Trajectory Planning for Heterogeneous Robot Teams

Mark Debord, Wolfgang Hönig, and Nora Ayanian

***Abstract*— We describe a trajectory planning method for heterogeneous mobile robot teams in known environments. We consider two core problems that arise with heterogeneous robot teams: asymmetric inter-robot collision constraints and varying dynamic limits. Asymmetric collision constraints are important for close-proximity flight of rotorcraft due to the downwash effect, which complicates spatial coordination. Varying dynamic limits complicate temporal coordination between robots and must be taken into account during planning. Our method builds upon a hybrid planner that combines graph-planning techniques with trajectory optimization and scales well to large homogeneous robot teams. We extend the hybrid planning approach to include the additional spatial and temporal coordination to support heterogeneous teams. Our method scales well with the number of robots and robot types and we demonstrate our approach on a team of 15 physical robots of 4 different types, including quadrotors and differential drive robots.**

I. INTRODUCTION

Trajectory planning for heterogeneous teams of robots is a core problem for many potential applications of multi-robot systems. To accomplish complex tasks, it could be beneficial for a team to be composed of different types of robots with varied capabilities. This complicates trajectory planning due to differing dynamics and mixed requirements for allowable interactions between robots. For example, downwash from rotorcraft is an effect that other nearby rotorcraft must consider in order to maintain stable flight, but ground robots are not necessarily affected by downwash. Figure 1 shows an example of a physical experiment in which many quadrotors of different sizes must fly in close proximity and thus be aware of other quadrotors' downwash while also considering the motion of ground robots.

In this work, we extend downwash-aware trajectory planning for large quadrotor teams [1] to heterogeneous teams of robots. The proposed method is centralized and is designed for *a priori* calculation of trajectories in known environments. The high-level structure of the original approach is retained. First, a graph-based planning method is used to compute a collision-free discretized schedule for all robots in the team. Then, a second parallelizable optimization stage refines these schedules into locally optimal smooth trajectories. Our extension introduces the capability of utilizing independent and asymmetric collision constraints between different pairs of robot types. Additionally, it generates trajectories that obey the dynamic limits of all robots types in the team while retaining temporal alignment. Like the original method, the one presented here scales well to teams with large numbers of robots, with additional processing time mostly due to offline roadmap generation.

All authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA.
Email: {mjdebord, whoenig, ayanian}@usc.edu
This work was supported by ONR N00014-14-1-0734.

Fig. 1. Heterogeneous robot team with ten small UAVs (blue, only eight visible), two medium UAVs (red), one large UAV (green), and two ground robots (yellow). The robots have to navigate through a cluttered environment, avoiding obstacles and taking asymmetric inter-robot constraints into account.

II. RELATED WORK

Homogeneous multi-robot motion planning problems have been addressed using a variety of approaches in the past, including graph-based and optimization-based methods. If the problem specification can be discretized into a graph, a discrete solution can be found with search-based algorithms [2]. Such algorithms can find solutions for hundreds of robots quickly, but ignore kinodynamic constraints. Another approach is creating a large optimization problem in the joint space of all robots [3]. Such formulations might include kinodynamic constraints, but do not scale very well to large robot teams. We extend an approach that combines graph-based search with trajectory optimization [1], [4]. This hybrid planning method works well with hundreds of homogeneous robots, considers kinodynamic constraints, and can take inter-robot constraints into account.

Motion planning for heterogeneous robots can employ optimization-based methods, graph-based methods, or reactive planners. Mixed-integer quadratic programs (MIQPs) can plan for a team of different quadrotors, taking their different sizes and aerodynamic effects between quadrotors into account [5]. This method has been applied to up to four physical quadrotors (three different sizes) and produces optimal trajectories, but does not scale well to a large number of robots and/or obstacles. Similar to our approach, inter-robot constraints are modelled asymmetrically, allowing large quadrotors to fly underneath smaller ones. Optimization-based methods can also be combined

with sequential planning to achieve better scalability [6]; this method, unlike our work, does not consider asymmetric interactions between differently sized robots and has not been demonstrated in 3D.

Graph-based planning can be combined with controller-based motion primitives to include complex dynamics during planning. Planning in the joint space of all robots can be used to model heterogeneous teams such as a collaboration between a UGV and a UAV [7], but it does not scale well to a large number of robots. Other approaches use graph-based planning for homogeneous robots that have different operating modes, such as flying cars [8]. Here, the motion planner can consider switching between driving and flying, but the resulting trajectories are not smooth and asymmetric inter-robot constraints are not considered.

Velocity obstacle approaches have also been applied to heterogeneous robot teams [9], but while such local planners can find solutions quickly, they may not find solutions in cluttered environments.

III. APPROACH

The first step is to specify the types of robots in the heterogeneous team and the geometric interaction models that are required to define collision-free trajectories. We then outline the major components of our approach and explain the major extensions to the previous work that enable heterogeneous teams.

A team of robots is considered heterogeneous if it consists of multiple robots with different physical capabilities or dynamic limits. We focus our efforts on two classes of robots, quadrotors and differential drive wheeled robots, however, the presented method applies for any differentially flat system. Each of these classes can be further delineated into *types* which may be of different sizes or have varied dynamic limits, such as maximum accelerations or velocities.

Quadrotors generate a fast-moving volume of air called downwash that impacts the ability of other quadrotors to fly directly below them. In the prior work this was addressed by modeling the collision volume of a quadrotor as an axis-aligned ellipsoid centered at the quadrotor's position [1]. This is not sufficient in the heterogeneous case, as the effect is asymmetric with respect to the sizes of the interacting quadrotors. For example, a large quadrotor is likely able to fly below a smaller one without difficulty, but the opposite is not true. Additionally, wheeled robots likely do not need to consider the downwash effect at all. Thus, we define independent collision volumes for every possible pair of interacting robot types.

