A Safe Cooperative Framework for Atmospheric Science Missions with Multiple Heterogeneous UAS using Piecewise Bézier Curves

S. Bilal Mehdi* Javier Puig-Navarro† Ronald Choe†
Venanzio Cichella* Naira Hovakimyan‡
University of Illinois at Urbana-Champaign, Urbana, IL, 61801, USA

Meghan Chandarana§
Carnegie Mellon University, Pittsburgh, PA, 15213, USA

Anna Trujillo¶ Paul M. Rothhaar‖ Loc Tran**
James H. Neilan** B. Allen Danette††
NASA Langley Research Center, Hampton, VA, 23681, USA

Autonomous operation of UAS holds promise for greater productivity of atmospheric science missions. However, several challenges need to be overcome before such missions can be made autonomous. This paper presents a framework for safe autonomous operations of multiple vehicles, particularly suited for atmospheric science missions. The framework revolves around the use of piecewise Bézier curves for trajectory representation, which in conjunction with path-following and time-coordination algorithms, allows for safe coordinated operations of multiple vehicles.

I. Introduction

With the increased interest in modeling and predicting the Earth's climate, and the growing need for more accurate weather predictions,[1–3] a better understanding of the underlying processes that drive our climate through field studies has gained critical importance.[4] In this regard and under the Science Mission Directorate (SMD), collection of atmospheric data, necessary for such understanding, is being conducted at NASA through missions that often rely on *i*) helium balloons; *ii*) manned aircraft such as Cessna 402B or NASA's P-3B aircraft; or *iii*) remotely piloted vehicles such as the Global Hawk. With the recent availability of relatively cheap multirotors, great interest has been shown in their use for science missions as well.

Although there is a variety of platforms available for atmospheric science missions, no current solution allows scientists to conduct a mission without an experienced crew being available at hand. This not only limits the number and types of missions that can be conducted, but also increases their cost. Thus, for a greater science and application productivity, autonomous operation of these platforms is important.[5]

Enabling autonomous operation for science missions, however, comes with several challenges. First, we point out that the requirements for atmospheric science missions are often different from the more classical

*Graduate Student, Dept. of Mechanical Science and Engineering, 1206 W. Green St., Urbana, IL, AIAA Student Member.
†Graduate Student, Dept. of Aerospace Engineering, 104 S. Wright Street, Urbana, IL, AIAA Student Member.
‡Professor, Dept. of Mechanical Science and Engineering, AIAA Associate Fellow.
§Graduate Student, Dept. of Mechanical Engineering, 5000 Forbes Ave., Pittsburgh, PA.
¶Senior Researcher, Crew Systems & Aviation Operations Branch, MS 492, AIAA Member.
‖Researcher, Dynamic Systems & Control Branch, MS 308.
**Researcher, Flight Software Systems Branch, MS 492.
††Senior Technologist for Intelligent Flight Systems, Crew Systems & Aviation Operations Branch, MS 492, AIAA Senior Member.

autonomy task of reaching a destination point given an initial point. From the perspective of mission planning, two primary challenges that differentiate atmospheric science missions are:

- *Flying in a geometric pattern:* for atmospheric data collection, scientists often require vehicles to fly in a predefined geometric pattern. The Atmospheric Tomography Mission[a] under NASA is an example of such a mission. By flying a DC-8 aircraft in a pattern, this mission provides concentration measurements of methane, tropospheric ozone, and black carbon aerosols at different altitudes.
- *Coordinated flight of heterogeneous vehicles:* in order to collect correlative data, or to calibrate sensors, science missions can require multiple heterogeneous vehicles to fly in coordination. An example of such operation would be NASA GRIP[b] mission, in which hurricane formation and intensification data was collected using three different kinds of aircraft (namely DC-8, B-57, and Global Hawk).

Taking these trajectory-generation requirements into consideration, this paper presents a piecewise Bézier curve based framework for the design and operation of atmospheric science missions that can provide scientists with the following capabilities

- Generate trajectories for the following tasks:
  - Given a set of initial positions, velocities (and accelerations), reach a final destination with a specific velocity (and acceleration) while satisfying a predefined inter-vehicle schedule. In this case, the final destination can either be a landing spot to reach at the end of the mission or, alternatively, the location of interest for correlative data acquisition.
  - Fly geometric patterns, possibly in formation. Some patterns of interest include *i*) straight vertical or helical curves where concentration of a gas can be measured at different heights using multirotors or fixed wing aircraft, respectively; *ii*) Boustrophedon-path like patterns to study the concentration of certain atmospheric components over a given region.
- Perform collision avoidance maneuvers using onboard algorithms.
- Autonomously follow paths while ensuring time coordination between the vehicles.

This paper is organized as follows: we begin with an introduction to Bézier curves in Section II, followed by an overview of the framework in Section III. The trajectory-generation algorithm, that serves as the backbone of the framework is provided in Section IV. Section V provides an overview of the collision-avoidance algorithm used in the proposed framework. The path-following and time-coordination algorithms are provided in Sections VI and VII, respectively. Finally, we provide a simulation that brings all these algorithms together in Section VIII and conclude the paper in Section IX.

II. Bézier Curves

In this section, we introduce Bézier curves and mention a few relevant properties and algorithms designed for them. An n-th degree Bézier curve is a polynomial curve defined completely by its control points $\bar{\mathbf{r}}_k$ as

$$\mathbf{r}(\zeta) = \sum_{k=0}^{n} \bar{\mathbf{r}}_k b_k^n(\zeta), \qquad \zeta \in [0,1],$$

where $b_k^n(\zeta)$ are the Bernstein polynomials given as

$$b_k^n(\zeta) = \binom{n}{k}(1-\zeta)^{n-k}\zeta^k, \quad \zeta \in [0,1].$$

An example Bézier curve is shown in Figure 1.

[a] http://science.nasa.gov/missions/atom/
[b] http://airbornescience.nsstc.nasa.gov/grip/

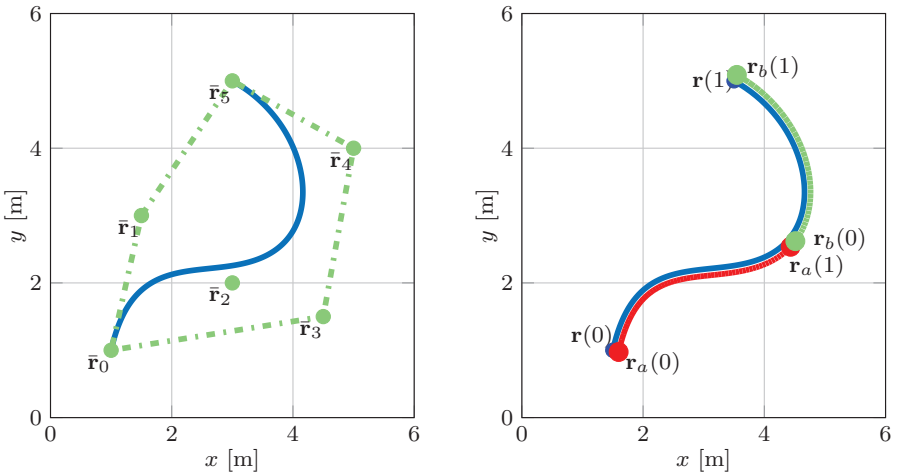

Figure 1: Left panel shows a Bézier curve in blue, defined by its control points $\bar{\mathbf{r}}_k$, $k \in \{0, 1, \ldots, 5\}$ shown in green. Right panel shows the application of the de Casteljau algorithm on the curve to obtain the curves shown in red and green. For illustrative purposes, the red and green curves have been shifted. In fact, they lie on top of the blue curve and $\mathbf{r}_a(1) = \mathbf{r}_b(0)$.

