Fast, On-line Collision Avoidance for Dynamic Vehicles using Buffered Voronoi Cells

Dingjiang Zhou[1], Zijian Wang[2], Saptarshi Bandyopadhyay[3], and Mac Schwager[2]

***Abstract*—This paper presents a distributed collision avoidance algorithm for multiple dynamic vehicles moving in arbitrary dimensions. In our algorithm, each robot continually computes its buffered Voronoi cell (BVC) and plans its path within the BVC in a receding horizon fashion. We prove that our algorithm guarantees collision avoidance for robots with single integrator dynamics. We show that our algorithm has computational complexity of $O(k)$, which is the same as that of the Optimal Reciprocal Collision Avoidance (ORCA) algorithm, and is considerably faster than model predictive control (MPC) and sequential convex programming (SCP) based approaches. Moreover, ORCA and MPC-SCP require relative position, velocity, and even other information, to be exchanged over a communication network among the robots. Our algorithm only requires the sensed relative position, and therefore is well suited for on-line implementation as it does not require a communication network, and it works well with noisy relative position sensors. Furthermore, we provide an extension of our algorithm to robots with higher-order dynamics like quadrotors. We demonstrate the capabilities of our algorithm by comparing it to ORCA in multiple benchmark simulation scenarios, and we present results of over 70 experimental trials using five quadrotors in a motion capture environment.**

***Index Terms*—Distributed Robot Systems, Collision Avoidance, Motion and Path Planning**

I. Introduction

In this paper we present distributed, reactive algorithms for groups of dynamic robots to avoid collision with one another. The algorithm can be executed on-line by each robot, and only requires the robots to know the relative *positions* of neighboring robots. This is in contrast to existing methods, which typically require *position* and *velocity* information, or more detailed trajectory information, to be shared between neighboring robots. Our method is advantageous because velocity is typically more difficult to measure in practice. Although it is possible to numerically differentiate (or filter) position to estimate velocity, this amplifies noise, and the result is subject to numerical errors, and can induce time lag. Our algorithm has the same collision avoidance guarantee, and the same computational complexity, as the state-of-the-art reciprocal velocity obstacle (RVO) algorithms [1]–[3], and executes on-line with comparable speed. However, it does not require velocity information to be communicated, sensed, or inferred between robots. This makes the algorithm well suited to collision avoidance in dynamic, large-scale, ad hoc vehicle groups with relative position sensing, but no wireless communication network. The algorithm is also readily extendable to arbitrary dimensions, and can be applied to collision avoidance for groups of micro aerial vehicles in 3D, or autonomous cars and boats in 2D.

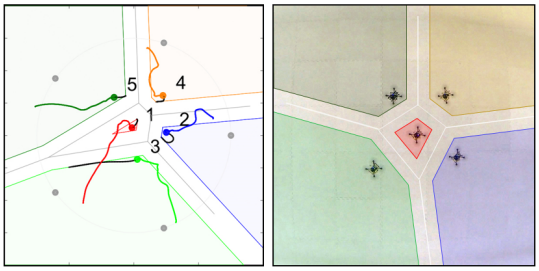


Fig. 1. Buffered Voronoi Cell used for collision avoidance.

To execute the algorithm, each robot computes its Voronoi cell, and plans a trajectory to the point in the cell that is closest to the robot's goal position. The robots update their Voronoi cells and re-plan their paths at each iteration, in a receding horizon fashion, until they reach their goal positions. Collision avoidance follows from a simple geometric principle—the Voronoi cells are disjoint from one another, thus ensuring that, if each robot stays entirely within its cell at the next time step, there cannot be a collision in all the future. The principle can be extended to heterogeneous robot groups, with each robot using a different policy to determine its motion.

We first introduce a *Buffered Voronoi Cell* (BVC), which retracts the edge of a Voronoi cell by a safety radius for one robot, so that if the robot's center point is in the BVC, its body will be entirely within the Voronoi cell, as illustrated in Fig. 1. We propose a receding horizon control approach to plan a path within the BVC while satisfying the relevant dynamic, positional, and input constraints. Furthermore, we show that this receding horizon controller can be solved analytically for a one step time horizon for very fast execution. We find that the one step horizon is well-suited for robots with integrator dynamics, while a multi-step horizon, is best suited for vehicles with more complex dynamics. We prove that collisions are avoided for both one step and multi-step algorithms for robots with integrator dynamics.

We show through three benchmark simulation scenarios that

Manuscript received: September 10, 2016; Revised December 4, 2016; Accepted January 16, 2017.

This work was supported in part by the SAIL-Toyota Center for AI Research, and NSF grant CNS-1330008. We are grateful for this support. Part of this research was carried out at the Jet Propulsion Laboratory (JPL), California Institute of Technology (Caltech), under a contract with the National Aeronautics and Space Administration (NASA).

[1]D. Zhou is with the Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA. zdj@bu.edu

[2]Z. Wang and M. Schwager are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305, USA. {zjwang, schwager}@stanford.edu

[3]S. Bandyopadhyay is with JPL, Caltech, Pasadena, CA 91109, USA. saptarshi.bandyopadhyay@jpl.nasa.gov

Digital Object Identifier (DOI): see top of this page.

our algorithm performs comparably to the Optimal Reciprocal Collision Avoidance (ORCA) algorithm from the RVO2 library [2], [4]. We also demonstrate the algorithm with higher order dynamics in simulations with eight quadrotors, and in experiments with five quadrotors in over 70 experimental trials in a motion capture environment.

