

Centralized Collision-free Polynomial Trajectories and Goal Assignment for Aerial Swarms[★]

Benjamin Gravell^{a,*}, Tyler Summers^a

^aDepartment of Mechanical Engineering, The University of Texas at Dallas, Richardson, TX, 75080 USA

ARTICLE INFO

Keywords:
Motion planning
multi-robot systems
collision avoidance

ABSTRACT

Computationally tractable methods are developed for centralized goal assignment and planning of collision-free polynomial-in-time trajectories for systems of multiple aerial robots. The method first assigns robots to goals to minimize total time-in-motion based on initial trajectories. By coupling the assignment and trajectory generation, the initial motion plans tend to require only limited collision resolution. The plans are then refined by checking for potential collisions and resolving them using either start time delays or altitude assignment. Numerical experiments using both methods show significant reductions in the total time required for agents to arrive at goals with only modest additional computational effort in comparison to state-of-the-art prior work, enabling planning for thousands of agents.

1. Introduction

In the burgeoning field of autonomous robotics, aerial robots are quickly becoming a useful platform for firefighting, police, search-and-rescue, surveillance, and product delivery. In the deployment of large fleets of such robots, trajectory plans must satisfy the competing criteria of safety and performance. Specifically, the safety requirement of avoiding vehicle-to-vehicle collisions and the performance requirement of minimizing time-in-flight are considered. As the number of robots increases, so too does the complexity of satisfying these requirements, warranting the development of computationally tractable methods to this end.

A wide class of traditional single-agent motion planning methods rely on discretization of the state space and definition of a related state-transition graph [1]. Optimal feasible paths are then found through a graph search [2–7] or other combinatorial solvers [8]. While these methods can solve the multi-agent planning problem e.g. as in [9–13], they become computationally intractable quickly as the number of agents increases, leading to an exponential growth of the search space dimensionality [8, 14]. Some methods have been explored which reduce the search space dimensionality [8, 15–17], but are unable to sufficiently reduce the complexity for large numbers of agents [18]. Other centralized planning approaches based on sequential mixed-integer linear optimization [19–21], sequential convex programming [22, 23], semidefinite programming [24], formation space-based velocity planning [25], or path improvement using revolving areas [26] can work well for relatively small teams

but do not scale well to large teams due to high computational complexity.

A complementary approach for motion planning is to use local decentralized feedback control laws to satisfy collision avoidance and dynamic constraints [27–35]. These require real-time sense-and-avoid capabilities, increasing the system cost and complexity. In general, these schemes lead to difficulties maintaining optimal or near-optimal performance and avoiding undesirable deadlock situations.


It was recently shown by [18] that when the robots are interchangeable, combining the assignment and planning problems facilitates finding collision-free trajectories. Therein a concurrent assignment and trajectory planning algorithm was proposed which tractably gave collision-free trajectories for large robot teams for sufficiently spaced start and goal locations. The trajectories generated by that approach are globally optimal with respect to a total *squared distance* metric and under the assumption of synchronized robot motion, which resulted in trajectories that can be significantly suboptimal in terms of total time-in-motion and may violate minimum velocity constraints associated with certain aerial robots. Additionally, it was asserted that coupling the assignment and trajectory planning was critical to avoid overwhelming computational complexity.

To overcome these shortcomings, in this work a similar problem setup and centralized planning setting to [18] is considered, but allows non-simultaneous trajectory end times and attempts to minimize the total *time-in-motion* via a strategy with partial coupling between goal assignment and trajectory planning. The proposed algorithm designs piecewise polynomial-in-time trajectories with physical feasibility and computational tractability guarantees; physical systems track the generated trajectories without colliding with each other, while the polynomial formulation enables key calculations such as the point-of-minimum-clearance to be executed via well-studied and efficient root finding algorithms. It is demonstrated that the proposed method significantly reduces total time-in-motion relative to [18] with only modest ad-

[★] This material is based on work supported by the United States Air Force Office of Scientific Research under award number FA2386-19-1-4073. This material was also supported by the Army Research Office and was accomplished under Grant Number: W911NF-17-1-0058.

Conflict of interest - none declared.

*Corresponding author

 benjamin.gravell@utdallas.edu (B. Gravell);

tyler.summers@utdallas.edu (T. Summers)

ORCID(s): 0000-0003-3302-0166 (B. Gravell); 0000-0002-0113-8912 (T. Summers)

ditional computational expense which becomes negligible for large numbers of agents. The authors' prior work [36] presented a similar approach, but required a simple kinematic model which assumed an unrealistic ability to instantaneously change positions vertically and instantaneously change velocities horizontally; these assumptions are removed in the present work. Further, more extensive simulations and an experimental implementation on a multi-robot testbed are provided.

The remainder of the paper is organized as follows. First a variation of the trajectory planning problems given in [18, 36] is defined, then a strategy in which goal assignment is performed based on initial trajectory plans is proposed, followed by a refinement step to resolve potential collisions using either time delays or altitude assignment. Throughout, constraints on the magnitude of time derivatives of position (speed, acceleration, jerk, etc.) are honored. Finally, simulated and physical experiments on a many-member quadrotor platform are presented that illustrate the effectiveness of the algorithms.

2. Preliminaries

This work begins by establishing mathematical and notational preliminaries; where applicable, the notation of [18] is followed. Unlike the authors' previous work [36] which restricted agents to 2-dimensional planes and assumed instantaneous switching between these planes, now a fully 3-dimensional Euclidean space is considered and accompanying trajectory generation algorithms developed which support implementation on a physical robotic system. To be clear, in this work all objects and geometry with physical extent reside within a common 3-dimensional Euclidean space. The set of integers between 1 and positive integer Z is denoted by $\mathcal{I}_Z \equiv \{1, 2, \dots, Z\}$, and the $Z \times Z$ identity matrix is denoted by I_Z . The following symbols are used for certain operators and objects:

\wedge : Logical "and"	\vee : Logical "or"
\cap : Set intersection	\cup : Set union
\emptyset : Empty set	\oplus : Minkowski sum
conv : Convex hull	

Vectors in \mathbb{R}^N are column vectors unless otherwise specified. The Euclidean norm of such a vector z is denoted as $\|z\|$ and defined as $\|z\| = \sqrt{\sum_{i=0}^N z_i^2}$. Consider the scenario where n agents begin at n start locations and move towards n goal locations in a 3-dimensional Euclidean space with a fixed origin and coordinates measured in a Cartesian coordinate system. The first two coordinates of this space are referred to as "horizontal" and the third coordinate as "vertical".

The i^{th} agent position is given by $x_i \in \mathbb{R}^3$, $i \in \mathcal{I}_n$ with the horizontal and vertical parts denoted as $x_{i,12} \in \mathbb{R}^2$ and $x_{i,3} \in \mathbb{R}^1$ respectively. The first four derivatives of position with respect to time are called velocity, acceleration, jerk, and snap. These and higher-order derivatives are collectively referred to as time derivatives. Derivatives with respect to time are notated either by dots or by numbers enclosed by

parentheses above the variable, e.g. $\dot{x}_i = \overset{(1)}{x}_i$ is velocity and $\ddot{x}_i = \overset{(2)}{x}_i$ is acceleration of agent i .

The collision volume of agent i is represented by a finite cylinder C_i of radius R_i and height H_i . Each cylinder is centered at $x_i \in \mathbb{R}^3$. The cylinders represent the safe collision volume around an agent; a collision occurs if and only if two cylinders intersect. The cylinders have orientations which remain fixed with the axial direction parallel to the world vertical direction (a "vertical" cylinder). The variable height of the cylinders relative to the radius is useful for modeling phenomena such as downwash from the rotors of a quadrotor vehicle. Let the largest radius and height of all vehicles be $R = \max_i(R_i)$ and $H = \max_i(H_i)$.

The i^{th} start location and j^{th} goal location are given by $s_i \in \mathbb{R}^3$, $i \in \mathcal{I}_n$ and $g_j \in \mathbb{R}^3$, $j \in \mathcal{I}_n$ respectively. The horizontal ground plane is the 2-dimensional set $\{x | x_3 = 0\}$. The agents operate in a region \mathcal{K} :

$$\mathcal{K} \equiv \text{conv}(\{s_i | i \in \mathcal{I}_n\} \cup \{g_j | j \in \mathcal{I}_n\}) \oplus C_\infty. \quad (1)$$

where C_∞ is a vertical cylinder with radius R whose horizontal coordinates of the center are on the origin and with vertical extent from the ground to positive infinity. An altitude $\mathcal{A}(\eta)$ at height η is the subset of \mathcal{K} within which any agent at that height is contained i.e. a 3-dimensional horizontal slab volume:

$$\mathcal{A}(\eta) = \mathcal{K} \cap \{x : |x_3 - \eta| \leq H/2\}$$

Define the $n \times 3$ goal matrix as

$$G = [g_1 \quad g_2 \quad \dots \quad g_n]^T.$$

Define the $n \times n$ boolean assignment matrix ϕ , which assigns agents to goals, as

$$\phi_{ij} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to goal } j \\ 0 & \text{otherwise} \end{cases}$$

Therefore row i of ϕG , denoted as $(\phi G)_i$, gives the goal location assigned to agent i . All agents are assigned to goals in a one-to-one mapping so

$$\phi^T \phi = I_n. \quad (2)$$

A polynomial $p(t) : \mathbb{R} \rightarrow \mathbb{R}$ of scalar t with degree d , with $d + 1$ coefficients α_i , is given by

$$p(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \dots + \alpha_d t^d = \sum_{i=0}^d \alpha_i t^i. \quad (3)$$

Minimizing a polynomial over a finite domain interval $[t_0, t_f]$ is a straightforward and computationally efficient procedure which follows from Fermat's theorem for stationary points from differential calculus. A description is provided in Algo. 5 for completeness since it will be called at various points throughout this work. Maximization is accomplished by the same algorithm by passing a negated polynomial. Also,

both minima and maxima can be found concurrently at little additional computational expense by a simple modification to the algorithm.

Similarly, finding the intervals of a finite domain interval $[t_0, t_f]$ over which a polynomial evaluates within a range of values i.e. finding the set

$$\{t \mid p(t) \in [p_-, p_+] \text{ and } t \in [t_0, t_f]\}$$

is also straightforward and efficient. The procedure begins by finding domain intervals where the polynomial evaluates above the lower limit p_- and below the upper limit p_+ (Alg. 6), then intersects those domain intervals with each other and the prescribed interval $[t_0, t_f]$ (Alg. 7). Note that Alg. 6 works identically for finding intervals where a polynomial is below an upper value p_+ by simply reversing inequalities.

The trajectory planning problem and the proposed solution techniques are now introduced.

3. Trajectory Planning Problem

The trajectory planning problem requires finding n instances of 3-dimensional trajectories which guide n agents from start to goal locations. The trajectories are given agent-wise by

$$\gamma_i(t) : [t_{0,i}, t_{f,i}] \rightarrow x_i, \quad i \in \mathcal{I}_n$$

and must satisfy the initial and terminal conditions

$$\gamma_i(t_{0,i}) = s_i, \quad i \in \mathcal{I}_n, \quad (4)$$

$$\gamma_i(t_{f,i}) = (\phi G)_i, \quad i \in \mathcal{I}_n. \quad (5)$$

Agents are considered to be quadrotors, whose center position dynamics linearized about the hover configuration are modeled as a quadruple integrator in horizontal directions (due to the rolling action which must precede lateral acceleration) and a double integrator in the vertical direction [37]:

$$\overset{(4)}{x}_{i,12}(t) = u_{\text{horz},i}, \quad \overset{(2)}{x}_{i,3}(t) = u_{\text{vert},i} \quad (6)$$

where $u_{\text{horz},i}$ and $u_{\text{vert},i}$ are control inputs. The dynamics are not used explicitly in terms of designing control inputs, but rather are used to motivate the choice of trajectory form, namely piecewise polynomials of a particular order.