A. Properties of Bézier Curves

Bézier curves were initially popularized by Pierre Bézier for being intuitive to design and modify. The properties that make these curves intuitive to use, also make them suitable for piecewise curve generation:

- A Bézier curve $\mathbf{r}(\zeta)$, for which all the control points are equal,
$$\bar{\mathbf{r}}_i = \bar{\mathbf{r}}_j \qquad i, j \in \{0, \cdots, n\},$$
is essentially a point, given as $\mathbf{r}(\zeta) = \bar{\mathbf{r}}_i$ for any $\zeta \in [0, 1]$.
- The initial and final points of a Bézier curve ($\mathbf{r}(0)$ and $\mathbf{r}(1)$) are equal to its first and last control points, respectively.
- The derivative of a Bézier curve of degree n:
$$\mathbf{q}(\zeta) = \frac{d\mathbf{r}(\zeta)}{d\zeta},$$
is a Bézier curve of degree $n-1$ with control points $\bar{\mathbf{q}}_k$:
$$\bar{\mathbf{q}}_k = n(\mathbf{r}_{k+1} - \mathbf{r}_k), \qquad k \in \{0, \cdots, n-1\}.$$
- From the above two properties, it follows that the initial and final derivative of a Bézier curve are specified as:
$$\mathbf{q}(0) = n(\mathbf{r}_1 - \mathbf{r}_0), \quad \text{and} \quad \mathbf{q}(1) = n(\mathbf{r}_n - \mathbf{r}_{n-1}),$$
respectively. A similar expression can be specified for initial and final second derivative.
- Addition of two Bézier curves with the same degree results in another Bézier curve of identical degree.

B. The de Casteljau Algorithm

The de Casteljau algorithm[6] can be used to subdivide a Bézier curve into two independent curves. Specifically, given an n^{th} degree Bézier curve and a scalar $\zeta_{\text{div}} \in (0, 1)$, the de Casteljau algorithm provides two independent Bézier curves $\mathbf{r}_1(\zeta)$ and $\mathbf{r}_2(\zeta)$ of the same degree such that

$$\mathbf{r}(\zeta) = \begin{cases} \mathbf{r}_1\left(\frac{\zeta}{\zeta_{\text{div}}}\right), & \zeta \in [0, \zeta_{\text{div}}], \\ \mathbf{r}_2\left(\frac{\zeta - \zeta_{\text{div}}}{1 - \zeta_{\text{div}}}\right), & \zeta \in [\zeta_{\text{div}}, 1]. \end{cases}$$

An illustration of this algorithm is provided in Figure 1.

C. Minimum Distance Calculation

Given two Bézier curves $\mathbf{r}_a(\zeta)$ and $\mathbf{r}_b(\zeta)$ with $\zeta \in [0,1]$, the minimum distance algorithm,[7] uses the de Casteljau algorithm[6] and the Gilbert-Johnson-Keerthi algorithm[8] to efficiently calculate

$$\min_{\zeta_a,\zeta_b\in[0,1]} ||\mathbf{r}_a(\zeta_a) - \mathbf{r}_b(\zeta_b)||, \quad \text{and} \quad \operatorname*{argmin}_{\zeta_a,\zeta_b\in[0,1]} ||\mathbf{r}_a(\zeta_a) - \mathbf{r}_b(\zeta_b)||,$$

without requiring any discretization.

III. The Framework

The control architecture adopted in this paper is primed for atmospheric science missions and enables coordinated flight of heterogeneous vehicles, tasked to fly to a target point and perform predefined flight patterns. The control structure is shown in Figure 2.

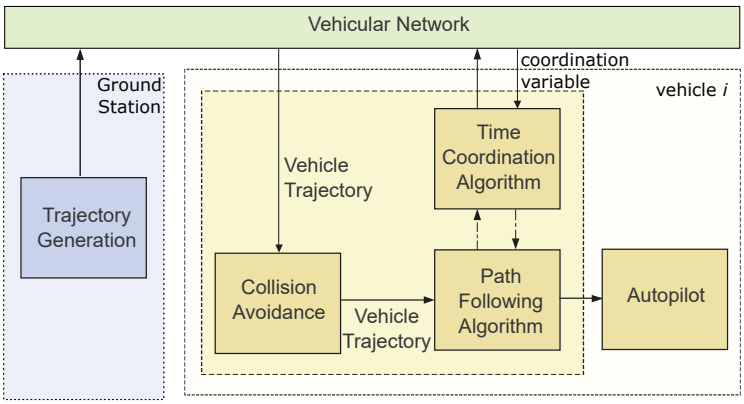

Figure 2: Overall architecture used by the framework that involves a ground station and multiple vehicles.

In Figure 2, the blue block represents a ground station that performs centralized trajectory generation (and possibly trajectory replanning) for all vehicles. Using the vehicle network, the ground station communicates the trajectory information to all vehicles.

The vehicles themselves are shown as the yellow block. Apart from receiving trajectory information from the ground station, these vehicles can communicate among one another for coordination during flight. The algorithms onboard the vehicles include the *i*) collision-avoidance algorithm that allows the vehicles to replan their own trajectories in case of an unexpected obstacle; and the *ii*) path-following and the time-coordination algorithms that allow the vehicle to follow the trajectory in a coordinated way.

IV. Trajectory Generation

In this section, we describe our method of generating trajectories that are of particular interest for atmospheric science missions. Mathematically, the objective of the trajectory-generation algorithm is to find a desired set of trajectories

$$\mathbf{p}_{d,i}(t_d) : [0, t^{\mathrm{f}}_{d,i}] \to \mathbb{R}^3, \qquad \forall i \in \{1, 2, \ldots, N\},$$

where N is the number of vehicles, $t^{\mathrm{f}}_{d,i} \in \mathbb{R}^+$ is the desired mission time for the ith vehicle, and $t_d \in [0, T_d]$ is the time-variable used during the trajectory-generation phase, with $T_d := \max\{t^{\mathrm{f}}_{d,1}, \ldots, t^{\mathrm{f}}_{d,N}\}$ being the desired overall mission duration.

