2019 American Control Conference (ACC)
Philadelphia, PA, USA, July 10-12, 2019

# Distributed Online Convex Programming for Collision Avoidance in Multi-agent Autonomous Vehicle Systems

Guohui Ding, Hadi Ravanbakhsh, Zhiyuan Liu, Sriram Sankaranarayanan, and Lijun Chen

*Abstract*— **We frame the collision avoidance problem of multi-agent autonomous vehicle systems into an online convex optimization problem of minimizing certain aggregate cost over the time horizon. We then propose a distributed real-time collision avoidance algorithm based on the online gradient algorithm for solving the resulting online convex optimization problem. We characterize the performance of the algorithm with respect to a static offline optimization, and show that, by choosing proper stepsizes, the upper bound on the performance gap scales sublinearly in time. The numerical experiment shows that the proposed algorithm can achieve better collision avoidance performance than the existing Optimal Reciprocal Collision Avoidance (ORCA) algorithm, due to less aggressive velocity updates that can better prevent the collision in the long run.**

## I. Introduction

Collision avoidance is an important problem for the safe operation of autonomous vehicle systems. In such a multi-agent system setting, partial observability and limited communication necessitate distributed control policies that are based only on information available locally at each agent. Moreover, effective collision avoidance requires real-time decisions that are based only on current and past information (as well as possibly limited prediction).

Indeed, many collision avoidance schemes start with these practical constraints on available information, and the resulting control schemes are highly scalable with low implementation complexity. A prominent example is the Optimal Reciprocal Collision Avoidance (ORCA) [13] algorithm, where each vehicle assumes that its neighbors will keep their current velocities within a certain prediction time window, and computes a sufficient collision-free condition in velocity. Then the vehicle will choose a collision-free velocity that is closest to the desired velocity.

One of the problems with these distributed real-time schemes such as the ORCA algorithm is that they may not be optimal from a systemwide perspective. For example, the vehicles may end up with a situation where most of them change their velocities excessively in order to avoid the collision. Another problem is that many of these schemes are myopic or greedy, and it is possible that a "best possible" action at the current time may get the vehicles into an unnecessarily prolonged collision avoidance phase when they have to frequently adjust their velocities. In this paper, we aim to alleviate the latter problem, focusing on the ORCA method.

This work was supported by NSF award No. 1646556.

All authors are with the Department of Computer Science, University of Colorado Boulder, CO 80309, USA. The email: firstname.lastname@colorado.edu.

Specifically, we frame the collision avoidance problem into an online convex optimization problem of minimizing the aggregate cost over the time horizon, where for each vehicle the collision-free velocity set at each time is computed in the same way as the ORCA algorithm and the cost function at each time is the difference from the desired velocity. We then propose a distributed real-time collision avoidance algorithm based on the online gradient algorithm for solving the resulting online convex optimization problem. We characterize the performance of the algorithm with respect to a static offline optimization, and show that, by choosing proper stepsizes, the upper bound on the performance gap scales sublinearly in time. Compared with the original ORCA algorithms, the decisions based on the gradient algorithm for minimizing the aggregate cost over the whole time horizon expect to result in less aggressive velocity updates that can better avoid collisions in the long run. This is confirmed by our numerical experiment.

The rest of the paper is organized as follows. Section II reviews some related work. Section III describes the ORCA algorithm, and Section IV reviews online convex programming. Section V presents the proposed distributed real-time collision avoidance algorithm and characterizes its performance analytically. Section VI evaluates the proposed algorithm numerically, and Section VII concludes the paper.

## II. Related Work

### A. Collision avoidance

Collision avoidance is a critical safety requirement in control design for autonomous vehicle systems or robotic systems; see, e.g., [1] for a survey on various collision avoidance approaches for unmanned vehicle systems, and [14] for a more recent survey on various strategies and methodologies for path-following and collision-free formation coordination, as well as communication issues involved.

Of particular relevance to this paper is [5] that introduces the concept of velocity obstacle representing the set of velocities that would result in a collision with a given vehicle at a given velocity within certain prediction time window, and [13] that proposes the Optimal Reciprocal Collision Avoidance (ORCA) algorithm based on velocity obstacles. The ORCA algorithm is a distributed strategy for cooperative collision avoidance between mobile agents; see Section III for a detailed description (as well as a brief discussion of its limitation).

There are lots of efforts on improving or extending the ORCA method; see, e.g., [7] for fast computation, [12] for a hybrid reciprocal velocity obstacle method, [2] for

978-1-5386-7926-5/$31.00 ©2019 AACC

a more general acceleration-velocity obstacle method, [11] for incorporating motion continuity constraints, and [4] for the integration of the ORCA method and model predictive control (MPC). In this paper, we integrate the ORCA method into the online convex programming framework, and use it to guide the design of the online algorithm to achieve better collision avoidance.

