V-RVO: Decentralized Multi-Agent Collision Avoidance using Voronoi Diagrams and Reciprocal Velocity Obstacles

Senthil Hariharan Arul[1] and Dinesh Manocha[2]

Abstract— We present a decentralized collision avoidance method for dense environments that is based on buffered Voronoi cells (BVC) and reciprocal velocity obstacles (RVO). Our approach is designed for scenarios with large number of close proximity agents and provides passive-friendly collision avoidance guarantees. The Voronoi cells are superimposed with RVO cones to compute a suitable direction for each agent and we use that direction for computing a local collision-free path. Our approach can satisfy double-integrator dynamics constraints and we use the properties of the BVC to formulate a simple, decentralized deadlock resolution strategy. We demonstrate the benefits of V-RVO in complex scenarios with tens of agents in close proximity. In practice, V-RVO's performance is comparable to prior velocity-obstacle methods and the collision avoidance behavior is significantly less conservative than ORCA.

I. INTRODUCTION

Recent advancements in multi-agent robotics support its use in last-mile delivery, warehouse inventory management, and urban surveillance. Many of these applications use a large number of robots (e.g., a few hundred) in a decentralized manner [1]. A key challenge in these scenarios is multi-agent navigation, which includes the computation of safe, collision-free paths between agents in a shared environment.

Prior multi-agent methods compute collision-free trajectories using centralized or decentralized methods. Centralized algorithms such as [2], [3], [4] can generate collision-free trajectories simultaneously for all agents by considering them as one composite system. These techniques can provide rigorous guarantees in terms of (probabilistic) completeness or deadlock avoidance. In the worst case, their computation time can increase exponentially with the number of agents [5], [6]. In practice, these methods are used for a few agents. In contrast, in a decentralized algorithm, each agent makes an independent decision to avoid an impending collision [7], [8], [9], [10]. As a result, these algorithms are scalable and can handle a large number of agents. However, guaranteeing collision avoidance for decentralized methods is non-trivial, especially for agents with higher-order dynamics constraints.

A. Prior Work

There is extensive work on developing decentralized collision-avoidance methods for multi-agent simulation, where each agent computes collision-free paths independently using local sensing of state. Our approach is based on the concept of velocity obstacles. Velocity Obstacle (VO) [11] computes a set of velocities that could result in a collision between agents or with dynamic obstacles. RVO [7] extends the VO concept by assuming that the circular agents share equal responsibility for collision avoidance by computing velocity cones for each pair of nearby agents. In the ORCA algorithm [8], the RVO constraints are linearized to reduce the feasible velocities to a convex set (i.e. ORCA constraints), and linear programming is used to quickly find a feasible solution. ORCA provides a sufficient condition for collision-free navigation for single integrator dynamics. The simplicity of the ORCA formulation makes it possible to extend that algorithm to elliptical agents [12], double integrators [13], linear agents [14], [15], non-holonomic agents [16], combining with statistical inference techniques [17], etc. However, the linear constraints in the velocity space tend to be overly conservative, especially when many agents are in close proximity or in dense scenarios with a large number of agents. As a result, there may be no feasible solution to the ORCA constraints and the resulting algorithms are unable to compute collision-free trajectories. As we take into account additional constraints corresponding to high-order dynamics or non-holonomic agents, the size of the feasible solution set decreases further and makes the approach even more conservative. Other methods have been proposed to accelerate the performance in crowded or challenging environments using interpolating bridges in the narrow passages [18]. In this paper, we present a new algorithm, V-RVO, that is less conservative than ORCA and has similar runtime performance.

Buffered Voronoi cell (BVC) [9] is an efficient decentralized method, which can compute collision-free trajectories for single integrator agents. In BVC, collision avoidance is performed by reasoning in the position space using only the agent's position information. Robots deviate from the goal direction only when the agents are in close proximity. This is in contrast to velocity obstacle methods, which identify that the current velocity would result in future collision and perform an avoidance maneuver. BVC's guarantees need not translate to high-order dynamics because the agent's bounded acceleration could regard a computed path as infeasible. An MPC planner that uses BVC and the braking distance for navigation in noisy scenarios has been proposed [19]. This is similar to our method where we use BVC and braking distance for *passive guarantees*. A decentralized multi-agent rapidly-exploring random tree (DMA-RRT) is

[1]Senthil Hariharan Arul is with the Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20740, USA sarul1@umd.edu

[2]Dinesh Manocha is with the Department of Computer Science, University of Maryland, College Park, MD 20740, USA dmanocha@umd.edu

presented in [20], where an RRT path planner and a token passing method are used for replanning.