One of the key contributions of this method is the use of piecewise Bézier curves for representing trajectories, where each curve is associated with a specific mission time subinterval. Specifically, for the i-th vehicle, we consider a sequence of i_m time instants in ascending order

$$\lambda^0_i = 0, \lambda^1_i, \lambda^2_i, \ldots, \lambda^{i_m}_i = t^{\mathrm{f}}_{d,i}.$$

Then for each interval $[\lambda_i^{j-1}, \lambda_i^j]$, we denote the j-th curve of the i-th vehicle's trajectory by $\mathbf{p}_{d,i}^j(t_d)$,

$$\mathbf{p}_{d,i}(t_d) = \mathbf{p}_{d,i}^j(t_d), \qquad t_d \in [\lambda_i^{j-1}, \lambda_i^j],$$

where $j \in \{1, 2, \cdots, i_m\}$. Furthermore, we represent $\mathbf{p}_{d,i}^j(t_d)$ as a Bézier curve

$$\mathbf{p}_{d,i}^j(t_d) = \sum_{k=0}^{n} \bar{\mathbf{p}}_{i,k}^j b_k^n \left(\frac{t_d - \lambda_i^{j-1}}{\lambda_i^j - \lambda_i^{j-1}} \right), \quad t_d \in [\lambda_i^{j-1}, \lambda_i^j],$$

where $\bar{\mathbf{p}}_{i,k}^j$ represents the k-th control point for the j-th curve of the i-th vehicle's trajectory.

Remark 1 *Note that each Bézier curve is a polynomial, and is therefore, C^∞. Thus, in order to achieve C^0 continuity of the trajectory $\mathbf{p}_{d,i}(t_d)$, we only need to ensure that $\mathbf{p}_{d,i}^j(\lambda_i^j) = \mathbf{p}_{d,i}^{j+1}(\lambda_i^j)$ for all $j \in \{1, \ldots, i_m - 1\}$. Using properties of Bézier curves, we conclude that this can be ensured by appropriately placing the first and last control point of each Bézier curve. Similarly, C^1 (or C^2) continuity can be achieved by appropriately placing the first and last two (or three) control points of each Bézier curve and so on.*

We use two different methodologies for generating each Bézier curve. As a result, a single trajectory may be comprised of multiple curves designed by utilizing different methodologies. Such generation methods are described in the following sections.

A. Optimization-based Bézier Curve Generation

The trajectory-generation algorithm adopted in this work is based on the methods described in [9,10]. Given the problem formulation above, this can be formally expressed as

$$\min_{\substack{\mathbf{p}_{d,i}^j \in \mathcal{P} \\ i=1,\ldots,N}} J(\mathbf{p}_{d,i}^j) = \min_{\mathbf{p}_{d,i}^j \in \mathcal{P}} \sum_{i=1}^{N} J_i(\mathbf{p}_{d,i}^j)$$

$$\begin{aligned} \text{subject to} \quad & \text{boundary conditions,} \\ & \text{spatial constraints,} \\ & \text{temporal constraints,} \\ & \text{dynamic constraints of the aircraft,} \\ & \text{mission-specific constraints,} \end{aligned}$$

where the $\mathcal{P}$ is a subset of Bézier curves. The cost function $J(\mathbf{p}_{d,i})$ can be any variable of interest that we wish to minimize such as: the total energy associated with the trajectories, an estimate of the fuel consumption, the deviation from the optimal velocity of the aircraft, or the mission time. Next, we present a simple classification of the set of constraints the trajectory may be subject to, along with some examples:

- *Boundary conditions* are the initial and final specifications on the trajectory. Hereof, we consider the initial and final position, the initial and final path directions, as well as the initial and final velocities. Some other boundary conditions that can be considered are the initial and final accelerations.
- *Spatial constraints* are the limits that shall be imposed to ensure a safe separation between a UAS and all the other elements in the environment. The most common examples are the minimum distance among vehicles, and the minimum distance between vehicles and obstacles.
- *Temporal constraints* are the type of assignments that impose some kind of schedule on the trajectories of the vehicles. Some examples are a relative inter-vehicle schedule, a desired time of arrival, or a desired window of arrival to the final destination.
- *Dynamic constraints of the vehicles* considered in this paper are a simplified version of the physical limits of the UAS. Hence, these assignments depend on the type of vehicle utilized during the mission. The types of vehicles, and the dynamic constraints considered are:
  - *Fixed-wing aircraft*, typically limited by a minimum and a maximum speed, acceleration, flight path angle, as well as a maximum turning and pitching rates.

– *Rotary-wing aircraft* typically fly within an envelope limited by the maximum excess thrust which determines the maximum total acceleration. Additionally, we can also impose limits on the velocity of the vehicle.

The optimization exploits the favorable geometric properties of Bézier curves and existing algorithms to solve the trajectory-generation problem described above. The benefits of using Bézier curves in this algorithm can be summarized as follows:

- Since Bézier curves are closed under integration and composition, the optimization procedure is able to take a decoupled approach. Specifically, the algorithm finds *i*) a quintic Bézier curve as the *spatial path*: a curve in space with no temporal specification, and *ii*) a quadratic Bézier curve as the *timing law* associated with the path: a function that captures the temporal specifications of the trajectory. This approach allows satisfaction of spatial and temporal constraints independently.
- By re-formulating the constraints into Bézier form, the constraint functions can be formulated as minimum distance problems and, therefore, solved using the minimum distance algorithm[7] for efficient constraint verification.

B. Bézier Curve Generation for Flying Geometric Patterns

Control points of Bézier cuves provide an intuitive control over the shape of the curve. Thus, trajectories for flying geometric patterns can often be easily designed using Bézier curves. Since, the exact design procedure varies with the pattern of interest, we show one pattern here as an example.

Specifically, in this section, we consider a Boustrophedon-like trajectory for a coverage task. Unlike a Boustrophedon path, however, we require the trajectory to be C^2-continuous (see Figure 3). Furthermore, we require the trajectory to have constant speed for as long as possible. Thus, we can design the trajectory in two steps: *i*) find constant velocity straight line Bézier curves and *ii*) connect those straight line segments with transition curves such that the trajectory is C^2-continuous.

Straight line segments: let us assume that the j-th segment for the i-th vehicle's trajectory is the straight portion that starts at $\bar{\mathbf{p}}^j_{i,0}$ and ends at $\bar{\mathbf{p}}^j_{i,n}$ (see Figure 3). Furthermore, assume that the desired velocity for this portion is given as $\mathbf{v}_{\text{nom}}$.