### B. Online convex programming

Online convex programming considers a setting where decisions have to be made based on the current and past information while trying to minimize certain aggregate cost over a finite or an infinite time horizon. It has broad applications in, e.g., prediction [10], [6], [8], [3] and online learning [9], [15]. See Section IV for a more detailed description of the online convex programming.

## III. Optimal Reciprocal Collision Avoidance

Optimal Reciprocal Collision Avoidance (ORCA) [13] is a velocity-based decentralized strategy for collision avoidance between multiple mobile agents (e.g., autonomous robots or vehicles), where all agents will follow the same strategy to select their actions and take the collision avoidance responsibility cooperatively. Each agent assumes that its neighbors will keep their current velocities within a certain prediction time window, and computes a sufficient collision-free condition (called *velocity obstacle (VO)* [5]). Then the agents will choose their velocities such that they get as close as possible to the desired velocities while simultaneously outside the VOs. Only the information on its neighbors' states is needed for each agent's decision.

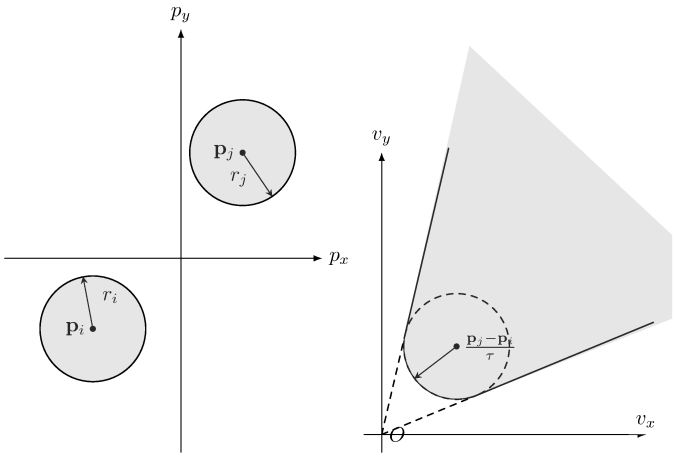
Consider a set $N$ of mobile agents, each $i \in N$ with a set $S_i^t$ of feasible states at time $t$. Denote by $\mathbf{x}_i^t = \{\mathbf{p}_i^t, \mathbf{v}_i^t\} \in S_i^t$ the state of agent $i$ at time $t$, where $\mathbf{p}_i^t = (x, y)_i^t$ is the two-dimensional position and $\mathbf{v}_i^t = (v_x, v_y)_i^t$ the two-dimensional velocity. For each agent $i$, the safe zone at time $t$ is specified by a ball $B(\mathbf{p}_i^t, r_i) = \{\mathbf{p} | \|\mathbf{p}_i^t - \mathbf{p}\|_2 \leq r_i\}$ centered at the current position $\mathbf{p}_i^t$ with a radius $r_i$.

Denote by $\tau$ the length of prediction time window. The velocity obstacle $VO_{i|j}^\tau$ of agent $i$ caused by agent $j$ during the prediction window is defined by:

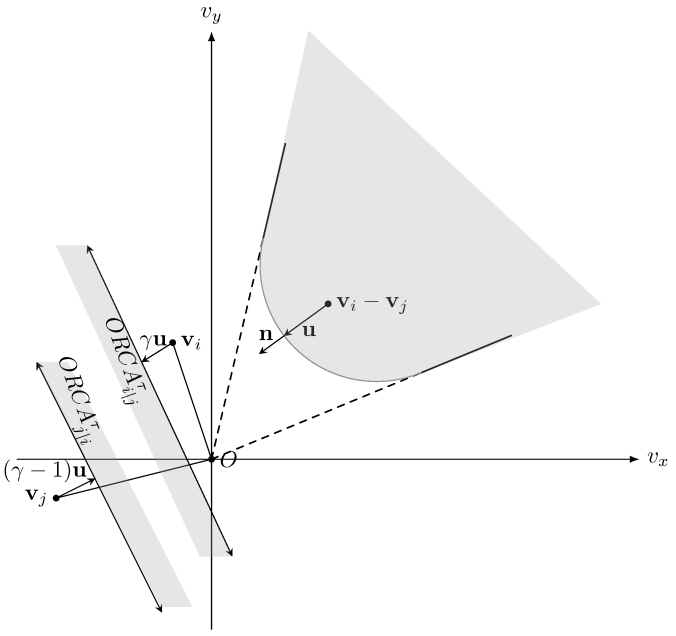
$$VO_{i|j}^\tau = \{\mathbf{v} | \exists t \in [0, \tau], t\mathbf{v} \in B(\mathbf{p}_j - \mathbf{p}_i, r_i + r_j)\}. \quad (1)$$