Other techniques for collision avoidance between multiple agents are based on a time-to-collision model [10], [21]. Another guaranteed technique is based on inevitable collision states (ICS) [22] and computes a set of agent states that have no collision-free trajectories for an infinite time horizon. ICS provides a theoretical guarantee in terms of collision avoidance, but this method can be very conservative and could regard the entire workspace as forbidden, even in the presence of a few agents [23]. [24] considers a relaxed safety condition known as passive motion safety, where agents cannot collide if they have a non-zero velocity.

B. Main Contributions

We present a novel multi-agent collision avoidance algorithm (V-RVO) that combines the benefits of velocity-space methods (e.g., velocity obstacles) and position-based methods (e.g., BVC). We use the velocity obstacle constraints for each agent to avoid a collision by reasoning over a time-horizon. At the same time, we exploit the characteristics of BVC to guarantee collision avoidance for higher-order dynamics.

In our method, each agent computes a buffered Voronoi cell (BVC) based on the neighbor's position information. We superimpose the RVO cones constraints on the BVC to compute an appropriate goal point on the BVC boundary. Next, we compute a *shrunken* BVC by considering the agent's (and the moving obstacles') stopping distance. We refer to this retracted Voronoi Cell as the *braking-aware buffered* Voronoi cell (baBVC). We compute the braking distance and control input by using the kinematic equations for each agent. Our V-RVO method computes a safe path for each agent inside the baBVC. We prove that baBVC has the property that any agent on the edges of baBVC can incorporate a braking maneuver to halt before reaching the BVC edges. We leverage the property that an agent's BVC is disjoint from its neighbors to provide the collision avoidance guarantees for agents with single and double integrator dynamics. Furthermore, we propose a simple deadlock resolution strategy. As compared to prior methods, V-RVO has the following benefits:

- V-RVO retains many benefits of ORCA, like handling high-order dynamics and anticipatory collision avoidance. The use of baBVC makes it possible to compute collision-free trajectories in dense scenarios, where ORCA is either overly conservative or fails (see Figs. 7).
- Unlike ORCA and BVC, V-RVO can provide safety guarantees for second order agents (Figure 6).

We have implemented our method and compared its performance with ORCA on agents with single- and second-order dynamics for dense scenarios. The average running time for V-RVO is a few milliseconds per agent and is about 3X slower than ORCA. In practice, V-RVO can compute collision-free trajectories for many challenging benchmarks with $25 - 70$ agents, where ORCA tends to fail.

TABLE I: Symbols and notation used in the paper

Notation	Definition
A_i	Refers to the ith agent
$\mathbf{p}_i$	2-D position of A_i $[p_{i,x}, p_{i,y}]$
$\mathbf{g}_i$	2-D goal position of A_i
$\mathbf{v}_i$	2-D velocity of A_i $[v_{i,x}, v_{i,y}]$
$\mathbf{u}_i$	Control Input of the A_i $[a_{i,x}, a_{i,y}]$
R_i	Radius of A_i's enclosing circle
$\mathcal{N}_i$	Neighbors of A_i
$\bar{\mathcal{N}}_i$	Neighbors sharing a Voronoi edge with A_i
$\mathcal{V}_i, \bar{\mathcal{V}}_i$	Voronoi and Buffered Voronoi cell for A_i
$\partial \bar{\mathcal{V}}_i$	Boundary of the set $\bar{\mathcal{V}}_i$
ϵ_p, ϵ_v	Small positive constants

II. BACKGROUND AND PROBLEM FORMULATION

In this section, we give an overview of various concepts used in our approach. The symbols and notations used in the paper are summarized in Table 1.

A. Problem Formulation

We consider an environment with N agents moving in a shared workspace $\mathcal{W} \subset \mathcal{R}^2$. Each agent A_i, where $i \in \{1, 2, ...N\}$, has its bounding geometric shape approximated as a circle of radius R_i. Two agents A_i and A_j are collision-free if

$$\|\mathbf{p}_i - \mathbf{p}_j\|_2 \geq R_i + R_j.$$

That is, the agents are collision-free if the inter-agent distance between A_i and A_j is greater than the sum of their radii. Our goal is to compute a collision-free path for each agent towards its goal position in a decentralized manner. Furthermore, our goal is to provide passive-friendly collision avoidance guarantees for agents with higher-order dynamics.

B. Assumptions

We assume each agent has the exact state information, including positions and velocities, of the neighboring agents. The environment may consist of other static and dynamic obstacles. The exact position of the dynamic obstacle is known, though no assumptions are made about its trajectories. We also assume that the maximum velocity and acceleration of such agents are known a priori.
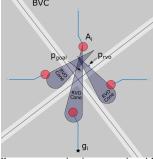

Fig. 1: The figure illustrates a navigation scenario with four agents moving to their antipodal positions. The three RVO cones, corresponding to three neighbors of A_i, are denoted by the *blue* conical regions and are superimposed onto the *gray* BVC region. The $\mathbf{p}_{goal}$ (green point) on the BVC boundary is in the direction of the goal ($\mathbf{g}_i$). Since this point lies in the RVO cone, we compute the closest point on the BVC boundary outside the RVO cones. Point $\mathbf{p}_{rvo}$ is denoted in *blue* and the agent is directed towards this point to avoid a collision.