A. Collision Model

Consider a team of N robots, where each robot is one of M types. The environment is defined as a set of convex obstacles $\mathcal{O}_1 \dots \mathcal{O}_{N_{obs}}$ within a boundary defined by a convex polytope $\mathcal{W}$. For each robot of type $k \in \{1 \dots M\}$ we define a convex volume $\mathcal{R}^k_{\mathcal{E}}(\mathbf{q})$ that represents the robot-environment collision volume for the k type robot at position $\mathbf{q} \in \mathbb{R}^3$. The set $\mathcal{F}^k$ describes the free configuration space

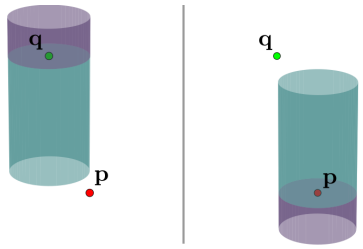


Fig. 2. An illustration of the cylindrical collision geometries for a large quadrotor of type k at position $\mathbf{q}$ and a small quadrotor of type l at position $\mathbf{p}$. In this case the tuple $R^{(k,l)}_{\mathcal{R}}(\mathbf{q}) : \langle r, a, b\rangle$ has $b > a$ to model the asymmetry between the downwash zones as shown on the left side. $R^{(l,k)}_{\mathcal{R}}(\mathbf{p})$ is symmetric to $R^{(k,l)}_{\mathcal{R}}(\mathbf{q})$ as shown on the right side.

for a robot of type k with respect to the environment. $\mathcal{F}^k$ is defined:

$$\mathcal{F}^k = (\mathcal{W} \backslash (\bigcup_{h=1}^{N_{obs}} \mathcal{O}_h)) \ominus \mathcal{R}^k_{\mathcal{E}}(\mathbf{0}), \tag{1}$$

where $\ominus$ denotes the Minkowski difference.

We define separate convex geometries to describe the unique collision constraints that exist between every robot type-pair. Consider a robot $r^{(i,k)}$ with index $i \in \{1 \dots N\}$ and type $k \in \{1 \dots M\}$ at position $\mathbf{q}$ and another robot $r^{(j,l)}$ at position $\mathbf{p}$. If $r^{(i,k)}$ is at position $\mathbf{q}$, then $r^{(j,l)}$ cannot occupy the convex volume $\mathcal{R}^{(k,l)}_{\mathcal{R}}(\mathbf{q})$. Likewise, if $r^{(j,l)}$ is at position $\mathbf{p}$, then $r^{(i,k)}$ cannot occupy the convex volume $\mathcal{R}^{(l,k)}_{\mathcal{R}}(\mathbf{p})$. Specifically, we say there is a collision between the two robots if $\mathbf{p} \in \mathcal{R}^{(k,l)}_{\mathcal{R}}(\mathbf{q})$, or equivalently, $\mathbf{q} \in \mathcal{R}^{(l,k)}_{\mathcal{R}}(\mathbf{p})$.

If k and l both specify types of quadrotors, then $R^{(k,l)}_{\mathcal{R}}(\mathbf{q})$ defines both the downwash and physical collision volume of the k type quadrotor with respect to l type. If either k or l specify a type of wheeled robot, then $R^{(k,l)}_{\mathcal{R}}(\mathbf{q})$ only specifies the physical collision volume.

For all cases of (k, l) type pairs, we parameterize the collision volumes with a tuple $R^{(k,l)}_{\mathcal{R}}(\mathbf{q}) : \langle r, a, b\rangle$ that specifies an axis-aligned cylinder of radius r where the top of the cylinder is located at $q_z + a$ and the bottom at $q_z - b$. The parameter r represents the minimum safe horizontal distance between the positions of the two robot types. The parameters a and b specify the minimum safe vertical distance for the l type robot above and below $\mathbf{q}$, respectively. These parameters are experimentally determined for real robots as described in Section VI.

Note that the definition of the cylinder for the (k, l) pair is symmetric with respect to the (l, k) pair. That is, if $R^{(k,l)}_{\mathcal{R}}(\mathbf{q}) : \langle r, a, b\rangle$, then $R^{(l,k)}_{\mathcal{R}}(\mathbf{p}) : \langle r, b, a\rangle$. Figure 2 shows an example of these cylinder definitions for the case of a large and small quadrotor. Also note that the cylinder model is an approximation of the downwash effect as it does not directly account for orientation or acceleration. To account for inaccuracies, we assume cylinder parameters are conservative.

Let $f^{(i,k)} : [0, T] \mapsto \mathbb{R}^3$ be the trajectory of robot $r^{(i,k)}$ where T represents the time that the last robot in the team

reaches its goal. Trajectories are considered collision free if there are no robot-environment collisions and no inter-robot collisions:

$$\begin{aligned} &f^{(i,k)}(t) \in \mathcal{F}^k && \forall i,\, 0 \leq t \leq T \\ &f^{(i,k)}(t) \notin \mathcal{R}_{\mathcal{R}}^{(l,k)}(f^{(j,l)}(t)) && \forall i \neq j,\, 0 \leq t \leq T. \end{aligned} \tag{2}$$

B. Problem Statement

Given the following:

- an environment specified by $\mathcal{W}$ and $\mathcal{O}_1 \dots \mathcal{O}_{N_{obs}}$;
- a set of N robots $r^{(i,k)}$, $i \in \{1 \dots N\}$ and $k \in \{1 \dots M\}$;
- start locations $\mathbf{s}^{(i,k)}$ and goal locations $\mathbf{g}^{(i,k)}$ for each robot;
- the robot-environment collision model $\mathcal{R}_{\mathcal{E}}^{k}(\cdot)$ for each robot type k and the inter-robot collision model $R_{\mathcal{R}}^{(k,l)}(\cdot)$ for each robot type-pair (k, l); and
- the maximum velocity $v_{\max}^{k}$ and acceleration limits $a_{\max}^{k}$ for each robot type k;

our goal is to compute T and a kinodynamically feasible trajectory $f^{(i,k)}$ that is collision-free according to (2) for each robot $r^{(i,k)}$ such that $f^{(i,k)}(0) = \mathbf{s}^{(i,k)}$ and $f^{(i,k)}(T) = \mathbf{g}^{(i,k)}$. This trajectory planning problem is often referred to as the *labeled* case, because each robot has its goal assigned a priori. We also consider the *k-color* case, where robots of the same type are interchangeable and allowed to swap goal assignments.