If agent $i$ has a velocity in $VO_{i|j}^\tau$ during the next $\tau$ time window, it will collide with agent $j$ who is not moving. If agent $j$ moves at velocity $\mathbf{v}_j$, we can consider the relative velocity $\mathbf{v}_i - \mathbf{v}_j$ of agent $i$; see Fig. 1a for a geometric illustration. By definition, $VO_{i|j}^\tau$ is a truncated convex cone with the following implications [4]:

- C1: If $\mathbf{v}_i - \mathbf{v}_j \in VO_{i|j}^\tau$, agent $i$ will collide with agent $j$ if they keep their current velocities. So agent $i$ will try to take an action $\mathbf{u}_i^t$ to escape from $VO_{i|j}^\tau$ as soon as possible.
- C2: If $\mathbf{v}_i - \mathbf{v}_j \notin VO_{i|j}^\tau$, it is safe for both agents to keep the current velocities.




(a) Velocity obstacle




(b) ORCA

Fig. 1: Velocity obstacle and the ORCA algorithm [4] [13].

Consider

$$\mathbf{u} = \arg \min_{\mathbf{v} \in \partial VO_{i|j}^\tau} \|\mathbf{v} - (\mathbf{v}_i - \mathbf{v}_j)\|_2 - (\mathbf{v}_i - \mathbf{v}_j). \quad (2)$$

If $\mathbf{v}_i - \mathbf{v}_j \in VO_{i|j}^\tau$, $\mathbf{u}$ is the "minimum" velocity change that allows $i$ to escape from $VO_{i|j}^\tau$; and if $\mathbf{v}_i - \mathbf{v}_j \notin VO_{i|j}^\tau$, $\mathbf{u}$ will be the "maximum" velocity change that does not lead to a collision. The optimal reciprocal collision avoidance velocity set $ORCA_{i|j}^\tau$ for agent $i$ with respect to agent $j$ [4] is defined as:

$$ORCA_{i|j}^\tau = \begin{cases} \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}_i + \gamma\mathbf{u}))^T \mathbf{n} \geq 0\}, & \text{if } \mathbf{v}_i - \mathbf{v}_j \in VO_{i|j}^\tau, \\ \{\mathbf{v} | (\mathbf{v} - (\mathbf{v}_i + \gamma\mathbf{u}))^T \mathbf{n} \leq 0\}, & \text{if } \mathbf{v}_i - \mathbf{v}_j \notin VO_{i|j}^\tau, \end{cases} \quad (3)$$

where $\mathbf{n} = \begin{cases} \frac{\mathbf{u}}{\|\mathbf{u}\|_2} & \text{if } \mathbf{v}_i - \mathbf{v}_j \in VO^\tau_{i|j} \\ -\frac{\mathbf{u}}{\|\mathbf{u}\|_2} & \text{if } \mathbf{v}_i - \mathbf{v}_j \notin VO^\tau_{i|j} \end{cases}$ is the outward normal vector pointing to the feasible set, and $\gamma \in (0, 1]$ is the responsibility factor. We see that $ORCA^\tau_{i|j}$ is actually a half-plane of the feasible collision-free velocities; see Fig 1b for a geometric illustration. The velocity set $ORCA^\tau_{j|i}$ can be calculated similarly. In the previous work, $\gamma = \frac{1}{2}$ which means that agents $i$ and $j$ take half of the collision avoidance responsibility respectively.

Agent $i$'s collision-free velocity set is given by the intersection $ORCA^\tau_i = \cap_{j \neq i} ORCA^\tau_{i|j}$, which is a polygon of the collision-free velocities considering all neighbors. Agent $i$ will then take the action by choosing a velocity in $ORCA^\tau_i$ that is closest to the desired velocity. The resulting ORCA algorithm for agent $i$ is summarized as Algorithm 1. At each time $t$, all agents will follow Algorithm 1 to update their velocities and positions.

**Algorithm 1** Original ORCA [13]

1: Calculate $VO^\tau_{i|j}$ (1) of agent $i$.
2: Calculate $ORCA^\tau_{i|j}$ based on (3).
3: Calculate collision-free velocity set $ORCA^\tau_i$

$$ORCA^\tau_i = B(\mathbf{0}, v_i^{max}) \cap \bigcap_{j \neq i} ORCA^\tau_{i|j}. \quad (4)$$

4: Compute new velocity $\mathbf{v}_i^{new}$ (and update position $\mathbf{p}_i^{new}$)

$$\mathbf{v}_i^{new} = \arg \min_{\mathbf{v} \in ORCA^\tau_i} \|\mathbf{v} - \mathbf{v}_i^{pref}\|_2, \quad (5)$$

$$\mathbf{p}_i^{new} = \mathbf{p}_i + \mathbf{v}_i^{new} \Delta t. \quad (6)$$

The division and sharing of responsibility as specified by Algorithm 1 leads to cooperative behaviors for collision avoidance. However, this algorithm does not provide any performance guarantee beyond collision avoidance. Indeed, the velocity update (5) is of "greedy" type, and it is possible that the current greedy choice will result in a situation of prolonged collision avoidance phase when the vehicle has to adjust its velocity frequently. Moreover, it may lead to frequent large changes in velocity, which may impact negatively the actuation and fuel efficiency of the vehicle and the state estimation by other vehicles. We will integrate the ORCA algorithm with online convex programming to alleviate these issues.