By choosing a whole number q sufficiently high and imposing constraints on the norm of $q - 1$ time derivatives, actuator constraints are honored. The particular choice for q in the case of quadrotors is established in Section 3.1.1. These constraints are encoded in a vector $\delta \in \mathbb{R}^{q-1}$ with $\delta_k > 0$ and applied as

$$\|\gamma_i^{(k)}(t)\| \leq \delta_k \text{ for } k \in \mathcal{I}_{q-1}. \quad (7)$$

Define the global start and end times for which motion may occur over all agents:

$$t_{0,\text{all}} = \min(t_{0,1}, t_{0,2}, \dots, t_{0,n}),$$

$$t_{f,\text{all}} = \max(t_{f,1}, t_{f,2}, \dots, t_{f,n}).$$

Ensure collision avoidance by requiring the collision volumes of all agent pairs to be disjoint during the period of possible motion:

$$\{x_i(t) \oplus C_i\} \cap \{x_j(t) \oplus C_j\} = \emptyset$$

$$\text{for } t : [t_{0,\text{all}}, t_{f,\text{all}}], \quad i \neq j \in \mathcal{I}_n. \quad (8)$$

Like the previous work in [36], the proposed method aims to minimize the total, or equivalently average, time-in-flight of all agents. This is a useful cost metric for many applications e.g. product delivery and emergency response. Therefore, the optimization problem seeks trajectories $\gamma^*(t) = [\gamma_1(t), \dots, \gamma_n(t)]$ and goal assignment ϕ^* that minimize total time-in-motion:

$$\gamma^*(t), \phi^* = \underset{\gamma(t), \phi}{\text{argmin}} \sum_{i=1}^n \int_{t_{0,i}}^{t_{f,i}} dt \quad (9)$$

subject to (2), (4), (5), (6), (7), (8)

Assumptions: The following assumptions are explicitly imposed as part of the problem formulation:

- (A1) Any assignment of agents to goals is permissible.
- (A2) The collision volume of each agent is the set of points contained in cylinder C_i .
- (A3) The effect of any dynamics model mis-specification, imperfect state knowledge, actuation error, and external disturbance are small enough such that the true physical extent of each agent is always fully contained inside the collision volume C_i .
- (A4) Continuity and satisfaction of upper bound constraints on $q - 1$ time derivatives of position is sufficient to ensure actuator constraints are honored.
- (A5) The region \mathcal{K} in (1) is devoid of any obstacles other than the agents themselves.
- (A6) The region \mathcal{K} in (1) has infinite positive vertical extent.
- (A7) All start and goal locations are fixed on a common ground plane and are spaced at least $2R$ apart:

$$s_{i,3} = g_{i,3} = 0 \quad \forall i \in \mathcal{I}_n$$

$$\|s_i - s_j\| > 2R \quad \forall i \neq j \in \mathcal{I}_n$$

$$\|g_i - g_j\| > 2R \quad \forall i \neq j \in \mathcal{I}_n$$

The modeling assumption of no uncontrolled obstacles in the operating space is not altogether unreasonable when considering the nearly empty airspace encountered at altitudes above tree tops, buildings, etc. in typical real-world flight scenarios. The use of cylindrical collision volumes renders orientation of each quadrotor irrelevant for the purpose of trajectory planning.

The solution to the global problem in (9) is ultimately not obtained exactly, but rather a suboptimal solution is found using (9) to guide generation of trajectories and goal assignment

by the approach proposed in the following subsections. The strategy for finding an approximate solution to this problem proceeds by temporarily ignoring the clearance requirements (8) which effectively reduces the domain of trajectories under consideration to the ground plane, choosing a function form for trajectories (piecewise polynomials) to reduce the problem to goal assignment, generating horizontal trajectories, then constructing vertical trajectories using refinement techniques which detect and resolve collisions. As a result, these trajectories will be shown to be feasible (e.g. collision-free) and computable after a finite number of operations by construction.

As the trajectory generation procedure based on piecewise polynomial functions is used throughout the goal assignment and collision resolution phases, the trajectory generation scheme is described next.

3.1. Trajectory Generation

The trajectory design is motivated by the observation that minimum-time trajectories along a long straight line with maximum speed constraints will naturally partition into three segments; acceleration, constant (max) speed, and deceleration. A similar idea has previously been suggested for point-to-point robot trajectory planning under the name "Linear Segments with Parabolic Blends" [38]. This idea is generalized to higher-order acceleration (blend) segments. During the acceleration segments, one or more time derivatives of order 2 and higher will be pushed to a constraint maximum, and during the constant max speed segment the higher order time derivatives will be zero. Although physical models involving friction (i.e. higher fidelity models than that assumed in (6)) theoretically allow only asymptotic approach of the maximum speed under actuation constraints e.g. the exponential approach of the speed of a particle in gravitational free-fall to a terminal speed, in practice it was found that the polynomial trajectories were sufficient for reference tracking.

This work does not attempt to optimize control effort during the acceleration segments since the control effort expended during the constant speed segment dominates e.g. due to air friction and by virtue of the relative duration of this segment over long horizontal paths. If deemed necessary, techniques such as minimum-snap trajectory design via quadratic programming [37] could be utilized to further decrease the control effort, possibly at the expense of trajectory duration and computational burden. Any techniques which return polynomial-in-time trajectory segments are fully compatible with the remainder of the proposed method.

This work also restricts trajectories to strictly piecewise vertical and horizontal straight-line paths, which permits simplified trajectory planning and collision resolution by treating trajectories as single-dimensional polynomials of time multiplied by a constant unit heading vector. A pair of tuples $\delta_{\text{horz},k}$ and $\delta_{\text{vert},k}$ are used and (7) is used with δ_k set to either $\delta_{\text{horz},k}$ or $\delta_{\text{vert},k}$ depending on whether $\gamma_i^{(k)}(t)$ is horizontal or vertical at t .

The acceleration segments are individualized polynomi-

als scaled from a base polynomial. The base polynomial is calculated only once at the beginning of the overall routine. Particular whole trajectories are generated by joining acceleration and constant speed segments. Generation of the base polynomial and individualized polynomials are described in the subsequent two subsections.

3.1.1. Base polynomial

Recalling the definition of a polynomial of degree d in (3) and the whole number q which represents the number of time derivatives on which constraints will be enforced, let $2q = d + 1$. It is evident that a given $2q$ -tuple of initial and terminal time derivative conditions ($2q$ total point constraints) uniquely specifies a polynomial of degree d so long as the problem is well-posed i.e. if a certain coefficient matrix A is invertible. To ensure continuity of position and $q - 1$ time derivatives at the endpoints, specify q constraints at $t = 0$ and q constraints at $t = T$. Due to the assumption on the dynamics in (6), by choosing reference trajectories which are piecewise polynomial with degree at least 4 and 2 respectively, open-loop control with sufficient control effort and the absence of disturbances would give perfect tracking. It is also desirable to make the segment transitions smooth to avoid discontinuous control signals. Choosing degree 9 would allow the specification of 5 endpoint time derivative constraints: position, speed, acceleration, jerk, and snap. However, to reduce the computational storage requirement for the trajectories during implementation on actual hardware and reduce computational effort during centralized trajectory planning, a degree of 7 is used. It was found that the difference between the degree 7 and 9 polynomials was extremely slight and in practice the reference tracking error was dominated by other noise sources. For comparison, degree 1 polynomials represent constant speed trajectories; this was effectively the approach taken in the authors' previous work [36]. The procedure for calculating the base polynomial is as follows:

1. Form the vector of endpoint conditions

$$b = [p(0), \dot{p}(0), \dots, \overset{(q)}{p}(0), \\ p(T), \dot{p}(T), \dots, \overset{(q)}{p}(T)]^T$$

2. Form the matrix of coefficients $A \in \mathbb{R}^{2q \times 2q}$ as

$$A_{ij} = \begin{cases} i! & \text{if } i = j \text{ and } i \leq q \\ 0 & \text{if } i \neq j \text{ and } i \leq q \\ \frac{(j-1)!}{(j-(i-q))!} T^k & \text{if } i - q \leq j \text{ and } i > q \\ 0 & \text{if } i - q > j \text{ and } i > q \end{cases}$$

where $k = (j - 1) + (i - q - 1)$. This follows from simple differentiation of polynomials and matching coefficients according to the endpoint constraints. As

an example, for $d = 7$ and $T = 1$ one has

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 0 & 2 & 6 & 12 & 20 & 30 & 42 \\ 0 & 0 & 0 & 6 & 24 & 60 & 120 & 210 \end{bmatrix}$$

3. Solve the system of linear equations $A\alpha = b$ to obtain the vector of polynomial coefficients α .

In this framework, other polynomial bases such as the orthogonal polynomials of Chebyshev or Legendre could be used to improve the conditioning of the A matrix [39] i.e. to encourage the singular values of the A matrix to remain clustered around unity and ensure numerical stability of the solution to $A\alpha = b$; for ever-higher degree polynomials the conditioning of the matrix in the monomial basis degrades. However, for simplicity, monomials are used since the error was found to be manageable on the problem instances encountered, i.e. for degree 7 polynomials. If position and the first $q - 1$ time derivatives are 0 at $t = 0$, the first q coefficients $\alpha_0, \dots, \alpha_q$ are also zero, which is evident from the partial diagonal structure of A . Indeed, it is desirable to create an acceleration polynomial which has $p(0) = 0, \dot{p}(0) = 0, p(T) > 0, \dot{p}(T) > 0$ and some higher-order time derivatives zero at both endpoints i.e.

$$b = [0, 0, \dots, 0, x_f, v_f, 0, \dots, 0]^T$$

Although this procedure will always generate a polynomial which satisfies the endpoint constraints, the behavior between the endpoints is governed by the duration T . In particular, there is a unique setting of T which ensures that both the position and velocity monotonically increase from the initial to terminal points, thus ensuring that the endpoints are where the minimum and maximum position and speed occur over the segment. This setting is

$$T = 2 \left| \frac{p(T) - p(0)}{\dot{p}(T) - \dot{p}(0)} \right| = 2 \left| \frac{p(T)}{\dot{p}(T)} \right|. \quad (10)$$

With this choice, as an additional benefit, the polynomial degree is reduced by 1 i.e. $\alpha_d = 0$. Although proving these facts for arbitrary degree polynomials is difficult, it is now shown that at least for $d = 7$, which is the case of interest in this work, that the given setting of T in (10) gives the desired behavior. Assuming, without loss of generality, that $p(0) = 0, \dot{p}(0) = 0, p(T) = 0.5, \dot{p}(T) = 1$, and by (10) set $T = 1$. Solving for the coefficients of the position polynomial obtain

$$p(t) = t^6 - 3t^5 + 2.5t^4$$

and differentiating, the acceleration is

$$\begin{aligned} \ddot{p}(t) &= 30t^4 - 60t^3 + 30t^2 \\ &= 30t^2(t - 1)^2 \end{aligned}$$

which is nonnegative for all t and thus on the interval $[0, T]$. Thus the velocity monotonically increases from 0 and so does the position, as desired. Attempting to show this for any other setting of T will fail; a proof of this fact is left to future work, noting that a product-of-squares argument (as here) is insufficient to prove a setting of T gives an acceleration which is somewhere negative.

It is emphasized that the base polynomial only needs to be calculated once at the beginning of the overall routine and can be scaled and translated (in time) as necessary for each particular trajectory. The base polynomials for vertical and horizontal trajectories are calculated separately to account for differing actuation constraints in each direction. In each case, a unit path length and terminal speed equal to the max speed ($p(0) = 0, \dot{p}(0) = 0, p(T) = 1, \dot{p}(T) = \delta_1, T = 2$) are used. This results in the polynomials $p_{\text{base,horz}}$ and $p_{\text{base,vert}}$.

3.1.2. Individualized polynomials

Once the base polynomials for an acceleration segment have been found, a piecewise polynomial (sub)trajectory may be generated which connects any two points x_0, x_f with a straight line path, subject to the time derivative constraints. Let the distance between x_0, x_f be $\ell_\Delta = \|x_f - x_0\|$. The whole piecewise polynomial trajectory for agent i with n_i pieces has the form

$$\gamma_i(t) = \left\{ p_{ik}(t) \hat{h}_{ik} \text{ if } t \in [t_{ik}, t_{ik+1}], k \in \mathcal{I}_{n_i-1} \right\}$$

where $\hat{h}_{ik} \in \mathbb{R}^3$ is a unit heading vector. In this work, this heading will either be horizontal $\hat{h}_{ik} = [a, b, 0]^T$ or vertical $\hat{h}_{ik} = [0, 0, 1]^T$ where a, b are dummy constants satisfying $a^2 + b^2 = 1$. Also, in this work these trajectories are comprised of 2- or 3-segment subtrajectories and 1-segment stationary wait segments. For notational compactness, let

$$\gamma_{ik} = \{p_{ik}, \hat{h}_{ik}, [t_{ik}, t_{ik+1}]\}$$

represent a polynomial trajectory segment which encodes a polynomial, a heading, and a time interval.

