Reciprocal Multi-Robot Collision Avoidance with Asymmetric State Uncertainty

Kunal Shah* k2shah@stanford.edu
Guillermo Angeris* angeris@stanford.edu
Mac Schwager schwager@stanford.edu

May 2021

Abstract

We present a general decentralized formulation for a large class of collision avoidance methods and show that all collision avoidance methods of this form are guaranteed to be collision free. This class includes several existing algorithms in the literature as special cases. We then present a particular instance of this collision avoidance method, CARP (Collision Avoidance by Reciprocal Projections), that is effective even when the estimates of other agents' positions and velocities are noisy. The method's main computational step involves the solution of a small convex optimization problem, which can be quickly solved in practice, even on embedded platforms, making it practical to use on computationally-constrained robots such as quadrotors. This method can be extended to find smooth polynomial trajectories for higher dynamic systems such at quadrotors. We demonstrate this algorithm's performance in simulations and on a team of physical quadrotors. Our method finds optimal projections in a median time of 17.12ms for 285 instances of 100 randomly generated obstacles, and produces safe polynomial trajectories at over 60hz on-board quadrotors. Our paper is accompanied by an open source Julia implementation and ROS package.

1 Introduction

Reliable collision avoidance is quickly becoming a mainstay requirement of any scalable mobile robotics system. As robots continue to be deployed around humans, assurances of safety become more critical, especially in high traffic areas such as factory floors and hospital corridors. We define a class of distributed collision avoidance methods, known as the reciprocally safe methods, which we prove are guaranteed to be collision free by construction. This class contains a number of well-known, published algorithms, providing an alternative proof of collision avoidance. We then present a special case of this class that allows a group of robots to avoid colliding with one another, even when each robot

*These authors contributed equally.

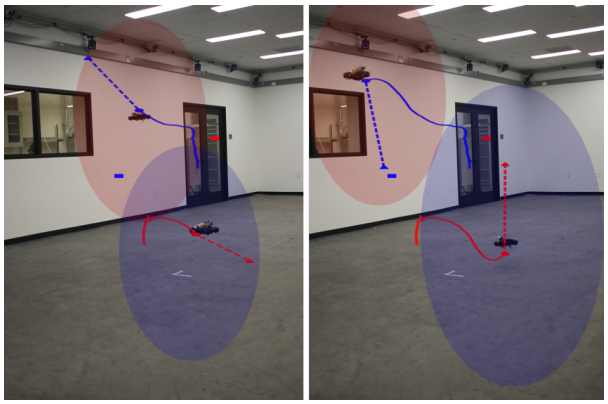


Figure 1: Two quadrotors executing a reciprocal collision avoidance maneuverer each agent (red, blue) maintains an ellipsoidal estimate of the other's position. The agents move towards their goal point (square) projecting (triangle) it into their safe-reachable set. The solid curve shows the trajectory history and the dotted curve shows the path to the projected point.

has its own (potentially noisy) estimates of other robots' states, as is common with noisy on-board sensors. The method additionally requires no explicit communication among the robots, nor does it require these estimates to be consistent with those of other robots. More specifically, we assume that each robot keeps an uncertainty set (*e.g.*, unions and intersections of ellipsoids) that contain the possible future locations of other robots.

The resulting policy is distributed in the sense that each robot *only* requires an estimate of the relative positions of the other robots. In other words, robots do not need to communicate their positions to one another to coordinate their actions. Each robot uses its own position estimates for the other robots to find a safe-reachable set for itself, which is characterized by a generalized Voronoi cell. Our algorithm then computes a projection onto this safe-reachable set, which we show reduces to an efficiently solvable convex optimization problem. We then extend the projection method to find smooth polynomial trajectories, instead of just a single point, which lie entirely inside the agents' safe-reachable set. Our method is amenable to fast convex optimization solvers. Using our method, a quadrotor maneuvering in a 3D space with 100 other quadrotos can compute its control action in approximately 17ms, including setup and solution time. Because the resulting method is reciprocally safe, we have the immediate implication that, if each robot uses this policy, mutual collision avoidance is guaranteed.

Summary. This paper is organized as follows. The remainder of this section discusses related work. The following section defines the class of 'reciprocally safe methods', proves that reciprocal collision avoidance is guaranteed for any method in this class, and then discusses some basic, but useful, extensions to this proof. The following section presents a particular instance of a reciprocally safe method, called 'CARP,' short for Collision Avoidance by Reciprocal Projections. This method is specifically constructed to guarantee safety, even in the presence of noisy sensor data, while also being fast to execute on-board, as the resulting trajectory can be computed by solving a small convex optimization problem. The following

section shows some benchmarks of the algorithm on synthetic data, a large team of simulated robots in both 2D and 3D, and, finally, on a team of quadrotors with noisy on-board data. We conclude with some thoughts and possible extensions. This paper extends the previous conference submission [ASS19] by adding a smooth polynomial trajectory generation method via the reciprocal projection framework as well as hardware demonstrations on a quadrotor platform. It also simplifies and extends the proof and the constructions given in the original paper.

1.1 Related work

The most closely related methods for fully distributed collision avoidance in the literature are the velocity obstacle (VO) methods, which can be used for a variety of collision avoidance strategies. These methods work by extrapolating the next position of an obstacle using its current position and velocity. One of the most common tools used for mutual collision avoidance is the Reciprocal Velocity Obstacles (RVO) [VdBLM08, VdBGLM11, VdBGS+16] method in which each agent solves a linear program to find its next step. The Buffered Voronoi Cell (BVC) method of [ZWBS17] provides similar avoidance guarantees, but does not require the ego agent to know other agents' velocities, which can be difficult to estimate accurately. The BVC algorithm opts instead for defining a given distance margin to compute safe paths. BVC methods have been coupled with other decentralized path planing tools [ŞHA19] in order to successfully navigate more cluttered environments, but require that the other agents' positions are known exactly.

Uncertainty. While both VO and BVC methods scale very well to many (more than 100) agents, they also require perfect state information of other agents' positions (BVC), or positions and velocities (RVO). In many practical cases, high accuracy state information, especially velocity, may not be accessible as agents are estimating the position of the same objects they are trying to avoid. Extensions to VO that account for uncertainty have been studied under bounded ([CHTM12]) and unbounded ([GSK+17]) localization uncertainties by utilizing chance constraints. While these have been extended to decentralized methods ([ZA19]), they assume constant velocity of the obstacles at plan time. Unbounded localization estimates, modeled as Gaussian Mixture Models, were combined with RVO in [AM21a], and have been used effectively, but these methods only estimate the velocity at plan time and do not consider how the velocity of the other agents may change over the horizon. Combined Voronoi partitioning and estimation methods have been studied for multi-agent path planning tasks [BCH14], but still require communication to build an estimate via consensus. The V-RVO method of [AM21b] is the most similar to our method in that it augments the safe reachable set by adding RVO style constraints to the BVC as well as deflating the safe-reachable regions based on higher order dynamics. In contrast to these, our method does not require any communication or velocity state information, nor does it require the true position of the other agents. Instead, the algorithm uses only an estimate of the current position of nearby agents and their reachable set within some time horizon.

Comparison. Our algorithm takes a desired trajectory or goal point (which can come from any source, akin to [VdBLM08, ZWBS17]), and returns a safe next step for an agent to take while accounting for both the uncertainty and the physical extent of other agents. The focus of this work is on fast, on-board refinement rather than total path planning. More specifically, while the algorithm presented could be used to reach a far away goal point, it is likely more useful as a small-to-medium scale planner for reaching waypoints in a larger, centrally or locally planned trajectory.

Higher order planners. Similarly, single agent path planners such as A* [HNR68] or rapidly-exploring random trees (RRTs) [LKJ01] can be applied to the multi-agent case, but solution times grow rapidly due to the exploding size of the joint state space. For example, graph-search methods can be partially decoupled [WC11] to better scale for larger multi-agent systems, but can explore the entire joint state-space in the worst case. Fast Marching Tree (FMT) methods [JSCP15] are similar to RRTs in that they dynamically build a graph via sampling. But, while FMT methods have better performance in higher dimensional systems, they still require the paths to be centrally calculated. A* can also be used in dense environments for decentralized multi-agent planning when combined with barrier functions [MJWP16], but require the true position of all other agents. While all of these methods require global knowledge and large searches over a discrete set, they can be used as waypoint generators that feed into our method—for use in, *e.g.*, cluttered environments.

