Indoor formation motion planning using B-splines parametrization and evolutionary optimization

Vincent Marguet, Cong Khanh Dinh, Florin Stoican, Ionela Prodan

Indoor formation motion planning using B-splines parametrization and evolutionary optimization

Vincent Marguet[a], Cong Khanh Dinh[a], Florin Stoican[b] and Ionela Prodan[a]

[a]Univ. Grenoble Alpes, Grenoble INP[†], LCIS, F-26000, Valence, France.
[†] Institute of Engineering and Management University Grenoble Alpes.
{vincent.marguet, cong-khanh.dinh, ionela.prodan}@lcis.grenoble-inp.fr,
[b]Faculty of Automatic Control and Computers, Politehnica University of Bucharest, Splaiul Independenţei nr. 313, sector 6, Bucharest, RO-060042, Romania florin.stoican@upb.ro

Abstract

Formation generation with connectivity maintenance under efficiency restrictions for a group of autonomous vehicles is a challenging problem. By planning trajectories offline, the vehicles can follow optimized paths, resulting in improved efficiency in terms of time, energy, and resource utilization. This paper introduces a coherent approach that leverages evolutionary computing, notably a differential evolutionary algorithm, along with B-spline parametrizations, to effectively coordinate multiple indoor nanodrones. Off-line trajectories for both the leader and followers are designed to enforce multiple constraints (i.e., position, velocity, angles, thrust, angular velocity, waypoint passing, obstacle avoidance). The proposed approach accommodates intricate maneuvers such as formation switching and obstacle avoidance, facilitated by a knot refinement procedure that minimizes conservatism in constraint enforcement. The theoretical results are validated in both simulation and experiments.

Keywords: Trajectories generation, Formation planning, B-spline curves, Knot refinement, Differential Evolution (DE), Multicopters.

1. Introduction

Motivation and related works: The applications of aerial drones, particularly multicopters, are steadily expanding, with increasing utilization in various sectors. They have been deployed in agriculture [1], infrastructure inspection [2], as well as in entertainment fields such as cinematography [3]. In certain applications, it has been demonstrated that the utilization of multiple drones flying in formation is advantageous from an economic standpoint. For example, in precision agriculture applications, it has been observed that the deposition amount of spray is greater when employing close formation drone spraying compared to sequential spraying [4]. Additionally, drones are frequently used for data collection and transmission tasks. Maintaining connectivity in this applications ensures the uninterrupted transmission of data from drones to ground stations or remote servers, enabling real-time analysis and decision-making [5], [6]. Hence, across a wide range of aerial drones applications, maintaining connectivity is essential for ensuring safe, reliable, and effective operation [7], [8].

Overall, the combination of nonlinear dynamics inherent to multicopters, high dimensionality, real-time requirements, and connectivity constraints renders the optimization problem for trajectory generation with operating and connectivity constraints highly challenging to solve. In the literature, both online and offline procedures have been proposed to address this issue.

The online approach [2],[9] offers the advantage of adapting trajectories in real-time by utilizing onboard components such as Light Detection and Ranging (LiDAR) technologies to assess the agent's actual position and its surrounding environment. However, online trajectory computation must be performed rapidly, with limited data and expensive onboard technologies. These additional constraints may restrict the number of considerations in solving the problem online.

Generating constrained trajectories offline offers several advantages [10, 11]:

i) it allows for more computationally intensive optimization algorithms to be employed, as there is no real-time constraint, leading to improved trajectory solutions;

ii) it allows the incorporation of complex environmental and mission constraints, which may be challenging to address in real-time scenarios;

iii) it provides the opportunity for trajectory refinement and post-processing, allowing for the optimization of additional objectives or the adjustment of trajectories based on updated information.