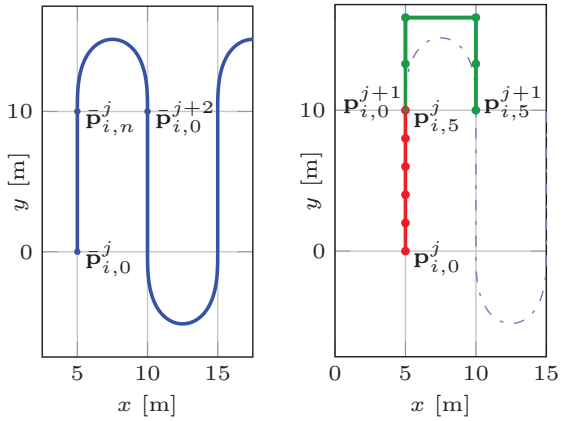

Figure 3: Left panel shows the desired path for the Boustrophedon-like trajectory. Right panel shows the desired control point locations to achieve such a trajectory.

Then, using the first three properties of Bézier curves described in Section IV-A, we conclude that the control points $\bar{\mathbf{p}}^j_{i,k}$ can be spread equally between $\bar{\mathbf{p}}^j_{i,0}$ and $\bar{\mathbf{p}}^j_{i,n}$. Specifically, we have

$$\bar{\mathbf{p}}^j_{i,k} = \bar{\mathbf{p}}^j_{i,0} + \frac{k}{n}(\bar{\mathbf{p}}^j_{i,n} - \bar{\mathbf{p}}^j_{i,0}), \qquad k \in \{0, 1, \ldots, n\}\,.$$

Furthermore, the length of time-interval associated with this segment can be calculated as

$$\lambda^j_i - \lambda^{j-1}_i = \frac{\bar{\mathbf{p}}^j_{i,n} - \bar{\mathbf{p}}^j_{i,0}}{||\mathbf{v}_{\text{nom}}||}.$$

Transition segments: now we assume that the $(j+1)$-th segment of the i-th vehicle's trajectory is the transition segment shown in Figure 3. Since we require C^2 continuity, the start and end position, velocity and acceleration of this segment is already specified. For example, the initial position, velocity and acceleration for this segment is $\bar{\mathbf{p}}^{j,n}_i$, $\mathbf{v}_{\text{nom}}$, and 0, respectively. Thus, setting the degree of this Bézier segment as $n = 5$, we can use properties of Bézier curves to conclude that

$$\bar{\mathbf{p}}^{j+1}_{i,k} = \begin{cases} \mathbf{p}^j_{i,n} + k\mathbf{v}_{\text{nom}}(\lambda^{j+1}_i - \lambda^j_i), & k \in \{0, 1, 2\}\,, \\ \mathbf{p}^{j+2}_{i,0} + (k-5)\mathbf{v}_{\text{nom}}(\lambda^{j+1}_i - \lambda^j_i), & k \in \{3, 4, 5\}\,. \end{cases}$$

Thus, we have 7 free variables[c] and 6 constraints. So we can arbitrarily choose the time interval for the segment $\lambda_i^{j+1} - \lambda_i^j$ and calculate the control points using the above equations.

V. Collision Avoidance

In the proposed framework, we adopt the collision-avoidance algorithm originally presented in [11]. This method is different from the trajectory-generation algorithm of Section IV in two primary ways. First, it is designed to run onboard vehicles and is, therefore, simpler and lighter in terms of computation. Second, the algorithm is run independently in all vehicles and does not utilize the knowledge of the trajectories of the other vehicles.

The algorithm can be divided into two primary steps: *i*) collision prediction and *ii*) trajectory replanning, discussed below.

A. Collision Prediction

Once vehicle i detects an obstacle with a predicted trajectory $\mathbf{p}_o(t_d)$, the algorithm onboard the vehicle utilizes $\mathbf{p}_{d,i}(t_d)$ and $\mathbf{p}_o(t_d)$ as well as the minimum distance algorithm to check for a possible collision in the future through the collision prediction steps described below:

1. A piecewise Bézier curve that represents the predicted separation between the vehicle and the obstacle is calculated
$$\mathbf{d}(t_d) = \mathbf{p}_{d,i}(t_d) - \mathbf{p}_o(t_d).$$
2. Using the separation Bézier curve, the minimum distance algorithm can be used to find
$$\min_{t_d \in [\lambda_i^{j-1}, \lambda_i^j]} ||\mathbf{d}(t_d)||, \qquad \operatorname*{argmin}_{t_d \in [\lambda_i^{j-1}, \lambda_i^j]} ||\mathbf{d}(t_d)||,$$
for all subintervals $[\lambda_i^{j-1}, \lambda_i^j]$ and $i \in \{1, 2, \ldots, i_m\}$. Thus,
$$d_{\min} = \min_{t_d \in [0, t_{d,i}^{\mathrm{f}}]} ||\mathbf{d}(t_{d,i})||,$$
can be found. If the minimum distance between the obstacle's estimated trajectory and the UAS' trajectory $d_{\min}$ is less or equal than a minimum safety distance d_{safe} then a collision is predicted.

B. Trajectory Replanning

In case vehicle i predicts a collision, it replans its trajectory by adding a detour $\Delta(t_d)$ to the original trajectory $\mathbf{p}_{d,i}(t_d)$. The procedure to calculate consists of the following steps:

1. Using a closed form formula, the algorithm first finds start and end time for the detour. In this paper, we denote the time for start and end of detour as t_d^{start} and t_d^{end}, respectively.
2. The algorithm then determines the time profile for the magnitude of the detour over the time interval $t_d \in [t_d^{\mathrm{start}}, t_d^{\mathrm{end}}]$. This time profile is calculated as a Bézier curve,
$$s(t_d) = \sum_{k=0}^{n} \bar{s}_k b_k^n \left(\frac{t_d - t_d^{\mathrm{start}}}{t_d^{\mathrm{end}} - t_d^{\mathrm{start}}} \right), \; t_d \in [t_d^{\mathrm{start}}, t_d^{\mathrm{end}}],$$
where $\bar{s}_k$ for the first and last three values of k, i.e. $k \in \{0, 1, 2, n-2, n-1, n\}$, are constrained to be zero. Then, through properties of Bézier curves described earlier, we guarantee that
$$\begin{aligned} &s(t_d^{\mathrm{start}}) = 0, \quad \left.\frac{ds(t_d)}{dt_d}\right|_{t_d = t_d^{\mathrm{start}}} = 0, \quad \left.\frac{d^2 s(t_d)}{dt_d^2}\right|_{t_d = t_d^{\mathrm{start}}} = 0, \\ &s(t_d^{\mathrm{end}}) = 0, \quad \left.\frac{ds(t_d)}{dt_d}\right|_{t_d = t_d^{\mathrm{end}}} = 0, \quad \left.\frac{d^2 s(t_d)}{dt_d^2}\right|_{t_d = t_d^{\mathrm{end}}} = 0. \end{aligned}$$
Now addition of this detour to the original trajectory will maintain C^2-continuity of the trajectory.