There exists an enormous body of literature in the field of multi-robot collision avoidance. The most relevant class of algorithms in our context are the highly influential RVO methods, as seen in [1], [2], [5]–[7], in which a common assumption that the velocities of neighboring robots are known. One of the most popular and powerful variants of these methods is the ORCA algorithm [2], [4]. Specifically, we show that our algorithm has the same computational complexity and similar empirical performance, but does not require relative velocity between robots. Our method relies on the computation of Voronoi diagrams, which are widely used in other path planning and collision avoidance works, but typically the robots move along the edges of a static Voronoi diagram [8]–[11]. Instead, our use of Voronoi cells is inspired by coverage control algorithms such as [12], [13]. However, our algorithm seeks to drive the robots to goal positions without collisions, rather than to provide sensor coverage. A similar algorithm appeared in [14], where a modified Voronoi cell was used to avoid collisions within a larger probabilistic swarm. Sequential Convex Programming (SCP) is another tool used in the literature for planning collision free paths between multiple dynamic vehicles, e.g., in [15], [16], "near optimal" collision free trajectories for the vehicles were found for avoiding collision. In [17], [18], a decentralized MPC-SCP algorithm for spacecraft swarms was presented, and in [19], a model predictive control (MPC) algorithm using SCP was proposed to deal with formation flight for a large number of agents. Unfortunately, SCP methods are not well suited to fast on-line implementation due to high computational complexity and communication requirements. Many other methods exist for collision and obstacle avoidance, for example methods based on artificial potential fields [20], mixed integer linear programming (MILP) [21]–[23], mixed integer quadratic programming (MIQP) [24], to name a few. All of these methods come with advantages and drawbacks, but most are not suited to on-line, distributed implementation due to computation or communication demands.

II. Problem Formulation

We use a Voronoi tessellation to plan collision-free paths for a group of robots in real-time, updating the Voronoi cells and the paths at each time step of execution. The Voronoi cells naturally separate the robots, thus guaranteeing no collision among the robots. To count for the physical size of a robot, the boundary of its Voronoi cell needs to retreat by a safety radius, and we call it a *Buffered Voronoi Cell (BVC)*. In this section, we formally define our problem, and present some key properties of the BVC.

Suppose we have a group of N robots, with positions denoted as $\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N$, and we define a safety radius for all robots as r_s. When the group of robots are in a configuration with no collision, we call that a *collision free configuration*.

Definition 1 (Collision Free Configuration): A collision free configuration for the group of N robots with identical safety radius r_s is one where the distance between robot i and robot j satisfies $\|\mathbf{p}_i - \mathbf{p}_j\| \geq 2r_\mathrm{s},\ \forall i, j \in \{1, 2, \cdots, N\},\ i \neq j$.

A. Buffered Voronoi Cell

For a group of N robots in a 2D plane $\mathbb{R}^2$, the general Voronoi cell [25] of robot i is defined as

$$\mathcal{V}_i = \{\mathbf{p} \in \mathbb{R}^2 | \|\mathbf{p} - \mathbf{p}_i\| \leq \|\mathbf{p} - \mathbf{p}_j\|, \forall j \neq i\}, \quad (1)$$

where $\|\cdot\|$ denotes the Euclidean distance. Equivalently, we can write (1) as

$$\mathcal{V}_i = \{\mathbf{p} \in \mathbb{R}^2 | \big(\mathbf{p} - \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\big)^\mathrm{T} \mathbf{p}_{ij} \leq 0, \forall j \neq i\}, \quad (2)$$

where $\mathbf{p}_{ij} = \mathbf{p}_j - \mathbf{p}_i$. To consider the safety radius r_s of the robots, we define the Buffered Voronoi Cell.

Definition 2 (Buffered Voronoi Cell): For a group of N robots in a 2D plane $\mathbb{R}^2$ with a collision free configuration, the Buffered Voronoi Cell (BVC) of robot i is defined as

$$\bar{\mathcal{V}}_i = \{\mathbf{p} \in \mathbb{R}^2 | \big(\mathbf{p} - \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\big)^\mathrm{T} \mathbf{p}_{ij} + r_\mathrm{s}\|\mathbf{p}_{ij}\| \leq 0, \forall j \neq i\}. \quad (3)$$

Lemma 1: (Properties of BVC) If the group of N robots with safety radius r_s is in a collision free configuration, then $\forall i, j \in \{1, 2, \cdots, N\}$, we have, (i) $\bar{\mathcal{V}}_i \neq \emptyset$; (ii) $\bar{\mathcal{V}}_i \subset \mathcal{V}_i$; (iii) $\forall \mathbf{p}'_j \in \bar{\mathcal{V}}_j,\ i \neq j,\ \|\mathbf{p}'_i - \mathbf{p}'_j\| \geq 2r_\mathrm{s}$; and (iv) $\bar{\mathcal{V}}_i \cap \bar{\mathcal{V}}_j = \emptyset$, $\forall i \neq j$.

Proof: (i) If the group of N robots is in a collision free configuration, i.e., $\|\mathbf{p}_{ij}\| = \|\mathbf{p}_i - \mathbf{p}_j\| \geq 2r_\mathrm{s}$, we can plug the position of robot i, $\mathbf{p}_i$, into (3) in Definition 2,

$$\big(\mathbf{p}_i - \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\big)^\mathrm{T} \mathbf{p}_{ij} + r_\mathrm{s}\|\mathbf{p}_{ij}\| = \frac{1}{2}\mathbf{p}_{ji}^\mathrm{T}\mathbf{p}_{ij} + r_\mathrm{s}\|\mathbf{p}_{ij}\|$$
$$= -\frac{1}{2}\|\mathbf{p}_{ij}\|^2 + r_\mathrm{s}\|\mathbf{p}_{ij}\| \leq -\frac{1}{2}(2r_\mathrm{s})\|\mathbf{p}_{ij}\| + r_\mathrm{s}\|\mathbf{p}_{ij}\| = 0.$$

According to Definition 2, $\mathbf{p}_i \in \bar{\mathcal{V}}_i$, thus $\bar{\mathcal{V}}_i \neq \emptyset$ since $\bar{\mathcal{V}}_i$ has at least one element, i.e., $\mathbf{p}_i$.

(ii) According to (3) in Definition 2, $\forall \mathbf{p}'_i \in \bar{\mathcal{V}}_i$, we have

$$\big(\mathbf{p}'_i - \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\big)^\mathrm{T} \mathbf{p}_{ij} \leq -r_\mathrm{s}\|\mathbf{p}_{ij}\| < 0,$$

which also satisfies (2), hence $\mathbf{p}'_i \in \mathcal{V}_i$. Therefore, we have $\bar{\mathcal{V}}_i \subset \mathcal{V}_i$.