Accordingly, the norm of the time derivatives has the simplified form

$$\left\| \dot{\gamma}_i^{(k)}(t) \right\| = \left\{ \left| \dot{p}_{ik}^{(k)}(t) \right| \text{ if } t \in [t_{ik}, t_{ik+1}], k \in \mathcal{I}_{n_i-1} \right.$$

It will be useful to keep in mind the spatial and temporal scaling formulas for polynomials:

$$\begin{aligned} c p(t) &= c \sum_{i=0}^d \alpha_i t^i \\ p(ct) &= \sum_{i=0}^d \alpha_i (ct)^i = \sum_{i=0}^d \alpha_i c^i t^i \\ \frac{1}{c} p(ct) &= c^{-1} \sum_{i=0}^d \alpha_i (ct)^i = \sum_{i=0}^d \alpha_i c^{i-1} t^i \end{aligned}$$

from which it follows that the derivatives satisfy:

$$\frac{d^k}{dt^k} (c p(t)) = c \frac{d}{dt} p(t) \quad (11)$$

$$\frac{d^k}{dt^k}(p(ct)) = c^k \frac{d^k}{d\tau^k}(p(\tau)) \quad (12)$$

$$\frac{d^k}{dt^k} \left(\frac{1}{c} p(ct) \right) = c^{k-1} \frac{d^k}{d\tau^k}(p(\tau)) \quad (13)$$

where $\tau = ct$.

First, temporal scaling is applied to the acceleration segment in order to ensure the terminal speed is the agent max speed so that

$$|\dot{p}(t)| \leq \delta_1 \text{ for } t \in [0, T]$$

with equality ensured exactly at $t = T$. The (absolute) maximum time derivative $\max_t(|\dot{p}(t)|)$ of the base polynomial is computed via Algorithm 5 with the interval $[0, T]$. The scale factor is found as $c = \frac{\max_t(|\dot{p}(t)|)}{\delta_1}$ then temporal scaling is applied as

$$p(t) \leftarrow p(ct) \quad \text{and} \quad T \leftarrow T/c$$

which achieves the proper scaling of speed per (12) and preserves the path length traversed. Next, scaling is applied to the acceleration segment in order to satisfy constraints on the higher time derivatives which are denoted by $\delta \in \mathbb{R}^{q-1}$ so that

$$|p^{(k)}(t)| \leq \delta_k \text{ for } t \in [0, T], k \in \mathcal{I}_{q-1}$$

with equality ensured in at least one derivative at one time. This minimizes the time taken to traverse the path by taking full advantage of the available time derivatives. The (absolute) maximum time derivatives $\max_t(|p^{(k)}(t)|)$ of the base polynomial are computed via repeated applications of Algorithm 5 with the interval $[0, T]$. Once the (absolute) maximum time derivatives have been identified, scale factors ψ_k associated with satisfying each time derivative constraint are found by

$$\psi_k = \left(\frac{\max_t(|p^{(k)}(t)|)}{\delta_k} \right)^{\frac{1}{k-1}} \text{ for } k \in \mathcal{I}_{q-1}. \quad (14)$$

The maximum of these scale factors is the only one that is needed to ensure all constraints are satisfied, so take $\psi_* = \max_k(\psi_k)$. The scaling is then applied by

$$p(t) \leftarrow \psi_* p(t/\psi_*) \quad \text{and} \quad T \leftarrow T\psi_*$$

which compresses the trajectory temporally and stretches it spatially in equal proportions such that the terminal speed remains the same, per (13), while honoring all higher order time derivative constraints, per (12).

Next, a determination of whether a middle constant speed segment is needed is made. This is accomplished by comparing the path length needed by the acceleration segment to reach max speed and (half) the actual path length between the physical endpoints i.e. if $2p(T) < \ell_\Delta$ then a constant speed segment is needed. This segment is trivial to calculate; it is simply a constant maximum speed segment whose duration

is simply $T_{cs} = \frac{\ell_\Delta - 2p(T)}{\delta_1}$. On the other hand, if $2p(T) \geq \ell_\Delta$ then no constant speed segment is needed and the acceleration segments must be scaled again to reduce their path length, in which case the maximum speed will not be attained.

The process continues with a spatial stretch in order to fit the path length exactly:

$$p(t) \leftarrow \frac{\ell_\Delta}{2p(T)} p(t)$$

This has the effect of strictly decreasing the time derivatives per (11) since the scale factor is less than 1. Then new scale factors are calculated similarly to (14) and a temporal stretch is applied to further optimize the trajectory by making full use of the available ‘‘capacity’’ of higher order time derivatives.:

$$\psi'_k = \left(\frac{\max_t(|p^{(k)}(t)|)}{\delta_k} \right)^{\frac{1}{k}} \text{ for } k \in \mathcal{I}_{q-1}, \psi'_* = \max_k(\psi'_k)$$

$$p(t) \leftarrow p(t/\psi'_*), \quad \text{and} \quad T \leftarrow T\psi'_*$$

The end result of this entire procedure is a piecewise polynomial (sub)trajectory with 2 or 3 pieces with the first q time derivatives continuous and satisfies all initial, terminal, and range constraints. See Fig. 1 for an illustrative example.

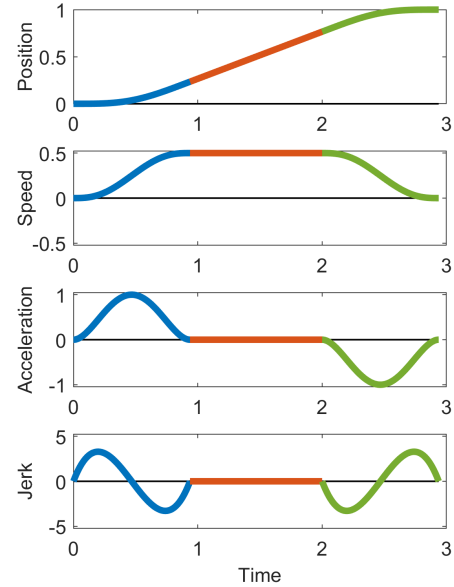


Figure 1: Plots of position and its time derivatives for an example piecewise polynomial subtrajectory generated by the proposed method. This example uses degree $d = 7$ to accommodate $q - 1 = 3$ time derivative constraints depicted as the upper and lower limits of the vertical axis and tightly bound the speed and at least one higher derivative (in this case acceleration). This example has 3 segments: acceleration, constant speed, and deceleration.

3.2. Goal Assignment

Having described the piecewise polynomial trajectory generation procedure, it is now possible to reduce the problem in (9) to one of a linear assignment (combinatorial) goal assignment problem by fixing the functional form of the trajectories. As in [36], if the collision avoidance constraint is ignored (8), an argument from the calculus of variations shows that trajectories which minimize the integral of dt , which is the time-in-motion, follow straight line paths and achieve the highest average speed possible while satisfying the boundary conditions and position derivative constraints. Thus the problem reduces to simply connecting each start to each goal with minimum-time trajectories on straight line paths, computing the time-in-motion incurred by each trajectory, and finding the goal assignment which minimizes total time-in-motion. If these minimum-time trajectories are replaced with constant velocities, as in [36], the problem amounts to minimizing the total *non-squared* distance. Unlike [36], motivated by Section 3.1, we now replace these minimum-time trajectories with piecewise polynomial trajectories, where the expression of the cost in terms of distance is more complicated and is driven by the size of the constraints on the time derivatives (which determine the base polynomial) relative to the distances.

Therefore, the optimal assignment is given by

$$\phi^* = \operatorname{argmin}_{\phi} \sum_{i=1}^n \sum_{j=1}^n \phi_{ij} C_{ij}$$

where the cost matrix C encodes the cost of assigning agent i to goal j . In accordance with (9), C contains the values of the time-in-motion taken by agent i to travel to goal j along a straight line. These times T_{ij} are found by calculating polynomial segment trajectories for agent i moving from start s_i to goal g_j by the procedure described earlier:

$$C_{ij} = T_{ij}, \quad i \in \mathcal{I}_n, j \in \mathcal{I}_n.$$

Due to the exceptionally simple form of the piecewise polynomial trajectories, calculating the $n(n-1)/2$ trajectories for each start-goal pair remains computationally tractable compared with the simplified case of constant velocity trajectories. This problem may be efficiently solved to optimality with a finite number of iterations using the well-known Hungarian algorithm [40, 41], which runs in $\mathcal{O}(n^3)$ time. Alternate algorithms such as the auction algorithm could also be used with the same time complexity, but with the benefit of parallelization [42, 43]. After solving the optimal assignment, the presumptive horizontal trajectories for each agent are simply chosen as those from the cost matrix generation which are selected by the optimal assignment.

A comparison with the C-CAPT algorithm of [18], which uses a cost function of the distance traveled squared, is given in [36]. The main disadvantages of the C-CAPT algorithm are that the speed of agents is limited due to the requirement of agents to start and arrive at goals at the same time, as well as a minimum separation spacing between starts and between goals of $2\sqrt{2}R$. The advantage of allowing asynchronous

goal arrival is highly dependent on the distribution of the start and goal locations; when some trajectory lengths are much larger than others, the ability to arrive earlier than other agents significantly improves utilization of the available actuation resources, e.g. speed. For many practical applications the service area includes goal locations which are both near and far from the start locations, which necessitates some agents to travel much longer than others, regardless of the goal assignment, so the advantage is substantial. The results of Section 6 demonstrate this advantage quantitatively, despite the minor degradation in flight times due to collision detection and resolution, which are discussed next.

4. Collision detection

Here the advantage of piecewise polynomial trajectories on straight line paths becomes apparent as the global minimum distance between any pair of agents across their entire trajectories becomes extremely easy and fast to compute. Additionally, the cylindrical collision volume representation synergizes with the restriction that paths are only vertical or horizontal and makes collision checking especially convenient and computationally efficient. Collisions at an instant of time are detected exactly by simply checking if both the radial separation is less than the sum of the radii and the vertical separation is less than the sum of the half-heights. Mathematically, the following equivalent conditions of collision between agents i and j hold:

$$C_i \cap C_j \neq \emptyset \leftrightarrow \begin{cases} \|x_{j,12} - x_{i,12}\| \leq R_i + R_j \\ \wedge |x_{j,3} - x_{i,3}| \leq \frac{H_i + H_j}{2}. \end{cases}$$

For a pair of points moving on straight-line paths whose positions are polynomials in time, the procedure in Alg. 1 is used to find the minimum separation distance.

Algorithm 1: Separation minimization

Input: Heading unit vectors \hat{h}_i and \hat{h}_j , polynomial trajectories $x_i(t) = p_i(t)\hat{h}_i$ and $x_j(t) = p_j(t)\hat{h}_j$ of degree d over a common time interval $\mathcal{T}_{ij} = [t_0, t_f]$.

Calculate relative position polynomial

$$x_{ij}(t) = p_j(t)\hat{h}_j - p_i(t)\hat{h}_i.$$

Calculate squared separation distance polynomial of degree $2d+1$ as $p_{ij}(t) = x_{ij}(t)^\top x_{ij}(t)$ whose coefficients are computed from multiplication and addition of the appropriate coefficients of $x_{ij}(t)$.

Minimize the squared separation distance using

Algorithm 5 with inputs $p_{ij}(t)$ and $[t_0, t_f]$.

Output: Minimum separation distance

$$d^* = \sqrt{\min_{t \in [t_0, t_f]} x_{ij}(t)^\top x_{ij}(t)}.$$

Consequently, Alg. 8 is used to check for a collision between a pair of agents for a single pair of polynomial segment

trajectories. The algorithm uses a default return of “false” and terminates immediately whenever “true” is returned; this “short-circuiting” dramatically improves computational speed. First, it is checked whether the intersection of the time intervals for the segments is nonempty; otherwise the segments are never active at the same time and no collision could occur. Then it is determined whether both agents are moving vertically, both horizontally, or one of each. Based on this, it is checked if (15) or (16) are satisfied, and if so Alg. 1 is used to obtain the relevant minimum separation distance and that distance is used in (15) or (16) to determine the presence of a collision.

