stegagno, and H. H. Blthoff, "Obstacle detection, tracking and avoidance for a teleoperated uav," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 2984–2990.

[18] R. Deits and R. Tedrake, "Computing large convex regions of obstacle-free space through semidefinite programming," in *Springer Tracts in Advanced Robotics*, vol. 107, pp. 109–124, 2015.

[19] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[20] A. Domahidi, A. U. Zgraggen, et al, "Efficient interior point methods for multistage problems arising in receding horizon control," in *IEEE Conference on Decision and Control*, 2012, pp. 668–674.

[21] A. Domahidi, "FORCES: Fast optimization for real-time control on embedded systems," *Available from http://forces. ethz*, 2012.