# Hold Or take Optimal Plan (HOOP): A quadratic programming approach to multi-robot trajectory generation

Sarah Tang, Justin Thomas, and Vijay Kumar

In this work, we present Hold Or take Optimal Plan (HOOP), a centralized trajectory generation algorithm for labeled multi-robot systems operating in obstacle-free, two-dimensional, continuous workspaces. Given a team of $N$ robots, each with $n^{th}$-order dynamics, our algorithm finds trajectories that navigate vehicles from their start positions to non-interchangeable goal positions in a collision-free manner. The algorithm operates in two phases. In the motion planning step, a geometric algorithm finds a collision-free, piecewise-linear trajectory for each robot. In the trajectory generation step, each robot's trajectory is refined into a higher-order piecewise polynomial with a Quadratic Program (QP). The novelty of our method is in this problem decomposition. The motion plan, through abstracting away robots' dynamics, can be found quickly. It is then subsequently leveraged to construct collision avoidance constraints for $N$ decoupled QPs instead of a single, coupled optimization problem, decreasing computation time. We prove that this method is safe, complete, and generates smooth trajectories that respect robots' dynamics. We demonstrate the algorithm's practicality through extensive quadrotor experiments.

## 2 Introduction

Multi-robot teams have become attractive solutions for many challenging problems, such as warehouse management [1], delivery [2, 3], construction [4], and interactive exhibits [5]. These settings require robots to have the fundamental capability of autonomously navigating amongst their neighbors. We propose a solution to this *labeled* multi-robot planning problem. In this formulation, robots begin at given start positions and must navigate to pre-assigned, non-interchangeable goal positions. This is in contrast to the unlabeled problem, where goals can be visited by any robot, and the k-color problem, where goals must be visited by any robot of a particular type designation.

Many multi-robot planning algorithms have been presented in the past decades. In this work, we develop a centralized algorithm which assumes the communication network is always connected. Centralized approaches can broadly be characterized as solutions for discrete or continuous workspaces. In the discrete domain, proposed algorithms have used sampling-based [6] or search-based [7] planners to search the joint configuration space of the team. These approaches can guarantee safety, optimality, and completeness, however, their complexities grow exponentially with the number robots and quickly becomes computationally prohibitive. Alternative algorithms have been proposed to retain algorithmic guarantees while improving computational efficiency, such as specialized multi-robot search variants that reduce the algorithm's branching factor [8]. In sub-dimensional expansion [9], robots nominally search in their individual configuration spaces and search in a joint space when necessary. Alternatively, the problem has been formulated as a Mixed Integer Program [10, 11] and solved with optimization techniques. Other approaches, such rule-based methods [12, 13] and decomposition into independent subproblems [14, 15], trade optimality for computational improvements.

A major disadvantage of discrete approaches is their limitation to planning paths as series of coordinated movements between graph nodes. These paths contain corner turns that are dynamically suboptimal or even infeasible for dynamic vehicles. Continuous algorithms, on the other hand, plan trajectories that designate robots' positions as well as their higher derivatives. One approach is to first fix robots' paths, allowing them to include intersections. Robots' velocity profiles are then designed such that vehicles will not collide [16, 17, 18]. Priority-based methods plan paths for each robot in a designated sequential order, and lower priority robots are responsible for avoiding higher priority robots [19, 20, 21, 22]. Both these approaches sacrifice completeness and optimality to more quickly find a safe solution — a poorly chosen path or priority designation can lead to an unsolvable problem. In addition, they have largely been applied to robots with first-order dynamics, while we hope to focus on planning for robots with general $n^{th}$-order dynamics.

Probabilistic approaches extend single-robot planners, such as RRT* [23] and Roadmaps [24], to multi-robot planning. Optimization-based approaches have also been applied to aerial robots for collision avoidance between vehicles [25, 26, 27], as well as for formation control in general [28].

As expected, there is a tradeoff between computational speed and solution optimality. In fact, the problem of navigating moving disks is NP-Hard [29]. With this in mind, our algorithm, Hold Or take Optimal Plan (HOOP), yields solutions that may be suboptimal in path length, but are more computationally efficient to find. At a high level, the algorithm designs trajectories where robots move directly to their goals ("take Optimal Plan") whenever possible and temporarily deviate from this plan ("Hold") to safely navigate congested regions. In particular, we note there are two major sources of complexity: the exponential growth of the robots' joint configuration space as the number of robots increases and the increase in the number of trajectory parameters as the order of the robots' dynamics increases. Our method handles these challenges in two separate steps.

In the motion planning step, we simplify the robots' full dynamic model to a kinematic one and plan a set of safe piecewise-linear trajectories. We introduce maneuvers called Circular HOlding Patterns (CHOPs), which are executed during "hold" phases to navigate robots past each other in collision-prone regions. Past centralized [13, 11] and decentralized [25, 30, 31] methods have yielded avoidance maneuvers that also resemble circular orbits. However, our collision resolution maneuver is constructed geometrically. Leveraging techniques from solutions to the unlabeled problem, both in obstacle-free [32] and cluttered environments [33], CHOPs return maneuvers that have sub-optimal path length, but can be constructed in polynomial time with respect to the number of robots instead of the typical exponential complexity.

In the trajectory generation step, we refine the previous step's output into a smooth trajectory. We use the piecewise-linear paths to partition the available workspace into convex regions. We then optimize over the space of higher-order trajectories that pass through a designated series of regions. Similar methods of freespace partitioning has been used in single-robot planning around obstacles [34, 35] as well as decentralized multi-robot planning [36, 37]. Our Quadratic Program (QP) formulation, similar to approaches taken to find minimum-snap trajectories for quadrotors [**?**, 38, 41], leads to a decoupled QP for each robot, as opposed to a joint optimization problem over all trajectories for the team.

We validate our algorithm with extensive experiments on a quadrotor testbed. Previous experimental validation of multi-robot algorithms has been limited to ground robots that can be approximated as kinematic, such as the Pioneer [19], iRobot Create [42], and Dr. Robot Jaguar Lite [43]. A number of algorithms have also been applied to quadrotors operating in planar environments [25, 44]. This work includes both planar and three-dimensional experiments.

This paper builds on earlier work in planning [45] and trajectory generation [46] for labeled multi-robot systems. This work presents algorithmic improvements that unify the two approaches into a single paradigm. While some of the experimental results were previously presented in a conference paper [47], this work includes a more extensive experimental study of the HOOP algorithm.

This paper is organized as follows. Section 3 will present a formal problem statement and Sect. 4 will give an algorithm overview. Sections 5 and 6 will detail the motion planning and trajectory generation steps, respectively. Section 7 will provide related proofs. Sect. 8 will present experimental results, and finally, Sect. 9 will give concluding remarks.

## 3 Preliminaries

Consider a team of $N \in \mathbb{N}$ robots, operating in an obstacle-free, two-dimensional workspace. Let the integer set $\mathcal{I}_N = \{1, 2, ..., N\}$ index each robot's related variables. Let $\mathbf{x}^i(t) \in \mathbb{R}^2$ denote the position of robot $i$ at time $t$ and $\mathbf{u}^i(t) \in \mathbb{R}^2$ denote its input. Each robot is represented with the state:

$$\mathbf{X}^i(t) = \begin{bmatrix} \mathbf{x}^i(t) & \dot{\mathbf{x}}^i(t) & ... & \frac{d^{n-1}}{dt^{n-1}}\mathbf{x}^i(t) \end{bmatrix}^\top \in \mathbb{R}^{2n}$$

and governed by $n^{th}$-order dynamics:

$$\frac{d^n}{dt^n}\mathbf{x}^i(t) = \mathbf{u}^i(t). \tag{1}$$

Each robot is modeled as a disk of radius $R$. We use $\mathcal{B}(\mathbf{x}^i(t))$ to represent the set occupied by robot $i$'s finite extent:

$$\mathcal{B}(\mathbf{x}^i(t)) = \{\mathbf{x} \mid \|\mathbf{x} - \mathbf{x}^i(t)\|_2 \leq R\}.$$

We consider two robots $i \neq j \in \mathcal{I}_N$ to be in collision if:

$$\|\mathbf{x}^i(t) - \mathbf{x}^j(t)\|_2 < 2R.$$

We assume robots have a maximum velocity $v_{max}$.

We use $\mathcal{M}^i(t) : \mathbb{R} \to \mathbb{R}^2$ to represent a *motion plan* for robot $i$. $\mathcal{M}^i(t)$ is a piecewise-linear trajectory:

$$\mathcal{M}^i(t) = \begin{cases} \mathcal{M}^i_0(t) & t^i_0 \leq t < t^i_1 \\ \mathcal{M}^i_1(t) & t^i_1 \leq t < t^i_2 \\ ... \\ \mathcal{M}^i_{m^i-1}(t) & t^i_{m^i-1} \leq t \leq t^i_{m^i} \end{cases}$$
$$= \begin{cases} \mathbf{x}^i_{d,0} + \frac{t-t^i_0}{t^i_1-t^i_0}(\mathbf{x}^i_{d,1} - \mathbf{x}^i_{d,0}) & t^i_0 \leq t \leq t^i_1 \\ \mathbf{x}^i_{d,1} + \frac{t-t^i_1}{t^i_2-t^i_1}(\mathbf{x}^i_{d,2} - \mathbf{x}^i_{d,1}) & t^i_1 \leq t \leq t^i_2 \\ ... \\ \mathbf{x}^i_{d,m^i-1} + \frac{t-t^i_{m^i-1}}{t^i_{m^i}-t^i_{m^i-1}}(\mathbf{x}^i_{d,m^i} - \mathbf{x}^i_{d,m^i-1}) & t^i_{m^i-1} \leq t \leq t^i_{m^i} \end{cases}.$$

Note that $\mathcal{M}^i(t)$ is a vector function. This trajectory can be characterized with the *trajectory breaktimes*, $\mathcal{T}^i = \{t^i_0, t^i_1, ..., t^i_{m^i}\}$ and *trajectory waypoints*, $\mathcal{X}^i = \{\mathbf{x}^i_{d,0}, \mathbf{x}^i_{d,1}, ..., \mathbf{x}^i_{d,m^i}\}$. Thus, the motion plan can alternatively be represented by the set $\mathcal{M}^i = \{\mathcal{T}^i,\ \mathcal{X}^i\}$. We use $\mathcal{M}^i$ to represent both the piecewise-linear trajectory and its break-times and waypoints. We use the phrase "the path of" to refer to the line segment traced out by the motion plan.

$m^i$ denotes the number of waypoints in robot $i$'s motion plan. We use $\mathcal{M}(t) = \{\mathcal{M}^i(t) \mid i \in \mathcal{I}_N\}$ to denote the set of all motion plans and $\mathcal{M}_s(t) = \{\mathcal{M}^i_s(t) \mid i \in \mathcal{I}_N\}$ to denote the set of all robots' $i^{th}$ trajectory segments. "Motion plan" will refer to both the entire set $\mathcal{M}(t)$ and a specific $\mathcal{M}^i(t)$.

We use $\gamma^i(t) : \mathbb{R} \to \mathbb{R}^2$ to represent a *trajectory* for robot $i$. $\gamma^i(t)$ is a piecewise polynomial of degree $M$. Explicitly:

$$\gamma^i(t) = \begin{cases} \gamma^i_0(t) & t^i_0 \leq t < t^i_1 \\ \gamma^i_1(t) & t^i_1 \leq t < t^i_2 \\ ... \\ \gamma^i_{m_i-1}(t) & t^i_{m^i-1} \leq t \leq t^i_{m^i} \end{cases}$$
$$= \begin{cases} \sum_{j=0}^M c^i_{0,j} t^i & t^i_0 \leq t < t^i_1 \\ \sum_{j=0}^M c^i_{1,j} t^i & t^i_1 \leq t < t^i_2 \\ ... \\ \sum_{j=0}^M c^i_{m^i-1,j} t^i & t^i_{m^i-1} \leq t \leq t^i_{m^i} \end{cases}.$$

Again, $\gamma^i(t)$ is a vector function, and $c^i_{s,j} \in \mathbb{R}^2$. We use $\gamma(t) = \{\gamma^i(t) \mid i \in \mathcal{I}_N\}$ to denote the set of all trajectories. We will use "trajectory" to refer to both a specific $\gamma^i(t)$ as well as the set $\gamma(t)$. Note that a motion plan is simply a trajectory where $M = 1$. However, we will use $\mathcal{M}^i(t)$ to refer explicitly to the output of the motion planning algorithm step and $\gamma^i(t)$ to refer to the output of the trajectory generation step.

Define $T = \max_{i \in \mathcal{I}_N} t^i_{m_i}$. A motion plan or trajectory, $\gamma$, is *safe* if:

$$\|\gamma^i(t) - \gamma^j(t)\|_2 \geq 2R \quad \forall i \neq j \in \mathcal{I}_N, \forall t \in [0, T].$$

The *labeled multi-robot planning problem* is explicitly stated as follows. Given a set of start positions, $\mathbf{s}^i \in \mathbb{R}^2$, and assigned goal positions, $\mathbf{g}^i \in \mathbb{R}^2$ for each robot, such that:

$$\begin{aligned} \|\mathbf{s}^i - \mathbf{s}^j\|_2 &\geq 2\sqrt{2}R \quad \forall i \neq j \in \mathcal{I}_N \\ \|\mathbf{g}^i - \mathbf{g}^j\|_2 &\geq 2\sqrt{2}R \quad \forall i \neq j \in \mathcal{I}_N, \end{aligned} \tag{2}$$

plan a set of trajectories $\gamma(t)$ such that:

```
Algorithm 1 $\gamma = \mathrm{HOOP}(\mathbf{s}, \mathbf{g}, n, N, R, v_{max})$
1: $\mathcal{M}$ := Plan Motions($\mathbf{s}, \mathbf{g}, N, R, v_{max}$)
2: $\gamma$ := Generate Trajectories($\mathcal{M}, n, N, R, v_{max}$)
```

1. Each robot begins and ends at the designated positions:

$$\gamma^i(0) = \mathbf{s}^i, \quad \gamma^i(T) = \mathbf{g}^i \quad \forall i \in \mathcal{I}_N.$$

2. $\gamma(t)$ is safe.

We use $\mathbf{s}$ and $\mathbf{g}$ to denote all start and goal positions for the team. For brevity, when not evaluating a time-varying variable at a specific time, we drop "$(t)$" from the notation.

## 4 Hold Or take Optimal Plan (HOOP): Algorithm overview

Algorithm 1 outlines the proposed algorithm, Hold Or take Optimal Plan (HOOP). The algorithm contains two main steps. Section 5 will detail the motion planning step, Line 1, which is outlined in Algorithm 5. Section 6 will detail the trajectory generation step, Line 2, outlined in Algorithm 7. Throughout this paper, we use := in algorithms to specify variable assignment and $\leftarrow$ to denote addition to a set.

## 5 Motion Planning

wIn this section, we present an iterative, geometric algorithm to find a safe motion plan for each robot. This path does not need to be dynamically feasible, as it will be further refined in the trajectory generation step. As a result, we temporarily assume robots have first-order dynamics (ie. $n = 1$).

### 5.1 Concurrent Assignment and Planning of Trajectories (CAPT): A solution to the unlabeled problem

Our motion planning algorithm takes inspiration from Concurrent Assignment and Planning of Trajectories (CAPT), a solution to the unlabeled planning problem [32], which will be described briefly here.

In the unlabeled problem, a team of $N$ robots must navigate to $N_g$ goals. However, it does not matter which robot arrives at which goal, as long as all goals are visited. In the context of this paper, we assume $N_g = N$. Solving this problem requires both assignment of goals to robots and generation of safe trajectories to those goals. However, the coupling of assignment and trajectory planning paradoxically decreases the problem complexity.

Let $\phi : \mathcal{I}_N \to \mathbb{R}^2$ represent the goal assignment, where $\phi^i = \mathbf{g}^j$ indicates $\mathbf{g}^j$ is assigned to the robot starting at $\mathbf{s}^i$. CAPT finds the assignment $\phi$ that minimizes the sum of the distances squared traveled by all robots, explicitly:

$$\underset{\phi}{\mathrm{argmin}} \sum_{i=1}^{N} \|\mathbf{s}^i - \phi^i\|_2^2. \tag{3}$$

This can be found using the Hungarian algorithm [48] in $O(N^3)$ time.

For first-order robots, the optimal trajectory between a start position, $\mathbf{x}_1$, and a goal position, $\mathbf{x}_2$ is found by optimizing the cost functional:

$$\underset{\gamma(t)}{\mathrm{argmin}} \int_0^T \|\frac{d}{dt}\gamma^i(t)\|_2^2 dt,$$

subject to:

$$\gamma^i(0) = \mathbf{x}_1, \ \ \gamma^i(T) = \mathbf{x}_2. \tag{4}$$

The solution to Eq. 4 is:

$$\gamma^i(t) = \mathbf{x}_1 + \frac{t}{T}(\mathbf{x}_1 - \mathbf{x}_2). \tag{5}$$

Given a CAPT assignment and $v_{max}$, the maximum time required for any robot to reach its goal is given by:

$$T = \max_{i \in \mathcal{I}_N} \frac{\|\mathbf{s}^i - \phi^i\|_2}{v_{max}} \tag{6}$$

Each robot travels from $\mathbf{s}^i$ to $\phi^i$ along a trajectory of the form given by Eq. 5 in time given by Eq. 6. If start and goals satisfy the separation constraints in Eq. 2, these trajectories are provably safe.