(iii) According to (3) again, $\forall \mathbf{p}'_i \in \bar{\mathcal{V}}_i$ and $\forall \mathbf{p}'_j \in \bar{\mathcal{V}}_j$,

$$\big(\mathbf{p}'_i - \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\big)^\mathrm{T} \mathbf{p}_{ij} + r_\mathrm{s}\|\mathbf{p}_{ij}\| \leq 0, \quad (4)$$

$$\big(\mathbf{p}'_j - \frac{\mathbf{p}_j + \mathbf{p}_i}{2}\big)^\mathrm{T} \mathbf{p}_{ji} + r_\mathrm{s}\|\mathbf{p}_{ji}\| \leq 0. \quad (5)$$

Since $\mathbf{p}_{ij} = -\mathbf{p}_{ji}$, $\|\mathbf{p}_{ij}\| = \|\mathbf{p}_{ji}\|$, the summation of (4) and (5) yields $(\mathbf{p}'_i - \mathbf{p}'_j)^\mathrm{T}\mathbf{p}_{ij} \leq -2r_\mathrm{s}\|\mathbf{p}_{ij}\|$, then by using the fact that $\|\mathbf{a}\| \cdot \|\mathbf{b}\| \geq \|\mathbf{a}^\mathrm{T}\mathbf{b}\|$, or $\|\mathbf{a}\| \geq \|\mathbf{a}^\mathrm{T}\mathbf{b}\|/\|\mathbf{b}\|$, we conclude that

$$\|\mathbf{p}'_i - \mathbf{p}'_j\| \geq \frac{\|(\mathbf{p}'_i - \mathbf{p}'_j)^\mathrm{T}\mathbf{p}_{ij}\|}{\|\mathbf{p}_{ij}\|} \geq \frac{2r_\mathrm{s}\|\mathbf{p}_{ij}\|}{\|\mathbf{p}_{ij}\|} = 2r_\mathrm{s}.$$

(iv) For any $\bar{\mathcal{V}}_i$ and $\bar{\mathcal{V}}_j$, $i \neq j$, consider an arbitrary point $\mathbf{p}'_i \in \bar{\mathcal{V}}_i$, we prove that $\mathbf{p}'_i \notin \bar{\mathcal{V}}_j$, $\forall j \neq i$. From (4), we have,

$$\big(\mathbf{p}'_i - \frac{\mathbf{p}_i + \mathbf{p}_j}{2}\big)^\mathrm{T} \mathbf{p}_{ij} \leq -r_\mathrm{s}\|\mathbf{p}_{ij}\|,$$

and plug $\mathbf{p}_i'$ into the definition of $\bar{\mathcal{V}}_j$ as in (3), we have

$$\begin{aligned}&\big(\mathbf{p}_i' - \frac{\mathbf{p}_j+\mathbf{p}_i}{2}\big)^{\mathrm{T}}\mathbf{p}_{ji} + r_{\mathrm{s}}\|\mathbf{p}_{ji}\|\\ \geq& r_{\mathrm{s}}\|\mathbf{p}_{ij}\| + r_{\mathrm{s}}\|\mathbf{p}_{ij}\| = 2r_{\mathrm{s}}\|\mathbf{p}_{ij}\|,\end{aligned}$$

which contradicts to the definition of $\bar{\mathcal{V}}_j$ in (3). Moreover, for an arbitrary point $\mathbf{p}_j' \in \bar{\mathcal{V}}_j$, we have $\mathbf{p}_j' \notin \bar{\mathcal{V}}_i$. Hence we conclude that $\bar{\mathcal{V}}_i \cap \bar{\mathcal{V}}_j = \emptyset$, $\forall i \neq j$. ■

III. BVC Collision Avoidance Algorithms

In this section, we start with describing our on-line receding horizon algorithm based on quadratic programming (QP) for a robot with general linear dynamics.

A. QP Based Collision Avoidance Algorithm

At each time step, the robot computes its BVC, plans a path within the BVC, then executes the first step along that path. The robot position, BVC, and planned path evolve together throughout the execution of the algorithm. Assuming that the planning horizon is T steps, and for robot i at each time instance, we denote the positions of the planned trajectory as $\bar{\mathbf{p}}_{i,1}$, $\bar{\mathbf{p}}_{i,2}$, $\cdots$, $\bar{\mathbf{p}}_{i,T}$, $\forall i \in \{1,2,\cdots,N\}$. The algorithm requires each robot to solve a QP at each time step, e.g., for robot i, the path and input (or velocity) are given by solving the following QP problem,

Problem 1 (QP Receding Horizon Path Planning):

$$\begin{aligned}\underset{\bar{\mathbf{p}}_1,\cdots,\bar{\mathbf{p}}_T}{\text{minimize}}\ \mathbf{J}_i = &\sum_{t=0}^{T-1}\Big((\bar{\mathbf{p}}_{i,t}-\mathbf{p}_{i,f})^{\mathrm{T}}\mathbf{Q}(\bar{\mathbf{p}}_{i,t}-\mathbf{p}_{i,f})\\ &+\mathbf{u}_{i,t}^{\mathrm{T}}\mathbf{R}\mathbf{u}_{i,t}\Big) + (\bar{\mathbf{p}}_{i,T}-\mathbf{p}_{i,f})^{\mathrm{T}}\mathbf{Q}_{\mathrm{f}}(\bar{\mathbf{p}}_{i,T}-\mathbf{p}_{i,f}),\end{aligned} \tag{6}$$

subject to

$$\bar{\mathbf{p}}_{i,t+1} = \mathbf{A}\bar{\mathbf{p}}_{i,t} + \mathbf{B}\mathbf{u}_{i,t},\ t=0,\cdots,T-1, \tag{7}$$