Nonconvex methods. Optimization methods that use sequential convex programming (SCP) [SHL+13, ASD12, MCH14] have also been studied for multi-agent path planning; however, these algorithms are still centralized and may exhibit slow convergence, making them unreliable for on-line planning. For some systems, these methods can be partially decoupled [CCH15], reducing computation time at the cost of potentially returning infeasible paths. Our method, in comparison, is fully decentralized and produces an efficient convex program for each agent. The solution of this program is a safe waypoint for the agent, which, unlike SCP methods, requires no further refinement.

2 Reciprocally safe methods

In this section we define a class of methods, called the reciprocally safe methods, and show that any method in this class is guaranteed to be collision-free, assuming the agents start out in a collision-free configuration. We will then show that this proof contains a number of results in the literature as special cases, and give some simple extensions, such as ensuring a minimum separation.

2.1 Method description

We start with n agents in an unbounded Euclidean space. For each agent $i = 1, \dots, n$, let $x_i(t) \in \mathbf{R}^d$ denote its position, where d is the dimension of the space, at time $t = 1, \dots, T$.

At each time step t, each agent i will have an estimate of possible future positions of every other agent j, given by some set $\mathcal{E}_j^i(t) \subseteq \mathbf{R}^d$, with $i, j = 1, \ldots, n$, and $j \neq i$. (Note that this set need not be bounded, nor closed for the proof.) We will assume that $\mathcal{E}_j^i(t)$ is *consistent*; *i.e.*, that agent j's position at time $t+1$ is always within the uncertainty set of agent i:

$$x_j(t+1) \in \mathcal{E}_j^i(t),$$

for every agent $i, j = 1, \ldots, n$ with $i \neq j$ and every given time $t = 1, \ldots, T-1$.

Reciprocally safe set. A set $P_i(t) \subseteq \mathbf{R}^d$ for agent i, at time t, is *reciprocally safe* if, for every other agent j with $j \neq i$, we have that

$$P_i(t) \cap \mathcal{E}_j^i(t) = \emptyset. \tag{1}$$

One way of interpreting $P_i(t)$ is as a set of positions which are known to be safe to move to, given the position estimates of every other agent. As before, we do not require this set be bounded or even closed.

We will then say that the n agents implement a *reciprocally safe method* if there exists reciprocally safe sets $P_i(t) \subseteq \mathbf{R}^d$ such that

$$x_i(t+1) \in P_i(t),$$

if $P_i(t)$ is nonempty, and $x_i(t+1) = x_i(t)$, otherwise, for each $i = 1, \ldots, n$ and $t = 1, \ldots, T-1$. Written out, we say that agents implement a reciprocally safe method whenever they move to a reciprocally safe position whenever such a position is available, and do not move otherwise. We will now show that any reciprocally safe method is collision-free, assuming the agents start from a collision-free configuration.

Proof. The proof is almost by definition. Consider any two distinct agents $i, j = 1, \ldots, n$ with $i \neq j$ at some time $t = 1, \ldots, T$. We will show that $x_i(t+1)$ and $x_j(t+1)$ always have positive distance from each other, assuming that $x_i(t)$ and $x_j(t)$ are some positive distance apart. First, if $P_i(t)$ is nonempty, then, by definition, we have

$$x_i(t+1) \in P_i(t),$$

but since the set $P_i(t)$ has empty intersection with the uncertainty set, by assumption,

$$P_i(t) \cap \mathcal{E}_j^i(t) = \emptyset,$$

then,

$$x_i(t+1) \notin \mathcal{E}_j^i(t),$$

and, finally, since the uncertainty set contains the true position of agent j at time $t+1$,

$$x_j(t+1) \in \mathcal{E}_j^i(t),$$

so $x_i(t+1)$ and $x_j(t+1)$ must have positive separation.

On the other hand, if $P_i(t)$ is empty, and $P_j(t)$ is nonempty, the proof follows similarly by replacing i with j. Finally, if both $P_i(t)$ and $P_j(t)$ are empty, then

$$x_i(t+1) = x_i(t), \quad \text{and} \quad x_j(t+1) = x_j(t),$$

so $x_i(t+1)$ and $x_j(t+1)$ have positive separation because $x_i(t)$ and $x_j(t)$ did, by assumption.

Because this is the case for any $i \neq j$ and every $t = 1, \ldots, T-1$, any reciprocally safe method is collision free as any two agents always have positive separation, assuming they have positive separation at $t = 1$.

2.2 Basic extensions

We outline some immediate extensions to the proof above.

Positive margin. In the previous proof, while there is a guarantee that the agents will be positively separated, the separation could be arbitrarily small. On the other hand, we can imagine that the agents are all required to have some amount of margin that is bounded from below. We will denote the required margin by some set $B \subseteq \mathbf{R}^d$, which we view as a "safe" region around the agent. Usually B will be a ball in d dimensions:

$$B = \{y \mid \|y\|_2 \leq \varepsilon\},$$

where $\varepsilon > 0$ is the desired minimal distance from all other agents, but we may generally take any set B we wish.

To guarantee this, we can strengthen condition (1) to

$$(P_i(t) + B) \cap \mathcal{E}_j^i(t) = \emptyset,$$

whenever the set $P_i(t)$ is nonempty. Here, we define

$$P_i(t) + B = \{y + z \mid y \in P_i(t),\ z \in B\},$$

as the set sum, or Minkowski sum, of $P_i(t)$ and B. Another way of stating this is, if the set $P_i(t)$ is nonempty, it must only contain points which have a margin of at least B from the future estimated positions. The resulting proof is similar to the base method and gives the stronger guarantee that

$$x_j(t+1) \notin x_i(t+1) + B,$$

for any two agents $i \neq j$. In the special case that B is an ε-ball, this would imply that agents i and j are at least ε distance apart.

Stopping set. The proof above requires the condition that $x_i(t+1) = x_i(t)$ if the set $P_i(t)$ is empty. This is, of course, not always possible for realistic agents such as drones, which cannot stop immediately. In this case, we can relax the condition $x_i(t+1) = x_i(t)$ to the following condition:

$$x_i(t+1) \in x_i(t) + S_i = \{x_i(t) + y \mid y \in S_i\},$$

where $S_i \subseteq \mathbf{R}^d$ is agent i's *stopping set* (similar to the braking set used in [AM21b]) which is assumed to be compact. We then strengthen the initial condition that $x_i(t)$ and $x_j(t)$ are positively separated for $i \neq j$, to

$$(x_i(t) + S_i) \cap (x_j(t) + S_j) = \emptyset,$$

and the reciprocal safety condition that $P_i(t)$ must satisfy when it is nonempty, to

$$(P_i(t) + S_i) \cap \mathcal{E}_j^i(t) = \emptyset,$$

for each $i, j = 1, \ldots, n$ with $i \neq j$. Positive separation between $x_i(t+1) \in x_i(t) + S_i$ and $x_j(t+1) \in x_j(t) + S_j$ is then always guaranteed in this case by a similar proof, since disjoint compact sets have positive separation. Note that this only guarantees one-step separation, from time t to time $t+1$. We can then guarantee positive separation at all times by requiring the additional condition that, if $P_i(t)$ and $P_i(t+1)$ are both empty, we have

$$x_i(t+2) = x_i(t+1),$$

i.e., agent i only needs to 'stop once.'

2.3 Example methods

There are several simple methods which satisfy the reciprocal safety property, and are therefore guaranteed to be collision free. We describe some basic examples in this section, which include some methods in the literature that are known to be collision free by other proof techniques, showing that this class is general enough to include a number of known results.