C. Overview of Approach

As described previously, the high level components of the presented method share the same structure as the prior work [1]. To extend the work to the heterogeneous case, several modifications to the discrete scheduling and trajectory optimization stages are necessary. In particular, the roadmap generation and conflict annotation phases are extended to account for the different free space definitions and velocity limits of each type of robot. Additionally, the construction of safe corridors in the trajectory optimization stage is generalized to account for the different collision volumes for each robot and robot type-pair. The following sections detail these modifications.

IV. Roadmap Generation, Conflict Annotation, Discrete Planning

The first phase of our hybrid planning method is to generate collision free *discrete schedules* for every robot in the team. A discrete schedule assigns each robot a line segment to traverse for a specific timestep; the segment might consist of a single point if the robot should be stationary during that timestep. The discrete schedule guarantees collision-free execution if the robots move along their segments. Each robot can traverse its segment using any velocity profile during the timestep, e.g., a differential-drive robot might turn in place before moving.

For homogeneous robot teams, a discrete schedule can be computed by first generating a roadmap and then solving a multi-agent path-finding problem on that roadmap. When applying this approach to motion planning for heterogeneous teams, three problems arise. First, the free space definition for each type of robot is different because of varying physical extent or different methods of locomotion. Second, the inter-robot conflicts are specific to the types of robots that are interacting, e.g., a small quadrotor cannot fly closely below a large quadrotor but it is able to fly closely to static obstacles or ground robots. Third, each robot type has different dynamic limits such as maximum velocities.

We address all three problems by constructing a *super roadmap*, which is the disjoint union of the roadmaps for the individual robot types. We show that the super roadmap can be used as a drop-in replacement for a regular roadmap as input to existing planning algorithms.

A. Super Roadmap Generation

A roadmap for robot type $k \in \{1 \dots M\}$ is an undirected connected graph of the environment $\mathcal{G}^k = (\mathcal{V}^k, \mathcal{E}^k)$, where each vertex $v \in \mathcal{V}^k$ corresponds to a location in $\mathcal{F}^k$ and each edge $(u, v) \in \mathcal{E}^k$ denotes that there is a linear path in $\mathcal{F}^k$ connecting u and v.

We generate the roadmap $\mathcal{G}^k$ given a representation of the environment and the shape of $\mathcal{R}_{\mathcal{E}}^{k}(\cdot)$ using the SPARS algorithm [10]. SPARS generates dense and sparse roadmaps such that the sparse roadmap is a subgraph of the dense one, while keeping any-pair shortest distances within a user-specified sub-optimality factor. Another parameter, Δ^k, controls the visibility radius and roughly corresponds to the average edge length of the generated sparse roadmap. We choose Δ^k to be smaller for slower robots, reflecting that they can travel shorter distances compared to faster ones in the same amount of time. Specifically, we generate $\mathcal{G}^k$ using:

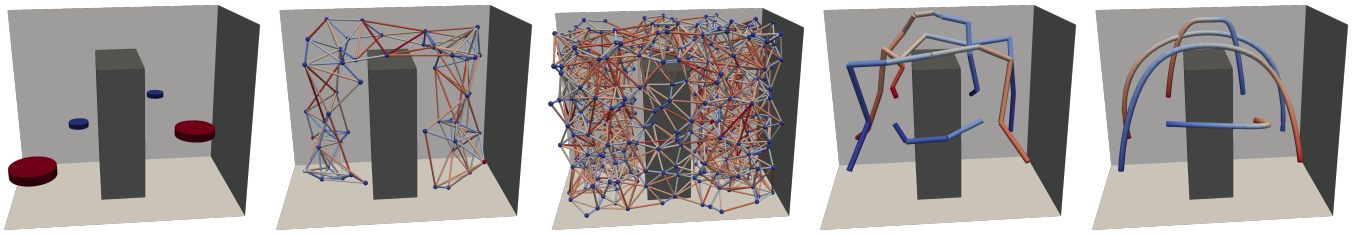
$$\Delta^k = \alpha v_{\max}^{k}, \quad \forall k \in \{1 \dots M\}, \tag{3}$$

where α corresponds to the sparsity of all roadmaps.

After executing SPARS, we add additional vertices that correspond to $\mathbf{s}^{(i,k)}$ and $\mathbf{g}^{(i,k)}$ to $\mathcal{V}^k$ for all $i \in \{1 \dots N\}$. We connect those vertices to neighboring vertices by adding additional edges to $\mathcal{E}^k$.

Consider an example with two small and two large quadrotors in an environment with an obstacle (see Fig. 3(a)), where each quadrotor has to move to the opposite side of the obstacle. The physical extent of the large UAVs forces them to fly over the obstacle (see roadmap in Fig. 3(b)), while the small UAVs can fly over or in front of the obstacle (see roadmap in Fig. 3(c)). In this example we also assume that the large UAVs can fly twice as fast as the small ones and select the Δ^k values accordingly. The resulting roadmap for the large UAV has fewer edges that are longer on average (166 edges with average length of $0.47\,\mathrm{m}$) compared to the roadmap for the small UAVs (1712 edges with average length of $0.28\,\mathrm{m}$).

For ground robots, we limit the roadmap generation to two dimensions. We can combine the separate roadmaps into a *super roadmap* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by computing the disjoint union or graph sum:



(a) Initial configuration. (b) Roadmap for large/fast UAVs. (c) Roadmap for small/slow UAVs. (d) Discrete trajectories. (e) Continuous trajectories.