C. Passive Safety Guarantees

Passive safety is a relaxed safety condition where the agent cannot collide while it is in motion [24]. A collision can occur only in situations where the agent is at rest and a moving obstacle collides with the agent. In *passive-friendly safety*, in addition to *passive safety*, the agent provides sufficient time for the moving obstacle to stop or replan to avoid collisions. However, a collision with a moving obstacle is possible if they plan on colliding. Passive-friendly safety also helps to operate safely in the presence of heterogeneous agents that do not use V-RVO.

D. Buffered Voronoi Cell (BVC)

We use the notion of computing Buffered Voronoi Cell (BVC), proposed by [9]. BVC is a contracted Voronoi region such that an agent whose center point is on the edge of a BVC has its bounding circle inside the Voronoi cell. Given $\mathcal{N}$ agents on a 2-D plane, the buffered Voronoi cell corresponding to an agent A_i is given as

$$\bar{\mathcal{V}}_i = \left\{ \mathbf{p} \in \mathcal{R}^2 \Big| \left(\mathbf{p} - \frac{\mathbf{p}_i + \mathbf{p}_j}{2} \right)^{\mathrm{T}} \mathbf{p}_{ij} + R_i \|\mathbf{p}_{ij}\| \leq 0, \forall j \neq i \right\}. \tag{1}$$

Here, $\mathbf{p_{ij}} = \mathbf{p_i} - \mathbf{p_j}$. The method plans a collision-free path for the agent A_i that is constrained to lie within $\bar{\mathcal{V}}_i$.

E. Reciprocal Velocity Obstacle

Reciprocal Velocity Obstacle (RVO) computes a set of relative velocities between two agents that can result in a future collision. Let us consider two agents A_i and A_j in a shared workspace $\mathcal{W}$. The RVO can be geometrically defined as

$$RVO_j^i(v_j, v_i) = \{\mathbf{v} | \exists t \in [0, \tau] :: t\mathbf{v} \in \mathbf{D}(\mathbf{p}_{ji}, R_{ij})\}. \tag{2}$$

Here, $\mathbf{D}(\mathbf{P}, r) = \{\mathbf{q} \mid \|\mathbf{q} - \mathbf{p}\| \leq r\}$ represents a disk of radius r and center $\mathbf{p}$. The variable $\mathbf{p}_{ji}$ represents the relative position given by $\mathbf{p}_j - \mathbf{p}_i$ and R_{ij} represents the combined radius given by $R_i + R_j$. If A_i chooses a velocity outside the RVO induced by A_j, the agent is guaranteed to be collision-free, provided the trajectories of both agents are governed by single-integrator dynamics.

III. V-RVO: OUR HYBRID NAVIGATION ALGORITHM

In this section, we describe our multi-agent navigation method. This includes computing the BVC for each agent, followed by RVO superposition and baBVC computation. Algorithm 1 highlights our proposed method.

A. RVO Superposition

In our method, we combine RVO constraints with BVC to compute a collision avoiding direction. Let us consider an agent A_i. At each time step, the agent constructs its BVC ($\bar{\mathcal{V}}_i$), and for every $A_j \in \mathcal{N}_i$ we construct an RVO cone w.r.t. each nearby agent. Because the RVO constraints (Equation II-E) are defined in the relative velocity space, each cone is transformed into the velocity space of A_i. The origin of the velocity space is the center of A_i. Thus, the RVO cones are superimposed onto the constructed BVC with $\mathbf{p}_i$ as the origin for the velocity space.

The union of the $\partial \bar{\mathcal{V}}_i$ sections outside the RVO cones provides safe directions of computing the velocity, as they are outside the RVO cones and within the BVC. We denote this union of $\partial \bar{\mathcal{V}}_i$ sections by $\partial \bar{\mathcal{V}}_{CF,i}$ and expressed as:

$$\partial \bar{\mathcal{V}}_{CF,i} = \partial \bar{\mathcal{V}}_i - \bigcup_{j \in \mathcal{N}_i} \partial \bar{\mathcal{V}}_i \cap RVO_j^i(v_j, v_i). \tag{3}$$

A point in the set $\partial \bar{\mathcal{V}}_{CF,i}$ with the least angular deviation from the goal direction is computed and is denoted by $\mathbf{p}_{rvo,i}$:

$$\mathbf{p}_{rvo,i} = \operatorname*{argmin}_{\mathbf{p} \in \partial \bar{\mathcal{V}}_{CF,i}} \arccos \left(\frac{\mathbf{p} \cdot (\mathbf{p}_i - \mathbf{g}_i)}{\|\mathbf{p}\| \cdot \|\mathbf{p}_i - \mathbf{g}_i\|} \right). \tag{4}$$

The agent A_i moves towards the $\mathbf{p}_{rvo,i}$ to avoid collisions with other agents. In order to provide passive-friendly guarantees, the agent's trajectory is planned such that it can brake and stop at the point $\mathbf{p}_{rvo,i}$. Figure 1 illustrates a simple scenario with four agents.