Trivial method. Perhaps the simplest of all possible reciprocally safe methods is the *trivial method*, which, for all agents $i = 1, \ldots, n$, defines $P_i(t)$ as

$$P_i(t) = \emptyset, \quad t = 1, \ldots, T-1,$$

and sets the position estimates to be all of $\mathbf{R}^d$ for any distinct agents $i, j = 1, \ldots, n$ with $i \neq j$,

$$\mathcal{E}_j^i(t) = \mathbf{R}^d, \quad t = 1, \ldots, T-1.$$

In other words, the agents' position estimates of each other are 'maximally bad,' *i.e.*, they include all of $\mathbf{R}^d$, and there is no safe position to move to, as $P_i(t) = \emptyset$ for each agent i and time t. It is follows that the sets $P_i(t)$ satisfy the reciprocal safety conditions (1), and

that the agents must always have $x_i(t+1) = x_i(t)$. Because of this, the agents never move from their starting locations, which is easily seen to be collision free, when starting from a collision free configuration.

While this example is not useful in practice, it is a good initial exercise to check the conditions necessary for a method to be reciprocally safe.

Buffered Voronoi cell. Another reciprocally safe method is the buffered Voronoi cell method of [ZWBS17]. In this method, the future position estimate that agent i has of agent j at time t, is given by

$$\mathcal{E}_j^i(t) = \{y \in \mathbf{R}^d \mid \|y - x_j(t)\|_2^2 + \varepsilon \leq \|y - x_i(t)\|_2^2\},$$

where $x_j(t)$ is the (known) position of agent j at time t, $x_i(t)$ is the position of agent i, and $\varepsilon > 0$ denotes some margin. In other words, the possible future positions of agent j are the set of points which are further from agent i than from j by at least ε. (We take a simpler approach here than that of [ZWBS17] for the sake of presentation, but the proof is nearly identical.) In this case, the set of 'safe' locations for agent i is defined similarly:

$$P_i(t) = \bigcap_{j \neq i} \{y \in \mathbf{R}^d \mid \|y - x_i(t)\|_2^2 + \varepsilon \leq \|y - x_j(t)\|_2^2\}.$$

The reciprocal safety condition can be readily verified, since for distinct agents i and j, and any point in the uncertainty region $y \in \mathcal{E}_j^i(t)$ satisfies

$$\|y - x_j(t)\|_2^2 + \varepsilon \leq \|y - x_i(t)\|_2^2,$$

by definition. So, y cannot be in the reciprocally safe set $P_i(t)$, since $y \in P_i(t)$ means that y also satisfies,

$$\|y - x_i(t)\|_2^2 + \varepsilon \leq \|y - x_j(t)\|_2^2 \leq \|y - x_i(t)\|_2^2 - \varepsilon,$$

a contradiction.

Any method that agent i uses to choose its next location within the set $P_i(t)$ is guaranteed to be collision free by the above proof. Additionally, because the set $P_i(t)$ is a convex set as it is the intersection of a number of convex sets, many convex optimization problems can easily include the constraint that the agent's future position must lie in this set. Because convex problems are almost always efficiently solvable, even with on-board computational constraints, finding feasible points that best satisfy some requirement can often be done in real time.

Method refinement. Very generally, we have the following additional result: if a method is reciprocally safe, any refinement of such a method is also always safe. That is, given a reciprocally safe method, where the agents $i \neq j$ have estimates $\mathcal{E}_j^i(t)$ and reciprocally safe sets $P_i(t)$ at each time t, we say a second method is a *refinement* of the first if it has estimates $\bar{\mathcal{E}}_j^i(t)$ that satisfy $\bar{\mathcal{E}}_j^i(t) \subseteq \mathcal{E}_j^i(t)$, and reciprocally safe sets $\bar{P}_i(t)$ that satisfy $\bar{P}_i(t) \subseteq P_i(t)$. This

implies that if the original method is reciprocally safe, *i.e.*, satisfies (1), then a refinement is also reciprocally safe as it also satisfies (1), and is therefore collision free. More intuitively, this is can be restated as the fact that having better estimates, or a more restricted action space, can never make an algorithm unsafe. This construction then immediately includes, for example, the safety results of [AM21b], where the position estimates are the buffered Voronoi cells given in the previous example, intersected with an additional reciprocal velocity obstacle (RVO) cone.

3 General method

In this section, we present a reciprocally safe method that generalizes the buffered Voronoi cells of [ZWBS17] to include measurement uncertainty and higher order dynamics. The resulting method relies on the solution of a small convex optimization problem at each time step that is unlikely to have a closed form, but can still be solved with on-board systems in under a millisecond with modern solvers.

To do this, we first introduce the idea of generalized Voronoi cells—a natural way of extending Voronoi cells from collections of points to collections of sets. In many cases, the resulting generalized Voronoi cells cannot be defined in terms of a finite number of closed form inequalities, but, because these sets are always convex, we can write new expressions that depend on a larger number of variables that do have closed forms.

3.1 Generalized Voronoi cells

Given a point $x \in \mathbf{R}^d$ and a collection of m sets $S_1, \ldots, S_m \subseteq \mathbf{R}^d$, we will define the *generalized Voronoi cell* of x with respect to the family S as

$$V(x, S) = \{y \in \mathbf{R}^d \mid \|y - x\|_2 \leq \min_{j=1,\ldots,m} \|y - S_j\|_2\}. \tag{2}$$

Here, we have defined $\|y - S_j\|_2$ to be the distance-to-set function:

$$\|y - S_j\|_2 = \inf_{z \in S_j} \|y - z\|_2,$$

for $j = 1, \ldots, m$. (If S_j is empty, we set the distance as $+\infty$, for convenience.) We can view the points in $V(x, S)$ as the set of points which are closer to x than to any point in any one of the sets S_j. Note that we can write

$$V(x, S) = \bigcap_{j=1}^{m} V(x, \{S_j\}).$$

We will make use of this fact later in what follows.

Comparison to Voronoi cells. The usual definition of the Voronoi cell is, given a family of points $x_1, \ldots, x_m \in \mathbf{R}^d$, the cell generated for the ith point is defined as the set of all points closer to x_i than to any other point x_j; *i.e.*,

$$\{y \in \mathbf{R}^d \mid \|y - x_i\|_2 \leq \min_{j=1,\ldots,m} \|y - x_j\|_2\}.$$

This is the special case of (2) where we take the generalized Voronoi cell of x_i with respect to S, and every set in the family S is a singleton; *i.e.*, $S_j = \{x_j\}$ for each $j = 1, \ldots, m$.

Convexity. Like the usual definition of a Voronoi cell, the generalized Voronoi cell of x with respect to S is convex, even when the sets S_j are not. To see this, note that we can write V as

$$V(x, S) = \bigcap_{j=1}^{m} \bigcap_{z \in S_j} \{y \in \mathbf{R}^d \mid \|y - x\|_2 \leq \|y - z\|_2\},$$

which is the intersection of a family of hyperplanes. (This follows by squaring both sides of the inequality and cancelling the $\|y\|_2^2$ term.) Because the intersection of convex sets is convex, and hyperplanes are convex sets, then $V(x, S)$ is always convex.

3.2 Projective method

We will now discuss a simple version of the method and its applications to single-integrator dynamics. Later in this section, we will also show how to extend this to more general dynamics and more general sets.

Projection. A common objective for an agent to optimize is its distance to some desired goal. We will call this goal point $x_i^g \in \mathbf{R}^d$, for agent i. At every time step t, the optimization problem for agent i is then:

$$\begin{array}{ll} \text{minimize} & \|y - x_i^g\|_2 \\ \text{subject to} & y \in P_i(t). \end{array}$$

The optimization variable here is $y \in \mathbf{R}^d$, while the problem data are the goal point $x_i^g \in \mathbf{R}^d$ and the reciprocally safe set $P_i(t)$. If a solution to the problem $y^\star$ exists, the agent takes a step towards $y^\star$, otherwise, if there are no feasible points, then the agent stops in place. (We will see extensions to the more general case where the agent has some stopping distance later in this section.)

Such a method is often called a "greedy" method, as the agent attempts to get as close as possible as it can to the goal position, while remaining safe. We refer to this specific way of picking the next possible point as a 'projective method' since a solution $y^\star$ is often called the projection of x_i^g onto the set $P_i(t)$. We will show how to construct reciprocally safe sets $P_i(t)$ for all agents $i = 1, \ldots, n$, assuming that each agent has some (potentially noisy) estimate of the future locations of all other agents.