The CAPT algorithm is computationally efficient, finding trajectories for up to 500 agents in less than 10 seconds [32]. Note, however, that robots' solution trajectories must be *synchronized*, that is, all robots must arrive at their goals simultaneously at $T$. Though we will impose this synchronicity constraint between certain waypoints, our labeled motion planning algorithm does not ultimately require robots to arrive at their goals simultaneously. We will refer to a constant-velocity, straight-line towards a goal as an "optimal trajectory".

### 5.2 A Motivating Example: Navigating towards labeled goals with the unlabeled solution

A key insight of provided by CAPT is that the coupling of goal assignment and trajectory generation can paradoxically increase computational efficiency. Unfortunately, in the labeled problem, goals are pre-assigned to robots and cannot be directly reassigned to conform to the minimum distance squared assignment necessary for CAPT's collision avoidance guarantees to hold. However, with appropriately chosen intermediate waypoints, we can navigate robots to their labeled goals by solving a series of unlabeled problems.

Figure 1 presents an example capturing this idea. Here, and throughout the remainder of this paper, circles represent start positions $\mathbf{s}$, and stars of the same color represent the assigned goals $\mathbf{g}$. The problem's goal assignment is not the minimum distance squared assignment. If robots attempt to execute optimal trajectories found using Eqs. 5 and 6 on this assignment, they collide at the outlined positions.

In Fig. 1b, we place two *intermediate waypoints*, denoted $\mathbf{x}_w$ and represented as black squares, into the workspace. Considering $\mathbf{s}$ as the set of start positions and $\mathbf{x}_w$ as the set of goals, we can apply CAPT and find optimal trajectories to the waypoints. Figure 1c pictures these trajectories.

We can then apply CAPT again, using $\mathbf{x}_w$ as the set of start positions and $\mathbf{g}$ as the set of goal positions. Figure 1d pictures these trajectories. The robots safely navigate to goals assigned by CAPT, which now coincides with the desired labeled assignment.

By deliberately choosing the location of the intermediate waypoints in this manner, we translate the labeled planning problem into two instances of the unlabeled problem, allowing us to leverage the computational benefits of the CAPT algorithm. The motion planning step of the HOOP algorithm uses this idea to place waypoints throughout the workspace to navigate robots towards their goals.

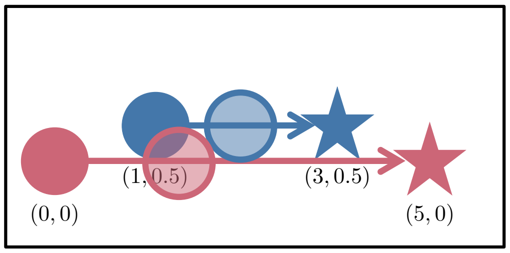
### 5.3 Circular HOlding Pattern (CHOP): A collision avoidance maneuver

In general, the problem of identifying sets of waypoints that transform a labeled planning problem to a series of unlabeled problems is hard. In this section, we present a geometric method for constructing these waypoint sets. Note that the identified waypoint sets are neither unique nor optimal, however, are computationally efficient to find even for large numbers of robots. The resulting maneuver will be referred to as a Circular HOlding Pattern (CHOP).

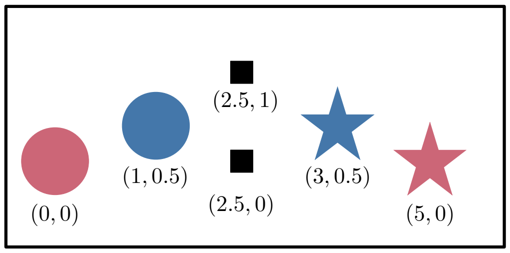
A CHOP, or "holding pattern", is a motion plan that safely navigates robots from a set of start positions, $\mathbf{s}_h$, to assigned goal positions $\mathbf{g}_h$. We characterize a CHOP with the parameters $h = \{\tau_{s,h}, \mathcal{R}_h, \mathbf{s}_h, \mathbf{g}_h, \mathbf{x}_{c,h}, N_h, R_h, \tau_{f,h}, \mathcal{M}_h\}$. These are defined in Table 1 and are determined using Algorithm 2. Throughout this section, Fig. 2 will be used as a demonstrative example.

#### 5.3.1 Determination of CHOP parameters.

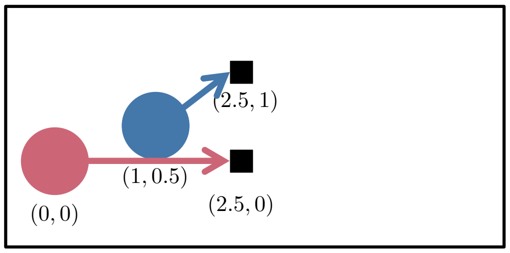
Assume robots begin at their designated start positions. Then, $\tau_{s,h} = 0$, $\mathcal{R}_h = \mathcal{I}_N$, $\mathbf{s}_h = \mathbf{s}$, $\mathbf{g}_h = \mathbf{g}$. These parameters are inputs to Algorithm 2, which defines the full CHOP. First, we define a set of *intermediate*



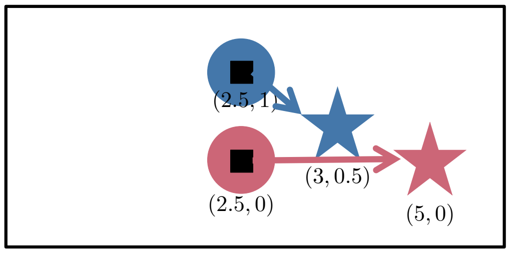
(a) Taking synchronized optimal trajectories to goals leads to a collision at the outlined positions.



(b) We place a two intermediate waypoints in the space, denoted $\mathbf{x}_w$ and pictured as black squares.



(c) We apply CAPT, using the sets $\mathbf{s}$ as start positions and $\mathbf{x}_w$ as goal positions. Robots follow the resulting optimal trajectories to intermediate waypoints.



(d) We apply CAPT again, using the sets $\mathbf{x}_w$ as start positions and $\mathbf{g}$ as goal positions. The resulting optimal trajectories navigate robots to their assigned goals.

Figure 1: Motivating example. Robots start at positions $\mathbf{s}$, indicated by circles, and must navigate to assigned goal positions $\mathbf{g}$, represented by stars of the same color. The placement of intermediate waypoints can transform the labeled multi-robot planning problem into two unlabeled problems.

Table 1: Definitions of CHOP parameters.

| | |
|---|---|
| $\tau_{s,h} \in \mathbb{R}$ | start time — time robots begin to move towards entry waypoints |
| $\mathcal{R}_h \in \mathbb{Z}$ | indices of robots in the CHOP |
| $\mathbf{s}_h, \mathbf{g}_h \in \mathbb{R}^{2\vert\mathcal{R}\vert}$ | set of start and assigned goal positions |
| $\mathbf{x}_{c,h} \in \mathbb{R}^2$ | center of CHOP |
| $N_h \in \mathbb{Z}$ | number of intermediate waypoints |
| $R_h \in \mathbb{R}$ | radius of CHOP |
| $\tau_{f,h} \in \mathbb{R}^{\vert\mathcal{R}\vert}$ | exit time for each robot — time each robot leaves its exit waypoint |

*waypoints* that robots will visit en-route to their goals. Line 3 defines the center, $\mathbf{x}_{c,h}$, as the average position of all start positions. Line 5 designates the number of intermediate waypoints, $N_h$, as twice the number of robots. Line 10 defines these waypoints as $N_h$ regular points along the circumference of a circle with center $\mathbf{x}_{c,h}$ and radius $R_h$.

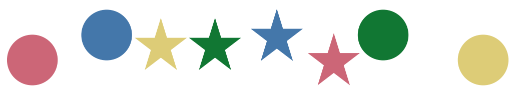
Line 6 calculates the radius itself. Since $\mathbf{x}_{c,h}$ is fixed, variation in $R_h$ allows for clearance guarantees between intermediate waypoints and goals. We find $R_h$ with:

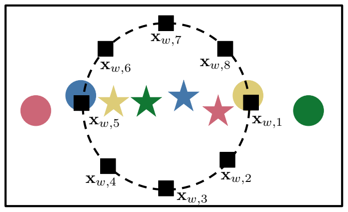
$$\min_{R_h \in \mathbb{R}} R_h \tag{7}$$

subject to:

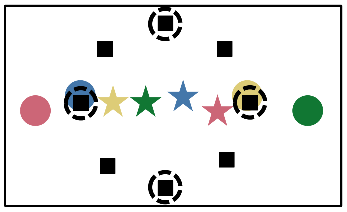
1. Every other intermediate waypoint is at least $2\sqrt{2}R$ away from each other.
2. Every intermediate waypoint is at least $2\sqrt{2}R$ away from all goals $\mathbf{g}_h$.
3. The path between any two adjacent waypoints is at least $2\sqrt{2}R$ away from all goals $\mathbf{g}_h$.

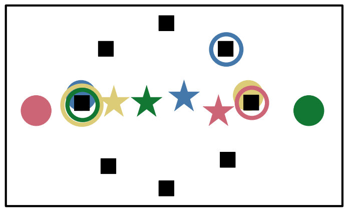
With the center and number of intermediate waypoints fixed, the constraints in Eq. 7, and hence $R_h$, can be found analytically. Note that start and goal positions can be both inside and outside the circle defined by

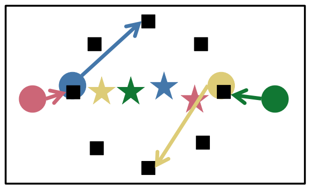

(a) Problem geometry.

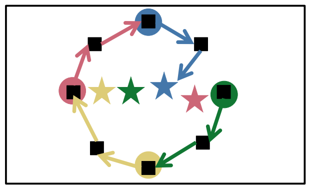


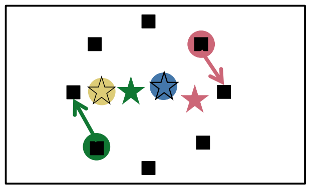
(b) We identify a set of intermediate waypoints, pictured as black squares.


(c) We identify a set of potential entry waypoints, indicated by black circles.


(d) We designate an exit waypoint and priority set for each robot. Each robot's assigned waypoint is indicated in a circle of the same color. In concentric sets, inner circles are higher priority.


(e) $\mathcal{M}_0$ for all robots. Vehicles move from their start positions to their assigned entry waypoints.


(f) $\mathcal{M}_1, \mathcal{M}_2$ for robots. Vehicles circle the holding pattern until they reach their exit waypoint.


(g) $\mathcal{M}_3, \mathcal{M}_4$ for robots. The yellow robot passes its exit waypoint because the green robot, with higher priority, has not exited.

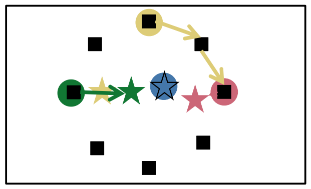

(h) $\mathcal{M}_5, \mathcal{M}_6$ for robots. Vehicles continue to circle until all robots have exited.

Figure 2: Construction and execution of a CHOP for a four-robot example. Robots start at circles and must navigate to stars of the same color.

$\mathbf{x}_{c,h}$ and radius $R_h$. Fig. 2a visualizes the placement of intermediate waypoints for the example problem as black squares.

### 5.3.2 Identification of entry waypoints.

Line 12 identifies a set of *entry waypoints*. Each robot will travel from their start position to one of these waypoints as a means of entering the CHOP.

### 5.3.3 Assignment of exit waypoints and priority sets.

Lines 14–16 assign an exit waypoint to each robot. Each robot will leave the CHOP and move towards its goal from its exit waypoint, which is chosen as the intermediate waypoint closest to its goal.

Lines 18–26 designate a *priority set*, a set of indices $\mathcal{P}^i$, for each robot. A robot cannot exit the CHOP until all robots in $\mathcal{P}^i$ have already exited. Conceptually, the condition in Line 22 forces robots whose goals are in the path of neighbors' trajectories to wait. This is similar to the priority designation used by [33] in the unlabeled domain.

Exit waypoint assignments are pictured in Fig. 2d, where each robot's waypoint is indicated by a circle of the same color. When concentric circles are present, inner circles indicate higher priorities. Explicitly, the priority sets are:

$$\mathcal{P}^{blue} = \mathcal{P}^{red} = \mathcal{P}^{green} = \emptyset$$
$$\mathcal{P}^{yellow} = \{green\}.$$

### 5.3.4 Derivation of a safe motion plan

**Algorithm 2** $h$ = Find CHOP$(\tau_{s,h}, \mathcal{R}_h, \mathbf{s}_h, \mathbf{g}_h, R, v_{max})$

```
1:  $N_m$ := size of $\mathcal{R}$
2:  // Find CHOP center.
3:  $\mathbf{x}_h := \frac{\sum_{i \in \mathcal{I}_{N_m}} \mathbf{s}_h^i}{N_m}$
4:  // Find number of intermediate waypoints.
5:  $N_h := 2N_m$
6:  $R_h$ := solve Eq. 7.
7:  // Identify intermediate waypoints.
8:  $\Delta\theta := \frac{2\pi}{N_h}$
9:  $\theta := \{(i-1)\Delta\theta \;\; \forall\, i \in \mathcal{I}_{N_h}\}$
10: $\mathbf{x}_w := \{\mathbf{x}_h + R_h[\cos(\theta_i) \;\; \sin(\theta_i)]^T \;\; \forall\, \theta_i \in \theta\}$
11: // Identify entry waypoints.
12: $\mathbf{X}_w := \{\mathbf{x}_{w,1}, \mathbf{x}_{w,3}, ..., \mathbf{x}_{w,N_h-1}\}$
13: // Assign exit waypoints.
14: for all $i \in \mathcal{I}_{N_h}$ do
15:     $\phi_g^i := \text{argmin}_{x_j \in \mathbf{x}_w} \|x_j - \mathbf{g}_h^i\|_2$
16: end for
17: // Determine priority sets.
18: $\mathcal{P}^i := \emptyset \;\; \forall i \in \mathcal{I}_{N_h}$
19: for all $i \in \mathcal{I}_{N_h}$ do
20:     for all $j \in \mathcal{I}_{N_h} \backslash i$ do
21:         $\psi = \phi_g^j + \beta(\mathbf{g}^j - \phi_g^j)$
22:         if $\exists \beta \in [0, 1]$ such that $\mathcal{B}(\mathbf{g}_h^i) \cap \mathcal{B}(\psi(\beta)) \neq \emptyset$ then
23:             $\mathcal{P}^i \leftarrow j$
24:         end if
25:     end for
26: end for
27: $(\mathcal{M}_h, \tau_{f,h})$ := Find CHOP Motion Plan$(\tau_{s,h}, \mathcal{R}_h, \mathbf{s}_h, \mathbf{x}_w, N_h, \mathbf{X}_w, \phi_g, \mathcal{P}, \mathbf{g}_h, v_{max})$
28: $h := \{\tau_{s,h}, \mathcal{R}_h, \mathbf{s}_h, \mathbf{g}_h, \mathbf{x}_{c,h}, N_h, R_h, \tau_{f,h}, \mathcal{M}_h\}$
```

Finally, in Line 27, we use the determined CHOP parameters to construct a corresponding motion plan. This mechanism is described in Algorithm 3, and Figs. 2e–2h illustrate the resulting motion plan on our running example.

To bring robots safely into the CHOP, Line 2 of Algorithm 3 runs CAPT using $\mathbf{s}_h$ as start positions and $\mathbf{X}_w$ as goal positions. The first trajectory segment of the motion plan is given by the trajectories returned by CAPT. Figure 2e illustrates this step.

$\mathbf{X}_w$ becomes a new set of start positions, $\mathbf{s}_{curr}$. Lines 7–14 then construct a new set of goal positions. For each robot in $\mathbf{s}_{curr}$, its goal is added to $\mathbf{g}_{curr}$ if its exit conditions are met. Otherwise, the intermediate waypoint that is adjacent to its current position, in the counter-clockwise direction, is added. Line 16 uses CAPT to find the next set of trajectory segments.

Lines 20–26 update the set $\mathcal{R}_g$ to contain robots that have arrived at their goals. As indicated by Lines 17–18, these robots will remain stationary at their goals. A new $\mathbf{s}_{curr}$ and $\mathbf{g}_{curr}$ is constructed using the robots still in the CHOP and CAPT is again used to obtain the next trajectory segments. This process continues until all robots have reached their goals. When a robot reaches its goal, Line 23 records its *exit time* $\tau_{f,h}^i$, the time it leaves its exit waypoint.

Fig. 2f pictures the resulting motion when no robots have met their exit condition; namely, robots will move around the CHOP in a counter-clockwise direction. Otherwise, as illustrated in Figures 2g and 2h, robots that have met their exit conditions will move to their goals while robots that have not will continue to rotate. Lemma 5.1 will prove this is the returned behavior.