Fig. 3. Example with two small and two large UAVs, one of each type on each side of the obstacle. The UAVs are tasked with moving to goal locations on the opposite side of the obstacle.

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) = \left(\bigcup_{k=1}^{M} \mathcal{V}^k, \bigcup_{k=1}^{M} \mathcal{E}^k \right). \tag{4}$$

Each vertex $v \in \mathcal{V}$ is associated with a position $\mathbf{q} \in \mathbb{R}^3$ and we denote this relationship by $\mathbf{q} = loc(v)$. The vertex set $\mathcal{V}$ is surjective to $\mathbb{R}^3$, i.e., a point in Euclidean space might correspond with multiple vertices (up to one for each robot type). Each vertex $v \in \mathcal{V}$ and edge $e \in \mathcal{E}$ have their respective robot type k associated and we refer to it by $k = type(v)$ and, with a slight abuse of notation, $k = type(e)$.

B. Conflict Annotation

The super roadmap $\mathcal{G}$ can be directly used for path planning for a single robot to avoid robot-obstacle collisions. When using multiple robots of either the same or different types, inter-robot constraints must be considered. An offline pre-processing step annotates the roadmap with potential inter-robot conflicts, by adding the following types of constraints:

Vertex-Vertex Constraints Two robots may not concurrently occupy two vertices which are in close proximity to each other. We can compute the collision set for each vertex $v \in \mathcal{V}$ by checking if a point is in the respective collision cylinder:

$$\begin{aligned} conVV(v) = \{u \in \mathcal{V} | \\ loc(u) \in \mathcal{R}_{\mathcal{R}}^{(type(v),type(u))}(loc(v))\}. \end{aligned} \tag{5}$$

Edge-Vertex Constraints One robot may not traverse an edge if a collision could occur with another stationary robot. A robot may become stationary during discrete planning if it is determined that to avoid collisions, it should wait at its current vertex rather than traverse an edge. Let $\hat{\mathcal{R}}_{\mathcal{R}}^{(k,l)}(e)$ be the convex hull that is created when $\mathcal{R}_{\mathcal{R}}^{(k,l)}(\cdot)$ is swept along edge e. We can compute the collision set for edge e by checking if the location associated with a vertex lies within that convex hull:

$$\begin{aligned} conEV(e) = \{v \in \mathcal{V} | \\ loc(v) \in \hat{\mathcal{R}}_{\mathcal{R}}^{(type(e),type(v))}(e)\}. \end{aligned} \tag{6}$$

Edge-Edge Constraints Two robots may not concurrently traverse two edges if a collision could occur during the traversal. Let $\hat{d}$ be the set of points comprising edge d. We can compute the collision set for each edge e by checking for intersection between the convex hull and the line segment that is defined by edge d:

$$conEE(e) = \{d \in \mathcal{E} \,|\, \hat{d} \subset \hat{\mathcal{R}}_{\mathcal{R}}^{(type(e),type(d))}(e)\}. \tag{7}$$

C. Discrete planning

The discrete path planning problem can now be formulated as an instance of *Multi-Agent Path-Finding with Generalized Conflicts* (MAPF/C) [1]. The inputs are the annotated roadmap and start and goal vertices for the individual robots. The output is a discrete trajectory for each robot such that all constraints are obeyed. A discrete trajectory $p^{(i,k)}$ for each robot $r^{(i,k)}$ is composed of a sequence of $K+1$ locations:

$$p^{(i,k)} = \mathbf{x}_0^{(i,k)}, \mathbf{x}_1^{(i,k)}, \ldots, \mathbf{x}_K^{(i,k)}, \tag{8}$$

where robots are synchronized in time and have to arrive at location $\mathbf{x}_n^{(i,k)}$ at timestep n. We define $\ell_n^{(i,k)}$ as the line segment between locations $\mathbf{x}_n^{(i,k)}$ and $\mathbf{x}_{n+1}^{(i,k)}$. Example discrete trajectories are shown in Fig. 3(d). Each edge is color-coded by the timestep n in which it will be traversed: blue at $n = 0$, gradually changing to red at $n = K$. The small UAV in the back flies on top of a large one (which is safe according to our collision model), while the small one in front has to keep a large vertical safety distance in order to pass below the other large UAV.

A robot is restricted to move on the roadmap for its own type by construction of the super roadmap using the disjoint union of the per-type roadmaps. As our solver, we use a variant of *Enhanced Conflict-Based Search* (ECBS) [11] that takes the generalized conflicts into account.

In the k-color case, robots of the same type may swap their goals. The super roadmap allows us to use existing task assignment algorithms, because there is no path between any vertices that belong to different robot types. For example, minimizing the maximum duration over all robots can be achieved by running the Threshold algorithm [12] prior to the ECBS execution.

V. Trajectory Optimization

Trajectory optimization is done using the same general process as the prior work for homogeneous teams [1]. First, collision-free corridors are generated from the discrete schedule for each robot for the entire duration of the plan. Second, independent trajectory optimizations occur for each robot within its corridor. Third, continuous refinement of the initial

trajectories is done by sampling the trajectories, recomputing the safe corridors, and re-optimizing the trajectories inside the new corridors. Finally, the trajectories are post-processed by uniformly scaling their duration in order to ensure the dynamic limits of all robots are obeyed. While the top-level process is similar, modifications to the construction of the corridors and the method for trajectory scaling are necessary to extend the process to heterogeneous teams. We start by introducing the definitions needed to describe these modifications.