$$\bar{\mathbf{p}}_{i,t} \in \bar{\mathcal{V}}_i,\ t=1,\cdots,T, \tag{8}$$

$$\bar{\mathbf{p}}_{i,0} = \mathbf{p}_i, \tag{9}$$

$$\|\mathbf{u}_{i,t,x}\| \leq u_{x,\max},\ t=0,\cdots,T-1, \tag{10}$$

$$\|\mathbf{u}_{i,t,y}\| \leq u_{y,\max},\ t=0,\cdots,T-1. \tag{11}$$

In Problem 1, the cost function $\mathbf{J}_i$ is a summation of intermediate state costs, input costs and a terminal cost. In (6), $\mathbf{p}_{i,f}$ is the goal position, $\bar{\mathbf{p}}_{i,0}$ to $\bar{\mathbf{p}}_{i,T}$ and $\mathbf{u}_{i,0}$ to $\mathbf{u}_{i,T-1}$ are the path, and inputs to be planned, respectively. The positive definite or semidefinite matrices $\mathbf{Q}$, $\mathbf{R}$, and $\mathbf{Q}_{\mathrm{f}}$ are weight factors to balance among the three costs. The decision variables for this standard QP problem are $\bar{\mathbf{p}}_{i,1}$ to $\bar{\mathbf{p}}_{i,T}$. The constraint (7) ensures that the path is feasible with the robot's dynamics. The constraint (8) restrains the planned path to be in the corresponding BVC of robot i. As (3) in Definition 2, (8) can be written explicitly in the form of a set of linear inequality constraint,

$$\bar{\mathbf{p}}_{i,t}^{\mathrm{T}}\boldsymbol{\epsilon}_j \leq 0,\ t=1,\cdots,T,\ j=1,2,\cdots,N,\ j\neq i, \tag{12}$$

where $\boldsymbol{\epsilon}_j$ is a 3×1 vector, corresponding to an edge of the BVC $\bar{\mathcal{V}}_i$. Constraint (9) makes sure the path starts from the robot's current position, and the lower and upper bounds for the input $\mathbf{u}_{i,t}$ are written component-wisely in (10) and (11).

B. Analytical Geometric Algorithm

The QP receding horizon path planning in Sec. III-A generates a control policy that is optimal over the planning horizon at the expense of solving a QP problem on-line at each time step. For the special case of an integrator robot with no intermediate cost terms, we can analytically solve the QP with a geometric algorithm which executes much faster than the QP, while collision avoidance is still guaranteed.

Consider the case where we only have the terminal cost, and neglect the intermediate state and the control input costs, i.e., we let $\mathbf{Q}=\mathbf{0}$, $\mathbf{R}=\mathbf{0}$, and keep $\mathbf{Q}_{\mathrm{f}}$ positive definite in (6). The constraints (7) to (11), however, are the same. This simplification can be regarded as a one-step greedy strategy that drives the robot to move to its goal position as soon as possible. With this simplification, the robot should move towards a point in the BVC that is closest to its goal position (which may be inside or outside the BVC at any time instant). This geometric special case is similar to an algorithm that first appeared in [14] for avoiding collisions as a part of larger swarm guidance algorithm. A naive method to find a closest point in a polygon is to check each edge and each vertex for a minimum, however, we present a novel approach to achieve the same goal here. The algorithm for finding this closest point in the BVC is outlined in Proposition 1 and Algorithm 1, by exploring some properties of Euclidean geometry.

Proposition 1: Let $\mathcal{V}=(\mathcal{E},\mathbf{e})$ represent a convex polygon in $\mathbb{R}^2$, where $\mathcal{E}$ is the set of edges and $\mathbf{e}$ is the set of vertices. For any point $\mathbf{g}\in\mathbb{R}^2$, the closest point $\mathbf{g}^*\in\mathcal{V}$ to $\mathbf{g}$ is either $\mathbf{g}$ itself, or on an edge $\mathcal{E}^*$ of $\mathcal{V}$, or is a vertex $\mathbf{e}^*$ of $\mathcal{V}$.

Algorithm 1 Finding Closest Point

Require: $\mathbf{g}$, $\mathcal{V}=(\mathcal{E},\mathbf{e})$
  $d_{\min}\leftarrow+\infty$, $\mathbf{g}^*\leftarrow\emptyset$
  for $\mathcal{E}_i$ **in** $\mathcal{E}$ **do**
    Let $\mathbf{g}_1$ and $\mathbf{g}_2$ represent the two vertices of $\mathcal{E}_i$
    $\theta_i = \mathrm{acos}\Big(\frac{(\mathbf{g}_1-\mathbf{g})^{\mathrm{T}}(\mathbf{g}_2-\mathbf{g})}{\|\mathbf{g}_1-\mathbf{g}\|\|\mathbf{g}_2-\mathbf{g}\|}\Big)$, $\lambda_i = -\frac{(\mathbf{g}_1-\mathbf{g})^{\mathrm{T}}(\mathbf{g}_2-\mathbf{g})}{\|\mathbf{g}_1-\mathbf{g}_2\|^2}$
    if $0\leq\lambda_i\leq 1$ **then**
      $\mathbf{g}_p = (1-\lambda_i)\mathbf{g}_1+\lambda_i\mathbf{g}_2$
      if $d_{\min}>\|\mathbf{g}_p-\mathbf{g}\|$ **then**
        $d_{\min}=\|\mathbf{g}_p-\mathbf{g}\|$, $\mathbf{g}^*\leftarrow\mathbf{g}_p$
      end if
    else
      if $d_{\min}>\|\mathbf{g}_1-\mathbf{g}\|$ **then**
        $d_{\min}=\|\mathbf{g}_1-\mathbf{g}\|$, $\mathbf{g}^*\leftarrow\mathbf{g}_1$
      end if
      if $d_{\min}>\|\mathbf{g}_2-\mathbf{g}\|$ **then**
        $d_{\min}=\|\mathbf{g}_2-\mathbf{g}\|$, $\mathbf{g}^*\leftarrow\mathbf{g}_2$
      end if
    end if
  end for
  if $\sum_{i=0}^{k}\theta_i = 2\pi$ **then**
    $d_{\min}=0$, $\mathbf{g}^*\leftarrow\mathbf{g}$
  end if
  return $\mathbf{g}^*$

In our formulation, for robot i, once the closet point $\mathbf{g}_i^*\in\bar{\mathcal{V}}$ to the goal position $\mathbf{p}_{i,f}$ is found, the control input $\mathbf{u}_{i,0}$ is calculated to move the robot i toward $\mathbf{g}^*$ at that time step.