**Remark 1** *In the case illustrated in Fig. 2f, CAPT will return two solutions: one where all robots rotate clockwise and another where all robots rotate counter-clockwise. We have chosen to always select the counter-clockwise solution. As both solutions result in the same cost, as given by Eq. 3, this does not introduce additional sub-optimality.*

**Algorithm 3** $(\mathcal{M}_h, \tau_{f,h})$ = Find CHOP Motion Plan$(\tau_{s,h}, \mathcal{R}_h, \mathbf{s}_h, \mathbf{x}_w, N_h, \mathbf{X}_w, \phi_g, \mathcal{P}, \mathbf{g}_h, v_{max})$

```
1:  // Move robots into the CHOP.
2:  $\mathcal{M}_0 := \text{CAPT}(\mathbf{s}_h, \mathbf{X}_w)$
3:  $\mathbf{s}_{curr} := \mathbf{X}_w$, $\mathcal{R}_g := \emptyset$, $j := 1$
4:  // Circle the CHOP and exit when possible.
5:  while $\mathcal{R}_g \neq \mathcal{R}_h$ do
6:    // Construct the goal set for robots still in CHOP.
7:    $\mathbf{g}_{curr} := \emptyset$
8:    for all $i \in \mathcal{R}_h \setminus \mathcal{R}_g$ do
9:      if $\mathbf{x}^i_{d,j} == \phi^i_g$ and $\mathcal{P}^i == \emptyset$ then
10:       $\mathbf{g}_{curr} \leftarrow \mathbf{g}^i_h$
11:     else
12:       $\mathbf{g}_{curr} \leftarrow$ counter-clockwise adjacent intermediate waypoint
13:     end if
14:   end for
15:   // Get next trajectory segment.
16:   $\mathcal{M}_{capt} := \text{CAPT}(\mathbf{s}_{curr}, \mathbf{g}_{curr})$
17:   $\mathcal{M}_{stationary} :=$ stationary trajectories for $i \in \mathcal{R}_g$
18:   $\mathcal{M}_j \leftarrow \mathcal{M}_{capt} \cup \mathcal{M}_{stationary}$
19:   // Record robots that arrived at goals.
20:   $\mathcal{R}_{g,new} := \emptyset$
21:   for all $i \in \mathcal{R}_h \setminus \mathcal{R}_g$ do
22:     if $\mathbf{x}^i_{d,j+1} == \mathbf{g}^i_h$ then
23:       $\mathcal{R}_{g,new} \leftarrow i$, $\tau^i_{f,h} := t_j$
24:     end if
25:   end for
26:   $\mathcal{R}_g \leftarrow \mathcal{R}_{g,new}$
27:   // Define new start set as robots not yet at their goals.
28:   $\mathbf{s}_{curr} := \mathbf{x}_{d,j+1} \setminus \{\mathbf{x}^i_{d,j+1} \mid i \in \mathcal{R}_g\}$
29:   $j := j + 1$
30: end while
```

Note that a robot's goal is only added to the set $\mathbf{g}_{curr}$ when both its exit conditions have been met. For example, in Fig. 2f, when the blue robot reaches its exit waypoint, it proceeds to its goal. The yellow robot is also at its exit waypoint, however, the green robot, which is in its priority set, has not exited. As a result, the yellow robot's goal is not added to $\mathbf{g}_{curr}$ and it continues around the CHOP.

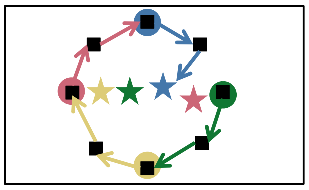
**Remark 2** *As the placement of intermediate waypoints and the definition of sets $\mathbf{s}_{curr}$ and $\mathbf{g}_{curr}$ at each step are deliberately chosen to produce the behavior illustrated in Fig. 2, the solution to each iteration of CAPT, with the exception of Line 2, is known a priori. Thus, in implementation, the motion plan $\mathcal{M}_h$ can be defined by running CAPT only once, improving computation speed.*

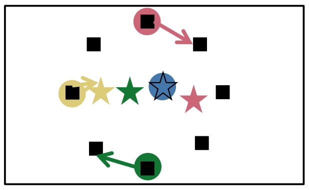
As an example, the motion plan corresponding to the blue robot in Fig. 2 is:

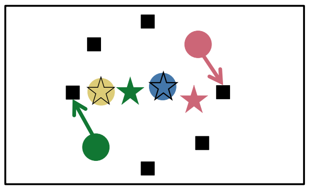
$$\mathcal{M}^{blue} = \begin{cases} \mathbf{s}^{blue} + \frac{t}{t_1}(\mathbf{x}_{w,7} - \mathbf{s}^{blue}) & 0 \leq t < t_1 \\ \mathbf{x}_{w,7} + \frac{t}{t_2 - t_1}(\mathbf{x}_{w,8} - \mathbf{x}_{w,7}) & t_1 \leq t < t_2 \\ \mathbf{x}_{w,8} + \frac{t}{t_3 - t_2}(\mathbf{g}^{blue} - \mathbf{x}_{w,8}) & t_2 \leq t < t_3 \\ \mathbf{g}^{blue} & t_3 \leq t < t_{12} \end{cases}$$

For this example, $\tau^{blue}_{f,h} = t_2$.

**Remark 3** *Figure 3 illustrates the necessity of prioritization. In this scenario, we remove the green robot from the yellow robot's priority set. In Fig. 3b, the yellow robot is allowed to exit to its goal. In Fig. 3d, we see that the green robot no longer has a safe trajectory to its goal. The priority set ensures that it will be possible for robots to remain at their goals once they've reached them.*


(a) $\mathcal{M}_1, \mathcal{M}_2$ for robots.


(b) $\mathcal{M}_3$ for robots. The yellow robot is allowed to exit to its goal.


(c) $\mathcal{M}_4$ for robots. The green robot progresses towards its exit waypoint.

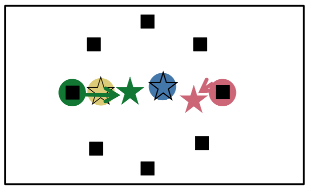

(d) $\mathcal{M}_5$ for robots. The green robot cannot safely exit to its goal.

Figure 3: Extension of Fig. 2 — motion plan when all priority sets are empty. This lack of prioritization leads to a collision.

*Note that Constraint 3 of Eq. 7 ensures that the priority condition in Line 22 of Algorithm 2 will never be true with respect to a robot moving between two intermediate waypoints. In other words, the CHOP radius is designed such that a robot stationary at its goal will not obstruct the motion of neighbors progressing around the CHOP.*

**Lemma 5.1** *In the motion plan returned by Algorithm 3, $\mathcal{M}_0$ brings all robots to the set of entry waypoints $\mathbf{X}_w$. Each robot then circles the CHOP until its exit conditions are met. The first time a robot's exit conditions are met, it will move to and remain stationary at its goal.*

**Proof 1** *Line 2 of Algorithm 3 explicitly brings all robots to the set of entry waypoints.*

*Consider sets $\mathbf{s}_{curr}$ and $\mathbf{g}_{curr}$ at the first iteration of Algorithm 3. $\mathbf{s}_{curr} = \mathbf{X}_w$ contains every other intermediate waypoint and $\mathbf{g}_{curr}$ contains waypoints adjacent to those in $\mathbf{X}_w$. Since $\mathbf{s}_{curr}$ and $\mathbf{g}_{curr}$ are disjoint, no robot can remain stationary in the CHOP. This remains true at all following times. Thus, unless a robot is at its goal, it will always move to another intermediate waypoint or to its goal.*

*Consider the sum of distances squared cost for the CAPT problem yielding $\mathcal{M}_s$ for any $s > 0$:*

$$\sum_{i \in \mathcal{R}_h \setminus \mathcal{R}_g} \|\mathbf{s}^i_{curr} - \phi^i_{curr}\|_2^2,$$

*where $\phi^i$ indicates the position in $\mathbf{g}_{curr}$ assigned to $\mathbf{s}^i_{curr}$ by CAPT. $\mathbf{s}^i_{curr}$ can only be an intermediate waypoint, while $\phi^i_{curr}$ can be either an intermediate waypoint or a goal.*

*Suppose $\phi^i_{curr}$ is an intermediate waypoint, $\mathbf{x}_{w,d}$. From Line 12 of Algorithm 3, $\mathbf{s}_{curr}$ must contain its adjacent waypoint in the clockwise direction, $\mathbf{x}_{w,d-1}$. Since $\mathbf{s}_{curr}$ only contains other intermediate waypoints, we know:*

$$\|\mathbf{x}_{w,d} - \mathbf{x}_{w,d-1}\|_2^2 \leq \|\mathbf{x}_{w,d} - \mathbf{s}^j_{curr}\|_2^2 \qquad \forall \mathbf{s}^j_{curr} \neq \mathbf{x}_{w,d-1} \in \mathbf{s}_{curr}.$$

*Suppose $\phi^i_{curr}$ is a goal, $\mathbf{g}^k$. Line 10 of Algorithm 3 ensures that $\mathbf{g}^k$ is in the goal set only when $\phi^k_g$, the exit waypoint for robot $k$, is in the start set and robot $k$ has met its exit criteria. Line 15 of Algorithm 2 ensures that:*

$$\|\mathbf{g}^k - \phi^k_g\|_2^2 \leq \|\mathbf{g}^k - \mathbf{s}^j_{curr}\|_2^2 \ \forall \mathbf{s}^j_{curr} \neq \phi^k_g \in \mathbf{s}_{curr}.$$

*From this, we see that an intermediate waypoint in $\mathbf{g}_{curr}$ must be reached from the clockwise adjacent intermediate waypoint and a goal in $\mathbf{g}_{curr}$ must be reached by its corresponding robot after it has fulfilled its exit criteria. Changing this assignment will attain a higher distance squared cost, so any possible permutation in the assignment will result in a total higher cost. The only except occurs when both the intermediate waypoints clockwise and counter-clockwise adjacent to $\mathbf{x}_{w,d}$ is in $\mathbf{s}_{curr}$, however, as stated in Remark 1, we choose the solution that causes robots to rotate counter-clockwise. Line 17 of Algorithm 3 ensures robots remain stationary at their goals after reaching them. Thus, CAPT iterations yield the described behavior.* ■

**Lemma 5.2** *Motion plans derived from CHOPs are safe.*

**Algorithm 4** $\mathcal{M}$ = CHOPs to Motion Plan $(\mathbf{s}, \mathbf{g}, \mathcal{H}, v_{max})$

```
1: for $i \in \mathcal{I}_N$ do
2:   $\mathcal{T}^i := \{0, \frac{\|\mathbf{s}^i - \mathbf{g}^i\|_2}{v_{max}}\}$, $\mathcal{X}^i := \{\mathbf{s}, \mathbf{g}\}$
3: end for
4: for $h \in \mathcal{H}$ do
5:   for $i \in \mathcal{R}_h$ do
6:     $\mathcal{T}^i \leftarrow \mathcal{T}^i_h, \mathcal{X}^i \leftarrow \mathcal{X}^i_h$
7:   end for
8: end for
```

**Proof 2** *Consider, at each time step, the robots not yet at their goals (ie. the set $\mathcal{R}_h \setminus \mathcal{R}_g$). The robots' trajectory segments are returned by CAPT and will be safe if the separation conditions in Eq. 2 are met.*

*For $\mathcal{M}_0$, points in $\mathbf{s}_{curr}$ satisfy Eq. 2 by assumption. Condition 1 of Eq. 7 guarantees $\mathbf{g}_{curr} = \mathbf{X}_w$, which contains every other intermediate waypoint, satisfies the separation conditions. Thus, $\mathcal{M}_0$ is safe.*

*Since at each iteration, all robots rotate or exit, in subsequent trajectory segments, $\mathbf{s}_{curr}$ will contain only intermediate waypoints that continue to satisfy the separation conditions. For the same reason, intermediate waypoints in $\mathbf{g}_{curr}$ will also satisfy the separation conditions with respect to each other. Condition 2 of Eq. 7 guarantees goals in $\mathbf{g}_{curr}$ are sufficiently separated from intermediate waypoints. Two goals in $\mathbf{g}_{curr}$ are sufficiently separated by assumption. Thus, all points in $\mathbf{g}_{curr}$ are also sufficiently separated and all CAPT assignments yield safe trajectories.*

*The remaining possibility of collision is between a moving robot and a stationary robot that has already arrived at its goal. Algorithm 2, Lines 18 –26 guarantee that robots moving from their exit waypoints to their goals will not collide with stationary robots. Eq. 7 guarantees that robots moving between intermediate waypoints will not collide with stationary robots. Thus, the motion plan is safe.* ■

## 5.4 Algorithm Definition

We can always coordinate robots in a single CHOP to safely navigate to goals. However, in the worst case, all robots will enter a CHOP that contains all goal positions. To mitigate this effect, HOOP allows robots to travel directly towards their goals ("take Optimal Plan") when possible and navigate past each other using CHOPs ("Hold") when necessary.

To begin, let $\mathcal{H}$ denote a set of CHOPs. Each CHOP $h \in \mathcal{H}$ contains robots $\mathcal{R}_h \subseteq \mathcal{I}_N$. Robots enter the CHOP from $\mathbf{s}_h$, their current positions at start time, $\tau_{s,h}$. Robot $r$ is in CHOP $h$ at time $t$ if $r \in \mathcal{R}_h$ and $t \in [\tau_{s,h}, \tau^r_{f,h}]$. A set $\mathcal{H}$ is *valid* if:

$$\{r \in \mathcal{I}_N \mid \exists h, \widetilde{h} \in \mathcal{H}, r \in \mathcal{R}_h \cap \mathcal{R}_{\widetilde{h}}, \tau_{s,h} \in [\tau_{s,\widetilde{h}}, \tau^r_{f,\widetilde{h}}]\} = \emptyset \tag{8}$$

$$\{r \in \mathcal{I}_N \mid \exists h, \widetilde{h} \in \mathcal{H}, r \in \mathcal{R}_h \cap \mathcal{R}_{\widetilde{h}}, \tau^r_{f,h} \in [\tau_{s,\widetilde{h}}, \tau^r_{f,\widetilde{h}}]\} = \emptyset \tag{9}$$

In other words, $\mathcal{H}$ is valid if every robot is in at most one CHOP at any given time.

Algorithm 4 translates a set $\mathcal{H}$ into a motion plan. In Line 2, each robot's motion plan is an optimal trajectory from its start to its goal. Each CHOP is then added to the motion plans of their constituent robots.

Algorithm 5 describes an iterative method to construct $\mathcal{H}$ and is detailed below. Fig. 4 presents a representative example of the algorithm steps.

### 5.4.1 Initialization with optimal trajectories

To begin, Lines 1–2 (of Algorithm 5) set each robot's motion plan to an optimal trajectory at $v_{max}$ to its goal, as pictured in Fig. 4a.

### 5.4.2 Identification of first collision

Line 3 identifies the first collision in the current motion plan. A collision occurs at time $t_c$ if there is a subset of robots $\mathcal{R}_c$ such that for all $i \in \mathcal{R}_c$, there exists $j \neq i \in \mathcal{R}_c$ such that:

$$\|\mathcal{M}^i(t_c) - \mathcal{M}^j(t_c)\|_2 < 2R.$$

Note that under this definition, two robots $i, j \in \mathcal{R}_c$ for which $\|\mathcal{M}^i(t_c) - \mathcal{M}^j(t_c)\|_2 \geq 2R$ can still be in collision if they simultaneously collide with a third robot. In general, collision-checking of trajectories is time-consuming. However, our motion plan consists of only constant-velocity trajectories, which allows us to conduct this check analytically.

Fig. 4b illustrates the first detected collision between the dark blue and green robots in the example problem. Note that the light blue, yellow, and purple robots collide as well, but this is not explicitly identified because it occurs later in time.

**Remark 4** *If multiple disjoint sets of robots collide simultaneously, we randomly choose one collision to resolve.*

### 5.4.3 Determination of CHOP parameters

If any collisions are found, Line 5 finds an appropriate set of parameters for the new CHOP. It also designates a set of CHOPs, $\mathcal{H}_{dup}$, that need to be removed for $\mathcal{H}$ to remain valid. This process is detailed in Algorithm 6.

For the remainder of Section 5.4.3, line numbers will refer to Algorithm 6. $\widetilde{h}$ denotes the new CHOP currently being constructed while $h$ denotes a CHOP in the current set $\mathcal{H}$.

The algorithm begins by constructing a CHOP consisting of only the robots in the collision. In Line 5, a start time is designated as the time closest to, but before, $t_c$ at which robots in $\mathcal{R}_{\widetilde{h}}$ will satisfy the CAPT separation condition. As shown in Line 6, these become the start positions from which robots enter the CHOP. Line 10 construct this CHOP. This is pictured for the example problem in Fig. 4c.