B. Braking-aware BVC

As mentioned in Section II-D, the BVC is generated by buffering the Voronoi cell edges by the agent's radius. Since an agent with single integrator dynamics can instantaneously change its velocity, the agent is guaranteed to be collision-free inside the BVC. For agents with double integrator dynamics the BVC is further buffered based on the minimum braking distance of the agent and the moving obstacle.

We use the kinematic equations of the agent to compute the buffering distance, as shown in Figure 2. Let us consider a time horizon t_h. The agent A_i moves to a position $\mathbf{p}_{rvo,i}$ as computed using RVO constraints. Since the agent has to halt at a point on the edge of the BVC to remain within its Voronoi cell, we plan a path from $\mathbf{p}_i$ to $\mathbf{p}_{rvo,i}$ using the kinematic constraints. Consider a third position $\mathbf{p}_{int,i}$ within the BVC such that the agent at $\mathbf{p}_{int,i}$ with a velocity $\mathbf{v}_{int,i}$ can brake and stop at $\mathbf{p}_{rvo,i}$ in time t_b. We compute the velocity $\mathbf{v}_{int,i}$ by solving the problem separately in X and Y coordinates.

$$s_{int,x} = v_{x,i} t_h + \frac{1}{2t_h} \big(v_{int,x,i} - v_{x,i} \big) t_h^2, \tag{5}$$

$$s_{stop,x} = \frac{v_{int,x}^2}{2a_{max}}. \tag{6}$$

Here, $s_{int,x}$ is the distance travelled while converting the velocity from $\mathbf{v}_{x,i}$ to $\mathbf{v}_{int,x}$, and $s_{stop,x}$ is the distance travelled when the agent decelerates to a stop. Using Equations (5) and (6) along with the relationship $s_x = s_{int,x} + s_{stop}$ (also there abs sum should be sx), we arrive at the following quadratic equation.

$$v_{int,x,i}^2 + (a_{max} t_h) . v_{int,x,i} + (u_{x,i} a_{max} t_h - 2a_{max} s). \tag{7}$$

The velocity $\mathbf{v}_{int,i}$ is computed from the quadratic equation by choosing the velocity with the least angular deviation from the goal direction. The position $\mathbf{p}_{int,i}$ can be computed using $\mathbf{v}_{int,i}$.

We assume low-velocity moving obstacles are present in the environment. The BVC is buffered by a distance d_{obs}
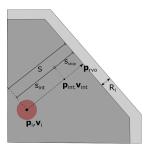
Fig. 2: We show an agent (red circle), Voronoi cell (gray region) and the BVC (dark gray) computed with the agent's radius. $\mathbf{p}_{int}$ and $\mathbf{v}_{int}$ are the position and velocity at an intermediate point computed from Equation 7. s_{int} and s_{stop} are the displacement vectors.

to provide enough stopping distance for the low-velocity moving obstacle, which is computed as:

$$d_{obs} = s_{\mathcal{O}} - \frac{\mathbf{v}_{max,obs}^2}{2a_{max}}. \qquad (8)$$

Here, $S_{\mathcal{O}}$ denotes the shortest distance between the obstacle's center and $\partial \bar{\mathcal{V}}_i$. We use this buffered BVC as the input from computing $\mathbf{p}_{rvo,i}$ in Section III-A. This helps provide passive-friendly collision avoidance guarantees. This results in a *contracted* BVC region, which we refer to as braking-aware buffered Voronoi cell (baBVC). We assume $S_{\mathcal{O}} \geq \frac{\mathbf{v}_{max,obs}^2}{2a_{max}}$ for this computation to maintain $d_{obs} \geq 0$.

C. Multi-Agent Navigation

In a multi-agent scenario, all agents compute the positions $\mathbf{p}_{rvo,i}$ and $\mathbf{p}_{int,i}$ as described in Sections III-A and III-B. For agents with single integrator dynamics, the positions $\mathbf{p}_{rvo,i} = \mathbf{p}_{int,i}$ as the velocity can be instantaneously changed to zero. The control input (velocity) applied to the agent is given as

$$\mathbf{u}_i = \|v_{max}\| \frac{\mathbf{p}_{rvo,i} - \mathbf{p}_i}{\|\mathbf{p}_{rvo,i} - \mathbf{p}_i\|}. \qquad (9)$$

If $\|\mathbf{p}_{rvo,i} - \mathbf{p}_i\| \leq 1$, we apply $\mathbf{u}_i = \mathbf{p}_{rvo,i} - \mathbf{p}_i$.