[c] This includes the time interval of the segment along with 6 free control points.

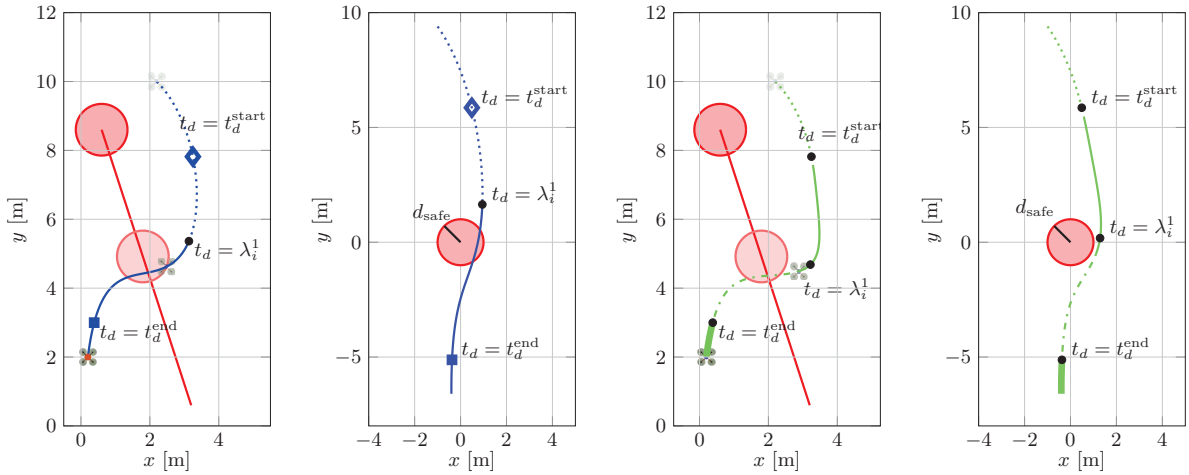

Figure 4: Illustration of the collision avoidance algorithm: first panel shows the original trajectory (blue) along with that of the obstacle (red). The separation curve is shown in the second panel. The third panel shows the replanned trajectory (green) with corresponding separation curve in the fourth panel (red). The diamond, square and thick black dots indicate t^{start}, t^{end} and λ's, respectively. Every unique line type indicates a single Bézier segment.

3. The detour is finalized by scaling with an appropriate vector. Specifically, the detour is found as

$$\Delta(t_d) = Ks(t_d)\mathbf{u}, \quad t_d \in [t_d^{\text{start}}, t_d^{\text{end}}],$$

where $\mathbf{u}$ is a unit vector that moves the vehicle away from the obstacle and K is a scaling coefficient that ensures collision avoidance.

4. Lastly, we add the detour to the original trajectory to obtain

$$\mathbf{p}_{d,i}^{\text{new}}(t_d) = \begin{cases} \mathbf{p}_{d,i}(t_d) + \Delta(t_d), & t_d \in [t_d^{\text{start}}, t_d^{\text{end}}], \\ \mathbf{p}_{d,i}(t_d), & \text{otherwise.} \end{cases}$$

Notice that the above equation is not a standard way of adding Bézier curves because of its piecewise nature. However, using the de Casteljau algorithm, we can separate segments of trajectory where addition takes place from the segments where it does not. Then, the above implementation can be performed by standard Bézier curve addition, that results in $\mathbf{p}_{d,i}^{\text{new}}(t_d)$ being a piecewise Bézier curve.

An illustration of this algorithm is provided in Figure 4.

We now summarize a few benefits this algorithm ensures through Bézier curve representation of trajectories:

1. Imminent collisions can be predicted fast and do not require any discretization.

2. As shown in [11], the algorithm provides bounds on the change in position, velocity and acceleration required for each collision avoidance maneuver. This can be used to guarantee a certain number of collision avoidance maneuvers without violating vehicle dynamic constraints.

3. By simply constraining the first and last three control points of the detour curve, the algorithm ensures C^2-continuity for the replanned trajectory.

4. The new detour trajectory found using this algorithm is also a piecewise Bézier curve, thus allowing seamless continuation of the mission.

VI. Path Following

The task of the path-following algorithm is to take in the trajectory information and produce commands that ensure that each vehicle converges to and follows its assigned path.[12–17] Due to the fundamental differences in the configuration of multirotors and fixed-wing aircraft, distinct flight control strategies are required for these types of vehicles. This section describes the integration of the path-following algorithms in [14, 17] into the proposed framework.

The underlying idea behind these algorithms is to track a virtual target that slides along the path. For this purpose, both algorithms follow a similar approach: a moving frame is attached to this virtual target and a generalized error vector $\mathbf{e}_{PF}(t)$[d] is defined. This generalized path-following error characterizes the position and attitude errors between the coordinate system attached to the virtual target and a frame attached to the vehicle. It should be noted that the definition of $\mathbf{e}_{PF}(t)$ as well as the corresponding control signals $\mathbf{u}_{PF}(t)$ generated are different for fixed-wing aircraft and multirotors, due to the differences in control strategies.

A. Objective

Formally, the purpose of the path-following algorithms is to produce a control signal $\mathbf{u}_{PF}(t)$ that ensures that the generalized path-following error remains within a small neighborhood of zero:

$$||\mathbf{e}_{PF}(t)|| \leq \kappa_0 e^{-\alpha t} + \kappa_\infty,$$

where λ, κ_0, and κ_∞ are known non-negative constants and κ_∞ defines the uniform ultimate bound (ideally $\kappa_\infty = 0$). In this sense, the control strategies to steer different types of vehicles along their path differ in the following aspects:

- The path-following algorithm for *fixed-wing aircraft* generates pitch-rate $q_c(t)$, yaw-rate $r_c(t)$ and speed $v_c(t)$ commands:
$$\mathbf{u}_{PF}(t) := [q_c(t), r_c(t), v_c(t)]^\top.$$
Note also that the algorithm does not explicitly generate a roll-rate command; instead, roll is left as a free parameter to be determined by the autopilot to achieve the desired yaw-rate command.

- While the path-following algorithm for *multirotors* produces roll $q_c(t)$, pitch $q_c(t)$, and yaw rate $r_c(t)$ commands, as well as a total thrust command $T_c(t)$:
$$\mathbf{u}_{PF}(t) := [p_c(t), q_c(t), r_c(t), T_c(t)]^\top.$$

Finally, the autopilot takes in these commands and transforms them into lower level commands such as throttle $\delta_{thr}(t)$, rudder $\delta_r(t)$, aileron $\delta_a(t)$, and elevator $\delta_e(t)$ deflections in the case of fixed-wing aircraft; and angular speed of propellers $\Omega_i(t)$ in the case of multirotors, where $i \in \{1, 2, \ldots, n_p\}$ and n_p is the total number of propellers.