We then identify existing CHOPs in $\mathcal{H}$ that will conflict with the new CHOP, $\widetilde{h}$. In particular, we check for three criteria. Lines 13–17 searches for robots whose current trajectories will collide with the new CHOP. For efficiency, we check if the paths of the robots' trajectories intersect with the circle defined by the center and radius of $\widetilde{h}$, which is a series of circle-line-segment intersection checks. In Figure 4c, the red robot violates this condition. We add the robots that violate this condition to the new CHOP and add their CHOPs to $\mathcal{H}_{dup}$ for removal.

Lines 19–23 remove CHOPs with respect to which any robot in $\mathcal{R}_{\widetilde{h}}$ violates the validity conditions in Eqs. 8 and 9. The robots in the removed CHOPs are then added to $\mathcal{R}_c$,

Finally, Line 25 identifies all CHOPs that occur after $\widetilde{h}$ and contain any subset of the robots in $\mathcal{R}_c$. While this is not required for safety, Lemma 5.4 demonstrates the necessity of this criterion for completeness. If any new robots and CHOPs were found, the algorithm iterates again with new parameters for $\widetilde{h}$ until no further changes are required.

### 5.4.4 Iteration of algorithm

Line 6 (of Algorithm 5) adds the new CHOP to $\mathcal{H}$ and Line 7 removes the CHOPs in $\mathcal{H}_{dup}$. The motion plan is iteratively refined until it is collision-free.

Figures 4d–4h illustrate iterations of Algorithm 5, as well as portions of Algorithm 6, for the example problem. Figure 4d illustrates the motion plan after the first iteration of Algorithm 5. In Fig. 4e, a collision between the light blue and yellow robots is found and subsequently resolved with the CHOP pictured in

**Algorithm 5** $\mathcal{M}$ = Plan Motions$(\mathbf{s}, \mathbf{g}, N, R, v_{max})$

```
1:  $\mathcal{H} := \emptyset$
2:  $\mathcal{M}$ := CHOPs to Motion Plan $(\mathbf{s}, \mathbf{g}, \mathcal{H}, v_{max})$
3:  $(t_c, \mathcal{R}_c)$ := Find First Collision$(\mathcal{M}, R)$
4:  while collision found do
5:      $(\tau_{s,\widetilde{h}}, \mathcal{R}_{\widetilde{h}}, \mathbf{s}_{\widetilde{h}}, \mathbf{g}_{\widetilde{h}}, \mathcal{H}_{dup})$ := Get Params$(\mathcal{M}, \mathcal{H}, t_c, \mathcal{R}_c, \mathbf{g}, R, v_{max})$
6:      $\mathcal{H} \leftarrow$ Find CHOP$(\tau_{s,\widetilde{h}}, \mathcal{R}_{\widetilde{h}}, \mathbf{s}_{\widetilde{h}}, \mathbf{g}_{\widetilde{h}}, R, v_{max})$
7:      $\mathcal{H} := \mathcal{H} \setminus \mathcal{H}_{dup}$
8:      $\mathcal{M}$ := CHOPs to Motion Plan $(\mathbf{s}, \mathbf{g}, \mathcal{H}, v_{max})$
9:      $(t_c, \mathcal{R}_c)$ := Find First Collision$(\mathcal{M}, R)$
10: end while
11: $\mathcal{M}$ := Resegment Motion Plan$(\mathcal{M})$
```

**Algorithm 6** $(\tau_{s,\widetilde{h}}, \mathcal{R}_{\widetilde{h}}, \mathbf{s}_{\widetilde{h}}, \mathbf{g}_{\widetilde{h}}, \mathcal{H}_{dup})$ = Get Params$(\mathcal{M}, \mathcal{H}, t_c, \mathcal{R}_c, \mathbf{g}, R, v_{max})$

```
1:  $merge := 1$
2:  $\mathcal{H}_{dup} := \emptyset, \mathcal{R}_{\widetilde{h}} := \mathcal{R}_c, t_s := t_c$
3:  while $true$ do
4:      // Find CHOP for current parameters.
5:      $\tau_{s,\widetilde{h}} := \text{argmax}_{t\in[0,t_s]} \|\mathcal{M}^i(t) - \mathcal{M}^j(t)\|_2 \geq 2\sqrt{2}R \; \forall j \neq i \in \mathcal{R}_{\widetilde{h}}$
6:      $\mathbf{s}_{\widetilde{h}} := \{\mathcal{M}^i(\tau_{s,\widetilde{h}}) \mid i \in \mathcal{R}_{\widetilde{h}}\}, \mathbf{g}_{\widetilde{h}} = \{\mathbf{g}^i \mid i \in \mathcal{R}_{\widetilde{h}}\}$
7:      if $merge == 0$ then
8:          break
9:      end if
10:     $\widetilde{h} :=$ Find CHOP$(\tau_{s,\widetilde{h}}, \mathcal{R}_{\widetilde{h}}, \mathbf{s}_{\widetilde{h}}, \mathbf{g}_{\widetilde{h}}, R, v_{max})$
11:     // Adjust $\widetilde{h}$'s parameters.
12:     // 1. Find intersections with $\widetilde{h}$'s circle.
13:     $t_a := \widetilde{t}_1, t_b := \max_{i\in\mathcal{R}_{\widetilde{h}}} \tau^i_{f,\widetilde{h}}$
14:     $l :=$ Paths of motion plans for robots $r \in \mathcal{I}_N \setminus \mathcal{R}_{\widetilde{h}}$ between $[t_a, t_b]$
15:     $\mathcal{R}_{\widetilde{h}} \leftarrow$ Robots whose paths in $l$ intersect a circle at $\mathbf{x}_{\widetilde{h}}$ with radius $R_{\widetilde{h}} + 2R$
16:     $\mathcal{H}_{dup} \leftarrow$ CHOPs that have robots whose paths in $l$ intersect a circle at $\widetilde{\mathbf{x}}_h$, radius $R_{\widetilde{h}} + 2R$
17:     $\mathcal{R}_{\widetilde{h}} \leftarrow$ Robots in $\mathcal{H}_{dup}$
18:     // 2. Include CHOPs that $\widetilde{h}$ conflicts with temporally.
19:     for $r \in \mathcal{R}_{\widetilde{h}}$ do
20:         $\tau^r_{min} := \min \; \tau_{s,\widetilde{h}} \cup \{\tau_{s,h} \mid h \in \mathcal{H}_{dup}\}$
21:     end for
22:     $\mathcal{H}_{dup} \leftarrow \{h \in \mathcal{H} \mid \exists\, r \in \mathcal{R}_h \cap \mathcal{R}_{\widetilde{h}}, \tau^r_{f,h} \geq \tau^r_{min}\}$
23:     $\mathcal{R}_{\widetilde{h}} \leftarrow$ Robots in $\mathcal{H}_{dup}$
24:     // 3. Include CHOPs that contain two or more common robots with $\widetilde{\mathcal{R}}$.
25:     $\mathcal{H}_{dup} \leftarrow \{h \in \mathcal{H} \mid |\mathcal{R}_{\widetilde{h}} \cap \mathcal{R}_h| \geq 2\}$
26:     // If there are changes, iterate again.
27:     if new elements were added to $\mathcal{R}_{\widetilde{h}}$ or $\mathcal{H}_{dup}$ then
28:         $\mathcal{R}_{\widetilde{h}} \leftarrow$ Robots in $\mathcal{H}_{dup}$
29:         $t_s := \min \; \tau_{s,\widetilde{h}} \cup \{\tau_{s,h} \mid h \in \mathcal{H}_{dup}\}$
30:         $merge := 1$
31:     else
32:         $merge := 0$
33:     end if
34: end while
```

Fig. 4f. Figure 4g illustrates a second collision between the purple and yellow robots. In this case, the collision occurs when the yellow robot is still executing a CHOP. This satisfies the condition in Line 2 of Algorithm 6. As a result, the CHOP between the light blue and yellow robots, pictured in Fig. 4f, is replaced. Figure 4h illustrates the final motion plan, which contains two separate CHOPs.

#### 5.4.5 Resegmentation into a disjoint motion plan

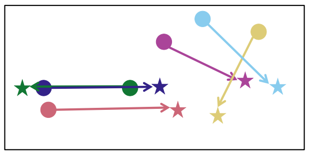
As a final step, in Line 11 of Algorithm 5, we redefine $\mathcal{M}$ such that it is *disjoint*, or that it satisfies the following:

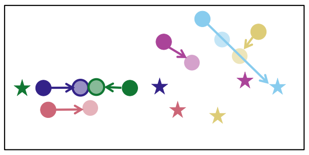
1. All robots in $\mathcal{M}$ share a common set of trajectory break-times:

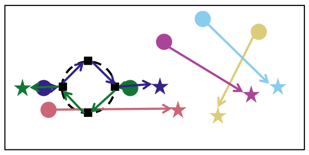
$$\mathcal{T}^i = \mathcal{T}^j \quad \forall i \neq j \in \mathcal{I}_N. \tag{10}$$

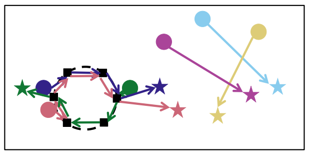
2. All paths at each time interval are at least $2R$ apart:

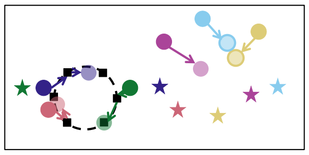
$$\min_{t_i, t_j \in [t_s, t_{s+1}]} \|\mathcal{M}^i_s(t_i) - \mathcal{M}^j_s(t_j)\|_2 \geq 2R \quad \forall i \neq j \in \mathcal{I}_N. \tag{11}$$

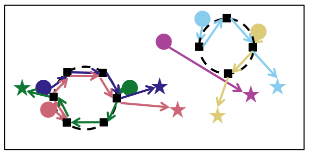

(a) Initially, all robots take direct trajectories to their goals (Algorithm 5, Line 2).

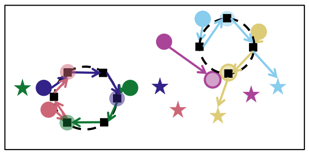

(b) The first collision is found between the dark blue and green robots (Algorithm 5, Line 3).


(c) Initial CHOP parameters are chosen to resolve the collision (Algorithm 6, Line 10, first iteration).


(d) The path of the red robot intersects the CHOP's circle (Algorithm 6, Line 15, first iteration). A CHOP containing the dark blue, green, and red robots is created instead. This satisfies all requirements in Algorithm 6 and a new motion plan is found.


(e) The first collision is found between the light blue and yellow robots (Algorithm 5, Line 9).


(f) Initial CHOP parameters are chosen to resolve the collision (Algorithm 6, Line 10, first iteration). This satisfies all requirements in Algorithm 6 and a new motion plan is found.


(g) The first collision occurs between the purple and yellow robots (Algorithm 5, Line 3).

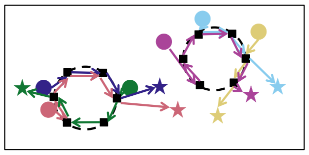

(h) Initial CHOP parameters are chosen to resolve the collision. However, this causes a conflict with the existing CHOP between the light blue and yellow robots (Algorithm 6, Line 22, first iteration). As a result, the purple, light blue, and yellow robots form one CHOP. This satisfies all requirements in Algorithm 6 and a new motion plan is found. This motion plan is safe, so Algorithm 5 terminates.

Figure 4: Example problem demonstrating Algorithm 5. Robots start at positions indicated by circles and must navigate to goals represented by stars of the same color.

We can turn any safe motion plan, $\mathcal{M}$, into a disjoint motion plan by redefining the trajectory break-times and waypoints. To satisfy Eq. 10, we first determine the set of trajectory break-times as the union of all trajectory break-times across all robots. For constant-velocity trajectory segments, we can easily derive the corresponding trajectory waypoints by evaluating the original motion plan at the appropriate break-times.

To illustrate this, consider a motion plan, $\mathcal{M}$, with:

$$\begin{aligned}
\mathcal{X}^1 &= \{\mathbf{x}^1_{d,0}, \mathbf{x}^1_{d,1}, \mathbf{x}^1_{d,2}\} \\
\mathcal{T}^1 &= \{t_0, t_1, t_4\} \\
\mathcal{X}^2 &= \{\mathbf{x}^2_{d,0}, \mathbf{x}^2_{d,1}, \mathbf{x}^2_{d,2}, \mathbf{x}^1_{d,3}\} \\
\mathcal{T}^2 &= \{t_0, t_2, t_3, t_4\}.
\end{aligned}$$

An equivalent motion plan with a common break-time set is:

$$\begin{aligned}
\widetilde{\mathcal{X}}^1 &= \{\mathbf{x}^1_{d,0}, \mathbf{x}^1_{d,1}, \mathcal{M}^1_1(t_2), \mathcal{M}^1_1(t_3), \mathbf{x}^1_{d,2}\} \\
\widetilde{\mathcal{T}}^1 &= \{t_0, t_1, t_2, t_3, t_4\} \\
\widetilde{\mathcal{X}}^2 &= \{\mathbf{x}^2_{d,0}, \mathcal{M}^2_0(t_1), \mathbf{x}^2_{d,1}, \mathbf{x}^2_{d,2}, \mathbf{x}^1_{d,3}\} \\
\widetilde{\mathcal{T}}^2 &= \{t_0, t_1, t_2, t_3, t_4\}.
\end{aligned}$$

For any pair of trajectory segments that violate Eq. 11, an additional break-time can be inserted at the appropriate time such that Eq. 11 will be satisfied for the partitioned trajectory segments.

**Remark 5** *Algorithm 5 guarantees safety of the returned motion plan by design, as it explicitly checks for and only returns safe solutions. It is instead more insightful to discuss the algorithm's completeness.*

**Lemma 5.3** *At the end of each iteration of Algorithm 5 (Line 8), $\mathcal{H}$ is valid.*

**Proof 3** *Suppose $\mathcal{H}$ is not valid and $\exists r \in \mathcal{I}_N$ such that:*

$$\exists h, \widetilde{h} \in \mathcal{H}, r \in \mathcal{R}_h \cap \mathcal{R}_{\widetilde{h}}, \tau^r_{f,h} \in [\tau_{s,\widetilde{h}}, \tau^r_{f,\widetilde{h}}].$$

*This fulfills Line 22 of Algorithm 6. Thus, $h \in \mathcal{H}_{dup}$, which is removed from $\mathcal{H}$ in Line 7 of Algorithm 5, resulting in a contradiction.*

*Alternatively, suppose:*

$$\exists h, \widetilde{h} \in \mathcal{H}, r \in \mathcal{R}_h \cap \mathcal{R}_{\widetilde{h}}, \tau_{s,h} \in [\tau_{s,\widetilde{h}}, \tau^r_{f,\widetilde{h}}].$$

*If $\tau_{s,h} \in [\tau_{s,\widetilde{h}}, \tau^r_{f,\widetilde{h}}]$, then $\tau_{s,h} \geq \tau_{s,\widetilde{h}}$. As a result, $\tau^r_{f,h} \geq \tau_{s,\widetilde{h}}$. This again fulfills Line 22 of Algorithm 6, leading to the removal of $h$ from $\mathcal{H}$ in Line 7 of Algorithm 5 and another contradiction.* ■

**Lemma 5.4** *Algorithm 5 is complete.*

**Proof 4** *Throughout this proof, we will use $\widetilde{h}$ to denote a newly formed CHOP and $h$ to denote CHOPs already in $\mathcal{H}$, as in Algorithms 5 and 6.*

*From Lemma 5.1, the solution where all robots enter a single CHOP is safe. Thus, a solution to Algorithm 5 always exists — to be complete, Algorithm 5 must always return a safe solution. Lemma 5.3 shows that $\mathcal{H}$ is valid between iterations of Algorithm 5, leading to a motion plan. Trivially, if the motion plan is safe, Algorithm 5 terminates. We will show that Algorithm 5 always terminates in a finite number of iterations.*

*Define an interaction as a pair of indices, $i, j \in \mathcal{R}$ such that $i, j \in \mathcal{R}_h$ for some CHOP $h \in \mathcal{H}$. Let $\mathcal{A}$ denote the set of all interactions in $\mathcal{H}$. For example, the set of interactions for the motion plan pictured in Fig. 4d is $\mathcal{A} = \{\{red, green\}, \{green, dark\ blue\}, \{red, dark\ blue\}\}$.*

*Line 25 of Algorithm 6 ensures each interaction appears in at most one CHOP in $\mathcal{H}$. As a result, $\mathcal{A}$ will never contain duplicate interactions and corresponds to a unique set $\{\mathcal{R}_h \mid h \in \mathcal{H}\}$.*

*We will first show that after each execution of Line 8, the number of interactions in $\mathcal{A}$ is strictly increasing.*

*Line 28 of Algorithm 6 adds all robots from $\mathcal{H}_{dup}$ to $\mathcal{R}_{\widetilde{h}}$, thus, no interactions will be removed from $\mathcal{H}$, and the number of interactions is non-decreasing.*

*Suppose the number of interactions remains the same between two iterations of Algorithm 5. This means $\mathcal{H}_{dup}$ contains exactly the same interactions as the new CHOP $\widetilde{h}$. Thus, $\mathcal{H}_{dup}$ contains only one CHOP $h$, where $h = \widetilde{h}$.*

*$\widetilde{h}$ is formed as a result of a collision between robots $\mathcal{R}_c$. Line 2 of Algorithm 6 ensures that $\mathcal{R}_c \in \mathcal{R}_{\widetilde{h}}$. Thus, $\mathcal{R}_c \in \mathcal{R}_h$ as well. However, if $\mathcal{R}_c \in \mathcal{R}_h$, then they have already coordinated their motions in a CHOP and the only way they can collide is if a subset of robots in $\mathcal{R}_c$ deviated from their optimal trajectories to their goals to enter another CHOP with robots not in $\mathcal{R}_h$. Explicitly, there is a CHOP $\bar{h}$ for which $\tau_{s,\bar{h}} \geq \tau_{s,h}$ and there exists nonempty sets $\mathcal{R}_1 = \mathcal{R}_{\bar{h}} \cap \mathcal{R}_c$ and $\mathcal{R}_2 = \mathcal{R}_{\bar{h}} \setminus \mathcal{R}_h$.*

*From Line 22 of Algorithm 6, $\bar{h} \in \mathcal{H}_{dup}$ and from Line 28, $\mathcal{R}_{\bar{h}} \in \mathcal{R}_{\widetilde{h}}$ and so, $\mathcal{R}_{\bar{h}} \in \mathcal{R}_h$. This contradicts the fact that $\mathcal{R}_2$ is nonempty.*

*As $\mathcal{A}$ is strictly increasing without duplicate interactions, within a finite number of iterations, $\mathcal{A}$ will contain all possible pairwise combinations of indices $i, j \in \mathcal{I}_N$. This corresponds to the case where all robots are in a single CHOP. This in turn corresponds to a safe motion plan, terminating Algorithm 5.* ■

## 6 Trajectory Generation

While the motion plan, $\mathcal{M}$, is safe for kinematic robots, it has a discontinuous velocity profile, making it potentially infeasible for robots with dynamics of order $n > 1$. Thus, we need to translate $\mathcal{M}$ into a set of dynamically feasible trajectories, $\gamma$, for the team.