## IV. Online Convex Programming

Convex programming seeks to minimize a convex cost function $f : S \to \mathbb{R}$ over a convex set $S$. An online convex programming (OCP) problem [15] is defined as follows.

*Definition 1:* An *online convex programming problem* includes a convex set $S \subseteq \mathcal{S}$ and a sequence of convex cost functions $f^t : S \to \mathbb{R}$ at each time or iteration $t = 1, 2, \cdots$.

Instead of a fixed feasible set $S$, there may be a different feasible set $S^t$ at each time $t$. For instance, in the collision avoidance problem the feasible collision-free velocity set $ORCA^\tau_i$ may be different at different times. The cost function $f^t$ captures how well the current decision performs. For example, for the collision avoidance problem the cost function can be chosen as the difference to the desired velocity $\mathbf{v}_i^{pref,t}$: $f_i^t(\mathbf{v}_i^t) := \|\mathbf{v}_i^t - \mathbf{v}_i^{pref,t}\|_2$ for agent $i$ at time $t$.

In the OCP problem, agents make the decisions based on the current and past information, while seeking to minimize the total cost $\sum_t f^t$. In such an online decision setting, an optimal solution is difficult to find because we can only have the current and past $f^t$ at time $t$ instead of all cost functions which make it difficult to find the optimal solution. However, based on the convexity structure of the problem, it is possible to design online algorithms to find a solution that is close to the offline optimum with a bounded gap. In the next section, we will integrate the ORCA method into the OCP framework to a design new collision avoidance algorithm, and characterize analytically and evaluate numerically its performance.

## V. Integration of ORCA and Online Convex Programming

Depending on the mission- or task-level objectives, different cost functions may be constructed. For example, if a mission/task prefers the vehicles to run at certain desired velocities, the cost function can be chosen as the difference to the desired velocity, and if however the goal is to arrive at destinations as early as possible, it can be chosen as the difference to the destination. In this section, we will consider velocity-based method, and integrate the ORCA algorithm and online convex programming with the cost function $f_i^t(\mathbf{v}_i^t) := \|\mathbf{v}_i^t - \mathbf{v}_i^{pref,t}\|_2$ introduced in the last section.

### A. Online algorithm

Notice that the original ORCA Algorithm 1 includes two parts. Steps (1-3) derive the ORCA feasible velocity set of agent $i$ constrained by its neighbors. Step (4) chooses "greedily" a velocity in the the ORCA set that is closest to the desired velocity. In the OCP setting, we will use steps 1-3 of Algorithm 1 to derive the convex feasible set $S_i^t := ORCA^\tau_i$ for agent $i$ at time $t$. Instead of the update (5), we will then use gradient descent method [15] to update the velocity. The application of gradient method will lead to smoother dynamics, as well as performance guarantee as will be seen next.

The resulting algorithm is summarized as Algorithm 2. Notice that in the gradient descent (8) we separate the stepsize into two parts $\alpha$ and $\eta_t$, corresponding to time-independent and time-dependent control parameters.

### B. Performance

We now characterize the performance of Algorithm 2 with respect to certain offline optimum, and examine how different choices of the stepsize will impact the performance of the algorithm.

Consider a time horizon of $T$, and define the total cost $C_i(T) := \sum_{t=1}^T f_i^t(\mathbf{v}_i^t)$ for agent $i \in N$. Denote by $C_i^{online}(T)$ the total cost from the solution generated by

**Algorithm 2** ORCA-OCP

1: Calculate $VO^{\tau}_{i|j}$ (1) of agent $i$.
2: Calculate $S^t_i = ORCA^{\tau}_i$ based on equations (3)-(4).
3: Construct the cost function $f^t_i$,

$$f^t_i(\mathbf{v}) = \|\mathbf{v} - \mathbf{v}^{pref,t}_i\|_2. \tag{7}$$

4: Update velocity using the gradient descent method

$$\mathbf{v}^{t+1}_i = P_{S^t_i}\left(\mathbf{v}^t_i - \alpha\eta_t \nabla f^t_i(\mathbf{v}^t_i)\right), \tag{8}$$

where projection $P_{S^t_i}(y) = \arg\min_{x\in S^t_i}\|x-y\|_2$.