Now segment pair collision detection is used to detect collisions between all pairs of full composite trajectories of the polynomial segment type described earlier using Alg. 2. The paths start and end at the start and goal locations on the ground and reside entirely within the planar region with infinite vertical extent passing through the line segment joining the start and goal i.e. $p_i(t) \in \{x | x_{12} \in \ell_i\} \forall t$. A “short-circuit” of the full polynomial segment collision check is then accomplished by first doing a computationally cheap check which helps quickly guarantee safety of many trajectory segment pairs. If the minimum distance between the two line segments of the trajectory pair is greater than the sum of agent radii, then a collision is impossible since there is no configuration of the agent centers within the assumed planar regions which gives an intersection since (15) is impossible to satisfy. If the minimum distance between the two line segments of the trajectory pair is not greater than the sum of agent radii, then collision detection using Alg. 8 is run. Using this fast preliminary check is critical to obtaining usable performance since, in all but the most highly congested scenarios, this check catches a large portion of segment pairs which are far apart spatially. The segment collision check is repeated for all polynomial trajectory segment pairs (for a single pair of agents). As soon as a collision is detected on a single pair of trajectory segments, the pair of agents is flagged as having a collision and the check progresses to the next pair of agents without finishing checking all remaining segments of the current pair of agents (another “short-circuit”). This process is repeated for each pair of agents, resulting in a symmetric boolean matrix of collision flags which can be represented by an upper triangular matrix or flattened vector to reduce the storage space by half. With exact collision results for the entire group of agents and trajectories in hand, the proposed methodology advances on to resolving the detected collisions.

5. Collision resolution

The overall trajectory generation proceeds by using the general collision detection scheme described in the previous section to determine which agents collide assuming they are all in the same altitude. After single-altitude collisions are detected, they are resolved by inserting vertical trajectories and time delays and/or additional altitudes.

Algorithm 2: All agents collision check

Input: Collection of n trajectories $\gamma_k(t)$ for $k = 1, \dots, n$.

foreach Pair of agents i, j **do**

 Calculate the minimum distance δ_{ij}^* between the two line segments joining the starts and goals e.g. via [44].

if $\delta_{ij}^* > R_i + R_j$ **then**

 | $B_{ij} \leftarrow \text{False}$

else

foreach Pair of segments γ_{im} in $\gamma_i(t)$ and γ_{jn} in $\gamma_j(t)$ **do**

 | $B_{ij} \leftarrow$ result of Alg. 8 with inputs γ_{im} and γ_{jn} .

 | **if** $B_{ij} = \text{True}$ **then**

 | **break**

Output: Boolean matrix $F \in \mathbb{S}^{n \times n}$ of collision flags.

5.1. Collision resolution via time delay

One way to resolve collisions is to send all agents first to a high holding altitude $\mathcal{A}_{\text{hold}}$, then after some delay times have agents descend vertically down, then move horizontally in a single traversal altitude $\mathcal{A}_{\text{trav}}$, then finally descend to the ground altitude \mathcal{A}_{gnd} at the goal location. In this scheme, a maximum of three altitudes are needed with a total height of $2.5H$ above the ground plane. By construction, given sufficient delay time on each agent that eventually all agents can complete their trajectories without colliding, since in the worst case an agent can simply wait in the holding altitude until all other agents have completed their trajectories and landed. See Fig. 2 for an illustration of this idea in the case when two identical agents must exchange positions. Although such a troublesome goal assignment would never be chosen by the goal assignment procedure in Sec. 3.2 since the reversal of the assignment gives a lower cost, it is conceptually useful simply to illustrate the ability of time delay to resolve collisions. The image shows a side-view with dashed lines representing paths and the table shows a sequence of positions that the agents pass through at generalized times t along the linear paths.

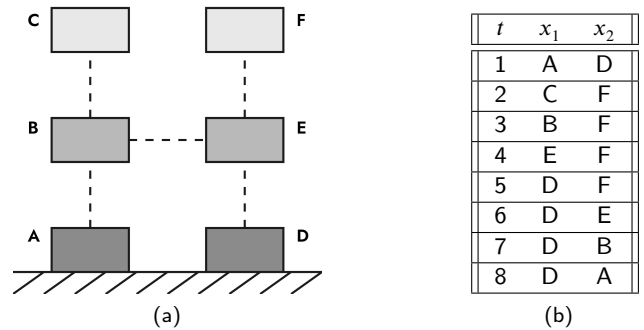


Figure 2: (a) Diagram and (b) table for a case which motivates the ability of time delay to resolve collisions.

The reason to have agents wait in a high holding altitude rather than on the ground is simply that the start and goal locations of two agents may be within a colliding distance of each other. If the somewhat weak restriction is imposed that

$$\|s_i - g_j\| \geq 2R \forall i \neq j \in \mathcal{I}_n$$

then the possibility of landing on top of another agent waiting on the ground is avoided and the holding altitude is unnecessary and agents can wait on the ground i.e. set $\eta_{\text{hold}} = 0$ so that the holding and ground altitudes coincide. In either case, the proposed method works the same way.

Choosing the height of the traversal and holding altitudes as $\eta_{\text{trav}} = \eta_{\text{gnd}} + H = H$ and $\eta_{\text{hold}} = \eta_{\text{trav}} + H = 2H$ ensures agents in different altitudes cannot collide. The traversal and holding altitudes are thus $\mathcal{A}_{\text{trav}} = \mathcal{A}(\eta_{\text{trav}})$ and $\mathcal{A}_{\text{hold}} = \mathcal{A}(\eta_{\text{hold}})$.

Each full trajectory is made up of 4 or 5 subtrajectories which are generated according to the procedure in Sec. 3.1 which have 2 or 3 polynomial segments each:

1. Vertical ascent from $\mathcal{A}_{\text{gnd}} \rightarrow \mathcal{A}_{\text{hold}}$:

$$s_i \rightarrow s_i + [0, 0, \eta_{\text{hold}}]^T \quad (17)$$

2. Stationary wait in $\mathcal{A}_{\text{hold}}$ for time τ_i :

$$s_i + [0, 0, \eta_{\text{hold}}]^T \rightarrow s_i + [0, 0, \eta_{\text{hold}}]^T \quad (18)$$

3. Vertical descent from $\mathcal{A}_{\text{hold}} \rightarrow \mathcal{A}_{\text{trav}}$:

$$s_i + [0, 0, \eta_{\text{hold}}]^T \rightarrow s_i + [0, 0, \eta_{\text{trav}}]^T \quad (19)$$

4. Horizontal movement within $\mathcal{A}_{\text{trav}}$:

$$s_i + [0, 0, \eta_{\text{trav}}]^T \rightarrow g_i + [0, 0, \eta_{\text{trav}}]^T \quad (20)$$

5. Vertical descent from $\mathcal{A}_{\text{trav}} \rightarrow \mathcal{A}_{\text{gnd}}$:

$$g_i + [0, 0, \eta_{\text{trav}}]^T \rightarrow g_i \quad (21)$$

In the case that agents wait on the ground, the subtrajectory in step 1 can be skipped and the agents ascend rather than descend in step 3.

The trajectory generation problem is now reduced to finding the set of time delays τ_i whose sum is minimum and also resolve all collisions while adhering to the trajectory generation framework described earlier:

$$\underset{\tau}{\text{minimize}} \sum_{i=1}^n \tau_i \quad (22)$$

$$\text{subject to (8), } \gamma(t) \leftarrow [(17), (18), (19), (20), (21)]$$

where (8) is the boolean collision avoidance constraint whose value is determined by the collision detection scheme in Sec. 4. This problem is nonconvex due to the collision avoidance constraint and has continuous decision variables, so a discretization scheme is used as an effective heuristic. The

Algorithm 3: Collision resolution via time delays

Input: Collection of n trajectories $\gamma_k(t)$ for $k = 1, \dots, n$.

for $i = 1, \dots, n$ **do**

Initialize $\tau_i = 0$

while Any $F \leftarrow \text{Alg. 2}(\gamma_k(t))$ for $k = 1, \dots, i$ **do**

$\tau_i \leftarrow \tau_i + \tau_\Delta$

Apply time delay τ_i to trajectory $\gamma_i(t)$

Output: Collection of n collision-free trajectories $\gamma_k(t)$ with included time delays τ_k for $k = 1, \dots, n$

heuristic begins by ordering the agents randomly, then for each agent the associated delay time is increased by an increment τ_Δ until collisions with all agents whose time delays have been fixed are resolved. This is repeated until each agent's delay time has been established. This procedure is expressed in Alg. 3.

Although this heuristic is not assured to find the global minimum of the problem in (22), the solutions found are empirically good and importantly are guaranteed to be found after a finite number of computations. To see this, consider the case depicted in Fig. 2 where each agent proceeds one-by-one; the first agent descends from the holding altitude and completes its full trajectory while all remaining agents remain in the holding altitude, then the second agent does the same and so forth until all agents have landed.

Let T_j for $j = 1, \dots, n$ be the times taken by each agent to execute trajectory segments 3, 4, 5 in (19), (20), (21), with T_j ordered from greatest to least. Then an upper bound on the number of increments of time delay increase for any other agent is $n_{\text{inc,max}} \leq \text{ceil}(T_1/\tau_\Delta)$ since for any greater time delay a collision is not possible, as explained earlier in the discussion of Fig. 2. Applying this argument iteratively shows that an upper bound on the number of increments for τ_i is $n_{\text{inc},i} \leq i \times n_{\text{inc,max}}$, and thus an upper bound on the total number of increments is

$$n_{\text{inc,tot}} = \sum_{i=1}^n n_{\text{inc},i} \leq \frac{n(n+1)}{2} n_{\text{inc,max}},$$

which is clearly $\mathcal{O}(n^2)$. In practice, many fewer increments are required than this conservative upper bound. The ordering of agents could likely be further improved i.e. according to some metric such as shortest time in horizontal flight, but it was found that random ordering gave good results.

5.2. Collision resolution via altitude assignment

Another way to resolve collisions is by finding an assignment to a set of altitudes and sending agents on trajectories that move horizontally only in these altitudes. The altitudes are given sufficient vertical separation to ensure clearance between agents in different altitudes regardless of horizontal position. Additional wait time and holding altitudes are introduced to resolve potential secondary collisions induced by the primary collision resolution.

There are m traversal altitudes $\mathcal{A}_{\text{trav},i}$ for $i \in \mathcal{I}_m$ and h holding altitudes $\mathcal{A}_{\text{hold},i}$ which are inserted between traversal altitudes and indexed to match the traversal altitudes, although in general $h \leq m - 1$. In this scheme, a maximum of n traversal altitudes and n holding altitudes are needed in addition to the ground altitude.

Define the $n \times m$ boolean altitude assignment matrix B , which assigns agents to altitudes, as

$$B_{ij} = \begin{cases} 1 & \text{if agent } i \text{ is assigned to altitude } j \\ 0 & \text{otherwise} \end{cases}$$

Therefore in row i of B , denoted as B_i , the index where $B_{ij} = 1$ gives the altitude assigned to agent i . Alternatively, in column j of B the indices where $B_{ij} = 1$ give the agents assigned to altitude j . All agents are assigned to altitudes in a one-to-many mapping, so

$$B^\top B = D_m$$

where D_m is an $m \times m$ diagonal matrix whose entry D_{ii} is the integer number of agents assigned to altitude i .