Since $\mathcal{M}$ is disjoint, we will drop the superscript $i$ from time variables and use $\mathcal{T}$ and $t_s$ to refer to the common set of break-times. Finally, we identify the time, $T^i_g$, and waypoint index, $m^i_g$, at which robots in $\mathcal{M}$ arrive at their goals.

We want to solve:

$$\begin{aligned}
\underset{\gamma}{\operatorname{argmin}} F &= \underset{\gamma}{\operatorname{argmin}}\ \mathcal{L}\left(t, \frac{d^n}{dt^n}\gamma^0, \frac{d^n}{dt^n}\gamma^1, ..., \frac{d^n}{dt^n}\gamma^N\right) \\
&= \underset{\gamma}{\operatorname{argmin}} \left(\sum_{i=1}^{N}\int_0^{T_g^i} \|\frac{d^n}{dt^n}\gamma^i\|_2^2\ dt\right) \\
&= \underset{\gamma}{\operatorname{argmin}} \left(\sum_{i=1}^{N}\sum_{s=0}^{m-1}\int_{t_s}^{t_{s+1}} \|\frac{d^n}{dt^n}\gamma^i\|_2^2\ dt\right) \qquad (12)
\end{aligned}$$

subject to:

1. Waypoint constraints: Robots must begin and end at their designated start and goal positions, at rest.

$$\begin{gathered}
\gamma^i(0) = \mathbf{s}^i,\ \ \gamma^i(T_g^i) = \mathbf{g}^i \\
\frac{d^k}{dt^k}\gamma^i(0) = 0,\ \ \frac{d^k}{dt^k}\gamma^i(T_g^i) = 0 \\
\forall k = \mathcal{I}_{n-1}, \forall i \in \mathcal{I}_N. \qquad (13)
\end{gathered}$$

2. Continuity constraints: Trajectories must be at least $n$ times differentiable.

$$\begin{gathered}
\frac{d^k}{dt^k}\gamma_s^i(t_{s+1}) = \frac{d^k}{dt^k}\gamma_{s+1}^i(t_{s+1}) \\
\forall s \in [0, m-1], k \in [0, n-1], i \in \mathcal{I}_N. \qquad (14)
\end{gathered}$$

3. Collision avoidance constraints: Trajectories must be safe.

$$\|\gamma^i - \gamma^j\|_2 \geq 2R\ \ \forall i \neq j \in \mathcal{I}_N. \qquad (15)$$

Recall robots have $n^{th}$-order dynamics, given by Eq. 1. The chosen cost functional will thus minimize the control input.

## 6.1 Decision Vector

Recall each trajectory has the form:

$$\gamma^i = \begin{cases} \sum_{j=0}^{M} c_{0,j}^i t^j & t_0 \leq t \leq t_1 \\ \sum_{j=0}^{M} c_{1,j}^i t^j & t_1 \leq t \leq t_2 \\ ... & \\ \sum_{j=0}^{M} c_{m^i-1,j}^i t^j & t_{m^i-1} \leq t \leq t_{m^i} \end{cases}.$$

First, we must determine the degree $M$ of the polynomial. A necessary condition for minimizing the cost functional in Eq. 12 is given by the Euler-Lagrange equation:

$$\frac{\partial \mathcal{L}}{\partial \gamma^i} + \sum_{k=1}^{n}(-1)^k \frac{d^k}{dt^k}\frac{\partial \mathcal{L}}{\partial \gamma^i} = 0\ \ \forall i \in \mathcal{I}_N.$$

where $\mathcal{L}$ is the cost function associated with the cost functional $F$. Evaluating this for the given $\mathcal{L}$ gives the condition:

$$\frac{d^{2n}}{dt^{2n}}\gamma^i = 0\ \ \forall i \in \mathcal{I}_N.$$

Integrating this equation shows that the optimal polynomials have order $2n - 1$.

Define the vector of coefficients of $\gamma_s^i$

$$\mathbf{c}_s^i = [(c_{s,0}^i)^\top\ \ (c_{s,1}^i)^\top\ \ ...\ \ (c_{s,2n-1}^i)^\top]^\top.$$

Note that each $c^i_{s,j} \in \mathbb{R}^2$ contains coefficients corresponding to the $x$ and $y$ components of $\gamma^i$. For each robot, we construct a decision vector composed of all coefficients of its piecewise-polynomial:

$$\mathbf{c}^i = [(\mathbf{c}^i_0)^\top \ \ (\mathbf{c}^i_1)^\top \ \ \dots \ \ (\mathbf{c}^i_{m-1})^\top]^\top.$$

For polynomials, the cost in Eq. 12 is quadratic with respect to the chosen decision vector [40]:

$$\int_0^{T^i_g} \| \frac{d^n}{dt^n} \gamma^i(t) \|_2^2 dt = \left(\mathbf{c}^i\right)^\top Q \left(\mathbf{c}^i\right). \tag{16}$$

Eqs. 13–14 are linear with respect to the decision vector:

$$A^i_{eq} \mathbf{c}^i = \mathbf{b}^i_{eq}. \tag{17}$$

Further note that Eqs. 16 and 17 are decoupled for each robot's decision vector $\mathbf{c}^i$. However, Eq. 15 is a nonlinear, nonconvex function that couple robots' decision vectors. Thus, to formulate the optimization problem as a QP (ie. optimization problem with a quadratic cost function and linear constraints), the challenge is to also impose collision avoidance constraints as decoupled, linear functions of each robot's decision vector.

## 6.2 Nominal Solution

First, we present a nominal solution that guarantees collision avoidance by inserting additional boundary conditions into the optimization problem. Consider any time interval $[t_s, t_{s+1}]$, robot $i$, and the modified optimization problem:

$$\underset{\gamma_i}{\operatorname{argmin}} \int_{t_s}^{t_{s+1}} \| \frac{d^n}{dt^n} \gamma^i_s \|_2^2 \, dt$$

subject to:

$$\begin{aligned} &\gamma^i(t_s) = \mathbf{x}^i_{d,s}, \ \ \gamma^i(t_{s+1}) = \mathbf{x}^i_{d,s+1} \\ &\frac{d^k}{dt^k}\gamma^i(t_s) = 0, \ \ \frac{d^k}{dt^k}\gamma^i(t_{s+1}) = 0 \ \ \forall k = \mathcal{I}_{n-1}. \end{aligned} \tag{18}$$

Here, homogeneous boundary conditions are imposed on the trajectory segment (ie. the values of all derivatives at $t_s$ and $t_{s+1}$ are 0). Solving Eq. 18 yields the solution:

$$\gamma^i_{s,nom} = \mathbf{x}^i_{d,s} + \beta_s(t)(\mathbf{x}^i_{d,s+1} - \mathbf{x}^i_{d,s}). \tag{19}$$

$\beta_s(t)$ is a polynomial of order $2n-1$ and satisfies the boundary conditions:

$$\begin{aligned} &\beta(t_s) = 0, \ \ \beta(t_{s+1}) = 1 \\ &\frac{d^k}{dt^k}\beta(t_s) = 0, \ \ \frac{d^k}{dt^k}\beta(t_{s+1}) = 0 \ \ \forall k = \mathcal{I}_{n-1}. \end{aligned}$$

We see that $\gamma^i_{s,nom}$ is simply a reparameterization of the optimal trajectory given by Eq. 5; that is, it represents the same path but varies the velocity profile.

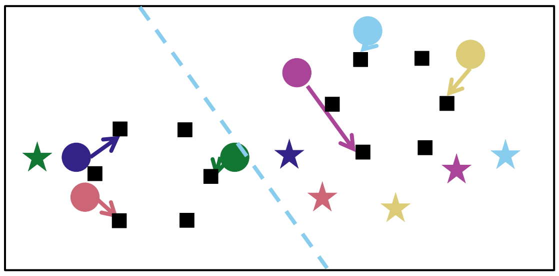
For each robot, we solve the optimization problem:

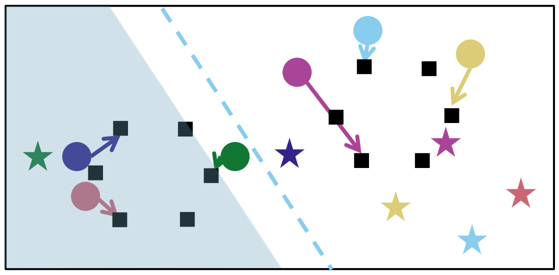
$$\underset{\gamma}{\operatorname{argmin}} \int_0^{T^i_g} \| \frac{d^n}{dt^n} \gamma^i \|_2^2 \, dt \tag{20}$$

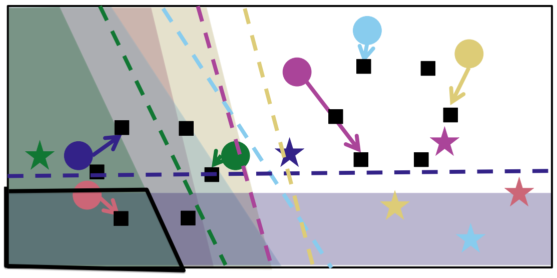
subject to:

$$\gamma^i(t_s) = \mathbf{x}^i_{d,s}, \ \ \gamma^i(t_{s+1}) = \mathbf{x}^i_{d,s+1} \tag{21}$$

$$\frac{d^k}{dt^k}\gamma^i(t_s) = 0, \ \ \frac{d^k}{dt^k}\gamma^i(t_{s+1}) = 0 \quad \forall k = \mathcal{I}_{n-1}, s \in [0, m-1]. \tag{22}$$


(a) Hyperplane for red robot with respect to light blue robot.


(b) Constraint of red robot to the corresponding half-plane.


(c) Half-plane constraints for the red robot with respect to all neighbors. The resultant convex region is outlined in solid black.

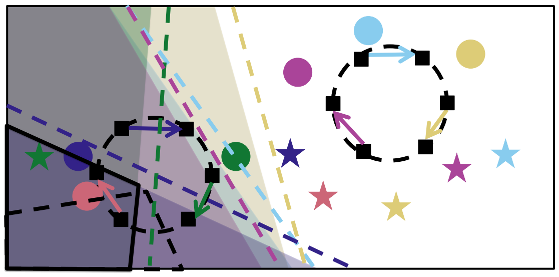

(d) Construction of convex region for red robot in next time interval, outlined in solid black. The convex region of the last time interval is indicated in dashed black.

Figure 5: Construction of convex feasible region for red robot for collision avoidance constraints. The color of each hyperplane and half-plane indicate the neighbor with respect to which the constraint is formed.

We will refer to this problem as the "nominal QP". It's solution is a trajectory that follows the same path as the motion plan, but trajectory segments are reparameterized to be dynamically feasible. We will refer to $\gamma_{nom} = \{\gamma^i_{nom} \mid i \in \mathcal{I}_N\}$ as the "nominal solution" and each $\gamma^i_{nom}$ as the "nominal trajectory". Note that $\beta_s(t)$ can be found once to quickly find all nominal trajectories.

It is easy to see that $\gamma_{nom}$ can always be found and satisfies the constraints of the original optimization problem, Eq. 12. The waypoint constraints, Eq. 13 are incorporated into Eq. 21. Eq. 14 is satisfied by the constraints in Eq. 22. Finally, since $\gamma_{nom}$ travels along the same path as $\mathcal{M}$, $\gamma_{nom}$ is safe.

## 6.3 Trajectory Smoothing

While $\gamma_{nom}$ is a valid solution to Eq. 12, it is inefficient, as robots must come to a complete stop at each trajectory waypoint. In this section, we present a method to linearize and decouple the collision avoidance constraints without adding additional boundary conditions. The resulting QP formulation yields a smooth, safe trajectory for each robot. Figure 5 illustrates this process for the example problem.

Consider a time interval, $[t_s, t_{s+1}]$, and a robot $i$. For any neighbor $j$, we can find a hyperplane separating $\mathcal{M}^i_s$ from $\mathcal{M}^j_s$. Denote this hyperplane with:

$$\mathbf{n}^{ij}_s \cdot (\mathbf{x} - \mathbf{b}^{ij}_s) = 0,$$

where $\mathbf{n}^{ij}$ is the normal of the hyperplane, pointing towards $\mathcal{M}^i_s$. This is shown for the red robot with respect to the blue robot in Fig. 5a. The region in the direction $\mathbf{n}^{ij}$ of the hyperplane defines a half-plane in the workspace.

Consider a constraint of the form:

$$\mathbf{n}^{ij}_s \cdot (\mathbf{x}_1 - \mathbf{b}^{ij}_s) \geq R.$$

The point $\mathbf{x}_1$ will be constrained to fall within the same half-plane as $\mathcal{M}^i_s$, at least $R$ away from the hyperplane. This region is pictured in Fig. 5b. If a second point, $\mathbf{x}_2$, is constrained by:

$$\mathbf{n}^{ji}_s \cdot (\mathbf{x}_2 - \mathbf{b}^{ji}_s) \geq R,$$

where $\mathbf{n}^{ji} = -\mathbf{n}^{ij}$ and $\mathbf{b}_s^{ji} = \mathbf{b}_s^{ij}$, we are guaranteed that:

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_2 \geq 2R.$$

With this in mind, we constrain the trajectory segment, $\gamma_s^i$, such that:

$$\mathbf{n}_s^{ij} \cdot (\gamma_s^i(t) - \mathbf{b}_s^{ij}) \geq R \quad \forall t \in [t_s, t_{s+1}].$$

This constraint can be approximated by constraining the trajectory segment at a series of sample times [38]. In this work, we apply this constraint using the method presented by [34]. The polynomial $\gamma_s^i$ can be alternatively represented as Bézier curve:

$$\gamma_s^i = \sum_{j=0}^{2n-1} p_{s,j}^i B_j.$$

Here, $B_j : \mathbb{R} \to \mathbb{R}$ are the Bernstein basis polynomials and $p_j \in \mathbb{R}^2$ are *control points*. In this representation, the control points exhibit the *Convex Hull Property*: the polynomial $\gamma_s^i$ is contained within the convex hull of the control points [49]. Thus, constraining the control points to a desired half-plane will in turn constrain the entire polynomial to that half-plane.

Define a decision vector of control points:

$$\mathbf{p}_s^i = [(p_{s,0}^i)^\top \quad (p_{s,1}^i)^\top \quad \ldots \quad (p_{s,2n-1}^i)^\top]^\top.$$

We can express $2n-1$ constraints:

$$\mathbf{n}_s^{ij} \cdot (p_{s,j}^i - \mathbf{b}_s^{ij}) \geq R \quad \forall j \in [0, 2n-1],$$

which are linear with respect to the control points, in the form:

$$A_{ineq,p,s}^{ij} \mathbf{p}_s^i \leq \mathbf{b}_{ineq,p,s}^{ij}.$$

Finally, there is a known linear relationship between the power basis coefficients, $\mathbf{c}_s^i$, and the control points, $\mathbf{p}_s^i$:

$$\mathbf{p}_s^i = T\mathbf{c}_s^i.$$

Thus, we can constrain $\gamma_s^i$ to a desired half-plane with $2n-1$ inequality constraints of the form:

$$A_{ineq,p,s}^{ij} T\mathbf{c}_s^i \leq \mathbf{b}_{ineq,p,s}^{ij}. \tag{23}$$

Applying this process with respect to all neighbors will constrain $\gamma_s^i$ to a convex region in the workspace in which all points of $\gamma_s^i$ will be at least $R$ away from all separating hyperplanes with neighbors. If all neighbors apply reciprocal constraints to $\gamma_s^j$, then all trajectory segments in time interval $t \in [t_s, t_{s+1}]$, will be guaranteed to satisfy constraint Eq. 15. The construction of this convex region is pictured in Fig. 5c.