Safe estimates. To simplify notation, we will write the algorithm for a single agent i at a given time step t. Because of this, we write x for $x_i(t)$, which is the agent's current position at time t, and $\mathcal{E}_j$ for $\mathcal{E}_j^i(t)$, which are the estimates agent i has of agent j at the next time step, $t+1$.

Using this notation, we can view the set $V(x, \mathcal{E})$ (*i.e.*, the set of points which are closer to x than they are to any $\mathcal{E}_j$) as a set of positions which agent i is guaranteed to reach before any agent j. More specifically:

$$V(x, \mathcal{E}) \cap \mathcal{E}_j = \emptyset, \tag{3}$$

for each $j \neq i$. We also note that this set is an overly-conservative set as there are many sets that also satisfy (3) that strictly contain $V(x, \mathcal{E})$.

A natural choice for a reciprocally safe set for agent i, is then

$$P_i(t) = V(x, \mathcal{E}),$$

since we know that any choice of $y \in P_i(t)$ is guaranteed to be safe, by construction. This implies that, if all agents i choose points within $P_i(t)$ (and simply stop if the set is empty) then the method is guaranteed to be collision free. The resulting optimization problem is:

$$\begin{array}{ll} \text{minimize} & \|y - x_i^g\|_2 \\ \text{subject to} & y \in V(x, \mathcal{E}), \end{array} \tag{4}$$

with variable $y \in \mathbf{R}^d$. Note that, since $P_i(t)$ is a generalized Voronoi cell, then $P_i(t)$ is convex. We will use this fact to give an efficient method for optimizing a goal function, when the sets $\mathcal{E}_j$ are the intersections and unions of well-known convex sets.

3.3 Projecting onto generalized Voronoi cells

We will show how to efficiently solve (4) by first reducing it to a problem over several constraints, each of which are simpler than the original. We then show how these constraints can be reduced to a number of inequalities which are easily compiled down to well-known conic constraints, when the uncertainty sets $\mathcal{E}_j$ are unions and intersections of ellipsoids and polygons. Similarly to the previous subsection, we will only consider agent i's position at time t, denoted simply as x and the uncertain estimates of the future positions of the other agents as $\mathcal{E}_j$ for $j = 1, \ldots, n$ with $j \neq i$.

A basic reduction. We first note that, given a problem of the form of (4), we can write the equivalent problem:

$$\begin{array}{ll} \text{minimize} & \|y - x_i^g\|_2 \\ \text{subject to} & y \in V(x, \{\mathcal{E}_j\}), \quad j \neq i. \end{array}$$

In other words, we have 'split' the single constraint $y \in V(x, \mathcal{E})$ to n constraints given by $y \in V(x, \{\mathcal{E}_j\})$ for each j. This follows from the fact that

$$V(x, \mathcal{E}) = \bigcap_{j \neq i} V(x, \{\mathcal{E}_j\}),$$

which is readily verified from (2). From this, it suffices to show how to write the constraint $y \in V(x, \{\mathcal{E}_j\})$ for a single set $\mathcal{E}_j$. We will write $\mathcal{E}_1$ for the set in question, with the understanding that $\mathcal{E}_1$ stands for any 'anonymous' set.

Convex sets. In general, the set $\mathcal{E}_1$ is defined by

$$\mathcal{E}_1 = \{z \in \mathbf{R}^d \mid f(z) \leq 0\}, \tag{5}$$

where $f : \mathbf{R}^d \to \mathbf{R}^r$ is a convex function. For example, in the case that $\mathcal{E}_1$ is an ellipse defined by (μ, Σ) where $\mu \in \mathbf{R}^d$ and $\Sigma \in \mathbf{S}^d_{++}$ is positive definite, we have that

$$f(z) = \frac{1}{2} z^T \Sigma^{-1} z + \mu^T z - 1.$$

On the other hand, if $\mathcal{E}_1$ is a polyhedron, it is defined by a number of affine inequalities; *i.e.*,

$$f(z) = Az - b,$$

where $A \in \mathbf{R}^{r \times d}$ and $b \in \mathbf{R}^r$. There are a number of other possible functions, such as indicator sets among many others, but we focus on these two cases as the most common types of sets.

Constraint rewriting. Given any set $\mathcal{E}_1$ defined by a function f, as in (5), the corresponding constraint is:

$$y \in V(x, \{\mathcal{E}_1\}).$$

From (2), this is true, if, and only if, y also satisfies

$$\|y - x\|_2^2 \leq \inf_{z \in \mathcal{E}_1} \|y - z\|_2^2.$$

We will rewrite this as

$$\|x\|_2^2 - 2y^T x \leq \inf_{f(z) \leq 0} \left(\|z\|_2^2 - 2y^T z \right), \tag{6}$$

by expanding the squared norm on both sides, cancelling like terms, and using the definition of $\mathcal{E}_1$. In general, it is unlikely that there is a closed form solution for the right hand side of the inequality, even in the special cases where $\mathcal{E}_1$ is an ellipse or a polyhedral set. To get around this, we will use a duality trick, introduced originally in [ASS19], to rewrite the right hand side as an unconstrained infimum. This new infimum has a simple analytical solution in the important cases where f is an affine function (when $\mathcal{E}_1$ is a polyhedron) and when f is a convex quadratic (when $\mathcal{E}_1$ is an ellipsoid). There are likely more applications of this method to more complicated functions f, but we focus on these two important cases. We encourage readers to apply this method to other functions f which may be useful in practice.

Weak duality. One simple approach to finding a reasonable replacement for (6) is to find an approximation of the right hand side that is concave and reasonably tight. One standard approach is by *Lagrange duality* [BV04], where the 'hard constraint' $f(z) \leq 0$ is relaxed to a linear penalty term with some weights $\lambda \in \mathbf{R}^n_+$, *i.e.*, inequality (6) is replaced with

$$\|x\|_2^2 - 2y^T x \leq \inf_z \left(\|z\|_2^2 - 2y^T z + \lambda^T f(z) \right). \tag{7}$$

This new infimum is unconstrained and sometimes, but not always, admits closed form solutions when the original does not. (In many cases, the closed form solutions, when they exist, are well-known.) For convenience, we will define the *dual function* g as

$$g(y, \lambda) = \inf_z \left(\|z\|_2^2 - 2y^T z + \lambda^T f(z) \right),$$

and note that, for any $\lambda \geq 0$ and y we have that

$$g(y, \lambda) \leq \inf_{f(z) \leq 0} \left(\|z\|_2^2 - 2y^T z \right), \tag{8}$$

which is known as *weak duality* [BV04]. Replacing inequality (6) with inequality (7) is often called a *restriction*: if y is feasible for some $\lambda \geq 0$ in (7) then y is also feasible for (6). In other words, the new constraint is at least as tight as the original. An important fact of the dual function g is that it is concave in its arguments, because $g(y, \lambda)$ is an infimum over a family of functions that are affine in y and λ. This implies that (7) is a convex inequality constraint.

Strong duality. Due to strong duality, the new constraint is, in fact, equivalent to the original. That is, for every y, there exists some $\lambda \geq 0$ such that inequality (8) holds at equality:

$$g(y, \lambda) = \inf_{f(z) \leq 0} \left(\|z\|_2^2 - 2y^T z \right).$$

Because of this, if y is feasible for (6) then there exists some λ such that y is also feasible for (7), and vice versa. In other words, replacing inequality (6) with (7) and solving this new problem (with $\lambda \geq 0$ as an additional variable) gives two equivalent problems in that a feasible y for the first corresponds to a feasible pair (y, λ), with the same objective value, for the second.

Dual functions for known sets. Given that both constraints are equivalent, the last remaining point is to write down the dual functions for some known sets. We will make use of the fact that the minimizer of a convex quadratic is

$$\inf_z \left(z^T P z + 2q^T z \right) = -q^T P^{-1} q, \tag{9}$$

for any positive definite matrix $P \in \mathbf{S}^d_{++}$ and vector $q \in \mathbf{R}^d$. This follows from an application of the first order optimality conditions.