For scenarios with double integrator agents, the control input ($\mathbf{u}_i$) is acceleration. A constant acceleration for the time horizon t_h is computed as follows:

$$\mathbf{u}_i = \frac{\mathbf{v}_{int,i} - \mathbf{v}_i}{t_h}. \qquad (10)$$

For double integrator agents, we also consider acceleration bounds. The path between $\mathbf{p}_{int}$ and $\mathbf{p}_i$ may not be a straight line. Thus, we verify that next position $\mathbf{p}'_{t,i}$ planned for every time step t in $[0...t_h]$ (considering control input from Equation 10) lies within $\bar{\mathcal{V}}_i$. t_h can be varied to account for the acceleration bound.

D. Collision-Free Guarantees

We prove that agents trajectories computed using V-RVO can provide *passive-safety* guarantees.

Theorem 1: Agents navigating in multi-agent scenarios using Algorithm I are guaranteed to be collision-free.

Proof: Let us consider N agents operating in a workspace $\mathcal{W}$. Let $\mathcal{V}_i$ be the Voronoi region corresponding to agent A_i. BVC region ($\bar{\mathcal{V}}_i$) is constructed by retracting the Voronoi

---

Algorithm 1 V-RVO Navigation Algorithm

Input: $\mathbf{p}_i, \mathbf{v}_i \quad i \in \mathcal{N}$
Output: $\mathbf{u}_i \quad i \in \mathcal{N}$
1: **for** $i \in \mathcal{N}$ **do**
2:     Compute BVC boundary ($\partial \bar{\mathcal{V}}_i$)
3:     Compute RVO cones
4:     $\partial \bar{\mathcal{V}}_{CF,i} = \partial \bar{\mathcal{V}}_i$
5:     **for** $j \in$ RVO Cones **do**
6:         $\partial \bar{\mathcal{V}}_{CF,i} \leftarrow \partial \bar{\mathcal{V}}_{CF,i} \cap$ BVC edge region not inside j^{th} RVO cones {// compute collision-free $\partial \bar{\mathcal{V}}$ section}
7:     **end for**
8:     $\mathbf{p}_{rvo,i} = \arg\min_{\mathbf{p} \in \partial \bar{\mathcal{V}}_{CF,i}} \arccos \left(\frac{\mathbf{p} \cdot (\mathbf{p}_i - \mathbf{g}_i)}{\|\mathbf{p}\| \cdot \|\mathbf{p}_i - \mathbf{g}_i\|} \right)$ {// position in $\partial \bar{\mathcal{V}}_{CF,i}$ with least angular deviation from goal direction}
9:     Compute $\mathbf{p}_{slow,i}, \mathbf{v}_{slow,i}$ from Equation (7)
10:    $\mathbf{u}_i = \frac{\mathbf{v}_{slow,i} - \mathbf{v}_i}{t_h}$ {// compute control input}
11: **end for**
12: **return** $\mathbf{u}_i$

---

edges by a distance equal to the agent's radius. Thus, $\bar{\mathcal{V}}_i \subset \mathcal{V}_i$ and BVC region between the agents is disjoint. That is, $\bar{\mathcal{V}}_i \cap \bar{\mathcal{V}}_j = \emptyset \quad \forall i, j \in N, j \neq i$.

Consider the agents at their initial position $\mathbf{p}_i \quad \forall i \in N$ (at $t = t_0$). Let the corresponding region occupied by the agent geometry be $\mathcal{A}(\mathbf{p}_i)$. Assuming agents are collision-free at $t = 0$, and from the definition of BVC

$$\|\mathbf{p}_i - \mathbf{p}_j\| \geq R_i + R_j \quad \forall j \in N/\{i\}$$
$$\implies \mathcal{A}(\mathbf{p}_i) \subset \mathcal{V}_i \iff \mathbf{p}_i \in \bar{\mathcal{V}}_i. \quad (11)$$

Provided A_i chooses its next position $\mathbf{p}'_i \in \bar{\mathcal{V}}_i$, then $\mathcal{A}(\mathbf{p}'_i) \subset \mathcal{V}_i$ and the agent continues to be collision free in the next time step.