Each full trajectory is made up of 4 or 6 subtrajectories which are generated according to the procedure in Sec. 3.1 which have 2 or 3 polynomial segments each:

1. Vertical ascent from $\mathcal{A}_{\text{gnd}} \rightarrow \mathcal{A}_{\text{trav},i}$:

$$s_i \rightarrow s_i + [0, 0, \eta_{\text{trav},i}]^\top \quad (23)$$

2. Stationary wait in $\mathcal{A}_{\text{trav},i}$ until global time t_1 :

$$s_i + [0, 0, \eta_{\text{trav},i}]^\top \rightarrow s_i + [0, 0, \eta_{\text{trav},i}]^\top \quad (24)$$

3. Horizontal movement within $\mathcal{A}_{\text{trav},i}$:

$$s_i + [0, 0, \eta_{\text{trav},i}]^\top \rightarrow g_i + [0, 0, \eta_{\text{trav},i}]^\top \quad (25)$$

4. Vertical descent from $\mathcal{A}_{\text{trav},i} \rightarrow \mathcal{A}_{\text{hold},i}$:

$$g_i + [0, 0, \eta_{\text{trav},i}]^\top \rightarrow s_i + [0, 0, \eta_{\text{hold},i}]^\top \quad (26)$$

5. Stationary wait in $\mathcal{A}_{\text{hold},i}$ for time τ_i :

$$s_i + [0, 0, \eta_{\text{hold},i}]^\top \rightarrow s_i + [0, 0, \eta_{\text{hold},i}]^\top \quad (27)$$

6. Vertical descent from $\mathcal{A}_{\text{hold},i} \rightarrow \mathcal{A}_{\text{gnd}}$:

$$g_i + [0, 0, \eta_{\text{hold},i}]^\top \rightarrow g_i \quad (28)$$

where the final 3 subtrajectories may be collapsed to a single vertical descent from $\mathcal{A}_{\text{trav},i} \rightarrow \mathcal{A}_{\text{gnd}}$.

Similarly to the collision resolution via time delays, the trajectory generation problem is now reduced to finding the altitude assignment and set of time delays τ_i which minimize the sum of flight times and also resolve all collisions while adhering to the trajectory generation framework described earlier:

$$\begin{aligned} & \underset{B, \tau}{\text{minimize}} \quad \sum_{i=1}^n \tau_i \\ & \text{subject to } (8), \gamma(t) \leftarrow [(23), (24), (25), (26), (27), (28)] \end{aligned}$$

5.2.1. Primary collisions

Primary collisions are those resulting from two agents moving horizontally in a shared altitude. These are resolved by altitude assignment. As a heuristic for minimizing the sum of flight times, one might seek to minimize the number of altitudes required so that time spent in vertical motion is minimized. However even finding the optimal altitude assignment which minimizes the number of altitudes is a hard nonconvex combinatorial problem, so a similar procedure as in collision resolution via time delays is used to find the altitude assignment B . Agents are prioritized randomly, then each agent is assigned the lowest altitude possible that resolves primary collisions with all previously assigned agents. If no such altitude exists, a new one is created at a height above the previous highest altitude by a vertical spacing of H . This is repeated for all agents. By construction, such an assignment guarantees that there will be no collisions during the horizontal movements. Alg. 4 documents this procedure using mathematical notation.

Algorithm 4: Altitude assignment

Input: Boolean collision flag matrix $F \in \mathbb{S}^{n \times n}$.

Initialize $m = 1$

for $i = 1, \dots, n$ **do**

for $j = 1, \dots, m$ **do**

if not any $F_{i,k}$ for $k \mid B_{k,j} == \text{True}$ **then**

$B(i, j) = \text{True}$

else if $j == m$ **then**

$m \leftarrow m + 1$

$B(i, m) = \text{True}$

else

Continue

Output: Boolean altitude assignment matrix

$B \in \mathbb{R}^{n \times m}$ that resolves primary collisions.

5.2.2. Secondary collisions

Although altitude assignment resolves primary collisions, the possibility remains of secondary collisions during the vertical descent movements down towards the goals on the ground. These are easily detected by the same collision detection scheme in Sec. 4. Secondary collisions are exhaustively partitioned into two types of collision: exit and entrance collisions. In practice, it was found that these secondary collisions were exceedingly rare, but nevertheless must be prevented.

Exit collisions In an exit collision, a descending agent is struck by another agent moving horizontally in the same traversal altitude. To resolve this, a simple enlargement of the collision radius is used. The longest time T_{exit} that any agent could take to exit its altitude is calculated; this is easily accomplished by generating a trajectory which descends vertically downwards by H (the spacing between two altitudes). This captures the effect of all position derivative constraints imposed on the agents. This is also conservative since some agents may not have to come to a full stop at the altitude below; some agents will continue descending and accelerating

which would reduce the time taken to exit the altitude, but this is ignored for simplicity. Next, the greatest distance L_{exit} that the fastest agent would traverse horizontally moving at maximum speed over the time T_{exit} is calculated. Then the collision radii of all agents are increased by $L_{\text{exit}}/2$. Thus by using the same collision detection scheme in Sec. 4 it is ensured that agents maintain an additional horizontal clearance of L_{exit} at all times, which by construction ensures that exit collisions are impossible.

Unfortunately this procedure requires the collision radii to be increased by an amount proportional to the maximum speed of the agents, but for agents with high maximum acceleration relative to the maximum speed, such as quadrotors, the detriment is not too severe. The enlargement of the collision radii is performed as the first step of the overall collision resolution, prior to finding the altitude assignment to resolve primary collisions.

Entrance collisions In an entrance collision, a descending agent enters a lower traversal altitude at the same time as another agent is moving horizontally underneath. To resolve an entrance collision, a holding altitude is placed between the descending agent’s traversal altitude and the next lowest traversal altitude (if one does not already exist). This gives the descending agent a place to wait while the other agent moves out of the way. Once the holding altitude has been inserted, new trajectories are generated and the entire check must begin again from the point where the altitude assignment was made. In particular, the offending descending agent is made to come to a full stop and wait in its (newly inserted) holding altitude. If an entrance collision still exists with this agent, delay time is added according to the same scheme as in Section 5.1. Again, by construction, given sufficient delay time all the possible collisions with agents at lower heights will be resolved since those agents can all land. Agents in the lowest traversal altitude will clearly not encounter this type of secondary collision, and so can complete their trajectories without collision. Arguing inductively, since the lowest agents have collision-free trajectories, and entrance collisions can be resolved for agents in each successively higher altitude, all entrance collisions can be resolved. Since there are a finite number of altitudes, it also follows that the time delays required are also finite. The roles of each agent in an entrance collision are distinguished by the collision detection algorithm simply by noting the heading vector of each agent. As a final remark, in the worst case n traversal altitudes and n holding altitudes are needed, and thus by construction the computations terminate in finite time.

To conclude the algorithmic development, Figure 3 gives a broad description of all the steps involved in the method and their relationships. Evaluation of the proposed schemes is presented next, both in computer simulations and in deployment on a physical testbed.

6. Simulation results

Both the computer simulations and physical experiments were based on the CrazySwarm, a hardware and software plat-

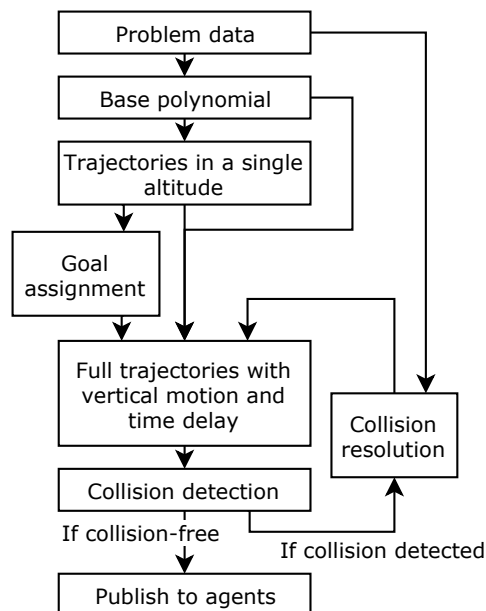


Figure 3: Diagram of information flow for trajectory generation.

Table 1
Kinematic constraints

Time derivative	Upper limit	
	Horizontal	Vertical
Speed (m/s)	0.2	0.2
Acceleration (m/s^2)	0.5	0.5
Jerk (m/s^3)	10	10

form that serves as a research testbed for quadrotor autonomy, which is described in more detail in Section 7. Throughout the simulations, the vehicle parameters used for trajectory planning were chosen to match the actual CrazySwarm platform used in the physical experiments. The Crazyflie quadrotor vehicles had a nominal outer diameter of 14 cm and height of 4 cm, while the diameter and height of the cylinders used for trajectory planning were enlarged to 30cm and 40cm respectively; see Section 7 for the rationale of this enlargement. The kinematic constraints imposed during trajectory generation for all simulations are listed in Table 6. These correspond to conservative values computed by scaling down the most aggressive values the physical CrazySwarm platform could experience without significant tracking error.

For the time delay increase rule, used by both algorithms discussed in Sections 5.1 and 5.2, an addition rule with an increment of $\tau_{\Delta} = 0.1s$ was used, which was found empirically to strike a nice balance between computation time and quality of solutions.

In order to analyze the performance of the proposed algorithms, Monte Carlo trials were performed with start and goal locations generated randomly with uniform probability over a square of side length S . In all trials all agents were identical so that collision volume dimensions were $R_i = R$, $H_i = H$ and position derivatives were $\delta_i = \delta$ for all $i \in \mathcal{I}_n$.

The number of agents n and the area density η were varied, where η is defined as the ratio of the summed area of all agents' projection onto the ground to the area on the ground that any projection could occupy:

$$\eta = \frac{A_{\text{agents}}}{A_{\text{space}}} = \frac{n\pi R^2}{S^2 + 4RS + \pi R^2}.$$

To ensure initial and terminal configurations were noncolliding, a minimum start-start and goal-goal separation distance of $2R$ was imposed. This led to an upper bound on the density, which occurs when the start locations are hexagonally close packed; proofs of this fact date back to Lagrange in 1773 with the first universally accepted proof delivered by Toth in 1942 [45]. For a separation of $2R$ the upper limit of density is $\frac{\pi}{2\sqrt{3}} \approx 0.9068$ and for a separation of $2\sqrt{2}R$ as in [18] the limit is $\frac{\pi}{4\sqrt{3}} \approx 0.4534$. For reference, a typical area density encountered in commercial aircraft traffic management is on the order of 10^{-5} [46]. For applications involving many unmanned aerial robots the traffic is considerably more dense, so simulations were performed over a wide range of densities.

6.1. Time delay distribution

Fig. 4 shows a histogram of time delays using the proposed time delay collision resolution method for $n = 1000$ agents and a high density of $\eta = 10^{-1/2} \approx 0.31$ for a single random Monte Carlo problem instance described in Section 6. This shows that, even when start and goal locations are very dense, most agents have zero or low-valued time delays.

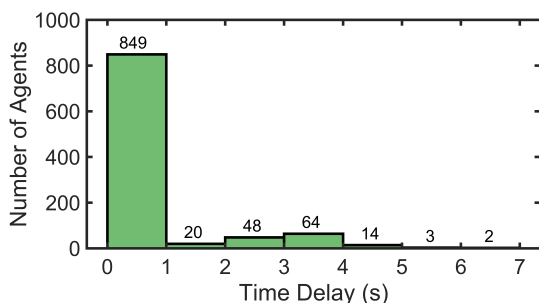


Figure 4: Histogram of time delays for 1000 agents with $\eta = 10^{-1/2} \approx 0.31$.

6.2. Number of altitudes required

Using altitude assignment, the number of altitudes required to resolve collisions was studied. Fig. 5 shows the number of flight altitudes (altitudes other than the ground) as a function of area density. As expected, the number of altitudes required grew as the density increased as a result of more potential collisions, but only a few altitudes were required even for highly dense scenarios.

6.3. Normalized flight times

To analyze the relative degradation in flight times due to avoiding collisions (altitude changes and time delays), the

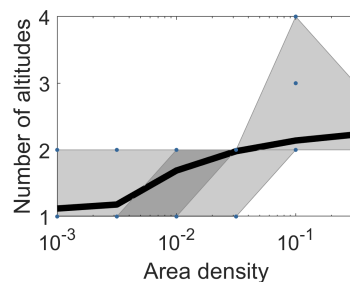


Figure 5: Number of flight altitudes required to resolve collisions as a function of area density. The minimum possible number of altitudes was 1. The number of agents was held constant at $n = 100$ and 100 trials were run at each density. Individual data for each agent in each trial are plotted as points. The mean, interquartile range (25th to 75th percentile), and full range (0th to 100th percentile) are plotted as a bold black line, dark shaded region, and light shaded region. The vertical axis is linear scaled and the horizontal axis is log scaled with base 10.

flight time data are normalized by dividing by the average time spent in horizontal motion for each trial. The time spent in horizontal motion can be viewed as unavoidable, since this is the minimum time which must be spent to reach the goals even if collisions were ignored.