In practice, our polygon is expressed as a group of linear inequalities as (12), hence $\bar{\mathcal{V}}_i$ is always convex. The computational complexity of Algorithm 1 increases as the number of Voronoi neighbors of the robot increases. In Algorithm 1, we iterate on each edge, and calculate an angle from $\mathbf{g}$ to the two vertices and a ratio. If the summation of all the angles is 2π, then $\mathbf{g}^* = \mathbf{g}$ is in the polygon $\mathcal{V}$, otherwise, $\mathbf{g}^*$ should either be a vertex, or on one edge, depending on the value of λ_i.

The key to increase the computational efficiency is to decrease the number of edges of the BVC of one robot, as seen from the above analysis. For typical Voronoi configurations, the number of Voronoi neighbors is small. Also, in practice a robot's sensing range will be limited, in which case only the robots that are close enough will be treated as neighbors, and corresponding edges will be created for its BVC.

C. Collision Avoidance Guarantee

For single integrator robots, the following Theorem illustrates that once the group of robots is in a collision free configuration, the robots will be in a collision free configuration for all time in the future, applying either of the on-line path planning algorithms in Sections III-A or III-B.

Theorem 1: If the group of N robots is in a collision free configuration, i.e., the initial positions $\mathbf{p}_{1,0}$, $\mathbf{p}_{2,0}$, $\cdots$, $\mathbf{p}_{N,0}$ satisfy $\|\mathbf{p}_{i,0} - \mathbf{p}_{j,0}\| \geq 2r_\mathrm{s}$, $\forall i \neq j$, then all future positions $\mathbf{p}_{1,k}$, $\mathbf{p}_{2,k}$, $\cdots$, $\mathbf{p}_{N,k}$, $k \in \mathbb{Z}^+$, are in a collision free configuration, i.e., $\|\mathbf{p}_{i,k} - \mathbf{p}_{j,k}\| \geq 2r_\mathrm{s}$, $\forall i \neq j$, if the control input is obtained by solving Problem 1.

Proof: We show (i) that a collision free configuration leads to a feasible QP, and (ii) that a one step execution of the resulting path leads to a new collision free configuration. These two facts applied in a mathematical induction and prove the theorem. Since the N robots are initially in a collision free configuration, according to (1) in Lemma 1, they have non-empty BVCs, $\bar{\mathcal{V}}_i \neq \emptyset$ and $\mathbf{p}_{i,0} \in \bar{\mathcal{V}}_i$, $\forall i \in \{1, 2, \cdots, N\}$. If $\mathbf{u}_{i,k} = 0$, $\forall i$, $\forall k$, which satisfies the constraints (10) and (11), then we know $\mathbf{p}_{i,t} = \mathbf{p}_{i,0} \in \bar{\mathcal{V}}_i$, $\forall i$, $t = 1, 2, \cdots, T$. Therefore, the Problem 1 is always feasible. Then, according to (3) in Lemma 1, $\forall i \neq j$, $\|\mathbf{p}_{i,t_i} - \mathbf{p}_{j,t_j}\| \geq 2r_\mathrm{s}$, $\forall t_i, t_j = 1, 2, \cdots, T$, which indicates that the paths computed by solving Problem 1 will not lead to an intersection. In particular, executing one step of the path leaves the robots in a collision free configuration. Hence the new BVC is non-empty, and the program at the next step is again feasible. By mathematical induction, our algorithms in Sec. III-A, and the special case in Sec. III-B, guarantee no collision among the robots for all time. ■

Remark 1: Note that the above result only requires that each robot stays in its BVC at the next time step. It does not matter what specific control policy the robots use. Hence the method can guarantee collision avoidance in heterogeneous groups, where each robot uses a different control policy. Indeed, the robots need not even know what the other robots' control polices are, which is in contrast to most other reciprocal collision avoidance strategies.

D. Dealing with Deadlock

Deadlock is a ubiquitous problem in distributed collision avoidance algorithms. Deadlock happens when some robots block each others' paths in such a way that at least one robot cannot reach its goal using its control algorithm. To our best knowledge, there is no existing algorithm can provably avoid deadlock without central computation or global coordination. Most distributed algorithms attempt to alleviate the problem through sensible heuristics. Similarly, here we propose two heuristic solutions that perform well in practice to alleviate deadlock phenomena.

First we propose a *right-hand rule* (or *left-hand rule*), a preventative strategy used before the deadlock happens, assuming the robots are in a 2D plane. By using the right-hand rule, each robot always chooses to detour from its right side when encountering other robots. Practically, a new constraint can be introduced into the QP problem. The simulation in Sec. IV-B applies the right-hand rule, for all robots at all time during the simulation.

Our second solution exploits the necessary condition of deadlock, and is used after the deadlock happens. As a result, if the robots can avoid satisfying the necessary condition, the deadlock can be prevented.

Proposition 2: For those robots whose goal positions are not inside their own BVCs, when in a deadlock configuration, each robot must be at the closest point to the goal position on its BVC. The closest point in the BVC of robot i, $\mathbf{g}_i^*$, to the goal $\mathbf{p}_{i,f}$, must be either (a) at a vertex, or (b) on an edge such that a line from $\mathbf{p}_{i,f}$ to $\mathbf{g}_i^*$ is perpendicular to this edge.

Proof: (Abbreviated) Deadlock only occurs when a robot is already located at $\mathbf{g}_i^*$, because otherwise the robot can move toward $\mathbf{g}_i^*$ for at least one step, meaning it is not a deadlock. The case (a) and (b) are self-evident given Proposition 1 and Algorithm 1. ■

Case (b) in Proposition 2, can be easily avoided because the perpendicular condition also requires the neighboring robot who shares the Voronoi edge to be on the line segment from $\mathbf{p}_{i,f}$ to $\mathbf{g}_i^*$. This condition is broken if the robot deviates to its right or left side a little to avoid the co-linear configuration. Therefore, we only aim to avoid case (a), i.e., staying at a vertex, when deadlock happens. Our heuristic solution is that when deadlock is detected and the robot is at the vertex, it then chooses one of its adjacent edges to detour along. In this way, the necessary condition of deadlock can be avoided, and furthermore, the deadlock can be avoided.