Polyhedral uncertainty. In the case where $\mathcal{E}_1$ is a polyhedron, we have that

$$f(z) = Az - b$$

for some $A \in \mathbf{R}^{r\times d}$ and $b \in \mathbf{R}^r$. In this case, the dual function is

$$g(y,\lambda) = \inf_z \left(\|z\|_2^2 - 2y^T z + 2\lambda^T(Az-b)\right),$$

where we have replaced λ with 2λ for convenience. Using (9) and the fact that $\|z\|_2^2 = z^T I z$, we get:

$$g(y,\lambda) = -\|A^T\lambda - y\|_2^2 - 2\lambda^T b,$$

as required.

Ellipsoidal uncertainty. The case where $\mathcal{E}_1$ is an ellipsoid with parameters $\mu \in \mathbf{R}^d$ and $\Sigma \in \mathbf{S}^d_{++}$ is rather similar. In this case, the function f is a quadratic and the dual function is

$$g(y,\lambda) = \inf_z \left(\|z\|_2^2 - 2y^T z + \lambda(z^T\Sigma z + 2\mu^T z - 1)\right).$$

Replacing, again, λ with 2λ for convenience. Collecting the quadratic, linear, and constant terms and applying (9), we have:

$$g(y,\lambda) = -(\lambda\mu - y)^T(I+\lambda\Sigma)^{-1}(\lambda\mu - y) - \lambda.$$

This function is concave, and its corresponding constraint is easily representable as a semidefinite constraint. On the other hand, most embedded solvers, often due to space and time restrictions, do not support semidefinite constraints. We can turn this into a constraint that is representable as a small family of second-order cone constraints, which are often more efficient in practice and can be handled by embedded solvers such as ECOS [DCB13].

Since Σ is positive definite, it has an eigendecomposition given by

$$\Sigma = VDV^T,$$

where $D \in \mathbf{R}^{d\times d}$ is a diagonal matrix and $V \in \mathbf{R}^{d\times d}$ is an orthogonal matrix such that $V^TV = VV^T = I$. Using this, we can write:

$$(I+\lambda\Sigma)^{-1} = (V(I+\lambda D)V^T)^{-1} = V(I+\lambda D)^{-1}V^T.$$

This implies that

$$g(y,\lambda) = -(V^T(\lambda\mu - y))^T(I+\lambda D)^{-1}(V^T(\lambda\mu - y)) - \lambda.$$

Because the matrix $I + \lambda D$ is diagonal, then:

$$g(y,\lambda) = -\sum_{i=1}^d \frac{(v_i^T(\lambda\mu - y))^2}{1+\lambda D_{ii}} - \lambda, \tag{10}$$

where v_i is the ith column of V. This expression is a sum of quadratic-over-linear terms, which are easily representable as second-order cone constraints [LVBL98] and are supported in most embedded solvers.

Unions and intersections of sets. If $\mathcal{E}_1$ is the union of a number of ellipsoids or polyhedra, then, as before, we can simply split $\mathcal{E}_1$ into its individual components and add each as a constraint. On the other hand, $\mathcal{E}_1$ can also be the *intersection* of ellipsoids and polyhedra; *i.e.*, $\mathcal{E}_1$ is the intersection of the polyhedra specified by $A \in \mathbf{R}^{r\times d}$ and $b \in \mathbf{R}^r$ (as the intersection of polyhedra results in another polyhedron) and the ellipsoids given by (μ_i, Σ_i) for $i = 1, \ldots, s$.

The derivation here is again nearly identical to the previous. We will write $\lambda_0 \in \mathbf{R}^r_+$ as the Lagrange multiplier for the polyhedral constraint, while we write $\lambda_i \geq 0$ for $i = 1, \ldots, s$ for each of the ellipsoidal constraints. The dual function is then:

$$g(y, \lambda) = -h\left(A^T\lambda_0 + \sum_{i=1}^{s} \lambda_i\mu_i - b, I + \sum_{i=1}^{s} \lambda_i\Sigma_i\right),$$

where the function $h : \mathbf{R}^d \times \mathbf{S}^d_{++} \to \mathbf{R}$ is the 'matrix fractional' function:

$$h(y, X) = y^T X^{-1} y.$$

Surprisingly, since the Σ_i are positive semidefinite, the corresponding equality constraint can also be written as a number of second order cone constraints, though this reduction is slightly more complicated and we do not present it here. See, *e.g.*, [LVBL98] for more information.

3.4 Safe quadrotor trajectory planning

For higher order dynamical systems such as quadrotors, it is more desirable to plan entire safe trajectories rather that just finding a single safe point. We can plan smooth polynomial trajectories for each agent i, such that the entire trajectory is inside each agent's reciprocally safe set, $P_i(t)$. After the polynomial trajectory is found, the required control inputs can be found via differential flatness [MK11]. In order to generate these trajectories and verify that they are safe, we require two extensions on the original CARP formulation: (1) a way to find a polynomial that is entirely inside the P_i, and (2) a method of expanding the uncertainty estimate $\mathcal{E}^i_j$ to include the stopping set of other quadrotors.

Bézier curves. Instead of the standard polynomial formulation, we use a Bézier curve to represent the polynomial trajectory. A Kth order 3-dimensional Bézier curve, $B(t) \in \mathbf{R}^3$, is defined by a set of $K + 1$ "control" points $c_k \in \mathbf{R}^d$, for $k = 0, \ldots, K$. We define the Kth order Bézier curve as a linear combination of Bernstein polynomials:

$$B(t) = \sum_{k=0}^{K} \binom{K}{k} (1-t)^{K-k} t^k c_k. \tag{11}$$

This curve has the property that it always lies in the convex hull of the control points, *i.e.*, $B(t) \in \mathbf{conv}\{c_0, \ldots, c_K\}$ for every $0 \leq t \leq 1$. To find a polynomial of this form that is entirely inside the safe region, we can add additional constraints to (4) to constrain each

control point to be inside the set P_i. Finally, since the derivatives of a Bézier curve are linear combinations of the control points, we can account for the initial and desired final dynamic state (consisting of the position, velocity, and acceleration) by adding additional linear constraints to problem (4).

Stopping margins. To include the stopping margins or each quadrotor, we simply expand the estimate $\mathcal{E}_j^i$, similar to [AM21b], by a sphere with radius $r = v_{\max}/2a_{\max}$, where $v_{\max}$ and $a_{\max}$ are the maximum velocity and acceleration of the quadrotor, respectively. This provides safe, yet conservative bound, for the stopping distance other agents will have during the planning horizon. In practice, if a velocity measurement is available then a more accurate inflation can be found. Since the original ellipsoidal estimate is only contains a point estimate of the other agents, we can expand the ellipsoid to account any number of arbitrary margins. Given a set of arbitrary ellipsoids $\mathcal{E}_0, \ldots, \mathcal{E}_n$, there exists an analytical method [BABD06, LZW16] that finds the smallest (in the sense of major axes length) external bounding ellipsoid $\bar{\mathcal{E}}$ for the Minkowski sum of all the ellipsoids $\bar{\mathcal{E}} \supset \sum_{i=0}^{n} \mathcal{E}_i$. Thus we can, in a procedural and consistent manner, combine different margins.

Persistently safe receding horizon controller. We extend problem (4) to account for additional constraints placed on the Kth order Bézier polynomial. To do this, we formulate the following optimization problem,

$$\begin{aligned} \text{minimize} \quad & \|c_K - x^{\mathrm{g}}\|_2^2 \\ \text{subject to} \quad & \|x\|_2^2 - 2c_k^T x + g(c_k, \lambda_k) + \lambda_k \leq 0, \quad k \in [K] \\ & x = c_0 \\ & v^0 = K(c_1 - c_0) \\ & v^f = K(c_K - c_{K-1}). \end{aligned} \tag{12}$$

The optimization variables are $c_k \in \mathbf{R}^3$ and $\lambda_k \geq 0$ for $k = 0, \ldots, K$, while x is the current position, and v^0 and v^f are the current and desired final velocities, respectively. For compactness, we write $[K] = \{0, \ldots, K\}$, and g to be defined as in (10). As written, this finds a 1 second long trajectory, but the section time for the trajectory can be changed by appropriately scaling the t variable in (11). Initial and final accelerations can also be added to (12). Since this method is a refinement of (4) (as any solution to (12) is feasible for (4) with $y = c_K$) then this method is guaranteed to be reciprocally safe. In the case where the optimization is infeasible, the agent can fall back to the previously calculated safe trajectory. Figure 2 shows an example instance of a polynomial trajectory being generated inside the safe-reachable area.

