Downwash-Aware Trajectory Planning for Large Quadcopter Teams

James A. Preiss, Wolfgang Hönig, Nora Ayanian, and Gaurav S. Sukhatme

***Abstract*— We describe a method for formation-change trajectory planning for large quadcopter teams in obstacle-rich environments. Our method decomposes planning into a discrete graph-based stage and a continuous refinement that converts the graph plans into smooth trajectories. We model downwash directly, allowing safe flight in dense formations. We show simulation results with up to 200 robots and a real-robot experiment with 32 quadcopters. To our knowledge, our approach is the first solution which can compute safe and smooth trajectories for hundreds of quadcopters in dense environments with obstacles in a few minutes.**

I. INTRODUCTION

Trajectory planning is a fundamental problem in multi-robot systems. Consider a team of N robots in an environment defined by convex polytope $\mathcal{W}$ and containing convex obstacles $\mathcal{O}_1 \dots \mathcal{O}_{N_{obs}}$, resulting in the obstacle-free configuration space $\mathcal{F}$ for a single robot. We are given a start position for each robot $s^i \in \mathcal{F}$ and a set of goal positions $G \subset \mathcal{F}, |G| = N$. We seek the following:

- An assignment of each robot to a goal position $g^{\phi(i)} \in G$, where ϕ is a permutation of $1, \dots, N$
- The total time duration $T \in \mathbb{R}_{>0}$ until the last robot reaches its goal
- For each robot r^i, a trajectory $f^i : [0, T] \to \mathcal{F}$ where $f^i(0) = s^i, \quad f^i(T) = g^{\phi(i)}$, f^i must be continuous up to the C^{th} derivative (where C is user-specified), and collisions are avoided at all times for all robot pairs.

In particular, we are interested in solving this problem for large teams of quadcopters in tight formations. To account for the downwash force generated by one quadcopter's air stream on another, we treat each robot as an axis-aligned ellipsoid of radii $0 < r_x = r_y \ll r_z$. Due to the differential flatness of quadrotor dynamics, we focus on planning smooth trajectories in 3D Euclidean space and ignore the yaw angle.

A large body of work has addressed this problem. Graph search approaches (e.g. [1]) are adept at dealing with maze-like environments and scenarios with high congestion. However, directly executing a graph plan results in a piecewise linear path, requiring the robot to fully stop at each graph vertex to maintain dynamic feasibility. Continuous approaches [2], [3] address this issue, but they often tightly coupled, solving one large optimization problem in which the decision variables define all robots' trajectories. These approaches are typically demonstrated on smaller teams and do not scale up to the size of team in which we are interested.

All authors are with the Department of Computer Science, University of Southern California, Los Angeles, CA, USA.

Email: {japreiss, whoenig, ayanian, gaurav}@usc.edu

This work was partially supported by the ONR grants N00014-16-1-2907 and N00014-14-1-0734, and the ARL grant W911NF-14-D-0005.

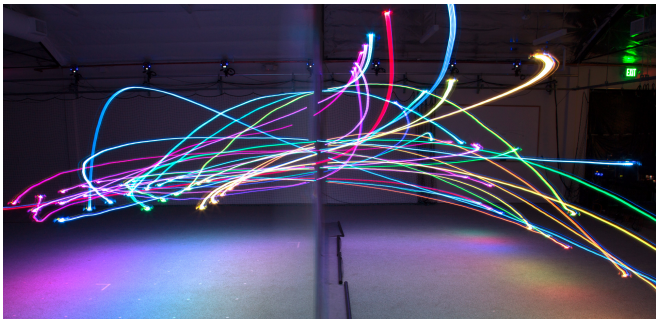

Fig. 1. Long exposure of 32 Crazyflie nano-quadrotors flying through a wall with windows, viewed from the edge of the wall.

Our approach decomposes the formation change problem into two steps. *Discrete planning* solves the goal assignment problem (generating ϕ) and computes a timed sequence of waypoints for each robot in a graph approximation of the environment. *Continuous refinement* uses the discrete plan as a starting point to compute a set of smooth trajectories, similar to [4] but adding support for three-dimensional ellipsoidal robots, environmental obstacles, and an anytime refinement stage to further improve the plan after generating an initial set of smooth trajectories.

II. DISCRETE PLANNING STAGE

We model unlabeled planning as a variant of the *unlabeled Multi-Agent Path-Finding* (MAPF) problem. We are given an undirected connected graph of the environment $\mathcal{G}_E = (\mathcal{V}_E, \mathcal{E}_E)$, where each vertex $v \in \mathcal{V}_E$ corresponds to a location in $\mathcal{F}$ and each edge $(u, v) \in \mathcal{E}_E$ denotes that there is a linear path in $\mathcal{F}$ connecting u and v. Obstacles are implicitly modeled by not including a vertex in $\mathcal{V}_E$ for each cell that contains an obstacle. At each discrete timestep, a robot can either wait at its current vertex or traverse an edge. We seek paths p^i such that the following properties hold:

P1: Each robot starts at its start vertex.
P2: Each robot ends at a unique goal location.
P3: At each timestep, each robot either stays at its current position or moves along an edge.
P4: There are no robots occupying the same location at the same time (*vertex collision*).
P5: There are no robots traversing the same edge in opposite directions (*edge collision*).
P6: Robots obey downwash constraints when stationary (*downwash vertex collision*).
P7: Robots obey downwash constraints while traversing an edge (*downwash edge collision*).

We consider a solution optimal if the makespan K is

TABLE I
RUNTIME FOR DIFFERENT EXAMPLES, SEE SECTION IV.