For a single integrator agent, the velocity ($\mathbf{v}_i$) can be instantaneously modified. Hence, the path between $\mathbf{p}_i$ and $\mathbf{p}'_i$ is a straight line ($\mathbf{p}_i \mathbf{p}'_i$). The agent's path is collision-free from the convexity of $\bar{\mathcal{V}}_i$ as

$$\mathbf{p}_i, \mathbf{p}'_i \in \bar{\mathcal{V}}_i \implies \mathbf{p}_i \mathbf{p}'_i \subset \bar{\mathcal{V}}_i \implies \mathcal{A}_i(\mathbf{0}) \oplus \mathbf{p}_i \mathbf{p}'_i \subset \mathcal{V}_i.$$
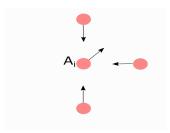
Here, $\mathbf{0}$ is the origin and $\oplus$ denotes Minkowski sum.
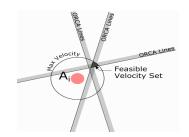
For double integrator agents, we compute a constant acceleration from Equation (10) applied for duration t_h. The acceleration moves the agent to an intermediate point $\mathbf{p}_{int,i}$ such that the agent can decelerate to a stop at $\mathbf{p}_{rvo,i} \in \partial \bar{\mathcal{V}}_i$. Computing the agent's future positions $\mathbf{p}'_{t,i}$ using (10) such that $\mathbf{p}'_{t_h,i} = \mathbf{p}_{int,i}$ and assuming a straight-line movement between time-steps. The agent's path is collision-free as

$$\mathbf{p}'_{t,i} \in \bar{\mathcal{V}}_i \implies \mathcal{A}(\mathbf{p}'_{t,i}) \subset \mathcal{V}_i \implies$$
$$\mathcal{A}_i(\mathbf{0}) \oplus \mathbf{p}'_{t-1,i} \mathbf{p}'_{t,i} \subset \mathcal{V}_i \quad \forall t \in [1...t_h]. \quad (12)$$

The above conditions when satisfied provides *passive* collision avoidance guarantees. Since all agents in the workspace $\mathcal{W}$ use V-RVO, the agents would brake to stop at a point on $\partial \bar{\mathcal{V}}_i$. Thus, agent do not collide with each other.

In the presence of low-velocity moving obstacles, the BVC is initially buffered as mentioned in Equation III-B. This provides a buffered distance for the moving-obstacle to brake without colliding. In this manner, we extend the passive guarantees to the passive-friendly.
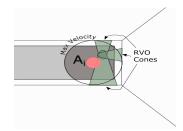
Fig. 3: (Left) image illustrates a navigation scenario with 4 agents (red disks) and their current velocities (denoted by arrows). (Center) image illustrates the feasible velocity set for A_i computed using linear ORCA constraints. In the (right) image the agent's local trajectory is computed using our V-RVO algorithm. The *green* conical region are the RVO constraints superimposed onto the BVC. The feasible regions and directions computed using V-RVO (shown in dark *grey*) is larger than that compared to ORCA (center image).
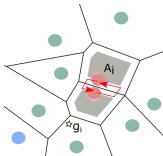

Fig. 4: A Dense scenario with 10 agents (circles). The agent A_i is deadlocked and unable to reach its goal location g_i as its neighbors have reached their goal position. The neighbors in HOLD mode are shown in *green* color. A_i chooses a neighbor and switches its location following the *red* path. The *gray* region corresponds to their respective BVCs.

IV. DEADLOCK

In this section, we present our distributed deadlock resolution strategy based on the constructed BVC. An illustrative example is provided in Figure 4, and the pseudo-code is summarized in Algorithm 2.

A. Deadlock Identification

Deadlock occurs in situations where multiple agents block each other such that one or more agents are unable to reach their goal, and instead remain stationary (with zero velocity) to avoid a collision. In our method, each agent chooses a point on the boundary of its baBVC and travels towards it. When an agent is in a deadlock, it is yet to reach its goal position, and it is at a point on $\partial \bar{\mathcal{V}}_i$ with a zero velocity.

B. Deadlock Resolution

To resolve deadlocks, we define three operating modes for the agents. Each agent A_i can be in one of the three modes: DEADLOCK, HOLD, or DEFAULT. In general, all agents are in the DEFAULT mode, where they are either moving towards their goal, avoiding collisions, or have reached their goal position. The agents in the HOLD mode have their velocities set to zero and thus remain at the current position until the mode is modified to DEFAULT. When an agent is identified to be in a deadlock (as mentioned in the Section IV-A), the agent's mode is modified to DEADLOCK.

Consider an agent A_i whose mode corresponds to DEADLOCK. If the agent A_i has neighboring agents with either DEADLOCK or DEFAULT modes with zero velocity, then it initiates a switch to potentially solve the deadlock.

- The agent initially identifies the neighboring agent (A_k) that is closest to the direction of the agent's goal location. Since A_i and A_k are neighboring agents, they share a common Voronoi edge.
- The states of the neighbors $\bar{\mathcal{N}}_i$ and $\bar{\mathcal{N}}_k$ are set to HOLD. Since the A_i and the agents in $\bar{\mathcal{N}}_i$ share Voronoi edges, setting the agents in $\bar{\mathcal{N}}_i \cup \bar{\mathcal{N}}_k$ to HOLD ensures that the union of BVCs of A_i and A_k could be used for the subsequent time steps.
- We plan a path between the agents A_i and A_k across their common Voronoi edge such that the positions of A_i and A_k are interchanged. (Algorithm 2, line 5).
- Once this position switch is completed, the agent state is reset to DEFAULT.