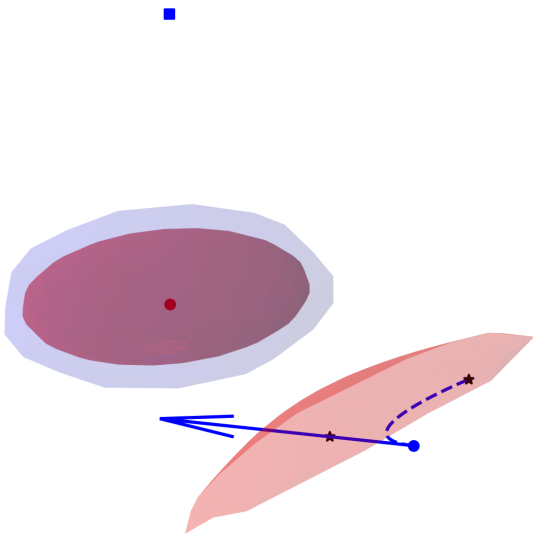


Figure 2: Polynomial trajectory (blue, dotted) generated for an agent (blue circle) with non zero initial velocity (blue arrow) to avoid the ellipsoidal estimate of the obstacle (red ellipsoid) expanded by a margin (blue ellipsoid). The agent goal position (blue square) and the Bézier control points (black stars) are also shown.

4 Results

4.1 Projection Implementation

To get an accurate estimate of the speed of the projection algorithm, the optimization problem outlined in (4) was implemented[1] in the Julia language [BEKS17] using the JuMP mathematical programming language [DHL17] and solved using ECOS [DCB13]. We generated 285 instances of the problem, each with 100 randomly generated ellipsoids in $\mathbf{R}^3$. Timing and performance results for generating and solving the corresponding convex program can be found in table 1. Figure 3 shows how the performance scales as the number of other agents increases. All times reported are on a 2.9GHz 2015 dual-core MacBook Pro.

Time	**Total (GC %)**
Minimum	13.20ms (00.00%)
Median	17.12ms (00.00%)
Mean	17.55ms (08.80%)
Maximum	36.77ms (10.59%)

Table 1: Timings with garbage collection (GC) as a percentage of time spent building and solving problems with 100 randomly generated 3D ellipsoids. Statistics are based on 285 instances and were obtained from the `BenchmarkTools.jl` [CR16] package.

[1] https://github.com/angeris/CARP.jl

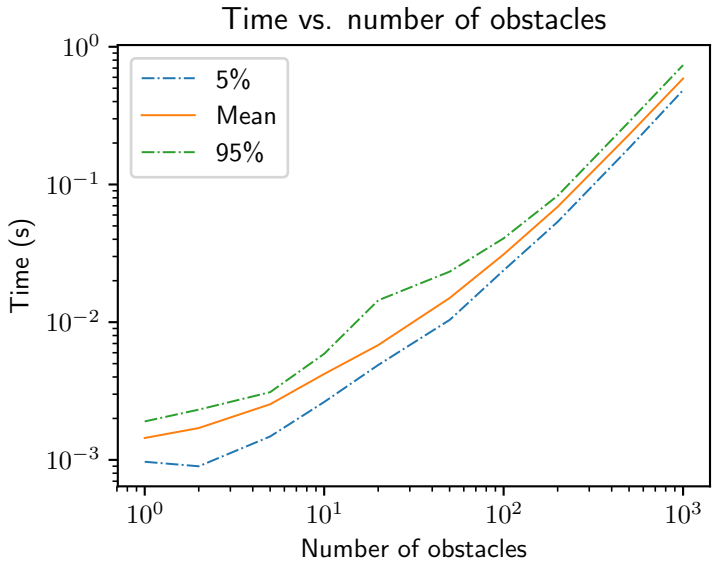


Figure 3: Graph showing total time for generating and solving optimization problem (4) as a function of the number of ellipsoids in the problem, when solved using the ECOS solver. Note the logarithmic scales on both axes.

4.2 Trajectory Simulations

The projection algorithm was implemented in both 2D and 3D with a varying number of agents. In this set up, each agent knows their own position exactly and maintains a noisy estimate of other agents' positions, with uncertainties represented as ellipsoids. This estimate is updated by a set-membership based filter [BR71, LZW16, SS19], a variant of the Kalman filter. We expand the uncertainty ellipsoid by a given margin to account for the robot's physical size. If this margin is also ellipsoidal, then a small ellipsoid which contains the Minkowski sum of the uncertainty ellipsoid and the margin can be found in closed form [LZW16, SS19]. This new bounding ellipsoid is used in the projection algorithm to account for a user defined margin, along with the uncertainty ellipsoid containing the noisy sensor information.

Figure 4 shows the minimum inter-agent distances for each agent in the simulation scenario mentioned above. The collision threshold was set to .4m, twice the radius of the agents. Although our method results in longer paths, it remains collision free, while RVO's paths result in collision.

Figure 5 shows six instances of a 3D simulation with 10 agents using the polynomial method. The agents start at the sides of a 10m $\times$ 10m $\times$ 10m cube and are constrained to a maximum speed of 6m/s and a maximum measurement error set to 1.0m.The agents, displayed as quadrotors, each have a bounding box of 0.45m $\times$ 0.45m $\times$ 0.2m and an additional ellipsoidal margin with an axis length of .3m in the $x$and y dimensions, and .7m in the z dimension. This margin effectively gives a buffer of 0.75m in the xy plane and a large buffer of .9m in z. We assume non-spherical margins in this simulation since, in the case of

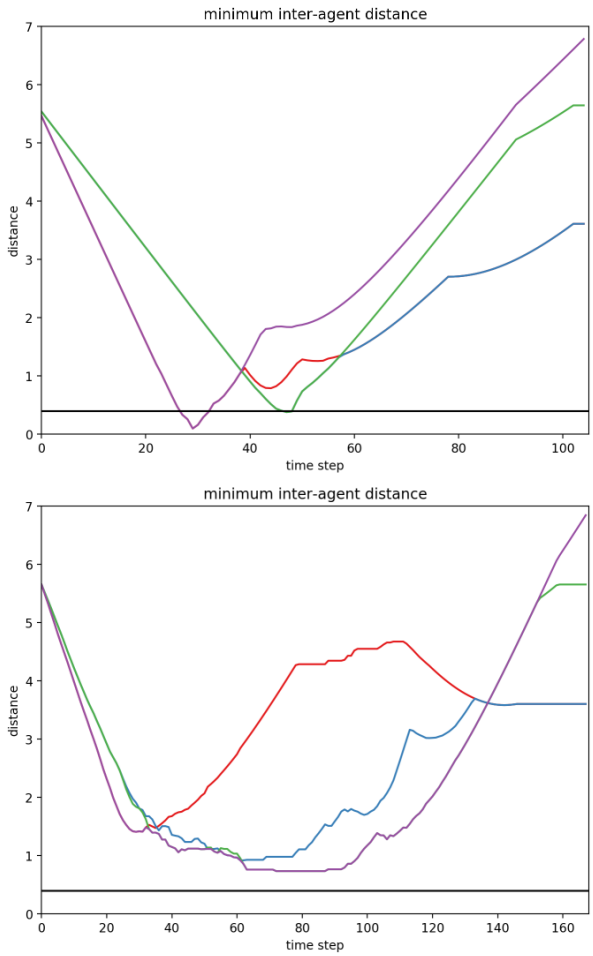


Figure 4: Inter-agent distances RVO (top) and our method (bottom). The RVO simulation results in inter-agent distances below the collision threshold (black line).

quadrotor flight, large margins in the z direction can prevent unwanted effects due to downwash ([PHAS17]). The minimum inter-agent distance during the simulation, as measured from the centers of the agents, was 1.6m.