Various works address the offline trajectories generation. For example, [12] propose deforming at run time spline trajectories pre-computed offline to accommodate changing environments and objectives. Note that, the trajectory computed during the optimization process needs to be stored, hence requiring a representation by a set of points. The simplest approach to obtain these points is to designate waypoints through which the drones must pass, as demonstrated in previous studies such as [1], where drones traverse straight lines between waypoints representing farm blocks. However, obstacles may exist between consecutive waypoints, necessitating a more sophisticated trajectory definition. A promising solution involves combinations of control points and specific functions that weight the importance of these points throughout the trajectory. Among these solutions, Bezier curves [13] and B-spline curves [14], [15], [16] are commonly utilized. It is noteworthy that these two types of curves are interconnected, and [17] demonstrate that a 2D B-spline curve can be locally converted into a Bezier curve with additional control points, offering the advantage of tighter convex hulls formed by the control points. [18] solve an optimization problem using precomputed B-spline basis functions and optimizing the control points coordinates to generate a feasible trajectory for a single multicopter passing through predifined waypoints. In the aforementioned study [16], B-spline curves describe the environment perceived by the LiDAR system integrated into the agent, underscoring the utility of such curves for representing complex environments. Indeed, B-spline curves possess numerous mathematical and geometrical properties that facilitate the design of smooth trajectories while avoiding obstacles, owing to the localized importance of each control point. These properties will be elaborated upon in the subsequent section.

Contributions: This study builds upon our preliminary results, where an optimization-based algorithm was developed for single multicopter trajectory generation, as outlined in [19]. The optimization problem minimized the input effort while satisfying various operational constraints. In this paper, we extend our research to address formation motion planning for multiple multicopters, with experimental validation. In addition to constraints on positions, velocities, angles, thrust, angular velocities, and waypoint passing, we introduce obstacle avoidance constraints and inter-agent conditions. These include communication constraints among multiple agents and the requirement to maintain and switch formations during flight. Unlike the connectivity maintenance approach proposed in [7], which involves solving Line of Sight constraints using Mixed Integer Programming (MIP) - known for its high computational complexity [20] - our contribution proposes a coherent integration of B-splines properties, knot refinement, and Differential Evolution (DE) computation.

Thus, our contribution addresses an inherently non-convex problem that traditional solvers struggle to solve. The proposed algorithm generates offline trajectories for multiple nanodrones while ensuring compliance with their dynamical constraints, waypoint passing, communication range, and formation switching. By leveraging the properties of B-splines, we efficiently formulate the cost in the evolutionary algorithm, leading to improved computation time and feasibility, when compared to existing methods [18]. In summary, we achieve the following:

i) solve a non-convex optimization problem with less conservative properties by leveraging knot refinement in the B-splines trajectories parameterization. This approach allows us to tighten the convex hulls containing the curve in the constraints formulation, leading to more accurate and efficient solutions;

ii) design an offline motion planning algorithm capable of generating multiple trajectories that adhere to various constraints, including unmanned aerial vehicle dynamics, waypoint constraints, communication constraints, forma-

tion constraints, and obstacle avoidance constraints;

iii) conduct a comparative analysis of the results obtained from our algorithm with those from other methods documented in the literature [18]. This comparison provides valuable insights into the effectiveness and efficiency of our approach;

iv) validate our proposed trajectories through experiments involving multiple indoor nanodrones, demonstrating the feasibility and practicality of our approach in real-world scenarios (https://tinyurl.com/CEPmarguetVID).

Organization: Section 2 provides a review of essential definitions and properties of the B-spline curve, knot refinement, introduces the multicopter model utilized in this study, and outlines the problem of optimal trajectory generation. In Section 3, the algorithm for solving a motion planning problem aimed at connectivity maintenance is detailed. Section 4 demonstrates the advantages of this approach through experimental results and comparisons. Finally, Section 5 summarizes the findings and draws conclusions.