Note that the solution described here cannot provably lead to all robots reaching their goals, because it can lead to the so-called *livelock* phenomenon, where robots oscillate indefinitely between a deadlock configuration and a deadlock avoidance behavior. However, we rarely observe this phenomenon in simulations or experiments.

E. Extension to 3D Space and Higher-Order Dynamics

Our algorithm can be naturally extended to 3D space, as long as the BVCs are expressed in 3D. Similar to the 2D case, the 3D BVC can also be written explicitly as a set of linear inequalities as in the form of (12), except that $\boldsymbol{\epsilon}_j$ is a 4×1 vector expressing a face of a BVC.

For higher order dynamical systems with input limits, one can always find a collision free configuration that will ultimately lead to collision (regardless of the collision avoidance algorithm). Consider two robots approaching each other at

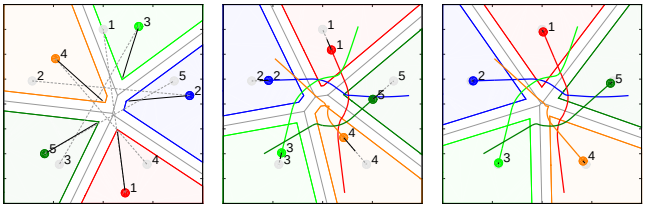
high speed at time t, with second or higher order dynamics, they will require a certain distance, $d_{\rm s} > r_{\rm s}$, to stop. Hence, there will be a collision free separation $2r_{\rm s} \leq \|\mathbf{p}_{i,t} - \mathbf{p}_{j,t}\| < 2d_{\rm s}$ for which no input will avoid a collision at some $t+\tau$ in the future, $\|\mathbf{p}_{i,t+\tau} - \mathbf{p}_{j,t+\tau}\| \leq 2r_{\rm s}$. However, we propose a heuristic to avoid this problem in practice. We set a safety radius $r_{\rm s}$ larger than the physical size of the robot, such that even if the safety radius is violated (due to the high order dynamics of the robot, or system noise), the physical collision will not occur. A small modification to Problem 1 can naturally account for this. We remove the initial condition constraint $\bar{\mathbf{p}}_{i,0} = \mathbf{p}_i$, and include it as part of the quadratic cost instead. Hence, if robot i is beyond its BVC by a small distance because of noise, momentum, etc., our algorithm can compute a path to bring it back to its BVC.

To illustrate the algorithm with higher order dynamics, we show a simulation with eight quadrotors with full nonlinear dynamics in 3D in Sec. IV-C, and experiments with five quadrotor robots in 2D in Sec. VI.

IV. Numerical Simulations

A. Simulation with QP Path Planning Algorithm

We first simulate five robots with single integrator dynamics in a 2D plane in MATLAB, with the QP solver implemented with CVXGEN [26], [27]. Simulation results are shown in Fig. 2. The five robots as well as their BVCs are plotted in same colors. The robots are initially distributed around the perimeter of a circle with a little random offset to break symmetry. The goal positions are also placed around the same circle. The dashed gray lines, which obviously intersect with each other, are the straight lines from the robots' current positions to their goal positions. The thick colored lines are the executed trajectories, and the thick dark lines are the planned paths from our algorithm. In our simulation, the system time step is $\Delta t = 0.25$ second, and the planning horizon is $T = 20$. The safety radius is $r_{\rm s} = 0.2$ meter, and the minimum distance among the robots during the simulation is $d_{\min} = 0.4$ meter.

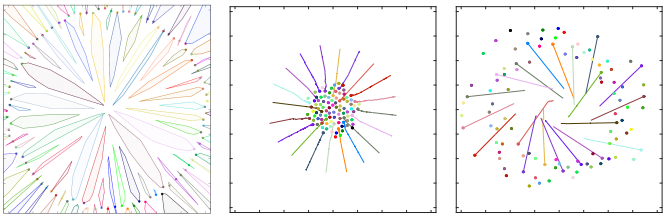


(a) Initial positions. (b) At step 160. (c) At step 240.

Fig. 2. Simulation with five robots with single integrator dynamics using the QP path planning algorithm.

B. Simulation with Geometric Algorithm

Our distributed algorithm is scalable to an arbitrary number of robots. In this section, we present a similar simulation as in Sec. IV-A with the number of robots increased to 100, while the control input is from the geometric algorithm as we described in Sec. III-B, and the right-hand rule is used to avoid deadlock. As in Sec. IV-A, the 100 robots are first distributed evenly on a circle with a radius of 20 meters, with a small random offset to break symmetry, as shown in Fig. 3(a). The BVCs for all robots are plotted in the same colors of the robots. Fig. 3(b) and 3(c) are the robots' configurations at step 160 and 480, respectively. The last 100 steps of the executed trajectories of only a few robots are plotted in order not to obscure the figures.

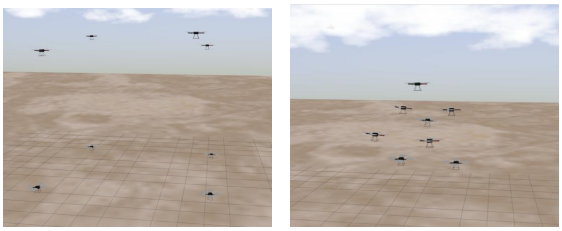


(a) Initial Configuration. (b) At step 160. (c) At step 480.

Fig. 3. Simulation with 100 robots with single integrator dynamics move to their goal configuration using the geometric algorithm from Sec. III-B.

C. Simulation in 3D Space with Quadrotors

We successfully validated our algorithm in 3D space using eight quadrotors in Gazebo, a well-known robotics simulator using the Open Dynamic Engine (ODE) as the underlying physics engine. To ensure the fidelity of the simulated quadrotor dynamics, we use the *hector quadrotor* package [28] in the Robot Operating System (ROS) that models the quadrotor quite comprehensively. The snapshots of the test scenario are shown in Fig. 4. The eight quadrotors, initially positioned at the corners of a 3D cube, are required to navigate to the corresponding diagonal corners. We use the geometric algorithm (Sec. III-B and Algorithm 1) on the 3D BVC (Sec. III-E), and the quadrotors are iteratively controlled to move to the newest closest points found by the geometric algorithm, at each time step. Our algorithm successfully generates reactive 3D paths in real-time for the quadrotors to avoid collision to each other.