B. Stability Guarantees

In order to prove the performance guarantees of the path-following algorithms for the two types of aircraft in discussion, the authors assume that each vehicle is assigned a trajectory that does not violate the dynamic constraints of the vehicle[e]. In [14,17], the authors prove that, given an ideal inner-loop tracking performance (the autopilot is capable of tracking $\mathbf{u}_{PF}(t)$ perfectly), these algorithms are locally exponentially stable with a guaranteed rate of convergence and a known domain of attraction ($\kappa_\infty = 0$). Whereas, for a non-ideal inner-loop tracking the path-following algorithms are locally uniformly ultimately bounded ($\kappa_\infty \neq 0$).

[d] Notice that the error vector is defined as a function of actual time t as opposed to mission time t_d used during trajectory generation. Section VII describes the relationship between actual (clock) time and mission time.

[e] In our framework, we guarantee constraint satisfaction through the trajectory-generation algorithm.

VII. Time Coordination

In this section, we provide a brief overview of the relative temporal constraint based algorithm originally presented in [17, 18]. The purpose of this section is to show how time-coordination algorithms fit into our framework useful for atmospheric science missions.

The key idea behind time coordination is to define a coordination variable γ_i for all vehicles $i \in \{1, 2, \ldots, N\}$ that maps the actual time to the desired mission time. Mathematically, this can be expressed as

$$\gamma_i : \mathbb{R}^+ \to [0, T_d], \quad \forall i \in \{1, \ldots, N\}\,.$$

The *commanded position* for vehicle i at time t is then defined as $\mathbf{p}_{c,i}(t) := \mathbf{p}_{d,i}(\gamma_i(t))$. Thus, at any time t, the parameter $\gamma_i(t)$ defines how far in time, the vehicle commanded position has progressed in its mission.

A. Objective

Focusing on relative temporal constraints, the objective of time-coordination algorithm is to ensure all vehicles are synchronized in the progression of their mission. Mathematically, we say that all vehicles are coordinated if the time-coordination error

$$\gamma_i(t) - \gamma_j(t) = 0, \quad \forall i, j \in \{1, \cdots, N\}\,, \quad i \neq j. \tag{1}$$

Furthermore, if

$$\dot{\gamma}_i(t) = 1, \quad \forall i \in \{1, \cdots, N\}\,, \tag{2}$$

then all vehicles are progressing along the mission at the desired speed.

B. Control Law

Let the evolution of $\gamma_i(t)$ be given by

$$\ddot{\gamma}_i = -b(\dot{\gamma}_i - 1) - a \sum_{j \in \mathcal{N}_i} (\gamma_i - \gamma_j) - \bar{\alpha}_i(\mathbf{e}_{PF,i})\,,$$
$$\gamma_i(0) = 0\,, \quad \dot{\gamma}_i(0) = 1\,,$$

where a and b are positive coordination control gains, while $\bar{\alpha}_i(\mathbf{e}_{PF,i})$ depends on the path-following error of the i-th vehicle.

The time-coordination algorithms assume typical communication capabilities, as originally introduced in [19]. If those assumptions are met, and the autopilot exhibits a non-ideal tracking performance, then the time-coordination error is ultimately bounded. Whereas, if the autopilot exhibits an ideal performance, the objectives of the time-coordination algorithm, Equations (1) and (2), are met exponentially with a guaranteed rate of convergence.[14, 18, 20]

VIII. Simulations

In this section, we provide a simulation for an atmospheric science mission that is of interest to NASA and the SMD.

A. Mission Description

We consider a mission where a scientist is interested in collecting correlative data around a source of a pollutant gas. To make the mission challenging, we assume that the location of the source of the pollutant is unknown and has to be located by flying vehicles in a search pattern over a region of interest. We assume this region of interest to be a convex polygonal area. Thus, the following steps need to be performed:

1. Divide the search area between the available vehicles.

2. Find trajectories for each vehicle to *i*) reach the search area, and then *ii*) fly in a search pattern.

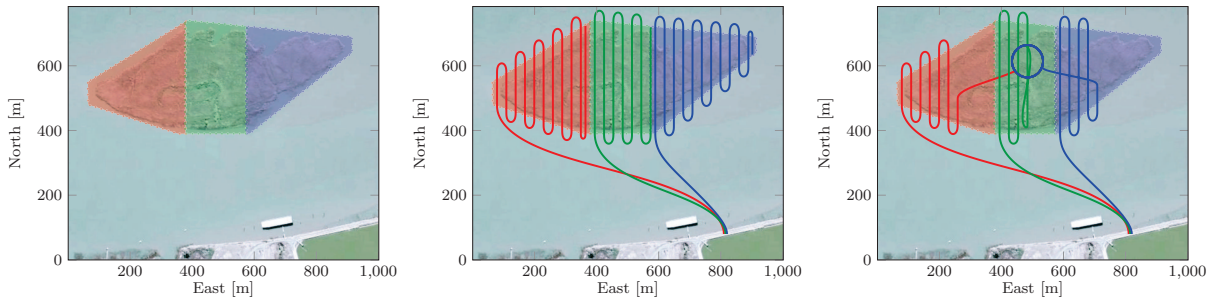

Figure 5: Left panel shows the island divided between the three vehicles for search. Middle panel shows the trajectories planned for the three vehicles to conduct the search mission. Right panel shows the trajectories used in the mission that involves replanning once the source of the pollutant has been located.

3. Once the source is located, replace the remaining search pattern with trajectories that allow each vehicle to *i*) reach the correlative data collection region, and then *ii*) fly in a spiral.

We assume to have 3 multirotor vehicles for the mission, that can fly at a maximum speed of 10 m/s and a maximum acceleration of $10\,\mathrm{m/s^2}$. Furthermore, we assume various obstacles to be present in the region, where a distance of $d_{\text{safe}} = 1\,\mathrm{m}$ is required to avoid them. Lastly, we require the trajectories to be at least C^1-continuous for good path-following performance.

B. Dividing the Search Region

The polygonal search region can be easily divided into three parts with equal area using parallel dividers. Finding the location of parallel dividers is trivial, therefore, we only show the results in Figure 5 (left panel) where the red, green and blue shaded regions show the search region for vehicle 1, 2, and 3 respectively. Here, we have divided the region using vertical dividers which is an arbitrary choice.

Notice that the use of parallel dividers ensures that all vehicles can reach their respective search regions without entering into a different vehicle's search region. Furthermore, if the vehicles leave their own search area along the North-South direction (as compared to East-West direction), they will not end up entering into the search region of a different vehicle. As we will see later, this enables separation between vehicles during the search operation.

C. Trajectory Generation for Search Mission

Initially, we need trajectories for the search part of the mission. Within the search mission part, we need vehicle trajectories to *i*) reach the search area and then *ii*) fly a Boustrophedon-like trajectory. The designed trajectories are shown in Figure 5.