6.3.1. Effect of collision resolution

From Fig. 6, for this class of random scenarios, it is evident that as the density increases, the total time taken increases as a larger portion of the time is spent moving vertically and waiting. From the zoomed portion, it is evident that the average induced degradation is manageable, being virtually negligible at low agent densities and peaking at around 60% worse than the lower bound at an agent density of $\eta = 10^{-1/2} \approx 0.316$ which represents a highly congested scenario as can be seen in Figure 8.

From Fig. 7 similar trends are observed as in Fig. 6 using the time delay collision resolution method, but with less time spent waiting, more time spent in vertical motion, and less time spent in total with the average induced degradation again virtually negligible at low agent densities and peaking at around 20% worse than the lower bound at a density of $\eta = 10^{-1/2} \approx 0.316$.

In Figures 8 and 9 example trajectories generated by the proposed algorithm are shown. Figure 8 gives a visualization of the scale of the area density, while Figure 9 demonstrates the ability of the proposed algorithm to plan trajectories for a large number of vehicles navigating between arbitrary locations.

6.3.2. Performance relative to alternate methods

For the purpose of comparing the proposed method to alternate methods e.g. that of [18], define the characteristic time t_c which is the time an agent would take to traverse the longest horizontal straight-line path within the space, which for a square space has length $\sqrt{2}S$. For a given trajectory plan, denote the time spent by agent i in horizontal motion

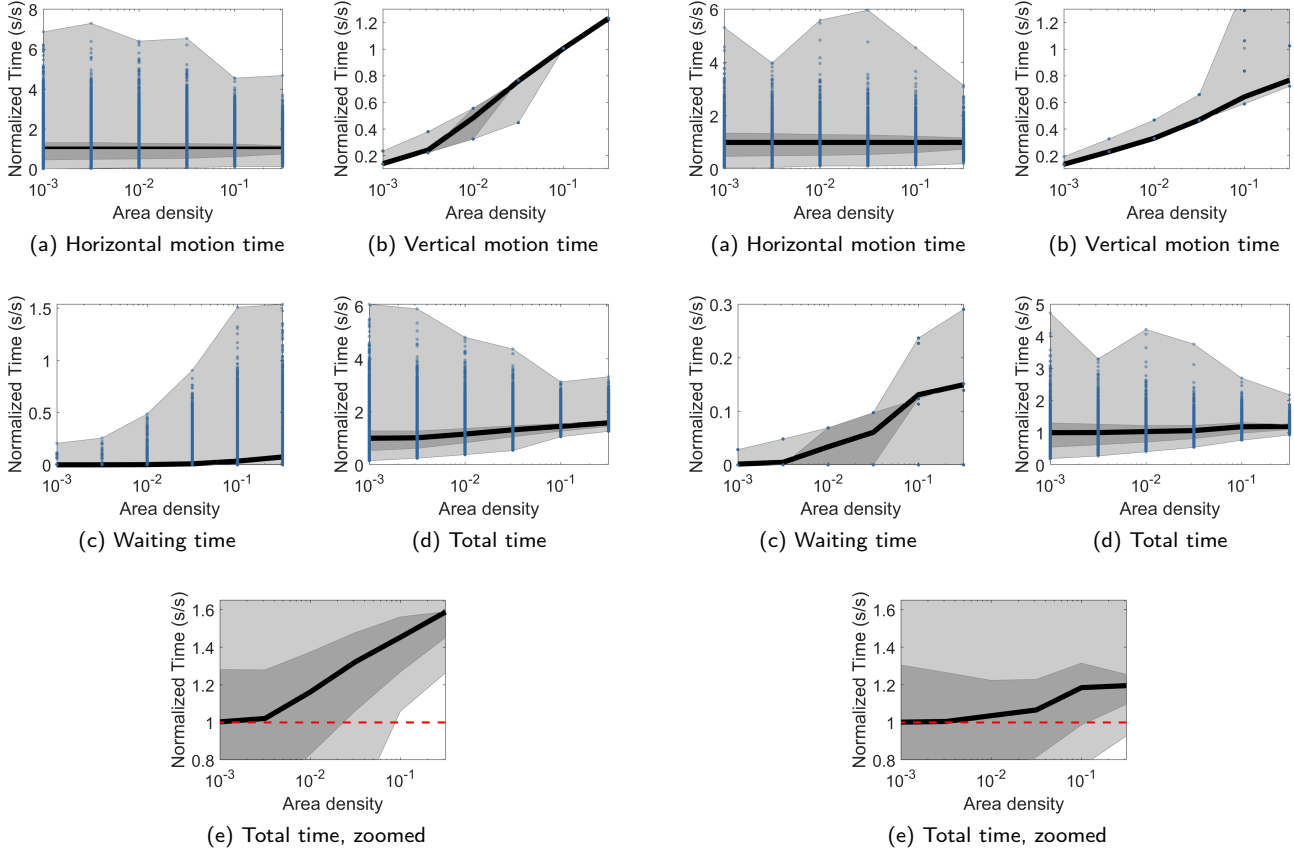


Figure 6: Time spent in (a) horizontal motion only, (b) vertical motion only, (c) waiting only, and (d) total, using collision resolution via time delay. A y-axis zoomed view of (d) is given in (e). The number of agents was held constant at $n = 100$ and 100 trials were run at each density. Individual data for each agent in each trial are plotted as points. The mean, interquartile range (25th to 75th percentile), and full range (0th to 100th percentile) are plotted as a bold black line, dark shaded region, and light shaded region. The vertical axis is linear scaled and the horizontal axis is log scaled with base 10.

and in waiting respectively as $t_{h,i}$ and $t_{w,i}$. Also define the characteristic normalized time in horizontal motion and in waiting t_p as

$$\tilde{t}_p = \frac{\frac{1}{n} \sum_{i=1}^n (t_{h,i} + t_{w,i})}{t_c}.$$

For simplicity, collisions were allowed for simulations using the approach of [18] to avoid imposing the $2\sqrt{2}R$ separation condition required for that approach to possess collision-free guarantees; this was conservative in the sense that the relative performance of the proposed methods relative to [18] was only degraded by this assumption. Also, for simplicity simple 1-degree (constant speed) polynomials were used for trajectory generation.

With respect to the t_p metric, plotted in Fig. 10, the proposed altitudes approach gave the best results for all densities. At low densities, the proposed time delay approach gave

Figure 7: Time spent in (a) horizontal motion only, (b) vertical motion only, (c) waiting only, and (d) total, using collision resolution via altitude assignment. A y-axis zoomed view of (d) is given in (e). The number of agents was held constant at $n = 100$ and 100 trials were run at each density. Individual data for each agent in each trial are plotted as points. The mean, interquartile range (25th to 75th percentile), and full range (0th to 100th percentile) are plotted as a bold black line, dark shaded region, and light shaded region. The vertical axis is linear scaled and the horizontal axis is log scaled with base 10.

nearly the same performance as the altitudes approach as a consequence of small time delays which vanish as the density goes to zero. At higher agent densities, the time delay approach result began to increase as the physical extent of the agents became more influential. Both proposed approaches performed better at all densities than the approach of [18].

6.4. Computation Time

Achieving low computation time is an important practical consideration for successful deployment of large robot teams. The simulations were implemented in MATLAB running on a desktop with an AMD Ryzen 7 2700X eight-core processor running at 3.7GHz. The results are given in Figures 11 and 12, where reasonable computation times for large teams are observed. Explicitly optimizing the code for performance or parallelization could decrease the computation times even further. The overall computation is split into three major

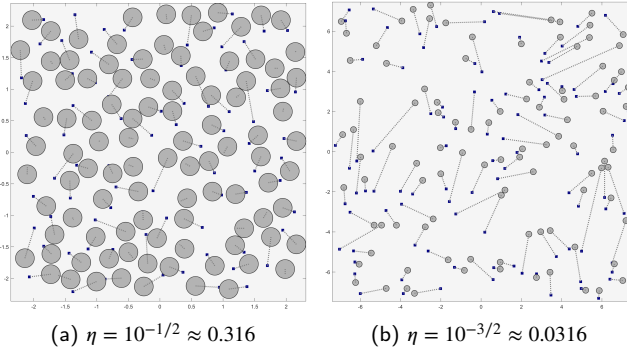


Figure 8: Example trajectories using collision resolution via time delays with 100 agents at two different area densities (η).

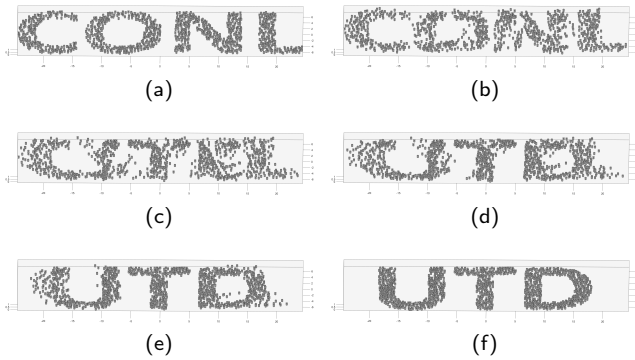


Figure 9: Example of 1000 agents spelling out the letters of the authors' lab and university by following collision-free trajectories generated by the proposed algorithm using collision resolution via time delays.

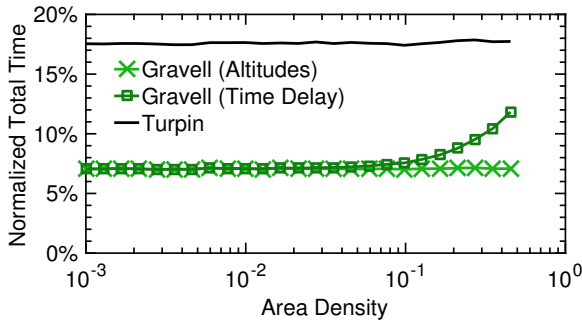
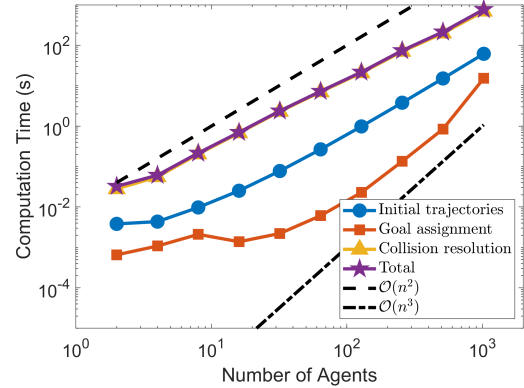


Figure 10: Normalized average flight times using the proposed collision resolution methods vs the approach of [18]. The mean value from 1000 trials over all $n = 100$ agents is plotted.

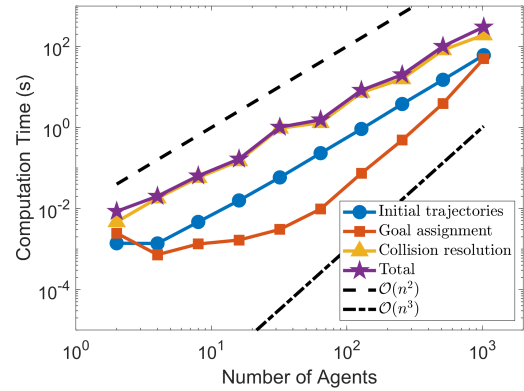
segments: the generation of initial trajectories which occurs when finding the cost matrix for input into the goal assignment, the Hungarian algorithm which actually does the goal assignment, and the combined collision detection and resolution steps. This encompasses nearly all of the computations, with the exception of the base polynomial generation and some post-processing steps which together take negligible time to execute.

The goal assignment computation time grew as $\mathcal{O}(n^3)$ as

expected from a standard computational complexity analysis [41]. The trajectory generation and collision resolution steps grew only as $\mathcal{O}(n^2)$ since the average number of pairwise trajectories and collisions grew with the number of pairs of agents. At ever higher agent numbers it is inevitable that the goal assignment will begin to dominate.



(a) High density $\eta = 10^{-1/2} \approx 0.316$



(b) Low density $\eta = 10^{-3/2} \approx 0.0316$

Figure 11: Computational time as a function of number of agents using various trajectories using the time delay collision resolution method. The area density was held constant at (a) $\eta = 10^{-1/2} \approx 0.316$ and (b) $\eta = 10^{-3/2} \approx 0.0316$. The number of agents varied from 2 to 1024 at each power of 2. The mean value from 10 trials at each number of agents is shown.

7. Experimental results

A series of experiments on physical hardware was performed to validate the performance and safety of the proposed approach.