**Remark 6** *The constructed convex region for each robot will be nonempty. Recall the Separating Hyperplane Theorem [50]: given two nonempty, convex sets that do not intersect, there exists* $\mathbf{n}$ *and* $\mathbf{s}$ *such that* $\mathbf{n} \cdot \mathbf{x} \leq \mathbf{s}$ *for all points* $\mathbf{x}$ *in one set and* $\mathbf{n} \cdot \mathbf{x} \geq \mathbf{s}$ *for all points* $\mathbf{x}$ *in the other.*

$\mathcal{M}$ *is guaranteed to be disjoint. Within any time interval, the paths of* $\mathcal{M}_s^i$ *and any neighbor* $\mathcal{M}_s^j$ *are non-intersecting convex sets in* $\mathbb{R}^2$*, and the hyperplane separating them will always exist. Furthermore, as the path of* $\mathcal{M}_s^i$ *is at least* $2R$ *away from any neighboring path, the convex region contains at least* $\mathcal{M}^i$ *and is therefore nonempty.*

We can easily apply this approach to every time interval of the piecewise-polynomial of robot $i$ to obtain a set of inequality constraints:

$$A_{ineq}^i \mathbf{c}^i \leq \mathbf{b}_{ineq}^i,$$

that constrain each trajectory segment to a designated convex region. Furthermore, consider the convex region for the interval $[t_{s+1}, t_{s+2}]$. As stated in Remark 6, this convex region will be nonempty. Furthermore, because $\mathcal{M}$ is continuous, there will be at least one point, $\mathbf{x}^i_{d,s+1}$, that is in both the convex region formulated at $[t_s, t_{s+1}]$ and $[t_{s+1}, t_{s+2}]$, guaranteeing a region of overlap between two consecutive convex regions. This is pictured in Fig. 5d. This overlap allows for optimization of a trajectory that smoothly moves through the designated convex regions.

As a result, each robot's trajectory can be found by solving the QP:

$$\underset{\mathbf{c}^i}{\operatorname{argmin}} \left(\mathbf{c}^i\right)^\top Q \left(\mathbf{c}^i\right)$$

subject to:

$$\begin{aligned} A^i_{eq}\mathbf{c}^i &= \mathbf{b}^i_{eq} \\ A^i_{ineq}\mathbf{c}^i &\leq \mathbf{b}^i_{ineq}. \end{aligned} \tag{24}$$

This can be solved by any commercial optimization software. We will refer to formulation as the "smooth QP" and its solution trajectories, $\gamma^i_{smooth}$, as "smooth trajectories".

**Remark 7** *We note that some robots will arrive at their goals before their neighbors. When a neighbor robot is at its goal, we construct convex region constraints with respect to the point* $\mathbf{g}^i$.

**Remark 8** *To improve the numerical stability of the smooth QP, we nondimensionalize each segment of* $\gamma^i_s$ *in time and evaluate it at* $\tau = \frac{t-t_s}{t_{s+1}-t_s} \in [0,1]$. *This also causes* $A^i_{eq}$ *and* $\mathbf{b}_e q^i$ *to be the same across all robots. This nondimensionalization is described in detail in [38]. Furthermore, the QP can be formulated in terms of any set of polynomial basis functions, such as Bernstein or Legendre polynomials. The polynomial order and/or the choice of derivative in the cost function can also be decreased for more numerical stability, at the expense of trajectory optimality and/or continuity. This might be a practical choice for higher-order systems. In particular, for quadrotors, a fourth-order dynamic system,* $5^{th}$*-order, minimum-jerk polynomials have been used instead of* $7^{th}$*-order, minimum-snap trajectories.*

The smooth trajectory has a number of benefits over the nominal. First, we eliminate all waypoint constraints except for those at times 0 and $T^i_g$. As a result, the robot is no longer required to travel exactly through each intermediate waypoint and the optimization often finds a significantly shorter path through its designated convex regions, mitigating the suboptimality of the motion planning step. This effect has similarly been seen in single-robot planning experiments [41].

## 6.4 Algorithm Definition

We formally state the trajectory generation algorithm in Algorithm 7. In short, Algorithm 7 finds the nominal trajectory and formulates the smooth QP for each robot. If a solution to the smooth QP is found, the robot will follow its smooth trajectories; otherwise, it will follow its nominal trajectory.

**Remark 9** *Note that* $\gamma^i_{nom}$ *satisfies all constraints of the smooth QP. The waypoint and derivative constraints in of the nominal QP satisfy the waypoint and continuity constraints in Eq. 12.* $\mathcal{M}^i$*, and therefore* $\gamma^i_{nom}$*, satisfies the convex region constraints. Thus, any subset of robots can follow their nominal trajectories without compromising the safety of the smooth trajectories of neighboring robots.*

Note that the constraint matrices, Line 4, the half-planes between robots, Line 8, and $\beta(t)$ in Eq. 19 can be cached between robots to avoid duplicate computations.

Finally, Line 18 checks each trajectory for violations of the maximum velocity constraint. Consider a trajectory $\gamma^i$ with break-times $\mathcal{T}$ and a trajectory $\widetilde{\gamma}^i$ with the same coefficients, but break-times $\widetilde{\mathcal{T}} = \alpha\mathcal{T}$, where $\alpha$ is a positive constant. $\widetilde{\gamma}^i$ will follow the same path as $\gamma^i$, but have derivative values $\frac{d^k}{dt^k}\widetilde{\gamma}^i = \frac{1}{\alpha^k}\frac{d^k}{dt^k}\gamma^i$ [38]. If any violations of the maximum velocity is found in any robot's trajectory, Line 18 finds the minimum $\alpha$ needed to adequately decrease the velocity. The break-time vector (which is common to all robots) is then modified to $\alpha\mathcal{T}$.

The solutions to Eq. 24 are locally optimal trajectories through the designated convex regions and will, in general, be a suboptimal solution to the optimization problem posed in Eq. 12. However, it has a number of important computational benefits. The convex region constraints are constructed entirely from the motion plan, eliminating the coupling between robots' trajectories required to impose Eq. 15 directly. This allows us

**Algorithm 7** $\gamma$ = Generate Trajectories($\mathcal{M}, n, N, R, v_{max}$)

```
1: $Q$ := construct cost function, Eq. 16
2: for $i \in \mathcal{I}_N$ do
3:     $\gamma^i_{nom}$ := find nominal trajectory
4:     $A^i_{eq}, b^i_{eq}$ := equality constraints from Eqs. 13–14
5:     $A^i_{ineq}, \mathbf{b}^i_{ineq} := \emptyset$
6:     for all $s \in [0, m-1]$ do
7:         for all robots $j \in \mathcal{I}_N, i \neq j$ do
8:             $A^i_{ineq}, \mathbf{b}^i_{ineq}$ ←add convex region constraint for robot $j$, time interval $[t_s, t_{s+1}]$ from Eq. 23
9:         end for
10:    end for
11:    $\gamma^i_{smooth}$ := solve Eq. 24
12:    if Eq. 24 solved successfully then
13:        $\gamma^i := \gamma^i_{smooth}$
14:    else
15:        $\gamma^i := \gamma^i_{nom}$
16:    end if
17: end for
18: $\gamma$ := adjust time scaling to satisfy $v_{max}$
```

to solve $N$ decoupled QPs for each robot's trajectories instead of a single, joint optimization problem, offering improved scalability to large teams. Furthermore, while a failure to solve a joint optimization problem leads to a failure to find any solution $\gamma$, in our decoupled formulation, robots who fail to find a smooth trajectory simply default to following their nominal trajectories without affecting their neighbors.

## 7 Algorithm Analysis

In this section, we discuss the properties of the HOOP algorithm (Algorithm 1). First, we prove two important properties.

**Theorem 7.1** *The HOOP algorithm is safe.*

**Proof 5** *As stated in Remark 5, a motion plan returned by Algorithm 5 is guaranteed to be safe. In the trajectory planning step, satisfying the inequality constraints given by Eq. 24 is a sufficient condition for safety of trajectories $\gamma$. For any robot $i$, if a solution $\gamma^i_{smooth}$ is successfully found, it must satisfy all inequality constraints and therefore be safe. If a solution is not found, the robot's trajectory is given by $\gamma^i_{nom}$, which also satisfies the inequality constraints in Eq. 24, and is therefore also safe.* ■

**Theorem 7.2** *The HOOP algorithm is complete.*

**Proof 6** *Lemma 5.4 shows that Algorithm 5 is complete, therefore $\mathcal{M}$ will always be found. From $\mathcal{M}$, nominal trajectories $\gamma_{nom}$ can always be found by a reparameterization. Thus, the algorithm is complete and in the worst case, $\gamma$ consists of only nominal trajectories.* ■

**Remark 10** *As $\gamma_{nom}$ requires vehicles to come to rest at trajectory waypoints, HOOP is not complete for robots, such as fixed-wing aircrafts, that cannot sustain zero velocity. However, this method can still be used to find smooth trajectories for such vehicles.*

**Remark 11** *HOOP is also not complete in applications where robots have a limited workspace. However, it is possible to computationally reject unsafe solutions by introducing half-plane constraints that model the workspace boundaries into the QPs in the trajectory generation step.*

We end with a discussion of the suboptimality and complexity of the HOOP algorithm.

The algorithm returns globally optimal trajectories only when no CHOPs are added in the motion planning step. In this case, each robot will take a minimum-cost, straight-line trajectory directly to its goal. This will typically not be true, and as a result, HOOP generally returns suboptimal trajectories. Qualitatively, the suboptimality of the algorithm is related to the problem density, as captured by the ratio:

$$d = \frac{2\pi R^2 N}{A_{workspace}},$$

where $A_{workspace}$ is the available workspace area. As $d$ decreases and more freespace is available, it becomes more likely that less, or no, holding patterns will be necessary to find a collision-free motion plan. As a result, the trajectories returned by HOOP will be optimal or close-to optimal. As $d$ increases, more holding patterns will need to be created, and the solution sub-optimality will increase as the number and size of the CHOPs in the solution increase. In the worst case, all robots must enter a single holding pattern.

We can quantify the algorithm's complexity. In the motion planning step, in the worst-case scenario, one robot gets added to a single, growing CHOP at each iteration. Thus, the motion planning step runs for at most $N$ iterations. In each iteration, a CHOP is constructed. Determination of CHOP parameters occurs in constant time. Assignment of entry waypoints, using the Hungarian Algorithm, is runs in $O(N^3)$ time. Assignment of exit waypoints is $O(N)$ and determination of priority sets is $O(N^2)$. Each step of the motion plan, an instance of CAPT, is constructed in $O(N^3)$. In the worst case, all robots are in a CHOP with $2N$ waypoints and have the same exit waypoint. The last robot must pass its exit waypoint $N$ times before exiting. Thus, the number of waypoints in the worst-case scenario is $O(N^2)$, for which the motion plan can be constructed in $O(N^5)$. This is an improvement over the exponential complexity of planning in a joint configuration space. Practically, computational complexity can be substantially improved as described in Remark 5.3.4.

Each segment of the nominal trajectory can be found in constant time. Since the worst-case number of waypoints scales as $O(N^2)$, the $N$ nominal trajectories can be found in $O(N^3)$.

In the smooth QP, each decision vector is of size $4nm$ — there are $2n$ coefficients in each of the $m$ two-dimensional trajectory segments. The number of constraints is at most:

$$4n + 2n(m-1) + (2n)(m)(N-1).$$

Each problem has $4n$ waypoint constraints, $2n(m-1)$ continuity constraints, and at most $(2n)(m)(N-1)$ convex region constraints. However, note that many of the convex region constraints become redundant and can be eliminated.

Qualitatively, we reduce the computational complexity and improve the scalability of our algorithm by addressing safety and dynamic feasibility in two separate steps. Many operations used in the motion planning step, such as translation of the set of CHOPs to a motion plan and collision checking, have analytical solutions for first-order robots (but become complex for higher-order vehicles with non-constant velocity profiles). We temporarily ignore the vehicles' dynamics to leverage this simplicity and quickly find a safe motion plan. Contrastingly, in the trajectory generation step, the dynamics are easily embedded in the optimization's cost function, but the problem becomes impractical in the presence of coupled collision avoidance constraints. We utilize the existing motion plan to avoid these coupled constraints when finding a smooth trajectory.

## 8 Experimental Application to a Quadrotor Team

We implement our algorithm in MATLAB on a 2014 15-inch Macbook Pro with a 2.5 GHz Intel Core i7 processor and 16 GB of RAM. We use Gurobi[1] for optimization. We experimentally validated our algorithm in a quadrotor testbed consisting of five Ascending Technologies (AscTec) Hummingbird[2] quadrotors operating in a Vicon[3] motion capture system, pictured in Fig. 6. The workspace is 3.5 m by 3.5 m in the $xy$-directions and 1.5 m in height. As shown in Fig. 7b, each robot carries an ODROID XU3 computer equipped with a wireless ethernet adaptor under it's body. The robot's rotor-tip-to-rotor-tip distance is 0.54 m and total mass, with onboard equipment and battery, is 0.703 kg.

Table 2 defines the variables used in our dynamic model and controller. The full coordinate-free state of the quadrotor is $\begin{bmatrix}\mathbf{r} & \dot{\mathbf{r}} & \mathbf{R} & \mathbf{\Omega}\end{bmatrix}^\top$, with input $u = \begin{bmatrix} f & \mathbf{M}\end{bmatrix}^\top$ and dynamics:

$$m(\ddot{\mathbf{r}} + g\mathbf{e}_3) = fR\mathbf{e}_3$$
$$\mathcal{I}\dot{\mathbf{\Omega}} + \mathbf{\Omega} \times \mathbb{I}\mathbf{\Omega} = \mathbf{M}.$$

[1]http://www.gurobi.com
[2]http://www.asctec.de/en/
[3]https://www.vicon.com/

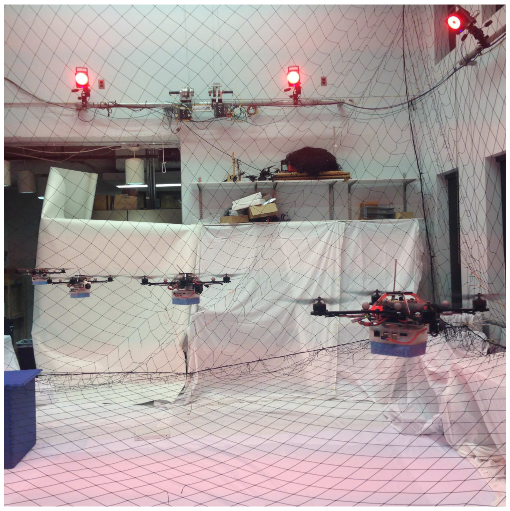


Figure 6: Hummingbird quadrotors flying in a Victon motion capture space.

Table 2: Variable definitions for the quadrotor dynamic model and controller.

| | |
|---|---|
| $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ | inertial frame unit coordinate axes |
| $\{\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3\}$ | body-frame unit coordinate axes |
| $g$ | gravity constant |
| $\mathbb{I} \in \mathbb{R}^{3\times 3}$ | inertia matrix |
| $m \in \mathbb{R}$ | mass |
| $\mathbf{r} \in \mathbb{R}^3$ | center of mass position |
| $R \in SO(3)$ | world-to-body rotation matrix |
| $\mathbf{\Omega} \in \mathbb{R}^3$ | body-frame angular velocity |
| $\psi \in \mathbb{R}$ | yaw angle |
| $f \in \mathbb{R}$ | thrust |
| $\mathbf{M} \in \mathbb{R}^3$ | moment in body-frame coordinates |
| $\mathbf{e}_r$, $\mathbf{e}_v$, $\mathbf{e}_R$, $\mathbf{e}_{\mathbf{\Omega}}$ | state errors |
| $k_r$, $k_v$, $k_R$, $k_{\mathbf{\Omega}}$ | positive controller gains |

This model is differentially flat, that is, the full quadrotor state can be expressed as smooth functions of a set of flat output variables and their higher derivatives. Specifically, the flat outputs are $\begin{bmatrix}\mathbf{r} & \psi\end{bmatrix}^\top$ [38]. As a result, any set of sufficiently smooth time-parametrized functions of the flat variables can be executed by the vehicle. We also note that the quadrotor is a fourth-order system.