To reach the search area, we use our optimization-based trajectory-generation method which ensures that all vehicles satisfy their dynamic constraints and maintain separation. We generate the trajectories such that all vehicles reach their search region from below with a final velocity of 5 m/s in the North-South direction. As we will shortly point out, search patterns for all vehicles begin at a velocity of 5 m/s in the North-South direction. Thus, C^1-continuity is maintained while the vehicles transition to the search pattern.

Once the curve that gets all the vehicles to the search region is available, we use the method discussed in Section IV-B to find Boustrophedon-like trajectories with $\mathbf{v}_{\text{nom}} = 5\,\mathrm{m/s}$. Notice that this geometric pattern ensures that vehicles fly either in their own search region, or slightly North or South of it. Thus, we are guaranteed separation between vehicles by design.

D. Trajectory Generation for Correlative Data Collection

Once the source of the pollutant gas has been located, we need to perform trajectory generation again. This time, we need trajectories for the vehicles to *i*) reach the data collection region and then *ii*) fly in a spiral pattern. The designed trajectories are shown in the rightmost panel of Figure 5.

To reach the data collection region, we use our optimization-based trajectory-generation method. We use the current position and velocity of all vehicles for the initial position and velocity constraint. Whereas, the final position of vehicles is chosen to be in a circle of radius 50 m around the pollutant source, separated

by 120 deg each. The final velocity is chosen such that the vehicles start the pattern with a velocity of 5 m/s moving in a clockwise motion[f].

For spiral pattern generation, we use the method detailed in [21]. Notice that separation between vehicles is guaranteed since vehicles are offset by an angle of 120 deg.

In order to show satisfaction of vehicle dynamic constraints, we show the speed requirement for each vehicle according to the planned trajectories in the top panel of Figure 6. Furthermore, we show the the separation between each vehicle in the bottom panel of Figure 6. In both panels, the shaded regions mark different phases of the mission: pink for reaching the search area; blue for searching the region; yellow for reaching the correlative data collection region; and white for correlative data collection.

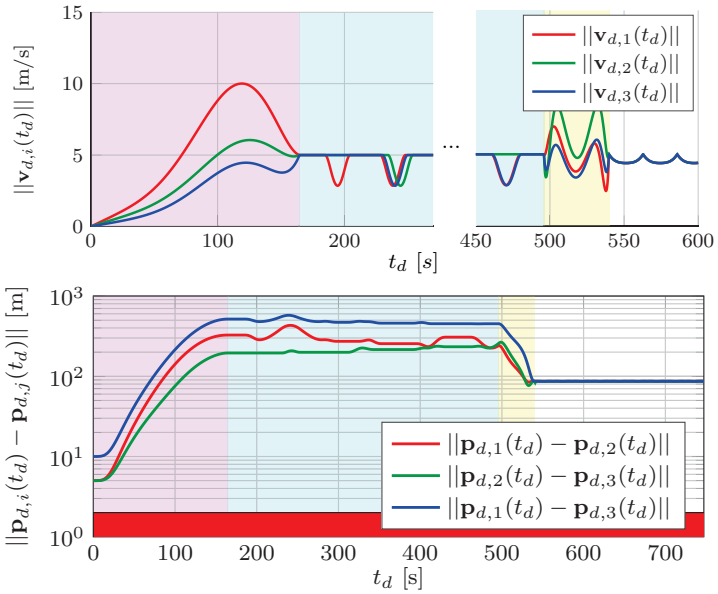

Figure 6: Top panel shows the desired speeds for all vehicles for the mission trajectories. Bottom panel shows the desired separation between the vehicles.

E. Collision Avoidance

We demonstrate collision avoidance using a static obstacle in the path of vehicle 1 before it reaches the search region. Using the procedure described in Section V, the vehicle replans its trajectory in order to avert the collision while modifying its trajectory only minimally as shown in Figure 7.

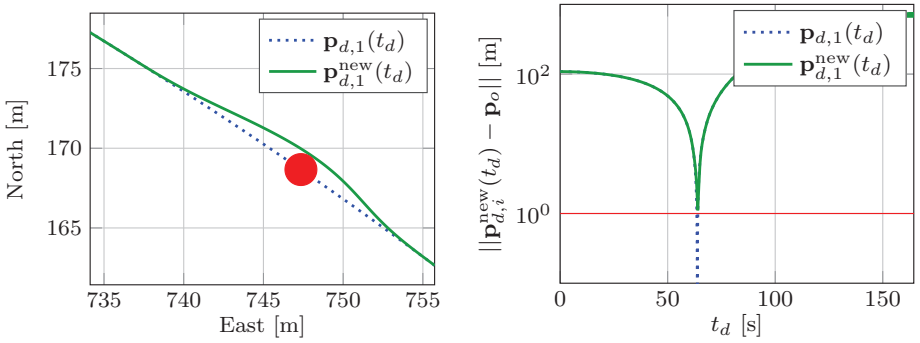

Figure 7: Panel on the left shows the initially planned trajectory along with the trajectory replanned for collision avoidance. Panel on the right shows the distance between the vehicle and the obstacle according to the original and replanned trajectory.

[f]Since the current position and velocity of the vehicles is chosen as the start of this curve, the vehicles can immediately shift from the search pattern to the curve while maintaining C^1-continuity. We currently ignore the time required for the trajectory replanning process which can be up to several seconds.

F. Path Following and Time Coordination

During mission execution, the vehicles use the path-following and time-coordination algorithms described in Sections VI and VII (and detailed in [14,15,17,18,20]). In order to show how these algorithms perform with the generated trajectories, we show the path-following and time-coordination errors in Figure 8, where it is clear that the path-following and time-coordination errors converge to a neighborhood of zero.

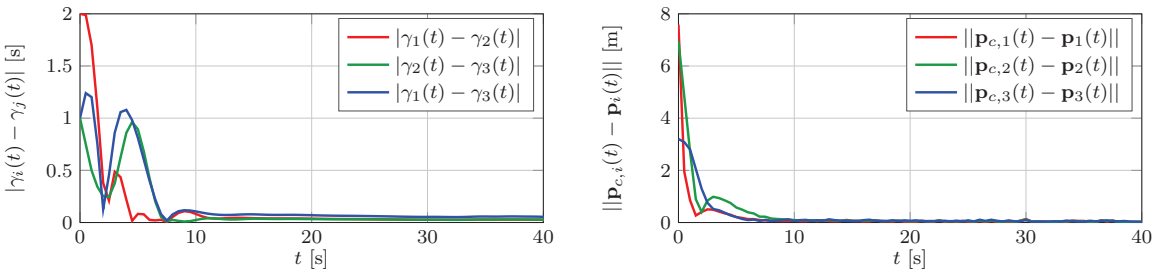

Figure 8: Top panel shows the time-coordination error between vehicles as the difference between the values of their coordination variable, whereas, the bottom panel shows the path-following error as the norm of the difference between commanded and actual position. For clearly showing the initial decrease in error, we limit the plot to the first 60s of the mission here. The error remains close to zero for the rest of the mission as well.