A. Definitions

We assign a time $t_n = n\Delta t$ to every step in the schedule determined by the discrete solver. Each of the K steps from the discrete solution specifies a time interval $[t_{n-1}, t_n]$. The parameter Δt is user defined and specifies an initial guess for the duration of the entire trajectory $T = K\Delta t$. Let $\mathcal{F}_n^{(i,k)}$ be the set of points defined by the trajectory of a robot $r^{(i,k)}$ during timestep n:

$$\mathcal{F}_n^{(i,k)} = \left\{ f^{(i,k)}(t) \mid t_n \leq t \leq t_{n+1} \right\}. \tag{9}$$

During both the initial trajectory optimization and continuous refinement stages, collision-free *safe corridors* are computed. The safe corridor for robot $r^{(i,k)}$ is defined as a sequence of convex polyhedra $\mathcal{P}_n^{(i,k)}, n \in \{1 \dots K\}$, such that if every robot travels in their respective $\mathcal{P}_n^{(\cdot,\cdot)}$ during timestep n they are guaranteed to be collision free for that timestep. The polyhedra $\mathcal{P}_n^{(i,k)}$ for a robot $r^{(i,k)}$ are specified as the intersection of $N-1$ half-spaces that separate $r^{(i,k)}$ from all other robots and N_{obs} half-spaces separating $r^{(i,k)}$ from all obstacles.

Let $\hat{\mathcal{R}}_{\mathcal{R}}^{(l,k)}(\mathcal{F}_n^{(j,l)})$ be the set of points defined by sweeping the cylinder specified by $\mathcal{R}_{\mathcal{R}}^{(l,k)}$ along $\mathcal{F}_n^{(j,l)}$. Also let the half-space separating $r^{(i,k)}$ from $r^{(j,l)}$ be denoted as $\mathcal{H}_{\mathcal{R}n}^{(i,j)}$. $\mathcal{H}_{\mathcal{R}n}^{(i,j)}$ is defined such that

$$\mathcal{H}_{\mathcal{R}n}^{(i,j)} \cap \mathcal{F}_n^{(i,k)} = \mathcal{F}_n^{(i,k)} \tag{10}$$

$$\mathcal{H}_{\mathcal{R}n}^{(i,j)} \cap \hat{\mathcal{R}}_{\mathcal{R}}^{(l,k)}(\mathcal{F}_n^{(j,l)}) = \emptyset. \tag{11}$$

Let $\mathcal{H}_{\mathcal{E}n}^{(i,h)}$ be a half-space separating $r^{(i,k)}$ from an obstacle $\mathcal{O}_h$ at step n. $\mathcal{H}_{\mathcal{E}n}^{(i,h)}$ is defined such that

$$\mathcal{H}_{\mathcal{E}n}^{(i,h)} \cap \mathcal{F}_n^{(i,k)} = \mathcal{F}_n^{(i,k)} \tag{12}$$

$$\mathcal{H}_{\mathcal{E}n}^{(i,h)} \cap \left(\mathcal{O}_h \oplus \mathcal{R}_{\mathcal{E}}^{k}(\mathbf{0})\right) = \emptyset. \tag{13}$$

where $\oplus$ denotes the Minkowski sum.

With the above definitions $\mathcal{P}_n^{(i,k)}$ is specified as

$$\mathcal{P}_n^{(i,k)} = \left(\bigcap_{j \neq i}^{N} \mathcal{H}_{\mathcal{R}n}^{(i,j)} \right) \bigcap \left(\bigcap_{h}^{N_{obs}} \mathcal{H}_{\mathcal{E}n}^{(i,h)} \right). \tag{14}$$

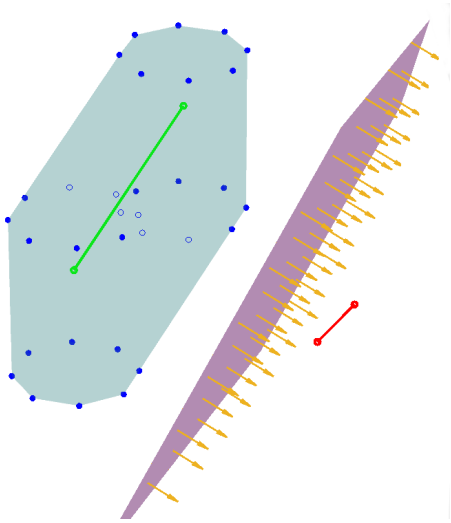


Fig. 4. An example of the the half-space computation for a robot $r^{(i,k)}$. The trajectory of $r^{(i,k)}$ is displayed in red on the right side, and another robot $r^{(j,l)}$'s trajectory is in green. The swept hull approximation for $\hat{\mathcal{R}}_{R}^{(l,k)}(\mathcal{F}_n^{(j,l)})$ is shown in blue. The separating hyper-plane is in purple with the arrows indicating the direction of the half-space

B. Safe Corridors

In the prior homogeneous planning work, the half-spaces defining the safe corridors were computed using a modified SVM in a stretched coordinate system to account for the uniform ellipsoidal collision models. In the heterogeneous case this method can no longer be used because of the non-uniform volumes that each robot occupies. Instead, we directly enumerate a vertex cloud specifying the time swept hull of the cylinders defined in Section III-A, then compute the half-spaces using a standard SVM.

Consider robots $r^{(i,k)}$, $r^{(j,l)}$, and a corresponding half-space $\mathcal{H}_{\mathcal{R}n}^{(i,j)}$. $\mathcal{H}_{\mathcal{R}n}^{(i,j)}$ is computed by first specifying two vertex sets $\mathcal{V}_n^{(i,k)}$ and $\mathcal{V}_n^{(j,l)}$ for each robot and then separating those vertex sets with a linear SVM. The set $\mathcal{V}_n^{(i,k)}$ is composed of the trajectory points sampled from $\mathcal{F}_n^{(i,k)}$, and the set $\mathcal{V}_n^{(j,l)}$ is constructed by generating a conservative approximation of the swept hull specified by $\hat{\mathcal{R}}_{R}^{(l,k)}(\mathcal{F}_n^{(j,l)})$.