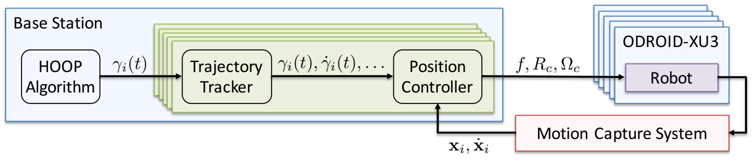
In our experiments, we maintain a constant yaw, $\psi = 0$, and a constant designated altitude. We use the HOOP algorithm, with fourth-order dynamics, to generate trajectories in the plane. Notationally, we will use $\mathbf{r}$ to represent a quadrotor's $xyz$ position and $\mathbf{x}$ to refer to its position in the $xy$ plane.

We determine the quadrotor inputs using a geometric controller [51], summarized below:

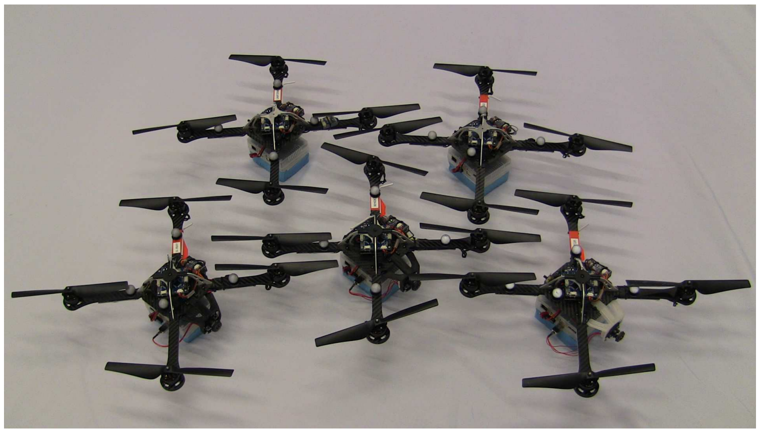
$$\begin{aligned} f &= (k_r \mathbf{e}_r + k_v \mathbf{e}_v + mg\mathbf{e}_3 + m\ddot{\mathbf{r}}_d) \cdot R\mathbf{e}_3 \\ \mathbf{M} &= k_R \mathbf{e}_R + k_{\mathbf{\Omega}} \mathbf{e}_{\mathbf{\Omega}} + \mathbf{\Omega} \times \mathbb{I}\mathbf{\Omega} - \\ &\qquad \mathbb{I}\left(\hat{\mathbf{\Omega}} R^\top R_c \mathbf{\Omega}_c - R^T R_c \dot{\mathbf{\Omega}}_c\right). \end{aligned}$$

Details regarding the calculation of the state errors is presented in [51]. It is worth noting that the desired quantities $\mathbf{r}_d$, $\dot{\mathbf{r}}_d$, $\ddot{\mathbf{x}}_d$ come directly from the input trajectories, while the other desired states, $R_c$ and $\dot{\mathbf{\Omega}}_c$, are calculated using differential flatness relations. The moment $M$ ultimately becomes a function of $\mathbf{r}^{(4)}$. We note that for fourth-order systems, HOOP plans $7^{(th)}$-order polynomial trajectories, which have continuous snap, confirming that our algorithm is compatible with quadrotor systems.

Figure 7a shows our experimental architecture. Our algorithm runs on a centralized base station computer that executes the HOOP algorithm and passes the generated trajectory for each robot to a trajectory tracker, which works with the position controller to calculate thrust and attitude commands. These are then sent



(a) System architecture of the quadrotor testbed. A base station computer runs the HOOP algorithm and a trajectory tracker and position controller for each robot. The robots each run an onboard attitude controller. The control loop is closed with feedback from the motion capture system.



(b) AscTec Hummingbird quadrotors. Each quadrotor carries an ODROID XU3 computer with a wireless ethernet adaptor.

Figure 7: Illustration of the experimental setup.

to the appropriate robot's onboard attitude controller. The motion capture system provides position and velocity feedback at 100 Hz.

We present a number of experiments validating the solution quality and robustness of the proposed algorithm. We then demonstrate one example application of our algorithm in a three-dimensional workspace. Video footage of these experiments can be found at https://youtu.be/812GUyc1lm0.

We quantify our results with a number of metrics. Let $\gamma(t) = \{\gamma_i(t) : \mathbb{R} \to \mathbb{R}^2 \quad \forall i \in [1, N]\}$ be the set of solution trajectories planned by the HOOP algorithm in the $xy$-plane. Let $K$ data points be collected at times $t[k]$. We use "velocity" to refer to the magnitude of a vehicle's velocity in the plane, namely:

$$v^i[k] = \|\dot{\gamma}^i(t[k])\|_2.$$

We will characterize a solution by the maximum velocity attained by any robot, namely:

$$v_{max} = \max_{i\in[1,N]} \left( \max_{k\in[1,K]} v^i[k] \right).$$

"Position error" will refer to the magnitude of the vehicle's position error in the plane, namely:

$$e^i[k] = \|\mathbf{x}^i[k] - \gamma^i(t[k])\|_2.$$

The average position error at a given time is the average error across all robots, namely:

$$e_{avg}[k] = \frac{\sum_{i=1}^{N} e^i[k]}{N}.$$

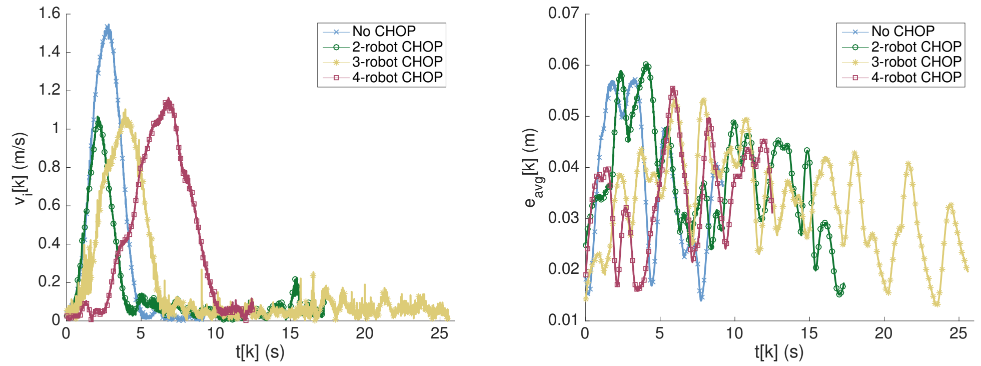
The average and maximum position error associated with an experiment is additionally averaged across time:

$$e_{avg} = \frac{\sum_{k=1}^{K} e_{avg}[k]}{K}$$
$$e_{max} = \max_{i\in[1,K]} e_{avg}[k].$$

Finally, "minimum separation" will refer to the minimum distance between the rotor tips of any two vehicles, namely:

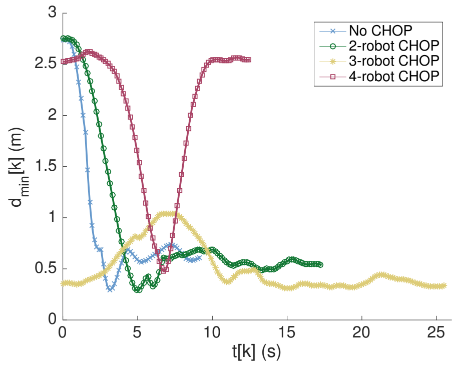
$$d_{min}[k] = \min_{i\in[1,N],j\in[1,N],i\neq j} \|\mathbf{x}^i[k] - \mathbf{x}^j[k]\|_2 - 0.54$$
$$d_{min} = \min_{k\in[1,K]} d_{min}[k].$$

## 8.1 Experimental Validation of Proposed Algorithm



(a) Velocity over time of fastest robot of each experiment.

(b) Average position error over time of each experiment.



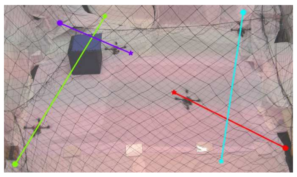
(c) Minimum separation over time of each experiment.

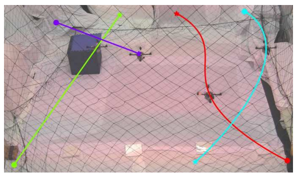
Figure 8: Performance results for four-robot experiments from one representative trial of each experiment.

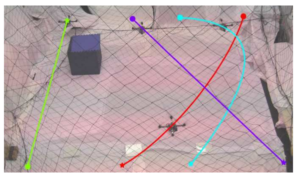
In this section, we present experiments validating the performance of the HOOP algorithm in a two-dimensional workspace. For all experiments, robots were held at a constant altitude of 1.5 m.

We first explored the performance characteristics of the algorithm by posing four problems with different geometries that require different amounts of interaction between four robots. These problems, and their solutions, are illustrated in Figs. 9a–9d. As before, robot's start positions are represented by circles and their goals are represented by stars of the same color. In Fig. 9a, the optimal trajectories are collision-free, therefore the robots' trajectories are straight-line paths to their goals (though their velocity profiles have been reparameterized). In Fig. 9b, the goal of the red robot is moved such that the red and blue robots must interact in a CHOP to avoid a collision. In Fig. 9c, the red, blue, and purple robots form a three-robot CHOP en-route to their goals, while the green robot continues to move directly to its goal. Finally, Fig. 9d illustrates the worst-case scenario, when all robots must enter a single, common CHOP to reach their goals. We see that, as expected, HOOP allows robots to take direct paths to their goals whenever possible and interact with neighbors only in congested regions of the workspace.

We reliably executed each solution over five trials. Figure 9 shows representative results from one trial of each experiment. Figure 8a displays the velocity profile taken by the robot that achieves the fastest velocity


(a) "No CHOP": All robots move directly to goals.


(b) "2-robot CHOP": Two robots form a CHOP while two robots move directly to goals.


(c) "3-robot CHOP": Three robots form a CHOP while one robot moves directly to goal.

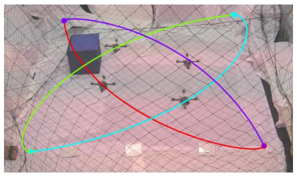

(d) "4-robot CHOP": All robots interact in a single CHOP.

Figure 9: Solution trajectories and performance data for four-robot experiments with varying geometries. Robots start at circles and must navigate to goal positions represented by stars of the same color. We show data from one representative trial of each experiment.

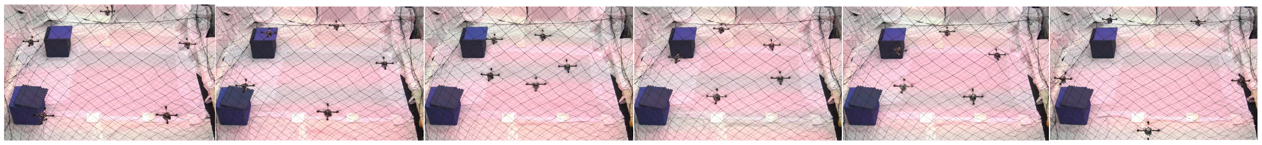


Figure 10: Snapshots of five quadrotors executing experiment seven, as described in Table 3.

in each problem. Clearly, even when a robot takes a straight-line trajectory to its goal, it travels with a non-constant velocity profile. Figure 8b shows the average position error over time, which we see is always below 0.06 m. Finally, Fig. 8c shows the minimum rotor-to-rotor separation between any two robots at a given time for each experiment. We see that in all experiments, robots come within 0.50 m of each other, yet still maintain a small position error.

Table 3: Performance statistics for teams of robots executing different CHOPs, calculated over five trials. CHOPs are characterized by $N$, the number of robots, $v_{max}$, the maximum forward velocity attained by any one robot, and $d_{min}$, the minimum rotor separation between any pair of neighbors. We report $e_{avg}$, the position error averaged across all robots and all trials, and $e_{max}$, the maximum position error.

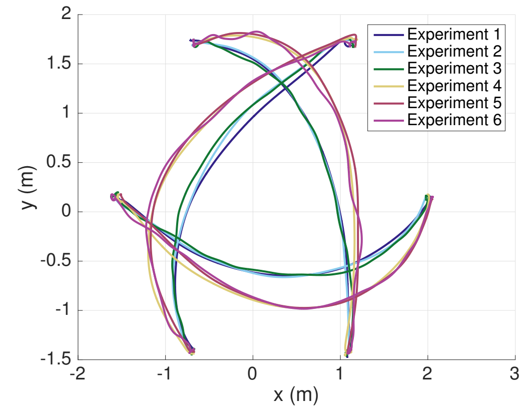
| No. | $N$ | $v_{max}$ (m/s) | $d_{min}$ (m) | $e_{avg}$ (m) | $e_{max}$ (m) |
|---|---|---|---|---|---|
| 1 | 3 | 0.61 | 0.69 | 0.045 | 0.12 |
| 2 | 3 | 1.4 | 0.70 | 0.047 | 0.12 |
| 3 | 3 | 1.9 | 0.67 | 0.053 | 0.14 |
| 4 | 3 | 0.63 | 1.1 | 0.045 | 0.11 |
| 5 | 3 | 1.2 | 1.2 | 0.052 | 0.13 |
| 6 | 3 | 1.7 | 1.3 | 0.060 | 0.14 |
| 7 | 5 | 1.1 | 0.28 | 0.039 | 0.14 |
| 8 | 5 | 1.5 | 0.57 | 0.045 | 0.14 |

Next, we verify the robustness of the system to unmodeled disturbances. In particular, we demonstrate accurate trajectory tracking in the presence of aerodynamic interactions between close neighboring vehicles, which has been shown to be a source of significant disturbances for quadrotors [52]. To this end, we present five trials each of eight experiments of robots executing a single CHOP at different velocities and proximities.

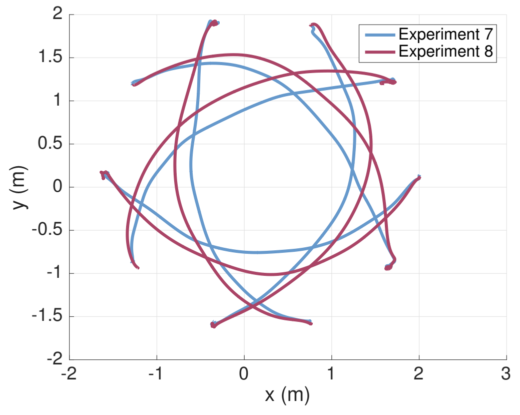
We characterize each CHOP with three parameters: $N$ — the number of robots, $v_{max}$, and $d_{min}$. The specific parameters of each experiment are listed in Table 3. Notably, in experiment seven, robots' rotors come within 0.28 m of each other. Snapshots of this maneuver are shown in Fig. 10.

Table 3 further reports errors in each experiment. In general, we see that the average error increases as robots attain higher velocities in their trajectories. However, the average position error is again always below 0.060 m. The maximum error is always below 0.14 m.

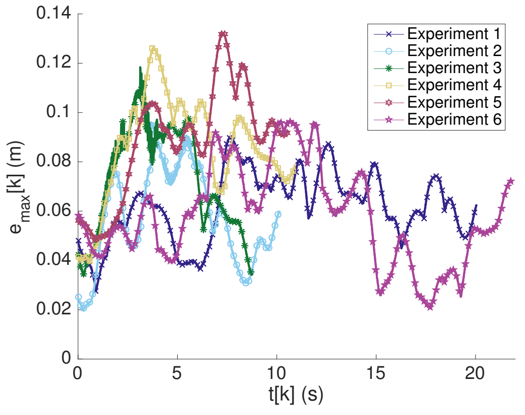
Figure 11 pictures data for one representative trial of each experiment. Figures 11a and 11b show the trajectories executed by robots in each experiment. Figures 11c–11f compare the maximum position error with the minimum separation between robots. As expected, the maximum position error increases when robots operates closer to their neighbors, subjecting them to increased aerodynamic disturbances. However,



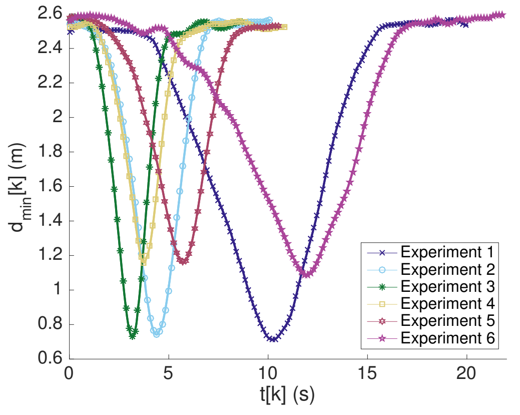
(a) Trajectories for three-robot experiments.



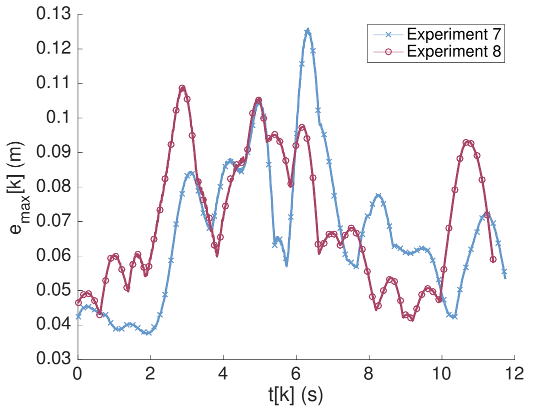
(b) Trajectories for five-robot experiments.