Algorithm 2. Denoted by $\mathbf{v}^*_i$ the static (offline) optimal solution that minimizes $\sum_{t=1}^T f^t_i(\mathbf{v}_i)$ subject to the same feasible collision-free velocity set $S^t_i$ at each time $t$ as Algorithm 2, and $C^{offline}_i(T) = \sum_{t=1}^T f^t_i(\mathbf{v}^*_i)$ the corresponding (offline) minimum cost. Following the standard performance characterization framework for online algorithms (see, e.g., [3], [15]), we define the *regret* of agent $i$ under Algorithm 2 as:

$$R_i(T) = C^{online}_i(T) - C^{offline}_i(T). \tag{9}$$

A smaller regret means a better performance of the online algorithm.

*Theorem 2:* Define the bounds for each agent $i \in N$:

$$\begin{aligned} D_i &= \max_t \max_{\mathbf{u},\mathbf{v}\in S^t_i} \|\mathbf{u}-\mathbf{v}\|_2, \\ G_i &= \max_t \max_{\mathbf{v}\in S^t_i} \|\nabla f^t_i(\mathbf{v})\|_2. \end{aligned}$$

If constant stepsizes are used, i.e., $\eta_t = 1$, the regret of agent $i$ under Algorithm 2 satisfies:

$$R_i(T) \le \frac{D_i^2}{2\alpha} + \frac{\alpha G_i^2 T}{2}, \tag{10}$$

and $\limsup_{T\to\infty}\frac{R_i(T)}{T} \le \frac{\alpha G_i^2}{2}$.

If time-dependent stepsizes $\eta_t = \frac{1}{\sqrt{t}}$ are used, the regret of agent $i$ under Algorithm 2 satisfies:

$$R_i(T) \le \frac{D_i^2\sqrt{T}}{2\alpha} + \frac{\alpha G_i^2}{2}\left(2\sqrt{T}-1\right), \tag{11}$$

and $\limsup_{T\to\infty}\frac{R_i(T)}{T} \le 0$. More generally, if diminishing stepsizes are chosen such that $\sum_{t=1}^T \eta_t = O(T^{1-\epsilon})$ with $\epsilon > 0$, then $\limsup_{T\to\infty}\frac{R_i(T)}{T} \le 0$.

*Proof:* For simplicity of presentation, we omit the index $i$. By equation (8), we have

$$\begin{aligned} &\|\mathbf{v}^{t+1} - \mathbf{v}^*\|_2^2 = \|P_{S^t}\left(\mathbf{v}^t - \alpha\eta_t\nabla f^t(\mathbf{v}^t)\right) - \mathbf{v}^*\|_2^2 \\ \le&\|\mathbf{v}^t - \alpha\eta_t\nabla f^t(\mathbf{v}^t) - \mathbf{v}^*\|_2^2 \\ =&\|\mathbf{v}^t - \mathbf{v}^*\|_2^2 + \alpha^2\eta_t^2\|\nabla f^t(\mathbf{v}^t)\|_2^2 - 2\alpha\eta_t\nabla f^t(\mathbf{v}^t)^T(\mathbf{v}^t - \mathbf{v}^*) \\ \le&\|\mathbf{v}^t - \mathbf{v}^*\|_2^2 + \alpha^2\eta_t^2\|\nabla f^t(\mathbf{v}^t)\|_2^2 - 2\alpha\eta_t(f^t(\mathbf{v}^t) - f^t(\mathbf{v}^*)), \end{aligned} \tag{12}$$

where the first inequality follows from the property of projection operator, and the last inequality follows from the convexity of $f^t(\cdot)$. By (12), we have

$$\begin{aligned} &f^t(\mathbf{v}^t) - f^t(\mathbf{v}^*) \\ \le&\frac{\|\mathbf{v}^t - \mathbf{v}^*\|_2^2 - \|\mathbf{v}^{t+1} - \mathbf{v}^*\|_2^2}{2\alpha\eta_t} + \frac{\alpha\eta_t\|\nabla f^t(\mathbf{v}^t)\|_2^2}{2} \\ \le&\frac{\|\mathbf{v}^t - \mathbf{v}^*\|_2^2 - \|\mathbf{v}^{t+1} - \mathbf{v}^*\|_2^2}{2\alpha\eta_t} + \frac{\alpha\eta_t G^2}{2}, \end{aligned} \tag{13}$$

from which we obtain

$$\begin{aligned} R(T) &= \sum_{t=1}^T \left(f^t(\mathbf{v}^t) - f^t(\mathbf{v}^*)\right) \\ &\le \sum_{t=1}^T \frac{\|\mathbf{v}^t - \mathbf{v}^*\|_2^2 - \|\mathbf{v}^{t+1} - \mathbf{v}^*\|_2^2}{2\alpha\eta_t} + \frac{\alpha G^2}{2}\sum_{t=1}^T \eta_t. \end{aligned} \tag{14}$$