To generate the swept hull vertices for $\mathcal{V}_n^{(j,l)}$ we first compute a polytope approximation of the cylinder specified by $R_R^{(l,k)}$ by computing the vertices of a circumscribed polygon with radius r and placing those vertices on both the top and bottom ends of the cylinder. $\mathcal{V}_n^{(j,l)}$ is then constructed by enumerating these cylinder approximations at points sampled from $\mathcal{F}_n^{(j,l)}$. In the first step of optimization, the points sampled from $\mathcal{F}_n^{(i,k)}$ and $\mathcal{F}_n^{(j,l)}$ are the endpoints of $\ell_n^{(i,k)}$ and $\ell_n^{(j,l)}$ from the discrete solution. During continuous refinement the samples from $\mathcal{F}_n^{(\cdot,\cdot)}$ are uniform evaluations of $f_n^{(\cdot,\cdot)}(t)$ over the corresponding interval.

Finally, $\mathcal{H}_{\mathcal{R}n}^{(i,j)}$ is computed by a linear SVM such that $\mathcal{F}_n^{(i,k)}$ lies entirely on the positive side of the separating hyper-plane and $\hat{\mathcal{R}}_{R}^{(l,k)}(\mathcal{F}_n^{(j,l)})$ lies entirely on the negative side. This processes is repeated for every robot pair at every timestep to specify the components of the corridor

TABLE I
ROBOT PROPERTIES.

Robot	Radius [m]	Height [m]	Weight [kg]	$v_{\max}$ [m/s]	$a_{\max}$ [m/s^2]
Small	0.08	0.06	0.033	1.7	6.2
Medium	0.14	0.12	0.124	2.0	8.5
Large	0.21	0.15	0.491	1.8	7.2
Ground	0.25	0.45	6.3	0.5	0.5

TABLE II
ROBOT INTERACTIONS. MINIMAL REQUIRED HORIZONTAL (r) AND VERTICAL DISTANCES (v) IN METERS DENOTED AS $\langle r, v \rangle$.

bottom \ top	Small	Medium	Large	Ground
Small	$\langle 0.2, 0.6 \rangle$	$\langle 0.3, 1.4 \rangle$	$\langle 0.35, 2.0 \rangle$	$\langle 0.33, 0.26 \rangle$
Medium	$\langle 0.3, 0.1 \rangle$	$\langle 0.3, 0.5 \rangle$	$\langle 0.4, 0.3 \rangle$	$\langle 0.39, 0.29 \rangle$
Large	$\langle 0.35, 0.2 \rangle$	$\langle 0.4, 0.2 \rangle$	N.A.	$\langle 0.46, 0.3 \rangle$
Ground	$\langle 0.33, 0.26 \rangle$	$\langle 0.39, 0.29 \rangle$	$\langle 0.46, 0.3 \rangle$	$\langle 0.5, 0.45 \rangle$

that partition the free space for every robot with respect to the rest of the team. Construction of half-spaces separating the robots from the environment follows the same procedure described in the prior work with the addition that every robot can have a separate specified size [1]. A visualization of an approximated swept cylinder hull and a separating hyperplane defining a half-space are given in Fig. 4.

C. Optimization

Trajectory optimization is identical to the prior homogeneous planning work [1], where we construct independent quadratic programs for each robot. In the following we outline this method. For each robot, we formulate a quadratic program with an optimization objective that minimizes the sum of integrated squared derivatives of the trajectories. The decision variables for the optimization are the control points of K sequential Beziér curves. We specify linear inequality constraints to bound the control points within the safe corridors and equality constraints to enforce starting and goal positions as well as continuity between each of the Beziér curves. The optimization can be repeated using the previous results as input for iterative cost improvement.

D. Dynamic Limits

Many nonholonomic mobile robots are differentially flat in position outputs, including quadrotors and differential drive robots [13], [14]. That means the control input to move the robots along a trajectory can be computed using the trajectory itself. Polynomial trajectories can be scaled in time; a longer trajectory has lower velocities and accelerations. Therefore, dynamic limits can be enforced by scaling all trajectories by a constant factor. We compute a trajectory stretching factor for each robot using a binary search approach. The maximum of all those factors is then applied uniformly to all trajectories and guarantees that the dynamic limits are fulfilled.

An example of the generated continuous trajectories is shown in Fig. 3(e); the trajectories are color-coded by time (blue means $t = 0$ and red means $t = T$). The small quadrotor in the back flies directly above the large one.

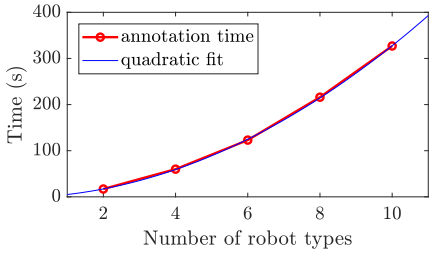


Fig. 5. Runtime of roadmap annotation for different numbers of robot types.

VI. EXPERIMENTS

For the discrete planning we implement roadmap generation and annotation in C++ using the SPARS implementation from the OMPL library [15] and an OctoMap [16] data structure. The half-space computation is implemented in Matlab using libSVM [17]. The remaining code (ECBS solver, trajectory optimization) is only slightly modified compared to our previous homogeneous planning work [1]. All simulations were executed on a PC running Ubuntu 16.04, with a Xeon E5-2630 2.2 GHz CPU and 32 GB RAM.