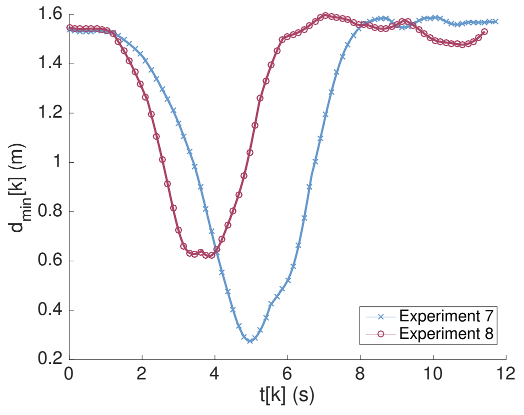
(c) Maximum position error over time for three-robot experiments.



(d) Minimum separation over time for three-robot experiments.



(e) Maximum position error over time for five-robot experiments.



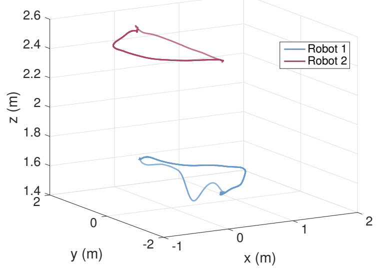
(f) Minimum separation over time for five-robot experiments.

Figure 11: Solution trajectories and performance data for robustness experiments. Experiment parameters are listed in Table 3. We show data for one representative trial of each experiment.

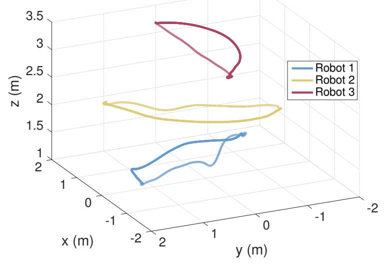
the error is always reasonably small, demonstrating our algorithm's ability to produce smooth, dynamically feasible trajectories that can be robustly executed by a real-world system.

## 8.2 Extension to Three-Dimensional Workspaces

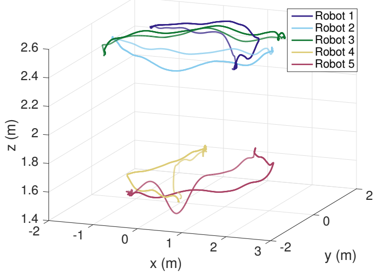
Finally, we demonstrate an extension of the HOOP algorithm to the three-dimensional workspace. For quadrotors posed with the labeled multi-robot planning problem, it might be tempting to simply stagger vehicles' altitudes instead of coordinating all vehicles in a common plane. However, the Federal Federal Aviation Administration (FAA) guidelines limit the altitudes of small Unmanned Aerial Systems (UASs) to below 400 ft [53], decreasing the available vertical workspace. Further, the downwash from quadrotors significantly perturbs the robots beneath them [52]. This effect has specifically been quantified for Hummingbird quadrotors; within a cylinder extending outwards by a radius of 0.5 m and downwards by a height of 1.5 m from a quadrotor [54], the $z$ displacement caused by an overhead vehicle is larger than 0.05 m and can be as much as 0.20 m.



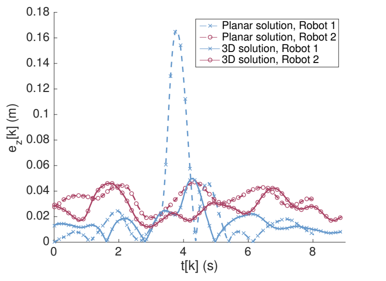
(a) Trajectories for two-robot experiment.



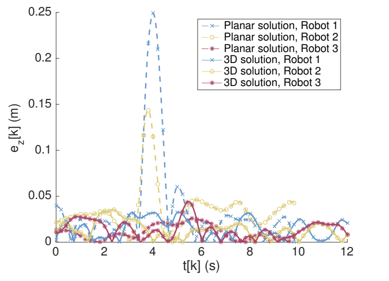
(b) Trajectories for three-robot experiment.



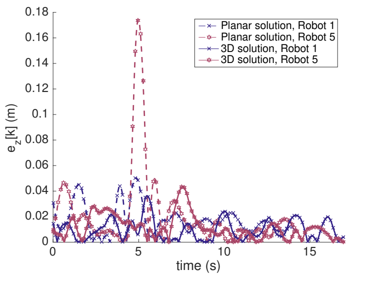
(c) Trajectories for five-robot experiment.



(d) Vertical position error over time for the two-robot experiment.



(e) Vertical position error over time for the three-robot experiment.



(f) Vertical position error over time for the five-robot experiment.

Figure 12: Solution trajectories and performance data for three-dimensional experiments from one representative trial.

Table 4: Performance statistics for three-dimensional experiments, calculated over five trials of each experiment. Each experiment is characterized by $N$, is the number of robots, $v_{max}$, the maximum forward speed attained by any one robot, and $d_z$, the minimum vertical separation between robots. We report average and maximum errors in the plane and in the vertical direction for the planar and 3D solutions.

| w $N$ | $v_{max}$ (m/s) | $d_z$ (m) | Planar | solution | errors | (m) | 3D | solution | errors | (m) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | $e_{z,avg}$ | $e_{z,max}$ | $e_{avg}$ | $e_{max}$ | $e_{z,avg}$ | $e_{z,max}$ | $e_{avg}$ | $e_{max}$ |
| 2 | 1.3 | 0.90 | 0.020 | 0.17 | 0.040 | 0.10 | 0.016 | 0.050 | 0.037 | 0.099 |
| 3 | 1.3 | 1.0 | 0.026 | 0.27 | 0.042 | 0.14 | 0.017 | 0.12 | 0.038 | 0.16 |
| 5 | 1.3 | 0.9 | 0.018 | 0.19 | 0.040 | 0.21 | 0.014 | 0.081 | 0.038 | 0.13 |

To mitigate this effect in scenarios where robots must fly in close altitude proximity to neighbors, we can apply the HOOP algorithm to coordinate robots at different altitudes to avoid directly flying above each other at all times. We conducted three experiments with five trials each, with teams of two, three, and five robots. In each instance, we first allowed robots to coordinate only with their coplanar neighbors — we will refer to this as the "planar solution". We compared this to allowing all robots in the three-dimensional workspace to coordinate using HOOP as if they were coplanar and subsequently execute their planned trajectories at their designated altitudes, thus ensuring no robots will through the downwash of another — we will refer to this as the "3D solution".

We conduct five trials of each solution for all three experiments. Figures 12a–11b show the trajectories executed by one trial of each experiment. For experiments with two and three robots, Figs. 12a and 12b, each robot flies at its own altitude. Thus, in the planar solution, all robots take straight-line, minimum-snap trajectories directly to their goal. For the five-robot experiment, Fig. 11b, three robots operate at a higher altitude than the remaining two. In the planar solution, a CHOP is formulated and executed independently at each height. Clearly, when executing the planar solutions, robots in lower planes show a significant error in $z$ when passing below their neighbors. In contrast, in the 3D solutions, robots' trajectories show significantly smaller altitude errors.

This effect is quantified in Figs. 12d–12f, which plot the vertical position errors over time for robots in the planar and 3D solutions. In each case, there is a clear, significant disturbance in the $z$ direction at around 4 s, when robots at lower altitudes pass below their neighbors. In Figs. 12d and 12f, we see that a robot is perturbed by around 0.10 m. This effect is even more severe in Fig. 12e, where Robot 1, the blue robot, is subject to the downwash of two higher neighbors and is perturbed by 0.20 m.

Table 4 lists error statistics for each experiment, calculated across all five trials. Note that here, we characterize each experiment with $d_z$, the altitude separation between robots. $e_{avg}$ and $e_{max}$ represent planar errors, whereas $e_{z,avg}$ and $e_{z,max}$ report altitude errors. Again, we see significant decreases in both average and maximum errors in altitude in using the 3D solution as opposed to the planar solution. We also note that planar errors are similar for both solutions.

In short, we see that disturbances from downwash, especially from multiple overhead neighbors, is potentially dangerous for quadrotors flying at staggered altitudes, as a sudden deviation downwards can cause unexpected collisions. Coordinating all robots with HOOP allows robots to avoid the column of downwash from neighbors and allow for successful coordination in the three-dimensional workspace.

## 9 Conclusions

In his work, we present Hold or take Optimal Plan (HOOP), a centralized algorithm for generating safe trajectories for the labeled multi-robot planning problem in obstacle-free, two-dimensional workspaces. This algorithm uses a geometric algorithm to quickly arrive at a motion plan, which consists of safe, piecewise linear trajectories for the team, assuming the vehicles have first order dynamics. It then refines each robot's motion plan into a smooth trajectory by formulating and solving a QP that explicitly incorporates the $n^{th}$-order dynamics. Our algorithm is safe, complete, and runs in polynomial, rather than exponential, time. We validate the method through extensive quadrotor experiments.

A known disadvantage of centralized multi-robot systems is their reliance on a connected network and their dependency on a single base station. With a view of overcoming this limitation, we have designed a number of basic components of HOOP to be conducive to a decentralized approach. Specifically, the CHOPs in the motion planning step are constructed analytically, and each robot only needs to know the start and goal positions of their neighbors to arrive at the same collision avoidance maneuver. In the trajectory generation step, the QPs for each robot are decoupled, and each robot can solve only for its own trajectory. Current research efforts are focused on developing a decentralized version of the HOOP algorithm that will allow robots to navigate using only local information about their neighbors.

## Acknowledgments.

We gratefully acknowledge the support of ONR grants N00014-09-1-1051 and N00014-09-1-103, NSF grant IIS-1426840, ARL grantW911NF-08-2-0004, ARO grant W911NF-13-1-0350, and Exyn Technologies. Sarah Tang is supported by NSF Research Fellowship Grant No. DGE-1321851.

## References

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," vol. 29, no. 1, pp. 9–20, 2008.

[2] D. Reisinger, “Watch amazon’s prime air complete its first drone delivery,” Fortune, December 2016. [Online]. Available: http://fortune.com/2016/12/14/amazon-prime-air-delivery/

[3] J. Stewart, “Google tests drone deliveries in project wing trials,” BBC News, December 2013. [Online]. Available: http://www.bbc.com/news/technology-28964260

[4] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus, “Ikeabot: An autonomous multi-robot coordinated furniture assembly system,” in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[5] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, “Image and animation display with multiple mobile robots,” *The International Journal of Robotics Research (IJRR)*, 2012.

[6] K. Solovey, O. Salzman, and D. Halperin, “Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning,” *The International Journal of Robotics Research (IJRR)*, 2016.

[7] M. Goldenberg, A. Felner, R. Stern, G. Sharon, N. Sturtevant, R. C. Holte, and J. Schaeffer, “Enhanced partial expansion A*,” *Journal of Artificial Intelligence Research*, 2014.

[8] T. Standley, “Finding optimal solutions to cooperative pathfinding problems,” in *AAAI Conference on Artificial Intelligence*, Atlanta, GA USA, July 2010.

[9] G. Wagner and H. Choset, “Subdimensional expansion for multirobot path planning,” *Artificial Intelligence*, 2015.

[10] J. Yu and S. M. LaValle, “Planning optimal paths for multiple robots on graphs,” in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[11] J. Yu and D. Rus, “An effective algorithmic framework for near optimal multi-robot path planning,” in *International Symposium on Robotics Research (ISRR)*, Sestri Levante, Italy, September 2015.

[12] B. de Wilde, A. W. ter Mors, and C. Witteveen, “Push and rotate: Cooperative multi-agent path planning,” in *International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, Saint Paul, MN, USA, May 2013.

[13] R. Luna and K. E. Bekris, “Push and swap: Fast cooperative path-finding with completeness guarantees,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Spain, July 2011.

[14] J. van den Berg, J. Snoeyink, M. Lin, and D. Manocha, “Centralized path planning for multiple robots: Optimal decoupling into sequential plans,” in *Robotics: Science and Systems (RSS)*, Seattle, WA USA, June 2009.

[15] T. S. Standley and R. E. Korf, “Complete algorithms for cooperative pathfinding problems.” in *International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Spain, July 2011.

[16] K. Kant and S. W. Zucker, “Toward efficient trajectory planning: The path-velocity decomposition,” *The International Journal of Robotics Research (IJRR)*, 1986.

[17] J. Peng and S. Akella, “Coordinating multiple robots with kinodynamic constraints along specified paths,” *The International Journal of Robotics Research (IJRR)*, 2005.

[18] T. Siméon, S. Leroy, and J. Laumond, “Path coordination for multiple mobile robots: a resolution-complete algorithm,” *IEEE Transactions on Robotics and Automation*, 2002.

[19] M. Bennewitz, W. Burgard, and S. Thrun, “Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots,” *Robotics and Autonomous Systems*, vol. 41, no. 2, pp. 89–99, 2002.

[20] S. Buckley, “Fast motion planning for multiple moving robots,” in *IEEE International Conference on Robotics and Automation (ICRA)*, Scottsdale, AZ USA, May 1989.

[21] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, 1986.

[22] J. van den Berg and M. Overmars, "Prioritized motion planning for multiple robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Edmonton, Canada, August 2005.

[23] M. Cap, P. Novak, J. Vokrnek, and M. Pvechoucek, "Multi-agent RRT*: Sampling-based cooperative pathfinding," in *International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, St. Paul, MN USA, May 2013.

[24] M. Peasgood, C. M. Clark, and J. McPhee, "A complete and scalable strategy for coordinating multiple robots within roadmaps," *IEEE Transactions on Robotics*, 2008.

[25] J.Alonso-Mora, T. Naegeli, R. Siegwart, and P. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, 2015.

[26] L. Pallottino, E. M. Feron, and A. Bicchi, "Conflict resolution problems for air traffic management systems solved with mixed integer programming," *IEEE Transactions on Intelligent Transportation Systems*, 2002.

[27] Z. Sunberg, M. Kochenderfer, and M. Pavone, "Optimized and trusted collision avoidance for unmanned aerial vehicles using approximate dynamic programming," in *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016.

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

[29] P. Spirakis and C. K. Yap, "Strong np-hardness of moving many discs," *IEEE Transactions on Robotics*, vol. 19, no. 1, pp. 55–59, 1984.

[30] K. E. Bekris, D. K. Grady, M. Moll, and L. E. Kavraki, "Safe distributed motion coordination for second-order systems with different planning cycles," *The International Journal of Robotics Research (IJRR)*, 2012.

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

[32] M. Turpin, N. Michael, and V. Kumar, "CAPT: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research (IJRR)*, 2014.

[33] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of interchangeable robots," *Autonomous Robots*, 2014.

[34] M. E. Flores, "Real-time trajectory generation for constrained nonlinear dynamical systems using non-uniform rational b-spline basis functions," Ph.D. dissertation, California Institute of Technology, 2008, ph.D.thesis.

[35] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016.

[36] P. Tournassoud, "A strategy for obstacle avoidance and its application to multi-robot systems," in *IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA USA, April 1986.

[37] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, 2017.

[38] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011, pp. 2520–2525.

[39] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *The International Symposium of Robotics Research (ISRR)*, Singapore, December 2013.

[40] ——, "Polynomial trajectory planning for quadrotor flight," in *IEEE International Conference on Robotics and Automation (ICRA)*, Karlsruhe, Germany, May 2013.

[41] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 2016.

[42] V. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams in complex environments using rapidly-exploring random trees," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 2011.

[43] A. Wiktor, D. Scobee, S. Messenger, and C. Clark, "Decentralized and complete multi-robot motion planning in confined spaces," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL USA, September 2014.

[44] S. Omidshafiei, A. akbar Agha-mohammadi, C. Amato, and J. P. How, "Decentralized control of partially observable markov decision processes using belief space macro-actions," in *IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA USA, May 2015.

[45] S. Tang and V. Kumar, "A complete algorithm for generating safe trajectories for multi-robot teams," in *International Symposium on Robotics Research (ISRR)*, Sestri Levante, Italy, September 2015.

[46] ——, "Safe and complete trajectory generation for large teams of robots with higher-order dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Deajeon, South Korea, October 2016.

[47] S. Tang, J. Thomas, and V. Kumar., "Safe navigation of quadrotor teams to labeled goals in limited workspaces," in *International Symposium on Experimental Robotics (ISER)*, Tokyo, Japan, October 2016.

[48] H. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics*, 1955.

[49] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. Berlin Heidelberg: Springer Verlag, 1997, ch. 2, pp. 47–80.

[50] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge, United Kingdom: Cambridge University Press, 2004, ch. 2, pp. 21–66.

[51] T. Lee, M. Leok, and N. H. McClamroch, "Control of complex maneuvers for a quadrotor UAV using geometric methods on SE(3)," vol. 15, no. 2, pp. 391–408, 2011.

[52] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar, "Influence of aerodynamics and proximity effects in quadrotor flight," in *International Symposium on Experimental Robotics (ISER)*, Quebec City, Canada, June 2012.

[53] FAA, "Overview of small uas notice of proposed rulemaking," February 2015.

[54] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed," *IEEE Robotics Automation Magazine*, 2010.