Example	N	N_{obs}	Grid Size	Discrete		Continuous	
				K	t_{dis}	t_1	t_{con}
USC	32	61	$13 \times 13 \times 5$	17 s	39 s	3 s	19 s
Maze50	50	441	$20 \times 13 \times 5$	26 s	60 s	9 s	57 s
Sort200	200	1320	$29 \times 29 \times 5$	19 s	541 s	36 s	239 s

minimal. If only the first five properties are considered and K is given, unlabeled MAPF can be solved in polynomial time by finding the maximum flow of a time-expanded flow-graph [1]. This maximum-flow problem can also be expressed as an Integer Linear Program (ILP), which allows us to add additional constraints for P6 and P7.

In order to find an optimal solution for an unknown K, we use a two-step approach. First, we find a lower bound for K, by ignoring the downwash constraints. We search the sequence $K = 1, 2, 4, 8, \ldots$ for a feasible K, and then perform a binary search to find the minimal feasible K. Because we ignore the downwash constraints, we can check the feasibility in polynomial time using the Edmonds-Karp algorithm on the time-expanded flow-graph. Second, we execute a linear search from the known lower bound, solving the fully constrained ILP.

III. CONTINUOUS REFINEMENT STAGE

In the continuous refinement stage, we convert the waypoint sequences p^i generated by the discrete planner into smooth trajectories f^i. We begin by finding *safe corridors* within the free space $\mathcal{F}$: The safe corridor for robot r^i is a sequence of convex polyhedra $\mathcal{P}^i_k, k \in \{1 \ldots K\}$, such that, if each r^i travels within $\mathcal{P}^i_k$ during time interval $[t_{k-1}, t_k]$, both robot-robot and robot-obstacle collision avoidance are guaranteed. The safe polyhedron $\mathcal{P}^i_k$ is the intersection of:

- $N-1$ half-spaces separating r^i from r^j for $j \neq i$
- N_{obs} half-spaces separating r^i from $\mathcal{O}_1 \ldots \mathcal{O}_{N_{obs}}$.

We compute these half-spaces by solving a modified version of the hard-margin support vector machine problem that accounts for the ellipsoid radii.

After computing safe corridors, we plan a smooth trajectory $f^i(t)$ for each robot, contained within the robot's safe corridor. We represent these trajectories as piecewise Bézier curves with one piece per time interval $[t_k, t_{k+1}]$. A degree-D Bézier curve is defined by a sequence of $D+1$ *control points* $y_i \in \mathbb{R}^3$ and a fixed set of Bernstein polynomials. The curve begins at y_0 and ends at y_D. In between, it is guaranteed to lie in the convex hull of all control points. Thus, when using Bézier control points as decision variables instead of monomial coefficients, constraining the control points to lie inside a safe polyhedron guarantees that the resulting polynomial will lie inside the polyhedron also.

We select an optimal Bézier trajectory by minimizing a weighted combination of the integrated squared derivatives:

$$\text{cost}(f^i) = \sum_{c=1}^{C} \gamma_c \int_0^T \left\| \frac{d^c}{dt^c} f^i(t) \right\|_2^2 dt$$

where the $\gamma_c \geq 0$ are user-chosen weights on the derivatives.

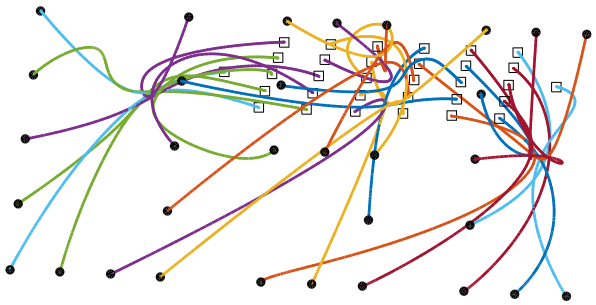

Fig. 2. Trajectory plans for "USC" formation change problem, executed on real robots. See section IV for details.

The cost can be expressed as a quadratic function of the control points. When combined with the linear corridor constraints, it forms a quadratic program. We solve one instance of this quadratic program per robot, in parallel.

We further improve these trajectories with an *iterative refinement* stage. We use the smooth trajectories to define a new corridor spatial decomposition based on sampled points along each polynomial piece, producing new safe corridors are roughly "centered" on the smooth trajectories rather than on the discrete plan. We then repeat the same optimization method to solve for a new set of smooth trajectories. Intuitively, iterative refinement provides a chance for the smooth trajectories to move further towards a local optimum that was not feasible under the original spatial decomposition.

IV. EXPERIMENTS

We evaluate our method on real robots with 32 Crazyflie nano-quadcopters and in simulation for larger, more obstacle-dense problems. A breakdown of the computation time for all problems is given in Table I.

In the real-robot task, the quadrotors begin in a grid in the $x-y$ plane, fly through a wall with three holes, and form the letters "USC" in the air. The discrete planner plans on a grid of $0.5\,\text{m}$ side length and finds a solution of $K = 17$ timesteps in 39 seconds (t_{dis}). The continuous planner needs three seconds to find the first set of smooth trajectories (t_1) and finishes six iterations of refinement after 19 seconds (t_{con}). After refinement, peak acceleration over all robots was reduced from 5.2 to 1.6 m/s^2 and peak angular velocity was reduced from 2.2 to 0.38 rad/s. The final set of 32 trajectories is shown in Fig. 2.

REFERENCES

[1] J. Yu and S. M. LaValle, "Planning optimal paths for multiple robots on graphs," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[2] 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 Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[3] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2012.

[4] S. Tang and V. Kumar, "Safe and complete trajectory generation for robot teams with higher-order dynamics," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.