A. Robot Characterization

We use four different types of robots in our experiments. The *small* robots are Bitcraze Crazyflie 2.0 nano-quadrotors, an open-hardware, open-source, and commercially available platform. Crazyflie 2.0 can fly standalone or, with an extension board, be used as a flight controller. We use off-the-shelf components for frame, motor, and power distribution to build two larger kinds of quadrotors (*medium* and *large*). We extend the Crazyswarm [18], an open-source solution to use many Crazyflie 2.0 quadrotors simultaneously, by adding support for heterogeneous quadrotor teams. Each of the quadrotors uses the same extended custom firmware that runs a non-linear trajectory tracking controller, extended Kalman filter for state estimation, and trajectory evaluation on-board at 500 Hz. As *ground* robot we use the Turtlebot2 platform equipped with a single-board embedded computer running Ubuntu 16.04 and ROS Kinetic. We implement a trajectory-tracking controller that runs on-board [19] at 50 Hz. A summary of the robot properties is given in Table I.

All experiments are conducted indoors using a motion capture system to provide state estimates to the robots. All robots fuse their external state estimate at approximately 100 Hz using an on-board EKF, but otherwise only receive high-level commands such as "start trajectory execution". The execution is distributed and clocks are initially synchronized using low-latency wireless broadcasts.

We use a figure-8 trajectory that can be stretched in time to empirically determine physically safe maximum velocity and acceleration limits for each robot platform. In each case, we stop if the Euclidean position error worsens significantly. All UAVs have similar dynamic limits of $v_{\max} \approx 1.8\,\mathrm{m/s}$, reaching roll/pitch angles of around 30°. The ground robot is significantly slower with $v_{\max} = 0.5\,\mathrm{m/s}$, see Table I for details.

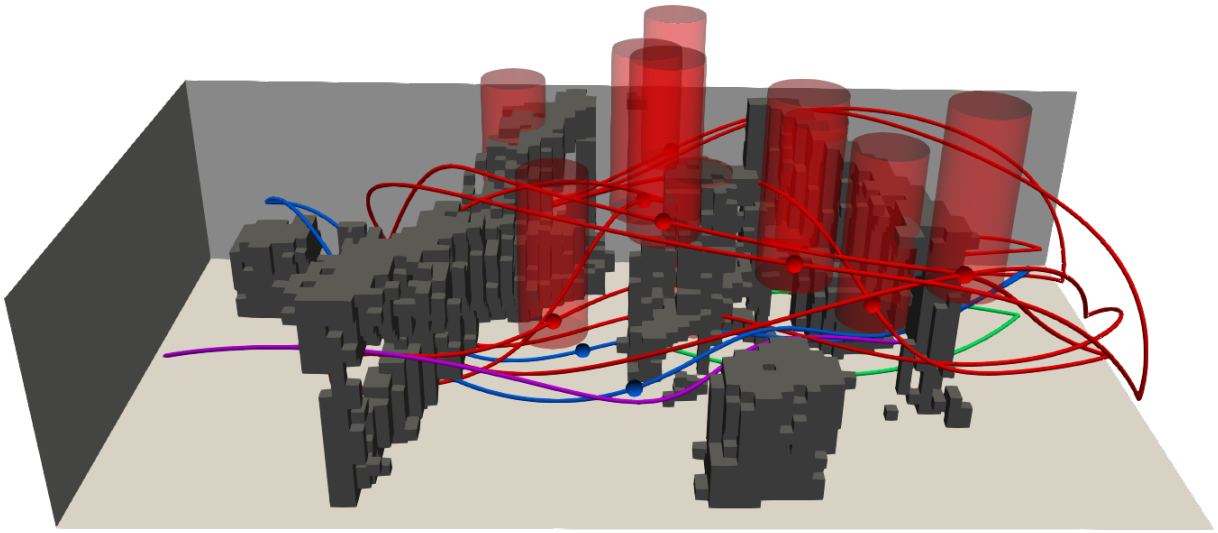


Fig. 6. Trajectories and small/medium collision-model for our physical experiment. The robot types are color-coded (red: small, blue: medium, purple: large, green: ground). The red cylinders denote $\mathcal{R}_{\mathcal{R}}^{(small,medium)}(\cdot)$ at a time during a simulated execution. There is no medium/small collision, because the blue spheres marking the position of the medium quadrotors are outside of all red cylinders.

For pairwise interaction, we find that two quadrotors hovering next to and on top of each other are worst-case scenarios and conducted experiments with all robot pairs except large/large, see Table II. For example, if the small UAV hovers on top of a medium UAV a vertical safety distance of $0.1\,\mathrm{m}$ is required, while in the opposite ordering the downwash effect necessitates a vertical safety distance of $1.4\,\mathrm{m}$. This results in a inter-robot collision model $R_{\mathcal{R}}^{(small,medium)}(\cdot) = \langle 0.3, 1.4, 0.1\rangle$ or equivalently $R_{\mathcal{R}}^{(medium,small)}(\cdot) = \langle 0.3, 0.1, 1.4\rangle$. This asymmetric effect has been noted in previous work [5], but does not occur with all UAV types. For example, the interaction between our medium and large quadrotors is nearly symmetric with $0.3\,\mathrm{m}$ and $0.4\,\mathrm{m}$ required vertical safety distances.

B. Scalability

We analyze the scalability of our approach in two experiments; first with respect to the number of robot types M and second with respect to the maximum ratio of velocity limits. We use $N = 50$ robots in a $28\,\mathrm{m} \times 12\,\mathrm{m} \times 12\,\mathrm{m}$ environment with obstacles where 25 robots start on each side and are tasked with swapping sides. We use an asymmetric collision model similar to the one we observed on real quadrotors, i.e. smaller UAVs can fly above bigger ones, but require a large vertical safety distance to fly below bigger ones (ranging from $0.02\,\mathrm{m}$ to $1.8\,\mathrm{m}$). We use up to ten different types, where the smallest one has radius $0.02\,\mathrm{m}$ and height $0.01\,\mathrm{m}$ and the biggest has radius $0.2\,\mathrm{m}$ and height $0.1\,\mathrm{m}$.