If constant stepsizes $\eta_t = 1$ are chosen, by (14) we obtain

$$\begin{aligned} R(T) &\le \frac{\|\mathbf{v}^1 - \mathbf{v}^*\|_2^2 - \|\mathbf{v}^{T+1} - \mathbf{v}^*\|_2^2}{2\alpha} + \frac{\alpha G^2 T}{2} \\ &\le \frac{\|\mathbf{v}^1 - \mathbf{v}^*\|_2^2}{2\alpha} + \frac{\alpha G^2 T}{2} \\ &\le \frac{D^2}{2\alpha} + \frac{\alpha G^2 T}{2}, \end{aligned}$$

and $\limsup_{T\to\infty}\frac{R(T)}{T} \le \frac{\alpha G^2}{2}$.

If diminishing stepsizes $\eta_t = 1/\sqrt{t}$ are chosen, rewrite inequality (14) as

$$\begin{aligned} &R(T) \\ \le&\frac{\|\mathbf{v}^1 - \mathbf{v}^*\|_2^2}{2\alpha} + \sum_{t=2}^T\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\frac{\|\mathbf{v}^t - \mathbf{v}^*\|_2^2}{2\alpha} \\ &- \frac{\|\mathbf{v}^{T+1} - \mathbf{v}^*\|_2^2}{2\alpha\eta_T} + \frac{\alpha G^2}{2}\sum_{t=1}^T \eta_t \\ \le&\frac{D^2}{2\alpha} + \sum_{t=2}^T\left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\right)\frac{D^2}{2\alpha} + \frac{\alpha G^2}{2}\sum_{t=1}^T \eta_t \\ =&\frac{D^2\sqrt{T}}{2\alpha} + \frac{\alpha G^2}{2}\sum_{t=1}^T \eta_t \\ \le&\frac{D^2\sqrt{T}}{2\alpha} + \frac{\alpha G^2}{2}(2\sqrt{T}-1), \end{aligned} \tag{15}$$

where the last inequality uses the following result [15]:

$$\sum_{t=1}^T \eta_t = \sum_{t=1}^T \frac{1}{\sqrt{t}} \le 1 + \int_{t=1}^T \frac{dt}{\sqrt{t}} \le 2\sqrt{T} - 1.$$

From the above inequality on $R(T)$, we have $\limsup_{T\to\infty}\frac{R(T)}{T} \le 0$. Further, by equation (15), if diminishing stepsizes are chosen such that $\sum_{t=1}^T \eta_t = O(T^{1-\epsilon})$ with $\epsilon > 0$, then $\limsup_{T\to\infty}\frac{R_i(T)}{T} \le 0$. ∎

Theorem 2 shows that the performance (i.e., regret) of Algorithm 2 depends on the choice of the stepsize. In particular, if certain diminishing stepsizes are chosen, the upper bound on the aggregate regret with respect to a (static) offline optimum scales sublinearly in time, and the upper bound on the average regret approaches zero as the time goes to infinity.

*Comment 1:* Even though Algorithm 2 is based on specific cost functions, the proof of Theorem 2 uses only general properties of convex functions. So, Theorem 2 applies to online gradient algorithms with any convex cost functions.

*Comment 2:* Algorithm 2 uses only current information, but can extend to incorporate past information (history); e.g., by replacing the "current" gradient $\nabla f_i^t(\mathbf{v}_i^t)$ with a weighted sum of "past" gradients $\nabla f_i^k(\mathbf{v}_i^k)$, $1 \leq k \leq t$. This may result in better performance bound, and be preferable in certain applications despite its higher implementation complexity. Also, we have focused on the velocity-based method. But the framework and algorithm design apply to the position-based method by choosing proper cost functions in position. We will pursue those extensions elsewhere.

## VI. Experiment

In this section, we implement the proposed ORCA-OCP algorithm, and evaluate its performance by comparing with the original ORCA algorithm.

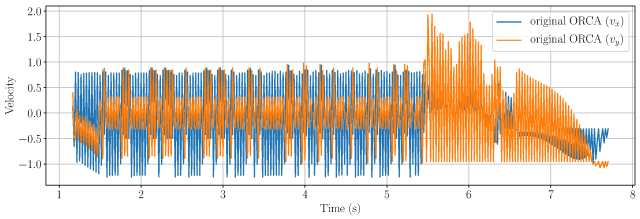
We consider a system of 5 mobile agents with the same safe zone radius of 0.2 (i.e., the same safe distance of 0.4). The initial positions are set to $1.5(\cos(\frac{2\pi}{5}i), \sin(\frac{2\pi}{5}i))$, $i = 1, \cdots, 5$, the desired velocities $(-\sin(\frac{2\pi}{5}i), -\cos(\frac{2\pi}{5}i))$, $i = 1, \cdots, 5$, and the maximal allowable speed $v_i^{\max} = 2$ for all agents. The numerical results reported below are for the parameter values $\gamma = 0.5$ (see equation (3)), $\alpha = 0.5$, and $\eta_t = 1/\sqrt{t}$.