Algorithm 2 Deadlock Resolution

1: **if** $\|\mathbf{p}_i - \mathbf{g}_i\|_2 \geq \epsilon_p$ and $\|\mathbf{v}_i\| \leq \epsilon_v$ **then**
2:     $Mode_i = $ DEADLOCK
3:     $k = \operatorname{argmin}_{j \in \mathcal{N}_i} \arccos \left(\frac{(\mathbf{p}_i - \mathbf{g}_i) \cdot (\mathbf{p}_i - \mathbf{p}_j)}{\|\mathbf{p}_i - \mathbf{g}_i\| \cdot \|\mathbf{p}_i - \mathbf{p}_j\|} \right)$
4:     $Mode_j = $ HOLD $\forall j \in \bar{\mathcal{N}}_i \cup \bar{\mathcal{N}}_k$
5:     SwitchAgent(i, k)
6:     $Mode_j = $ DEFAULT $\forall j \in \bar{\mathcal{N}}_i \cup \bar{\mathcal{N}}_k \cup i \cup k$
7: **end if**

V. RESULT

In this section, we describe our implementation and highlight the performance of V-RVO in different scenarios. We also compare with ORCA.

A. Experimental Setup

We perform our evaluation on a 2.7 GHz Quad-Core Intel Core $i7$ processor with 16 GB of memory. The agents and obstacles in our scenarios are circular agents of radius 0.25m. The agents have maximum velocity and acceleration of $2m/s$ and $1m/s^2$ respectively. We use a time step of 0.1 seconds for our simulation results.

B. Simulation Results

We evaluate our method on simulation in a circular scenario, where each agent moves to its antipodal goal position. We vary the number of agents and increase the density in the scenario. Figures (5) and (6) illustrates the agent trajectories for single and double integrator dynamics, respectively.

In addition, we compared V-RVO and ORCA on a scenario with 70 agents (Fig. 7). While V-RVO can handle such
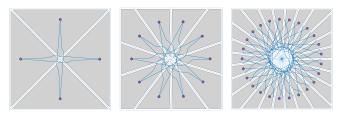
Fig. 5: Collision-free trajectories for single integrator dynamics ($\dot{\mathbf{p}} = \mathbf{u}$). The sub images represent scenarios with 4 agents (left), 10 agent (center), and 25 agents (right).
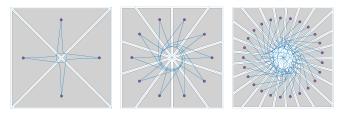

Fig. 6: Collision-free trajectories for double integrator dynamics ($\ddot{\mathbf{p}} = \mathbf{u}$). The sub images represent scenarios with 4 agents (left), 10 agent (center), and 25 agents (right). ORCA cannot compute collision-free trajectories for the 25-agent scenario.

complex scenario, ORCA results in collisions. This is due to the fact that ORCA constraints are not satisfied and it computes the closest velocity, which may not guarantee collision-free navigation.
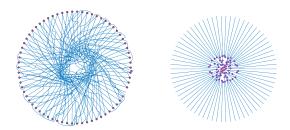

Fig. 7: Trajectories for a scenario with 70 agents. V-RVO (left) can generate collision-free trajectories for all the agents. ORCA (right) results in collisions, as shown with red agents.

We also evaluated V-RVO in a scenarios with agents approaching with head on collisions. Figure 8 illustrates the collision-free trajectories.

C. Deadlock Resolution

We evaluate the performance of our deadlock resolution method on the formation benchmark. There are 16 agents and they are initially placed on the circle perimeter equally spaced. The final positions of the agents are arranged in a 4×4 grid formation (see Fig. 10). Our deadlock resolution method works well and the agents can reach the goal position in Fig. 10 (right).

D. Scalability

We compare the time required to compute the control inputs for the agent in scenarios with a team size ranging from 5 to 70 agents. Each agent considers only its neighbors within its sensing region for the computation. The average

Fig. 8: We highlight the agent trajectories computed using V-RVO with 4 pairs. The pink regions denote the BVCs associated with each agent during each time step. The left, center, and right images show the BVCs and the trajectories in chronological order.
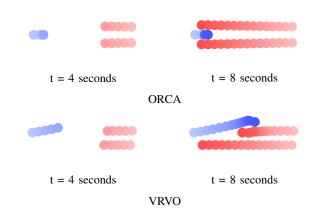