7.1. System description

A proprietary branch of the CrazySwarm system was used, which encompasses both hardware and software [47]. State estimation was accomplished by taking position measurements of infrared (IR) markers on each quadrotor with an external Vicon camera system. The point cloud of these individual position measurements were then resolved into

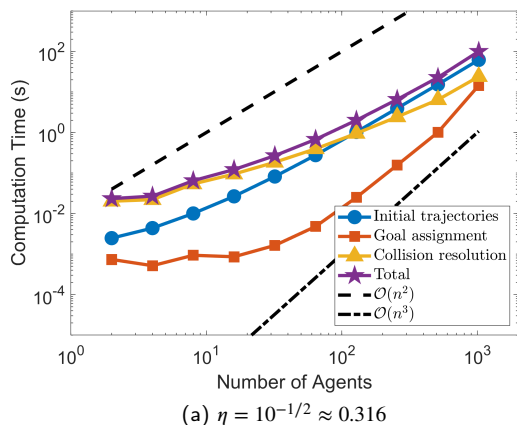


Figure 12: Computational time as a function of number of agents using various trajectories using the altitudes collision resolution method. The area density was held constant at $\eta = 10^{-1/2} \approx 0.316$. The number of agents varied from 2 to 1024 at each power of 2. The mean value from 10 trials at each number of agents is shown.

body coordinate frame (state) estimates using the object tracker portion of the CrazySwarm software. These state estimates were then used by the CrazySwarm software to generate feedback control signals, which were then broadcast over wireless radios to the flying vehicles and electrically converted to motor voltages, completing the feedback loop. The controller used was the standard “Mellinger” controller implemented by the CrazySwarm package, which is a modified version of the nonlinear reference-tracking controller proposed by [37] which takes advantage of differential flatness of the quadrotor. See the documentation at <https://github.com/TSummersLab/crazyswarm> for further details of the software and hardware setup.

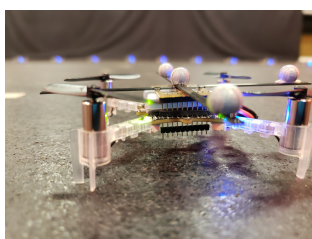


Figure 13: Photograph of a single Crazyflie quadrotor vehicle.

7.2. Trajectory tracking errors

One immediate practical issue was reference trajectory tracking in the presence of noise; although the planned trajectories could be followed perfectly in the absence of disturbances, the presence of disturbances precludes this possibility. Natural sources of disturbances included ambient air currents from air conditioner vents, downwash from other agents, ground aerodynamic effects, other unmodeled dynamics, and sensor (camera) noise. By simple enlargement of the collision volumes and ensuring bounds on the position

error of all agents, collision avoidance remained guaranteed. Feedback control within the CrazySwarm package ensured the position deviation of each agent from the desired position remained small at all times. In particular, it was found from the experiments that during all trajectory traversals that the radial position error was bounded by 8 cm and the vertical position error was bounded by 7cm; the plot in 17 demonstrates satisfaction of these bounds. Additionally, it was found that the effects of downwash were accounted for by extending the bottom of the collision cylinder by an additional 20cm. Thus, by choosing a collision cylinder with dimensions enlarged by these amounts relative to the physical dimensions of the quadrotor i.e. diameter of $14 + 2 \times 8 = 30$ cm and height of $4 + 2 \times 7 + 20 = 40$ cm, the vehicle was guaranteed to always be strictly contained within the collision volume, maintaining collision avoidance guarantees. This can be observed from Figure 16; the trajectory generation tightly respected the collision constraints, as the minimum clearance approached zero without becoming negative. Likewise, during the physical experiment the agents did not experience any collisions, as evidenced by the strictly positive clearance. This was true for all experiments.

7.3. Experiment description and findings

One experiment (“X20”) is now presented with $n = 20$ agents moving in a 2m by 3m room from start locations randomly selected from a grid with 0.5m spacing to goals arranged in an “X” configuration roughly 2.8m across the widest section; see Figures 14 and 15. In this experiment the time delay collision resolution method was used. This was for the practical reason that the height of the room limited the number of usable altitudes; in outdoor environments the height of the flyable space would be much greater. The kinematic constraints imposed during trajectory generation were the same as for the simulation results i.e. those listed in Table 6.

The results in Figures 14, 15, 16, 17 demonstrate that the proposed method reliably generated trajectories that could be successfully tracked by a physical quadrotor team and executed in a reasonable time frame with guaranteed absence of collisions.

Videos demonstrating the experiment described in this paper as well as several others are available at <https://youtu.be/OapaAQAGWDE>. The code which implements the algorithms described in this work and which supports both the virtual simulations and physical experiments can be found in <https://github.com/TSummersLab/cannon-tags>.

8. Conclusion

This work demonstrated tractable centralized methods for solving the goal assignment and inter-agent-collision-free trajectory planning problem for multiple robots. The assignment of agents to goals achieved a low total time-in-motion, and the resulting polynomial-in-time trajectories took full advantage of (possibly heterogeneous) speed capabilities. The results of numerical simulations revealed promising decreases in the total time with only mild increases in the

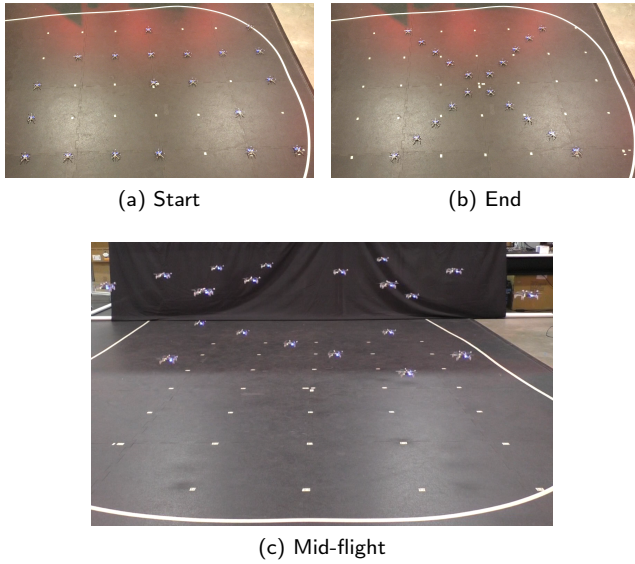


Figure 14: Photographs of experimental setup at the (a) start configuration, top view, (b) end configuration, top view, and (c) mid-flight, side view.

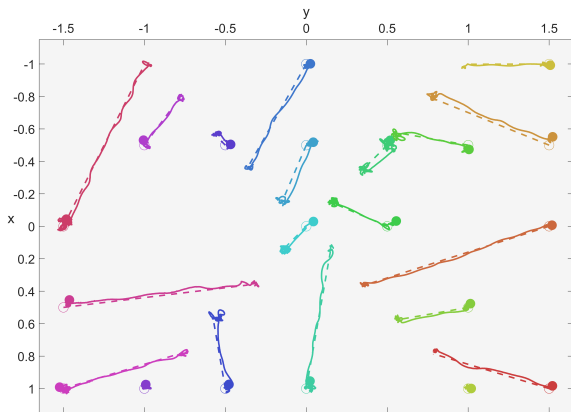


Figure 15: Top-down view of agent centers (filled dots) and trajectories at the beginning of the X20 experiment. Desired trajectory paths shown as dashed lines and actual paths realized by the physical vehicles are shown as solid lines.

computation time over existing approaches, allowing faster task completion in practical terms. The proposed algorithm also allowed us to eliminate restrictions present in other methods such as enforcement of synchronized start and end times and minimum separation of start and goal locations.

Future work is envisioned where the proposed framework would be used as a high-level centralized planner, combined with other decentralized techniques for dealing with lower-level local obstacles and disturbances. The ability to use different altitudes i.e. all three spatial dimensions is crucial to the proper working of the proposed approach; operating spaces limited to a single 2D plane are not supported. Future work will investigate using curved (polynomial) paths to alleviate this issue while retaining tractability.

Future work also includes extension to agents with more

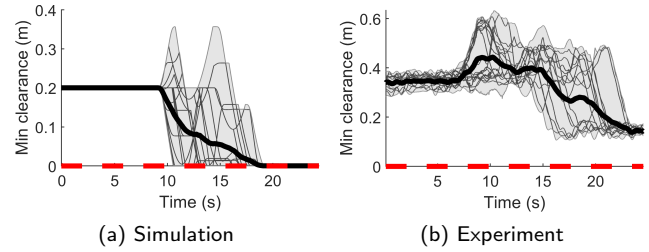


Figure 16: Minimum clearance between each agent and all other agents vs time, showing (a) clearance using the enlarged collision volumes on the disturbance-free simulation trajectories (b) clearance using the actual vehicle boundary volume on the noisy realized physical paths. The minimum clearance of each agent is plotted as a thin line, the mean as a thick line, the range between minimum and maximum shaded in grey, and zero as a dashed line.

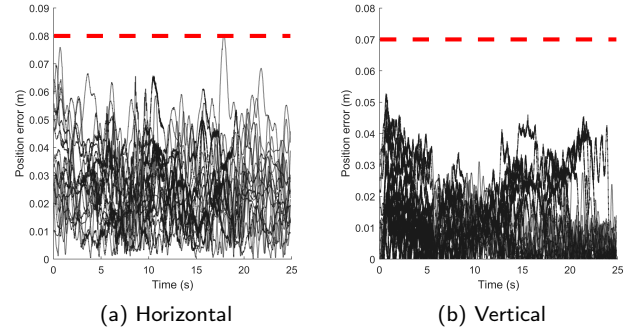


Figure 17: Position errors over time for all agents during the X20 experiment with (a) horizontal and (b) vertical components shown. Upper bounds used for trajectory planning are shown as dashed lines.

complex dynamics and/or motion constraints, dealing with uncontrolled obstacles, combining time delays with altitudes, reassigning goals dynamically to further reduce would-be collisions, and a parallel implementation to decrease solve times. Investigation of the setting when there are more goals than agents and the setting of multiple stages is also warranted, both requiring dynamic goal assignment and replanning.

Regarding the hardware implementation, refinements to the localization and state estimation furnished by the camera system as well as using more sophisticated controllers which account for downwash and ground effects [48, 49] could further reduce the magnitude of the actual position errors and allow shrinkage of the collision volumes.

A. Algorithms

Algorithm 5: Polynomial minimization

Input: Polynomial $p(t)$, interval $[t_0, t_f]$.
 Differentiate $p(t)$ to obtain $\dot{p}(t)$ of degree $d - 1$. This operation amounts to d integer multiplications from the exponents of t .
 Find roots t_i of $\dot{p}(t)$ by calculating eigenvalues of a companion matrix, as with the MATLAB roots function.
 Discard spurious solutions with imaginary part $\text{Im}(t_i) \neq 0$ or $t_i \notin [t_0, t_f]$.
 Append the endpoints t_0 and t_f to the list t_i .
 Evaluate $p_i = p(t_i)$.
Output: Minimum $p^* = \min(p_i)$ and associated time $t^* = \text{argmin}(p_i)$.

Algorithm 6: Polynomial above value

Input: Polynomial $p(t)$, lower value p_- .
 Find roots t_i of $p(t) - p_-$.
 Discard spurious t_i with imaginary part $\text{Im}(t_i) \neq 0$.
if $\{t_i\} = \emptyset$ **then**
 if $p(0) \geq p_-$ **then**
 $\mathcal{T}_{i-} \leftarrow [-\infty, \infty]$
 else
 $\mathcal{T}_{i-} \leftarrow \emptyset$
else
 Sort the roots t_i in ascending order.
 for $i = 1, 2, \dots, |\{t_i\}| + 1$ **do**
 $t_{lwr} \leftarrow t_{i-1-}$
 $t_{upr} \leftarrow t_i$
 $t_{eval} \leftarrow (t_{lwr} + t_{upr})/2$
 if $i = 1$ **then**
 $t_{eval} \leftarrow t_i - 1$
 $t_{lwr} \leftarrow -\infty$
 else if $i = |\{t_i\}| + 1$ **then**
 $t_{eval} \leftarrow t_i + 1$
 $t_{upr} \leftarrow \infty$
 Evaluate $p_{i-} = p(t_{eval})$.
 if $p_{i-} \geq p_-$ **then**
 $\mathcal{T}_{i-} \leftarrow [t_{lwr}, t_{upr}]$
 else
 $\mathcal{T}_{i-} \leftarrow \emptyset$
Output: Domain intervals $\mathcal{T}_{i-} \neq \emptyset$.