Fig. 2 shows the evolution of one agent's (Agent-1) velocity when it enters the collision avoidance phase (CAP) and needs to adjust velocity frequently. We see that, compared with the original ORCA algorithm, the ORCA-OCP algorithm achieves better collision avoidance as the agent stays shorter in the CAP. This is expected, as the ORCA-OCP algorithm has smoother (or less aggressive) velocity update, as well as tries to minimize an aggregate cost over the whole time horizon so as to better prevent the collision in the long run. This is also confirmed by Fig. 3a which shows that all agents stay in the CAP significantly shorter under the ORCA-OCP algorithm. Further, Fig. 3b shows the means, standard deviations, and maximums/minimums in velocity. We see that the ranges of velocity under the two algorithms are similar, but the deviations are smaller under the ORCA-OCP algorithm. This implies that the ORCA-OCP algorithm achieves better collision avoidance, not because of lower speeds but due to smoother or less aggressive velocity updates.

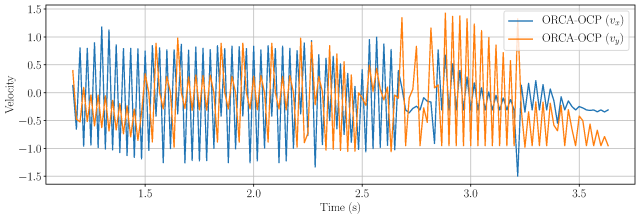
Fig. 4 shows the trajectories of the agents under the original ORCA algorithm and the ORCA-OCP algorithm over the time periods of $[0.0s, 8.6s]$ and $[0.0s, 4.6s]$, respectively. We can see the new algorithm brings less excessive collision avoidance actions than the original ORCA algorithm so that the CAP could be finished earlier. Besides, Fig. 5 shows the evolution of average regret of each agent, which is consistent with Theorem 2.

## VII. Conclusion

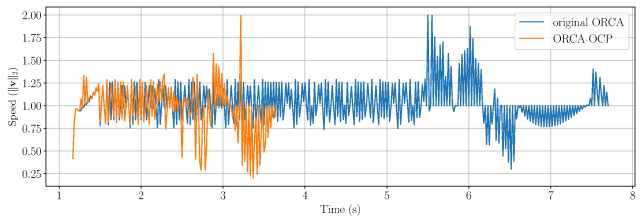
We have integrated the online convex programming (OCP) framework with the optimal reciprocal collision avoidance (ORCA) algorithm, and proposed a new distributed collision avoidance algorithm based on the online gradient algorithm for solving a proper online convex optimization problem. We characterize analytically the performance of the proposed




(a) $v_x$ and $v_y$ under the original ORCA algorithm.

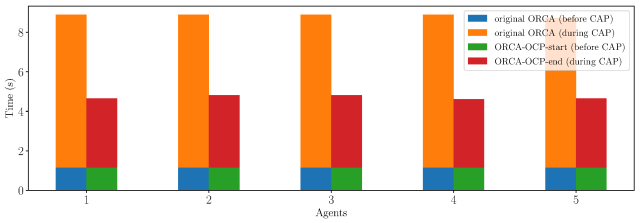



(b) $v_x$ and $v_x$ under the ORCA-OCP algorithm

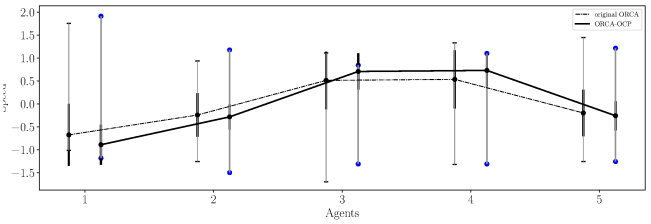



(c) Speed ($\|\mathbf{v}\|$) comparison between the two algorithms.

Fig. 2: Velocity of one agent under the ORCA-OCP algorithm and the original ORCA algorithm when it enters the collision avoidance phase (CAP).

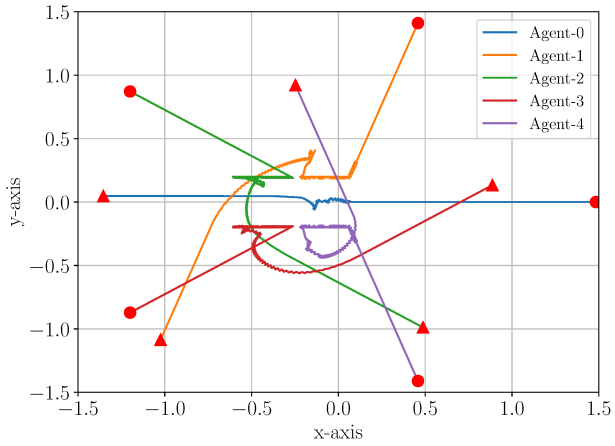



(a) Durations in the collision avoidance phase (CAP).