In the first experiment, all robots have the same velocity limits. We vary the number of robot types $M \in \{2, 4, 6, 8, 10\}$ and attempt to keep the difficulty of the problem similar by always using the full range of robot types, e.g., in case of $M = 2$, we use 25 robots of the smallest size and 25 robots of the largest size distributed equally. We find that the roadmap annotation step varies roughly quadratically with M (see Fig. 5), while the other steps stay mostly constant (ECBS: $2\,\mathrm{s}$, two iterations of optimization: $60\,\mathrm{s}$). Each of the roadmaps belonging to a single robot type has about 800 vertices and 2500 edges, because the desired Δ^k is constant. Thus, the super roadmap contains approximately a factor of M more vertices and edges compared to a single roadmap. The annotation step requires checking every pair of entities in the roadmap, resulting in a quadratic runtime in M. However, this computation is a preprocessing step – robots can be assigned new start/goal locations without the need to re-run the roadmap annotation. The planning time itself only depends on the number of robots and the "hardness" of the problem. Here, "hardness" refers to the number of generalized conflicts described in Section IV-B. The number of conflicts might increase for large collision cylinders, but this is difficult to quantify.

In the second experiment, we use two robot types (the largest and smallest) and reduce the maximum velocity limit of the smallest robot type using factors $\{1, 2, 3, 4\}$. Lower velocities necessitate the generation of more vertices and edges for that roadmap type, e.g., the roadmap for the case where the small robot is four times slower has 28 times more vertices and 7 times more edges compared to the large robot. As before, this results in a runtime increase during the roadmap annotation ($23\,\mathrm{s}$ to $518\,\mathrm{s}$ for the two extreme cases). The resulting plans require more discrete timesteps because of the smaller robots moving slower, resulting in longer runtimes for both ECBS ($3\,\mathrm{s}$ to $7\,\mathrm{s}$) and optimization ($43\,\mathrm{s}$ to $460\,\mathrm{s}$).

C. Physical Experiment

We set up an obstacle course in a space of $9\,\mathrm{m} \times 4.5\,\mathrm{m} \times 2\,\mathrm{m}$ where robots must swap positions using ten small, two medium, and one large quadrotors, and two ground robots. We create an OctoMap representation of the environment using an RGB-D camera that is tracked by

our VICON motion capture system. The roadmap generation step takes $30\,\mathrm{s}$ to $120\,\mathrm{s}$ per roadmap, where each roadmap has 140 to 330 vertices and 260 to 1300 edges, depending on robot size, dynamic limits, and roadmap dimensions (2D or 3D). We merge the individual roadmaps to our super roadmap and annotate it with potential vertex-vertex, vertex-edge, and edge-edge conflicts in $4\,\mathrm{s}$, resulting in 820 vertices and 2700 edges. Most conflicts are edge-edge conflicts with a mean of 77 conflicting edges per edge due to the large swept volumes created by our inter-robot collision model. Our discrete planner can compute discrete trajectories in less then $1\,\mathrm{s}$, and the four iterations of spatial partitioning and optimization takes about $15\,\mathrm{s}$. After stretching the trajectories according to our dynamic limits ($2\,\mathrm{s}$), the resulting trajectories are $16\,\mathrm{s}$ long. The trajectories and the small/medium collision model are shown in Fig. 6. The robots safely execute the generated trajectories simultaneously, see Fig. 1 for a snapshot and the supplemental video for the full execution.

VII. CONCLUSION

We present a motion planning method that can compute safe trajectories for heterogeneous robot teams. Our approach considers asymmetric inter-robot spatial constraints as well as the different dynamic limits for each robot type in the team. Prior alternative methods either do not scale well for large teams, may get stuck in non-trivial environments, or have not been shown to work in 3D. In contrast, we have shown that our method scales well for at least 50 robots in simulation and have physically demonstrated trajectories for 15 robots in a 3D obstacle-rich environment. To our knowledge, our method is the first method that efficiently solves heterogeneous trajectory planning problems for large robot teams.

In the future we plan to improve the roadmap generation for reduced runtime, reduced conflicts during annotation, and better repeatability. We also plan to investigate methods for online settings or cases where each robot has to fulfill multiple goals.

REFERENCES

[1] W. Hönig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics, Special Issue on Aerial Swarm Robotics*, 2018, to appear.

[2] A. Felner, R. Stern, S. E. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. R. Sturtevant, G. Wagner, and P. Surynek, "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges," in *Symposium on Combinatorial Search (SOCS)*, 2017, pp. 29–37.

[3] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1917–1922.

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

[5] 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)*, 2012, pp. 477–483.

[6] D. R. Robinson, R. T. Mar, K. Estabridis, and G. Hewer, "An efficient algorithm for optimal trajectory generation for heterogeneous multi-agent systems in non-convex environments," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1215–1222, 2018.

[7] J. Butzke, K. Gochev, B. Holden, E. Jung, and M. Likhachev, "Planning for a ground-air robotic system with collaborative localization," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 284–291.

[8] B. Araki, J. Strang, S. Pohorecky, C. Qiu, T. Naegeli, and D. Rus, "Multi-robot path planning for a swarm of robots that can both fly and drive," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 5575–5582.

[9] J. Alonso-Mora, T. Naegeli, R. Siegwart, and P. A. Beardsley, "Collision avoidance for aerial vehicles in multi-agent scenarios," *Autonomous Robots*, vol. 39, no. 1, pp. 101–121, 2015.

[10] A. Dobson and K. E. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 1, pp. 18–47, 2014.

[11] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Symposium on Combinatorial Search (SOCS)*, 2014, pp. 19–27.

[12] R. E. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*. Society for Industrial and Applied Mathematics, 2009.

[13] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

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

[15] I. A. Şucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012, software available at http://ompl.kavrakilab.org.

[16] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013, software available at http://octomap.github.com.

[17] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[18] J. A. Preiss*, W. Hönig*, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3299–3304, software available at https://github.com/USC-ACTLab/crazyswarm.

[19] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1990, pp. 384–389.