(a) Initial configuration. (b) Snapshot of the process.

Fig. 4. Simulation with eight quadrotors in Gazebo. The goal position of one quadrotor is to move from one corner to the corresponding diagonal corner of a 3D cube.

V. Comparison with Other Collision Avoidance Algorithms

We note that SCP-based algorithms will not perform competitively against neither our algorithm nor ORCA, because they require multiple steps of convex programming [15], [16] be performed in a loop until some criteria are reached, and many variants require the exchange of entire planned paths between robots at each time step [17], [18].

Hence, in this section we mainly compare the computation performance of our algorithm to the most promising existing algorithms, ORCA [2], [4]. Our algorithm is position-based, requiring only position information from neighboring robots, while the ORCA algorithm is velocity-based, requiring both position and velocity information. Because of this difference, our algorithm is more suitable and robust for implementation when robots have only on-board sensors, especially when there is no communication network, or the communication capability is limited. Proximity sensors such as laser range finders, sonar sensors, etc., have considerable noise on position sensing, and then velocity estimated from these sensors will be significantly corrupted, hence in this case, our algorithms, i.e., position-based algorithms, are more suitable.

A. Computational Complexity

We compare the computational complexity between our algorithms and the ORCA algorithm for each robot, assuming that each robot has found the k neighboring robots in its vicinity. The ORCA algorithm takes $O(k)$ time to create ORCA lines from the k neighbors. Then, it solves a velocity in $O(k)$ time by adding the constraints one by one in random order while keeping track of the current optimal new velocity [2]. Therefore, the overall computational complexity is $O(k)$ for the ORCA algorithm. In our QP based receding horizon algorithm, constructing a BVC takes $O(k)$ time, while solving a standard QP problem takes $O(m^3)$ time, where m is the number of decision variables, and it is linear to the number of neighboring robots, i.e., k, as well as the planning horizon T. Hence the total computational complexity is $O(k^3)$ for one robot, worse than that of the ORCA algorithm. On the other hand, with our geometric solution in Algorithm 1, we see that the complexity is apparently $O(k)$, i.e., comparable to the ORCA algorithm.

B. Case Study

In this section, we compare the performance of our algorithms versus the ORCA algorithm, using three cases with 100 robots. The first case is the same with that in Sec. IV-B and Fig. 3. The second and third cases are to swap positions with 100 robots in 2D plane, as in Fig. 5. The three cases represent particularly challenging collision avoidance problems, and are considered as performance benchmarks. Our algorithm is implemented in MATLAB, while the ORCA algorithm for these cases is implemented in C++ with RVO2 library [4]. We set the same parameters for all simulations. The execution steps for our benchmarks are listed in Table I.

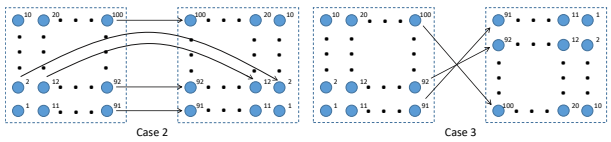


Fig. 5. Case 2 and 3 for performance benchmarks.

As a summary, our geometric solution gives a performance comparable to the ORCA algorithm. Though the QP based RHC (or MPC) path planning algorithm has a slower convergence rate, but for dealing with high order dynamics, we believe it has its advantages (See Sec. VI).

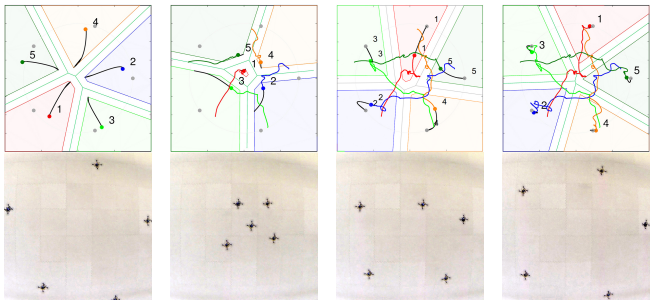
TABLE I
EXECUTION STEPS FOR THE THREE CASES.

	QP algorithm	Geometric algorithm	ORCA algorithm
Case 1	755	569	612
Case 2	360	355	361
Case 3	1670	704	509

VI. EXPERIMENTS WITH MICRO AERIAL VEHICLES

In this section, we validate our QP based RHC path planning algorithm in experiments with five quadrotors in an OptiTrack motion capture environment. The quadrotors are controlled to remain in a horizontal plane, and use our algorithm in 2D plane to navigate in the plane. The low-level control is implemented in MATLAB, in the same framework as described in our previous work [29]–[31].

In our experiments, we choose the QP receding horizon algorithm, instead of the geometric algorithm, as we find it to be better suited to vehicles with complex dynamics, and to show that it is still fast enough for real-time collision avoidance. We implement our algorithm with CVXGEN with a planning horizon $T = 30$, and a system time step $\Delta t = 0.25$ second. We let the initial and final configurations be the same as with that in the simulation in Sec. IV-A. We conducted a total of 79 experimental trials without collisions or deadlocks before losing one quadrotor. Fig. 6 shows a typical trial in our experiments, that the five quadrotors reach their goal configuration in 30 seconds. Due to system noise and aerodynamic effects, the trajectories are not as smooth as that in our simulation, however, it certainly proves the robustness of our algorithm.

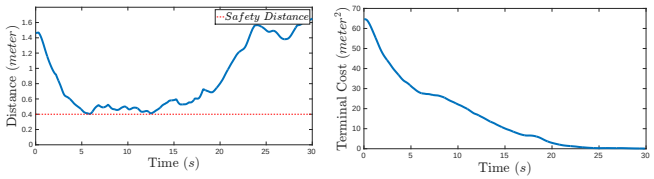


(a) At 0 second. (b) At 10 second. (c) At 20 second. (d) At 30 second.

Fig. 6. Experiment with five micro aerial vehicles reconfiguring their formation, using our QP based on-line receding horizon path planning algorithm.