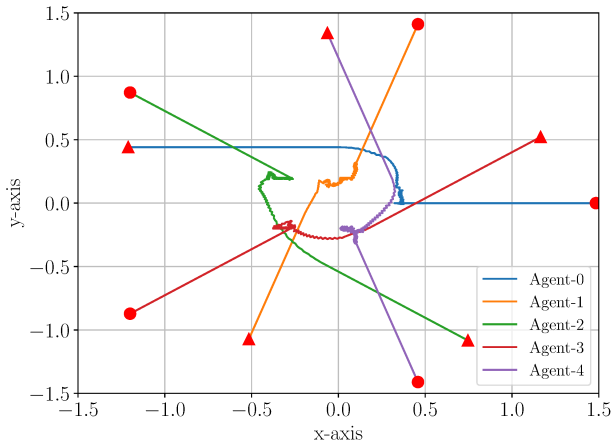



(b) Velocity statistics.

Fig. 3: Durations in the collision avoidance phase (CAP) and velocity statistics.




(a) Trajectories of all agents using Algorithm 1 (the solid circles indicate the starting positions) during $[0.0s, 8.6s]$.




(b) Trajectories of all agents using Algorithm 2 (the solid circles indicate the starting positions) during $[0.0s, 4.6s]$.

Fig. 4: Trajectories using two algorithms.

algorithm in terms of regret with respect to a static offline optimum. The numerical experiment shows that the new algorithm achieves better collision avoidance than the original ORCA algorithm, because of less aggressive velocity updates that can better prevent the collision in the long run.

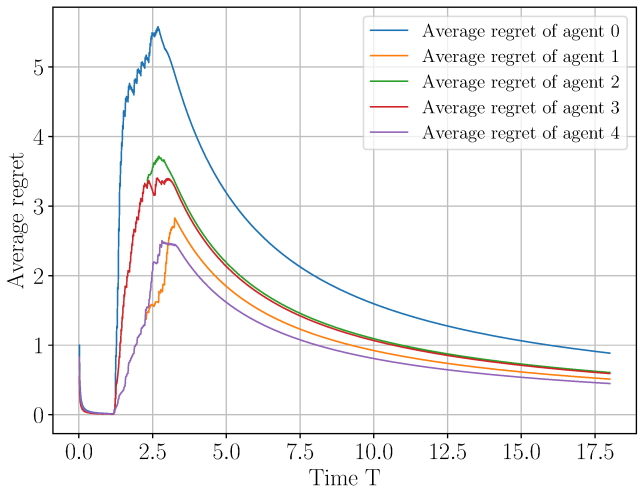



Fig. 5: Average regrets of all agents.

## REFERENCES


[1] BM Albaker and NA Rahim. A survey of collision avoidance approaches for unmanned aerial vehicles. In *technical postgraduates (TECHPOS), 2009 international conference for*, pages 1–7. IEEE, 2009.

[2] Daman Bareiss and Jur van den Berg. Generalized reciprocal collision avoidance. *The International Journal of Robotics Research*, 34(12):1501–1514, 2015.

[3] Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, New York, NY, USA, 2006.

[4] Hui Cheng, Qiyuan Zhu, Zhongchang Liu, Tianye Xu, and Liang Lin. Decentralized navigation of multiple agents based on orca and model predictive control. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 3446–3451. IEEE, 2017.

[5] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772, 1998.

[6] Yoav Freund and Robert E Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

[7] Stephen J Guy, Jatin Chhugani, Changkyu Kim, Nadathur Satish, Ming Lin, Dinesh Manocha, and Pradeep Dubey. Clearpath: highly parallel collision avoidance for multi-agent simulation. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 177–187. ACM, 2009.

[8] Mark Herbster and Manfred K Warmuth. Tracking the best linear predictor. *Journal of Machine Learning Research*, 1(Sep):281–309, 2001.

[9] Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

[10] Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

[11] Martin Rufli, Javier Alonso-Mora, and Roland Siegwart. Reciprocal collision avoidance with motion continuity constraints. *IEEE Transactions on Robotics*, 29(4):899–912, 2013.

[12] Jamie Snape, Jur Van Den Berg, Stephen J Guy, and Dinesh Manocha. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 27(4):696–706, 2011.

[13] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[14] Youmin Zhang and Hasan Mehrjerdi. A survey on multiple unmanned vehicles formation control and coordination: Normal and fault situations. In *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pages 1087–1096. IEEE, 2013.

[15] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.