4.3 Hardware Demonstrations

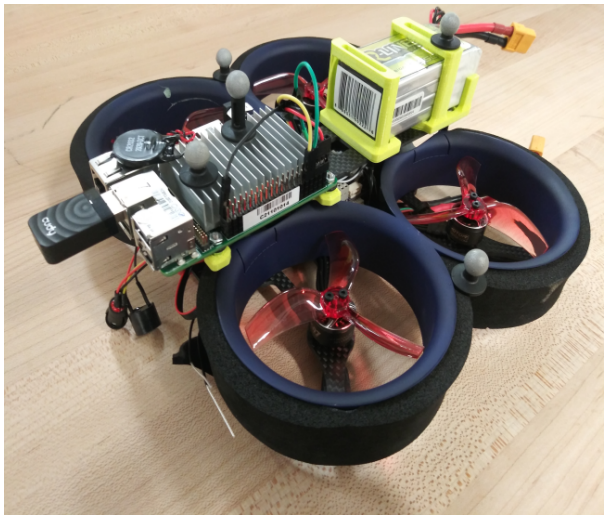
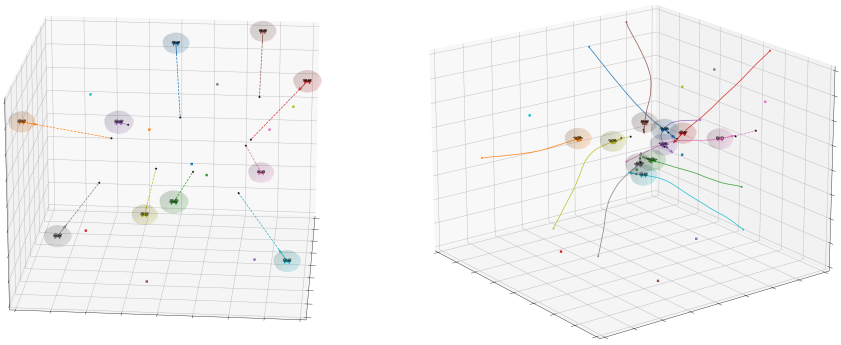


Figure 6: Quadrotor equipped with a PX racer and UP Board companion computer.

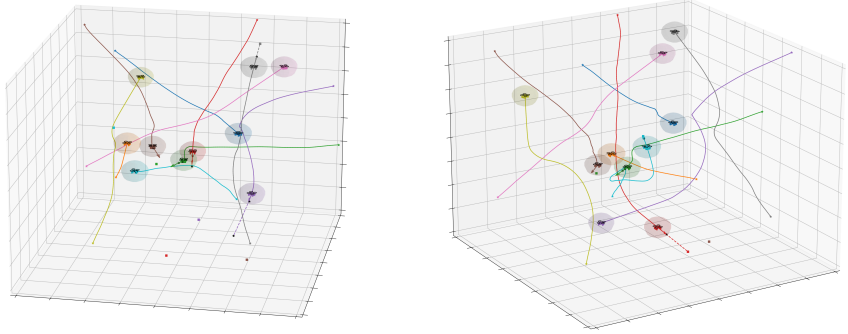
We implemented[2] our collision avoidance the method on a set of 2 PX4 based quadrotor platforms (figure 1) to demonstate our method on a real robotic system[3]. Each quad (see figure 6) is outfitted with a x86 based UP board and is tracked via a motion capture system. Each agent is sent its own position and maintains a noisy ellipsoidal estimate other agents' positions, which are filtered by a set-membership filter [BR71, LZW16, SS19] and updated via noise-corrupted measurements. Planning and closed loop trajectory control, as well as estimation and filtering, are all done on-board. The planner is updated at 80hz, while the filter is run at 100hz. Low-level control is done with a feed-forward PID velocity controller. We implemented both the direct goal projection method as well as the polynomial trajectory generation method. The quadrotors measure 23cm by 23cm by 2.3cm and are given an additional 30cm margin in the xy plane and a 60cm margin in the z axis. Figures 7 and 8 show the distance between agents over the course of the flight, as well as the final trajectory and the intermediate trajectories for the projection method (figure 7) and the polynomial method (figure 8).

[2] https://github.com/StanfordMSL/carp_ros

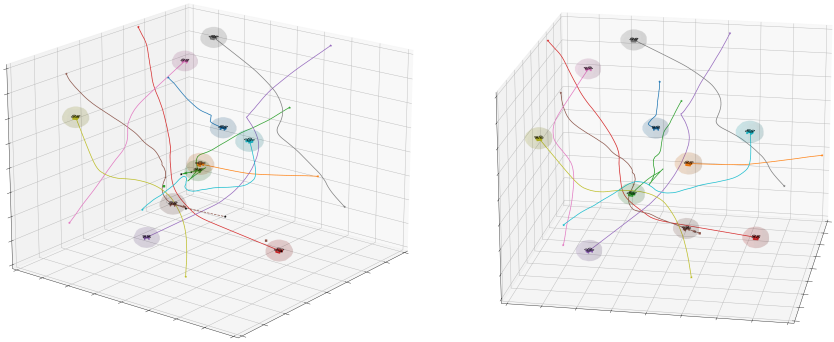
[3] A video of the simulations and experiments can be found at https://youtu.be/oz-bMovG4ow.



$t = 0$ $t = 60$



$t = 120$ $t = 180$



$t = 210$ $t = 250$

Figure 5: Six time instances of a 3D simulation of 10 agents. Each agent adds an ellipsoidal margin (shown) elongated in the z-axis to account for downwash affects.

5 Closing Remarks

In this work, we presented a scalable system that can work with simple or complex, distributed or centralized high-level planners to provide safe trajectories for a team of agents. Under the assumptions stated, we showed that collision avoidance is guaranteed provided each agent follows this method. However, we observe practical collision avoidance behavior even if only the ego agent follows this method. Computational performance results and simulations provide evidence that this algorithm can potentially be used in safety-critical applications for mobile robots with simple dynamics. Our open source library can also be used directly as a ROS package.

Acknowledgments. This work was supported in part by the Ford-Stanford Alliance program, and by DARPA YFA award D18AP00064. Guillermo Angeris is supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-1656518.

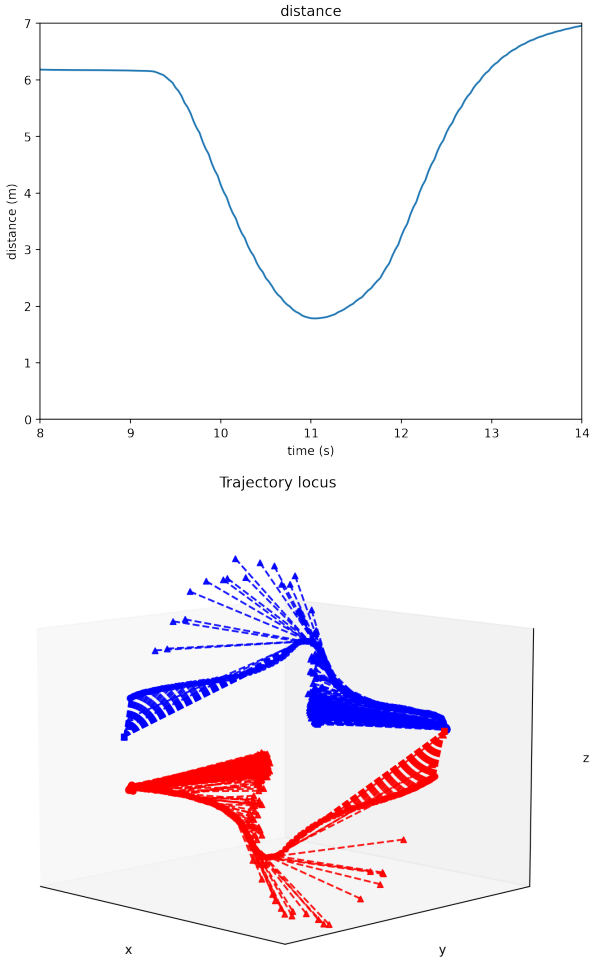


Figure 7: Projection only method: Inter-agent distance (top) and trajectory locus (bottom) for two quadrotors. The minimum distance between agents is 1.78m. The locus plot show the reprojected point (triangle) and goal point (square) over the course of the flight.