We set the safe radius of our quadrotors in our experiment as $r_s = 0.2$ meter to count for noise, aerodynamics, and higher order dynamical effects. During one example experiment, the minimal distance between the quadrotors stays larger than the safety distance (0.4 meter) all the time, as shown in Fig. 7(a). The total terminal cost of the five quadrotors eventually converges to zero, as shown in Fig. 7(b), indicating the arrivals of the robots at their goal configuration.

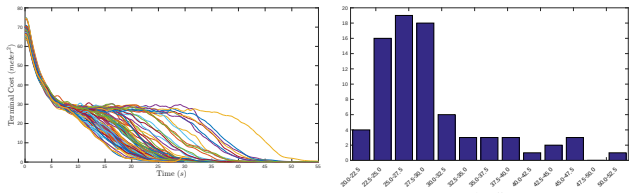
Fig. 8(a) shows that the total terminal cost of all quadrotors monotonically decreases over time from all 79 experimental trials. The histogram of time for achieving convergence to the goal configuration is shown in Fig. 8(b), in which the average



(a) Minimum distance. (b) Terminal costs for all robots.

Fig. 7. Algorithm performance in an experiment trial.

convergence time is 29.3 seconds, and the standard deviation is 6.4 seconds.



(a) Summation of terminal cost. (b) Histogram of execution time.

Fig. 8. Statistics of 79 experimental trials.

VII. Conclusion

In this paper, we introduce efficient distributed algorithms for a group of dynamic vehicles to avoid collision with one another, based on the Buffered Voronoi Cell. Our algorithm gives comparable performance to the existing RVO, or ORCA algorithm, and collision avoidance is guaranteed among the robots, however it requires only position information from neighboring robots. In the future, we will consider static obstacles in the environment, and will pursue an experimental implementation with on-board computation and sensing.

References

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

[2] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *The 14th International Symposium of Robotics Research (ISRR)*. Springer, 2011, pp. 3–19.

[3] A. Giese, D. Latypov, and N. M. Amato, "Reciprocally-rotating velocity obstacles," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 3234–3241.

[4] J. van den Berg, S. J. Guy, J. Snape, M. C. Lin, and D. Manocha, "Rvo2 library: Reciprocal collision avoidance for real-time multi-agent simulation," Available at http://gamma.cs.unc.edu/RVO2/, 2016.

[5] J. Alonso-Mora, A. Breitenmoser, P. Beardsley, and R. Siegwart, "Reciprocal collision avoidance for multiple car-like robots," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, May 2012, pp. 360–366.

[6] D. Wilkie, J. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, October 2009, pp. 5573–5578.

[7] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, August 2011.

[8] S. A. Bortoff, "Path planning for uavs," in *Proceedings of the 2000 American Control Conference (ACC)*, vol. 1, no. 6. IEEE, 2000, pp. 364–368.

[9] P. Bhattacharya and M. L. Gavrilova, "Roadmap-based path planning-using the voronoi diagram for a clearance-based shortest path," *IEEE Robotics Automation Magazine*, vol. 15, no. 2, pp. 58–66, 2008.

[10] S. Garrido, L. Moreno, and D. Blanco, "Voronoi diagram and fast marching applied to path planning," in *2006 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2006, pp. 3049–3054.

[11] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path planning for mobile robot navigation using voronoi diagram and fast marching," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, October 2006, pp. 2376–2381.

[12] J. Cortés, S. Martínez, T. Karatas, and F. Bullo, "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, April 2004.

[13] L. C. Pimenta, V. Kumar, R. C. Mesquita, and G. A. Pereira, "Sensing and coverage for a network of heterogeneous robots," in *2008 IEEE 47th Conference on Decision and Control (CDC)*. IEEE, December 2008, pp. 3947–3952.

[14] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic swarm guidance using optimal transport," in *2014 IEEE Conference on Control Applications (CCA)*. IEEE, October 2014, pp. 498–505.

[15] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: science and systems*, vol. 9, no. 1. Citeseer, 2013, pp. 1–10.

[16] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research (IJRR)*, vol. 33, no. 9, pp. 1251–1270, 2014.

[17] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Decentralized model predictive control of swarms of spacecraft using sequential convex programming," in *Procedings of the 23rd AAS/AIAA Space Flight Mechanics Meeting*, Kauai, Hawaii, 2013, pp. 1–20.

[18] ——, "Model predictive control of swarms of spacecraft using sequential convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 6, pp. 1725–1740, 2014.

[19] D. Morgan, G. P. Subramanian, S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic guidance of distributed systems using sequential convex programming," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, September 2014, pp. 3850–3857.

[20] D. E. Chang, S. C. Shadden, J. E. Marsden, and R. Olfati-Saber, "Collision avoidance for multiple agent systems," in *2003 IEEE 42th Conference on Decision and Control (CDC)*, Maui, Hawaii, 2003, pp. 539–543.

[21] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *2001 European Control Conference (ECC)*. IEEE, September 2001, pp. 2603–2608.

[22] A. Richards, T. Schouwenaars, J. P. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[23] J. How, E. King, and Y. Kuwata, "Flight demonstrations of cooperative control for uav teams," in *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, Chicago, Illinois, September 2004, pp. 20–23.

[24] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2012, pp. 477–483.

[25] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*. John Wiley & Sons, 2009, vol. 501.

[26] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control," *IEEE Control Systems*, vol. 31, no. 3, pp. 52–65, 2011.

[27] J. Mattingley and S. Boyd, "Cvxgen: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[28] J. Meyer, A. Sendobry, S. Kohlbrecher, U. Klingauf, and O. Von Stryk, "Comprehensive simulation of quadrotor uavs using ros and gazebo," in *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2012, pp. 400–411.

[29] D. Zhou and M. Schwager, "Vector field following for quadrotors using differential flatness," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014, pp. 6567–6572.

[30] ——, "Virtual rigid bodies for coordinated agile maneuvering of teams of micro aerial vehicles," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2015, pp. 1737–1742.

[31] ——, "Assistive collision avoidance for quadrotor swarm teleoperation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016, pp. 1249–1254.