Algorithm 7: Polynomial interval restriction

Input: Polynomial $p(t)$, domain interval $\mathcal{T}_{\text{glob}} = [t_0, t_f]$, range interval $[p_-, p_+]$.
 Find domain intervals \mathcal{T}_{i-} and \mathcal{T}_{j+} by calling Alg. 6 with inputs $p(t)$ and p_- or p_+ respectively.
 Initialize $k \leftarrow 1$
foreach Pair \mathcal{T}_{i-} and \mathcal{T}_{j+} **do**
 $\mathcal{T}_k \leftarrow \{\mathcal{T}_{i-} \cap \mathcal{T}_{j+}\} \cap \mathcal{T}_{\text{glob}}$
 $k \leftarrow k + 1$
Output: Time intervals $\mathcal{T}_k \neq \emptyset$.

Algorithm 8: Segment pair collision check

Input: Heading unit vectors \hat{h}_i and \hat{h}_j , polynomial trajectories $x_i(t) = p_i(t)\hat{h}_i$ and $x_j(t) = p_j(t)\hat{h}_j$ of degree d over time intervals $\mathcal{T}_i = [t_{i,0}, t_{i,f}]$ and $\mathcal{T}_j = [t_{j,0}, t_{j,f}]$.
 Calculate shared time interval $\mathcal{T}_{ij} = \mathcal{T}_i \cap \mathcal{T}_j$.
if \mathcal{T}_{ij} not empty **then**
 if $\hat{h}_{i,3} = 0$ and $\hat{h}_{j,3} = 0$ **then**
 if (16) true for $x_i(\tau), x_j(\tau), \tau \in \mathcal{T}_{ij}$ **then**
 Obtain d^* from Alg. 1 with inputs $\hat{h}_{i,12}, \hat{h}_{j,12}, x_i(t) = p_i(t)\hat{h}_{i,12}, x_j(t) = p_j(t)\hat{h}_{j,12}, \mathcal{T}_{ij}$.
 if (15) true for $\|x_{j,12} - x_{i,12}\| = d^*$ **then**
 return True
 else if $\hat{h}_{i,12} = 0$ and $\hat{h}_{j,12} = 0$ **then**
 if (15) true for $x_i(\tau), x_j(\tau), \tau \in \mathcal{T}_{ij}$ **then**
 Obtain d^* from Alg. 1 with inputs $\hat{h}_{i,3}, \hat{h}_{j,3}, x_i(t) = p_i(t)\hat{h}_{i,3}, x_j(t) = p_j(t)\hat{h}_{j,3}, \mathcal{T}_{ij}$.
 if (16) true for $|x_{j,3} - x_{i,3}| = d^*$ **then**
 return True
 else
 Calculate the time interval(s) $\mathcal{T}'_{ij,k}$ where (16) is satisfied using Alg. 7.
 if $\hat{h}_{i,3} = 0$ **then**
 $\hat{h} = \hat{h}_i$
 else
 $\hat{h} = \hat{h}_j$
 foreach $\mathcal{T}'_{ij,k}$ **do**
 Obtain boolean b from Alg. 8 with inputs $\hat{h}, \hat{h}, x_i(t) = p_i(t)\hat{h}, x_j(t) = 0\hat{h}, \mathcal{T}'_{ij,k}$.
 if b **then**
 return True
 return False
Output: Boolean of collision.

References

- [1] P. Švestka, M. H. Overmars, Probabilistic path planning, Springer Berlin Heidelberg, Berlin, Heidelberg, 1998, pp. 255–304.
- [2] E. W. Dijkstra, A note on two problems in connexion with graphs, *Numer. Math.* 1 (1959) 269–271.
- [3] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Transactions on Systems Science and Cybernetics* 4 (1968) 100–107.
- [4] W. Wang, W.-B. Goh, Multi-robot path planning with the spatio-temporal A* algorithm and its variants, in: Proceedings of the 10th International Conference on Advanced Agent Technology, AAMAS'11, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 313–329.
- [5] S. Koenig, M. Likhachev, Incremental A*, in: T. G. Dietterich, S. Becker, Z. Ghahramani (Eds.), *Advances in Neural Information Processing Systems 14*, MIT Press, 2002, pp. 1539–1546.
- [6] S. Koenig, M. Likhachev, D. Furcy, Lifelong planning A*, *Artificial Intelligence* 155 (2004) 93–146.
- [7] A. Stentz, Optimal and efficient path planning for unknown and dynamic environments, *International Journal of Robotics and Automation* 10 (1993) 89–100.
- [8] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, Cambridge, U.K., 2006.
- [9] M. Turpin, K. Mohta, N. Michael, V. Kumar, Goal assignment and trajectory planning for large teams of interchangeable robots, *Autonomous Robots* 37 (2014) 401–415.
- [10] G. Sharon, R. Stern, A. Felner, N. R. Sturtevant, Conflict-based search for optimal multi-agent pathfinding, *Artificial Intelligence* 219 (2015) 40–66.
- [11] A. Adler, M. de Berg, D. Halperin, K. Solovey, Efficient multi-robot motion planning for unlabeled discs in simple polygons, *IEEE Transactions on Automation Science and Engineering* 12 (2015) 1309–1317.
- [12] K. Solovey, J. Yu, O. Zamir, D. Halperin, Motion planning for unlabeled discs with optimality guarantees, in: Proceedings of Robotics: Science and Systems, Rome, Italy, 2015.
- [13] W. Höngig, J. A. Preiss, T. K. S. Kumar, G. S. Sukhatme, N. Ayanian, Trajectory planning for quadrotor swarms, *IEEE Transactions on Robotics* 34 (2018) 856–869.
- [14] M. Erdmann, T. Lozano-Perez, On multiple moving objects, in: Proceedings. 1986 IEEE International Conference on Robotics and Automation, volume 3, 1986, pp. 1419–1424.
- [15] G. Wagner, M. Kang, H. Choset, Probabilistic path planning for multiple robots with subdimensional expansion, in: 2012 IEEE International Conference on Robotics and Automation, 2012, pp. 2886–2892.
- [16] G. Wagner, H. Choset, Subdimensional expansion for multirobot path planning, *Artificial Intelligence* 219 (2015) 1 – 24.
- [17] K. Solovey, O. Salzman, D. Halperin, Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning, in: *Algorithmic Foundations of Robotics XI*, Springer, 2015, pp. 591–607.
- [18] M. Turpin, N. Michael, V. Kumar, CAPT: Concurrent assignment and planning of trajectories for multiple robots, *The International Journal of Robotics Research* 33 (2014) 98–112.
- [19] T. Schouwenaars, B. De Moor, E. Feron, J. How, Mixed integer programming for multi-vehicle path planning, in: *European Control Conference*, IEEE, 2001, pp. 2603–2608.
- [20] A. Richards, J. P. How, Aircraft trajectory planning with collision avoidance using mixed integer linear programming, in: *American Control Conference*, volume 3, IEEE, 2002, pp. 1936–1941.
- [21] A. Alonso-Ayuso, L. F. Escudero, F. J. Martín-Campo, Multiobjective optimization for aircraft conflict resolution. a metaheuristic approach, *European Journal of Operational Research* 248 (2016) 691 – 702.
- [22] F. Augugliaro, A. P. Schoellig, R. D'Andrea, Generation of collision-free trajectories for a quadcopter fleet: A sequential convex programming approach, in: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 1917–1922.
- [23] Y. Chen, M. Cutler, J. P. How, Decoupled multiagent path planning via incremental sequential convex programming, in: *2015 IEEE International Conference on Robotics and Automation*, 2015, pp. 5954–5961.
- [24] E. Frazzoli, Z.-H. Mao, J.-H. Oh, E. Feron, Resolution of conflicts involving many aircraft via semidefinite programming, *Journal of Guidance, Control, and Dynamics* 24 (2001) 79–86.
- [25] S. Kloder, S. Hutchinson, Path planning for permutation-invariant multirobot formations, *IEEE Transactions on Robotics* 22 (2006) 650–665.
- [26] I. Solomon, D. Halperin, Motion planning for multiple unit-ball robots in R^d , in: *International Workshop on the Algorithmic Foundations of Robotics*, Springer, 2018, pp. 799–816.
- [27] C. W. Warren, Multiple robot path coordination using artificial potential fields, in: *Robotics and Automation*, 1990. Proceedings., 1990 IEEE International Conference on, IEEE, 1990, pp. 500–505.
- [28] P. Fiorini, Z. Shiller, Motion planning in dynamic environments using velocity obstacles, *The International Journal of Robotics Research* 17 (1998) 760–772.
- [29] J. van den Berg, M. Lin, D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, in: *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1928–1935.
- [30] J. van den Berg, S. J. Guy, M. Lin, D. Manocha, *Reciprocal n-Body Collision Avoidance*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 3–19.
- [31] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, P. Dubey, Clearpath: Highly parallel collision avoidance for multi-agent simulation, in: *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '09, ACM, New York, NY, USA, 2009, pp. 177–187.
- [32] S. J. Guy, J. Chhugani, S. Curtis, P. Dubey, M. Lin, D. Manocha, Pedestrians: A least-effort approach to crowd simulation, in: *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 2010, pp. 119–128.
- [33] M. Cap, P. Novak, A. Kleiner, Finding near-optimal solutions in multi-robot path planning, *CoRR abs/1410.5200* (2014).
- [34] M. Cap, P. Novak, A. Kleiner, M. Selecký, Prioritized planning algorithms for trajectory coordination of multiple mobile robots, *IEEE Transactions on Automation Science and Engineering* 12 (2015) 835–849.
- [35] D. Panagou, M. Turpin, V. Kumar, Decentralized goal assignment and safe trajectory generation in multirobot networks via multiple lyapunov functions, *IEEE Transactions on Automatic Control* 65 (2020) 3365–3380.
- [36] B. Gravell, T. Summers, Concurrent goal assignment and collision-free trajectory generation for multiple aerial robots, *IFAC-PapersOnLine* 51 (2018) 75 – 81. IFAC Workshop on Networked & Autonomous Air & Space Systems NAASS 2018.
- [37] D. Mellinger, V. Kumar, Minimum snap trajectory generation and control for quadrotors, in: *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 2520–2525.
- [38] M. W. Spong, M. Vidyasagar, *Robot dynamics and control*, John Wiley & Sons, 2008.
- [39] D. Mellinger, A. Kushleyev, V. Kumar, Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams, in: *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 477–483.
- [40] H. W. Kuhn, The Hungarian method for the assignment problem, *Naval Research Logistics Quarterly* 2 (1955) 83–97.
- [41] J. Munkres, Algorithms for the assignment and transportation problems, *Journal of the Society for Industrial and Applied Mathematics* 5 (1957) 32–38.
- [42] D. P. Bertsekas, D. A. Castañón, The auction algorithm for the transportation problem, *Annals of Operations Research* 20 (1989) 67–96.
- [43] D. P. Bertsekas, D. A. Castañón, Parallel synchronous and asynchronous implementations of the auction algorithm, *Parallel Comput.* 17 (1991) 707–732.
- [44] V. J. Lumelsky, On fast computation of distance between line segments, *Information Processing Letters* 21 (1985) 55–61.
- [45] L. F. Toth, Über die dichteste kugellagerung, *Mathematische Zeitschrift* 48 (1942) 676–684.

- [46] I. C. A. Organization, Doc 4444 - Procedures for Air Navigation Services - Air Traffic Management, 16 ed., International Civil Aviation Organization, 2016.
- [47] J. A. Preiss, W. Hönig, G. S. Sukhatme, N. Ayanian, CrazySwarm: A large nano-quadcopter swarm, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3299–3304.
- [48] D. Yeo, E. Shrestha, D. A. Paley, E. M. Atkins, An empirical model of rotorcraft UAV downwash for disturbance localization and avoidance, in: AIAA Atmospheric Flight Mechanics Conference, 2015, p. 1685.
- [49] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azzadenesheli, A. Anandkumar, Y. Yue, S.-J. Chung, Neural lander: Stable drone landing control using learned dynamics, arXiv preprint arXiv:1811.08027 (2018).