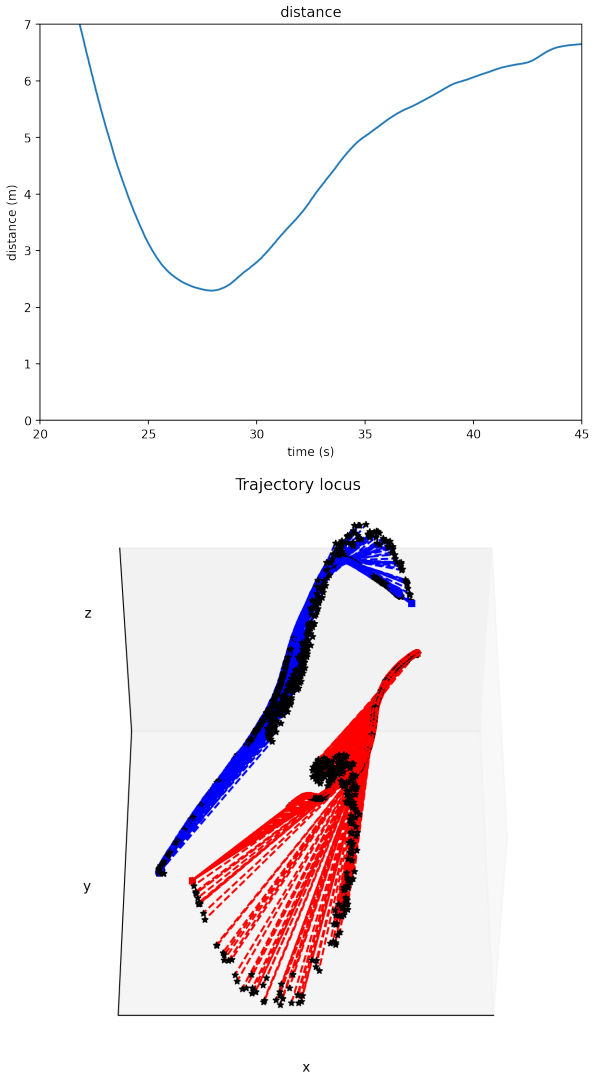


Figure 8: Bézier polynomial method: Inter-agent distance (top) and trajectory locus (bottom) for two quadrotors. The minimum distance between agents is 2.29m. The locus plot show the trajectory (dotted) and control points (star) as well as the final goal point (square) over the course of the flight.

References

[AM21a] Senthil Hariharan Arul and Dinesh Manocha. Swarmcco: Probabilistic reactive collision avoidance for quadrotor swarms under uncertainty. *IEEE Robotics and Automation Letters*, 6(2):2437–2444, 2021.

[AM21b] Senthil Hariharan Arul and Dinesh Manocha. V-rvo: Decentralized multiagent collision avoidance using voronoi diagrams and reciprocal velocity obstacles. *arXiv preprint arXiv:2102.13281*, 2021.

[ASD12] F. Augugliaro, A. P. Schoellig, and R. D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1917–1922, Oct 2012.

[ASS19] Guillermo Angeris, Kunal Shah, and Mac Schwager. Fast Reciprocal Collision Avoidance Under Measurement Uncertainty. In *2019 International Symposium of Robotics Research (ISRR)*, Hanoi, Vietnam, 2019.

[BABD06] Y. Becis-Aubry, M. Boutayeb, and M. Darouach. A stable recursive state estimation filter for models with nonlinear dynamics subject to bounded disturbances. In *Proceedings of the 45th IEEE Conference on Decision and Control*, pages 1321–1326, Dec 2006.

[BCH14] S. Bandyopadhyay, S. Chung, and F. Y. Hadaegh. Probabilistic swarm guidance using optimal transport. In *2014 IEEE Conference on Control Applications (CCA)*, pages 498–505, Oct 2014.

[BEKS17] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[BR71] D. Bertsekas and I. Rhodes. Recursive state estimation for a set-membership description of uncertainty. *IEEE Transactions on Automatic Control*, 16(2):117–128, Apr 1971.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[CCH15] Yufan Chen, Mark Cutler, and Jonathan P How. Decoupled multiagent path planning via incremental sequential convex programming. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5954–5961, 2015.

[CHTM12] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen. Collision avoidance under bounded localization uncertainty. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1192–1198, Oct 2012.

[CR16] Jiahao Chen and Jarrett Revels. Robust benchmarking in noisy environments. *arXiv preprint arXiv:1608.04295*, 2016.

[DCB13] Alexander Domahidi, Eric Chu, and Stephen Boyd. Ecos: An socp solver for embedded systems. In *2013 European Control Conference (ECC)*, pages 3071–3076. IEEE, 2013.

[DHL17] Iain Dunning, Joey Huchette, and Miles Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[GSK+17] B. Gopalakrishnan, A. K. Singh, M. Kaushik, K. M. Krishna, and D. Manocha. Prvo: Probabilistic reciprocal velocity obstacle for multi robot navigation under uncertainty. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1089–1096, Sep. 2017.

[HNR68] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968.

[JSCP15] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research*, 34(7):883–921, 2015.

[LKJ01] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[LVBL98] Miguel S. Lobo, Lieven Vandenberghe, Stephen Boyd, and Hervé Lebret. Applications of second-order cone programming. *Linear algebra and its applications*, 284(1-3):193–228, 1998.

[LZW16] Yushuang Liu, Yan Zhao, and Falin Wu. Ellipsoidal state-bounding-based set-membership estimation for linear system with unknown-but-bounded disturbances. *IET Control Theory & Applications*, 10(4):431–442, feb 2016.

[MCH14] Daniel Morgan, Soon-Jo Chung, and Fred Y Hadaegh. Model predictive control of swarms of spacecraft using sequential convex programming. *Journal of Guidance, Control, and Dynamics*, 37(6):1725–1740, 2014.

[MJWP16] X. Ma, Z. Jiao, Z. Wang, and D. Panagou. Decentralized prioritized motion planning for multiple autonomous uavs in 3d polygonal obstacle environments. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 292–300, 2016.

[MK11] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE international conference on robotics and automation*, pages 2520–2525. IEEE, 2011.

[PHAS17] J. A. Preiss, W. Hönig, N. Ayanian, and G. S. Sukhatme. Downwash-aware trajectory planning for large quadrotor teams. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 250–257, Sep. 2017.

[ŞHA19] Baskın Şenbaşlar, Wolfgang Hönig, and Nora Ayanian. Robust trajectory execution for multi-robot teams using distributed real-time replanning. In *Distributed Autonomous Robotic Systems*, pages 167–181. Springer, 2019.

[SHL+13] John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: science and systems*, volume 9, pages 1–10. Citeseer, 2013.

[SS19] Kunal Shah and Mac Schwager. Multi-agent cooperative pursuit-evasion strategies under uncertainty. In *Distributed Autonomous Robotic Systems*, pages 451–468. Springer, 2019.

[VdBGLM11] Jur Van den Berg, Stephen Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. *Springer Tracts in Advanced Robotics*, 70:3–19, 04 2011.

[VdBGS+16] Jur Van den Berg, Stephen Guy, Jamie Snape, Ming Lin, and Manocha. Rvo2 library: Reciprocal collision avoidance for real-time multi-agent simulation, 2016.

[VdBLM08] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 1928–1935. IEEE, 2008.

[WC11] G. Wagner and H. Choset. M*: A complete multirobot path planning algorithm with performance bounds. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3260–3267, Sep. 2011.

[ZA19] H. Zhu and J. Alonso-Mora. Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783, 2019.

[ZWBS17] Dingjiang Zhou, Zijian Wang, Saptarshi Bandyopadhyay, and Mac Schwager. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. *IEEE Robotics and Automation Letters*, 2(2):1047–1054, 2017.