Fig. 9: Image illustrates a three agent scenario where two *red* agents travel from right to left, while the *blue* agent travels from left to right. The trajectory of the agents are shown using colored disks where the agent's positions for recent time steps are denoted by darker color shades. While using ORCA, the *blue* agent is stays at the same position for several time steps due to ORCA's conservative nature. In contrast, while using VRVO the *blue* agent deviates to avoid a collision.

time per agent is in the range $1 - 2$ms and the overall performance is almost linear in the number of agents

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a new decentralized method based on Voronoi diagrams and RVO that computes a collision-free path for agents operating in a shared workspace with other agents, static and dynamic obstacles. Our V-RVO algorithm provides passive collision avoidance guarantees and we demonstrates its performance on agents with first and second-order dynamics. We also presented a deadlock resolution strategy and demonstrates its benefits over prior velocity-obstacle method.

Our method has some limitations. Our approach can be conservative and not complete in terms of always finding a collision-free trajectory (if that exists). Furthermore, our method does not guarantee deadlock resolution, similar to prior decentralized methods. Our Voronoi and baBVC implementations are not optimized and can be accelerated. As a part of our future work, we plan to explore extending our deadlock resolutions and further improve the performance. We would also like to handle agents with non-holonomic constraints or imprecise state information (e.g., due to sensor errors).
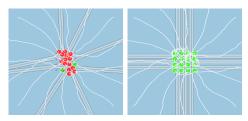
Fig. 10: We highlight the performance of V-RVO algorithm without deadlock avoidance (left) and with deadlock avoidance (right). The adjacent goal locations in the grid are 1.25m apart. The agents in *green* have reached there goal, while the agent in *red* are deadlocked, as shown with their white trajectories.
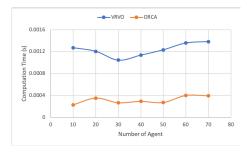

Fig. 11: Average time per agent taken by V-RVO vs ORCA, as we vary the number of agents between $5 - 70$. V-RVO has additional overhead of computing Voronoi regions and baBVC and is about $3X$ slower than ORCA.

References

[1] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," *arXiv preprint arXiv:2005.07371*, 2020.

[2] S. Tang and V. Kumar, "A complete algorithm for generating safe trajectories for multi-robot teams," in *Robotics Research*. Springer, 2018, pp. 599–616.

[3] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[4] M. Hamer, L. Widmer, and R. D'andrea, "Fast generation of collision-free trajectories for robot swarms using gpu acceleration," *IEEE Access*, vol. 7, pp. 6679–6690, 2019.

[5] K. Solovey, O. Salzman, and D. Halperin, "Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning," *The International Journal of Robotics Research*, vol. 35, no. 5, pp. 501–513, 2016.

[6] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer, "Enhanced partial expansion a," *Journal of Artificial Intelligence Research*, vol. 50, pp. 141–187, 2014.

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

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

[9] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, online collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[10] B. Davis, I. Karamouzas, and S. J. Guy, "Nh-ttc: A gradient-based framework for generalized anticipatory collision avoidance," *arXiv preprint arXiv:1907.05945*, 2019.

[11] 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.

[12] A. Best, S. Narang, and D. Manocha, "Real-time reciprocal collision avoidance with elliptical agents," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 298–305.

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

[14] J. Van Den Berg, D. Wilkie, S. J. Guy, M. Niethammer, and D. Manocha, "Lqg-obstacles: Feedback control with collision avoidance for mobile robots with motion and sensing uncertainty," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 346–353.

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

[16] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, *Optimal Reciprocal Collision Avoidance for Multiple Non-Holonomic Robots*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 203–216. [Online]. Available: https://doi.org/10.1007/978-3-642-32723-0_15

[17] S. Kim, S. J. Guy, W. Liu, D. Wilkie, R. W. Lau, M. C. Lin, and D. Manocha, "Brvo: Predicting pedestrian trajectories using velocity-space reasoning," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 201–217, 2015.

[18] L. He, J. Pan, and D. Manocha, "Efficient multi-agent global navigation using interpolating bridges," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4391–4398.

[19] H. Zhu and J. Alonso-Mora, "B-uavc: Buffered uncertainty-aware voronoi cells for probabilistic multi-robot collision avoidance," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 162–168.

[20] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4956–4961.

[21] Z. Forootaninia, I. Karamouzas, and R. Narain, "Uncertainty models for ttc-based collision-avoidance," in *Robotics: Science and Systems*, vol. 7, 2017.

[22] T. Fraichard and H. Asama, "Inevitable collision states. a step towards safer robots?" in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 1, 2003, pp. 388–393 vol.1.

[23] S. Bouraine, T. Fraichard, and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in unknown dynamic environments," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 174–179.

[24] K. Macek, D. A. V. Govea, T. Fraichard, and R. Siegwart, "Towards safe vehicle navigation in dynamic urban scenarios," 2009.