G. Real Flight Tests

Apart from simulations, we have also performed flight tests using multirotors. For these flight tests, we generated spiral trajectories for flying spiral trajectories in coordination. A picture showing a flight test in progress is shown in Figure 9.

Figure 9: Flight tests in progress.

IX. Conclusion

In this paper, we have presented a piecewise Bézier curve based framework for cooperative missions particularly suited for atmospheric science missions. We have discussed how properties of Bézier curves help trajectory generation and replanning while ensuring several desirable behaviors including smoothness, satisfaction of dynamic constraints, separation between vehicles and collision avoidance. We have discussed how path-following and time-coordination algorithms fit into our framework in order to achieve coordinated behavior. Lastly, we have provided a simulation example for a challenging science mission that required vehicles to fly to different destinations and fly several geometric patterns.

References

[1]M. Rosenzweig and C. R. Udry, "Forecasting profitability," National Bureau of Economic Research, Working Paper 19334, August 2013. [Online]. Available: http://www.nber.org/papers/w19334

[2]P. M. Cox, R. A. Betts, C. D. Jones, S. A. Spall, and I. J. T. Totterdell, "Acceleration of global warming due to carbon-cycle feedbacks in a coupled climate model," *Nature*, pp. 184–187, 2000.

[3]N. A. Rayner, D. Parker, E. Horton, C. Folland, L. V. Alexander, D. P. Rowell, E. C. Kent, and A. Kaplan, "Global analyses of sea surface temperature, sea ice, and night marine air temperature since the late nineteenth century," *Journal of Geophysical Research*, vol. 108, pp. 1–20, 2003.

[4]T. B. Ryerson, A. E. Andrews, W. M. Angevine, T. S. Bates, C. A. Brock, B. Cairns, R. C. Cohen, O. R. Cooper, J. A. de Gouw, F. C. Fehsenfeld, R. A. Ferrare, M. L. Fischer, R. C. Flagan, A. H. Goldstein, J. W. Hair, R. M. Hardesty, C. A. Hostetler, J. L. Jimenez, A. O. Langford, E. McCauley, S. A. McKeen, L. T. Molina, A. Nenes, S. J. Oltmans, D. D. Parrish, J. R. Pederson, R. B. Pierce, K. Prather, P. K. Quinn, J. H. Seinfeld, C. J. Senff, A. Sorooshian, J. Stutz, J. D. Surratt, M. Trainer, R. Volkamer, E. J. Williams, and S. C. Wofsy, "The 2010 California research at the nexus of air quality and climate change (CalNex) field study," *Journal of Geophysical Research*, vol. 118, pp. 5830–5866, 2013.

[5]S. S. Wegener, S. M. Schoenung, J. Totah, D. Sullivan, J. Frank, F. Enomoto, C. Frost, and C. Theodore, "UAV autonomous operations for airborne science missions," in *AIAA 3rd "Unmanned Unlimited" Technical Conference, Workshop and Exhibit, Infotech@Aerospace Conferences*, Chicago, IL, September 2004, AIAA 2004-6416.

[6]R. T. Farouki, *Pythagorean-Hodograph Curves*. Berlin Heidelberg: Springer-Verlag, 2008.

[7]J.-W. Chang, Y.-K. Choi, M.-S. Kim, and W. Wang, "Computation of the minimum distance between two Bézier curves/surfaces," *Computers & Graphics*, vol. 35, no. 3, pp. 677–684, June 2011.

[8]E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal of Robotics and Automation*, vol. 4, no. 2, pp. 193–203, April 1988.

[9]R. Choe, V. Cichella, E. Xargay, N. Hovakimyan, A. C. Trujillo, and I. Kaminer, "A trajectory-generation framework for time-critical cooperative missions," Boston, MA, August 2013, AIAA 2013-4582.

[10]R. Choe, J. Puig-Navarro, V. Cichella, E. Xargay, and N. Hovakimyan, "Trajectory generation using spatial Pythagorean Hodograph Bézier curves," Kissimmee, FL, January 2015, AIAA 2015-0597.

[11]S. B. Mehdi, R. Choe, and N. Hovakimyan, "Multiple collision avoidance through trajectory replanning using piecewise Bézier curves," in *54th IEEE Conference on Decision and Control*, Osaka, Japan, December 2015.

[12]A. P. Aguiar and J. P. Hespanha, "Position tracking of underactuated vehicles," vol. 3, June 2003, pp. 1988–1993.

[13]S. Al-Hiddabi and N. McClamroch, "Tracking and maneuver regulation control for nonlinear nonminimum phase systems: application to flight control," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 780–792, November 2002.

[14]V. Cichella, I. Kaminer, E. Xargay, V. Dobrokhodov, N. Hovakimyan, A. Aguiar, and A. António M. Pascoal, "A Lyapunov-based approach for time-coordinated 3D path-following of multiple quadrotors," in *2012 IEEE 51st Annual Conference on Decision and Control (CDC)*, 2012, pp. 1776–1781.

[15]V. Cichella, E. Xargay, V. Dobrokhodov, I. Kaminer, A. M. Pascoal, and N. Hovakimyan, "Geometric 3D path-following control for a fixed-wing UAV on SO(3)," in *AIAA Conference of Guidance, Navigation and Control conference*, 2011.

[16]I. Kaminer, A. M. Pascoal, E. Hallberg, and C. Silvestre, "Trajectory tracking for autonomous vehicles: An integrated approach to guidance and control," vol. 21, no. 1, pp. 29–38, JanuaryFebruary 1998.

[17]E. Xargay, "Time-critical cooperative path-following control of multiple unmanned aerial vehicles," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana-Champaign, Illinois, USA, August 2013.

[18]V. Cichella, I. Kaminer, V. Dobrokhodov, E. Xargay, R. Choe, N. Hovakimyan, A. P. Aguiar, and A. M. Pascoal, "Cooperative path-following control of multiple multirotors over faulty networks," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 945–957, July 2015.

[19]M. Arcak, "Passivity as a design tool for group coordination," vol. 52, no. 8, pp. 1380–1390, August 2007.

[20]V. Cichella, R. Choe, S. B. Mehdi, E. Xargay, N. Hovakimyan, V. Dobrokhodov, and I. Kaminer, "Trajectory generation and collision avoidance for safe operation of cooperating UAVs," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2014.

[21]Y. J. Ahn and H. O. Kim, "Approximation of circular arcs by Bézier curves," *Journal of Computational and Applied Mathematics*, vol. 81, pp. 145–163, 1997.