Notations: Throughout the paper, the following notations are employed: vectors and matrices are denoted in bold, T_f is the final time of the flight that starts at 0s; $\mathbf{z}_i, \dot{\mathbf{z}}_i, \ddot{\mathbf{z}}_i, \dddot{\mathbf{z}}_i, \mathbf{z}_i^{(4)}$ correspond to the trajectory of agent i and its derivatives; $\mathbf{z}$ refers to the trajectories of all the agents; n is the number of control points describing a B-spline curve; $\mathbf{B}_{d,\boldsymbol{\xi}}(t)$ are the B-spline basis functions of degree d and knot vector $\boldsymbol{\xi}$; τ_k is the $(k+1)$-th time instant of the knot vector $\boldsymbol{\xi}$; ${}^xP_k^i$ is the coordinate on x-axis of the $(k+1)$-th control point of agent i; $\mathbf{p}_\ell^{wp}$ is the position of the ℓ-th way-point associated to the time t_ℓ; the number of waypoints is N_{WP}; the transpose of a matrix $\mathbf{X}$ is denoted $\mathbf{X}^\top$.

2. Prerequisites and problem formulation

2.1. B-spline curves in trajectory generation

A B-spline curve is described as a linear combination of control points and B-spline basis functions. In our setting, the trajectory of an agent i is defined as a B-spline curve:

$$\mathbf{z}_i(t) = \sum_{k=0}^{n-1} \mathbf{P}_k^i B_{k,d,\boldsymbol{\xi}}(t) = \mathbf{P}^i \mathbf{B}_{d,\boldsymbol{\xi}}(t), \forall t \in [0, T_f], \quad (1)$$

with $\mathbf{P} = \begin{bmatrix}\mathbf{P}_0^i & \dots & \mathbf{P}_{n-1}^i\end{bmatrix}$ the matrix which gathers the control points, $\mathbf{B}_{d,\boldsymbol{\xi}}(t) = \begin{bmatrix}B_{0,d,\boldsymbol{\xi}}(t) & \dots & B_{n-1,d,\boldsymbol{\xi}}(t)\end{bmatrix}^\top$ the basis vector.

B-splines of degree up to $d \leq m-2$ are defined recursively over the knot sequence $\boldsymbol{\xi} = \{0 = \tau_0 \leq \tau_1 \leq ... \leq \tau_m = T_f\}$:

$$B_{k,1,\boldsymbol{\xi}}(t) = \begin{cases} 1, & t \in [\tau_k; \tau_{k+1}[\\ 0, & \text{otherwise,} \end{cases} \quad (2a)$$

$$B_{k,d,\boldsymbol{\xi}}(t) = \frac{t-\tau_k}{\tau_{k+d}-\tau_k} B_{k,d-1,\boldsymbol{\xi}}(t) + \frac{\tau_{k+d+1}-t}{\tau_{k+d+1}-\tau_{k+1}} B_{k+1,d-1,\boldsymbol{\xi}}(t). \quad (2b)$$

The following properties [21] hold:

P1) B-splines basis functions have a local support:

$$B_{k,d,\boldsymbol{\xi}}(t) = 0, \quad \forall t \notin [\tau_k; \tau_{k+d+1}). \quad (3)$$

P2) B-splines basis functions partition the unity:

$$\sum_{k=0}^{n-1} B_{k,d,\boldsymbol{\xi}}(t) = 1, \quad \forall t \in [\tau_0; \tau_m] \quad (4a)$$

and

$$B_{k,d,\boldsymbol{\xi}}(t) \geq 0, \quad \forall t \in [\tau_0; \tau_m]. \quad (4b)$$

P3) The 'r' order derivatives of B-spline basis functions are linear combinations of B-splines of lower degree, i.e. there exists a matrix $\mathbf{M}_{d,d-r}$ such that:

$$\mathbf{B}_{d,\boldsymbol{\xi}}^{(r)}(t) = \mathbf{M}_{d,d-r} \mathbf{B}_{d-r,\boldsymbol{\xi}}(t). \quad (5)$$

P4) $B_{k,d,\boldsymbol{\xi}}(\tau_l) \in C^{d-\mu_l}$ at $\tau_l \in \boldsymbol{\xi}$ with multiplicity μ_l and C^∞ otherwise. A function of class C^k is a function that has a k-th derivative that is continuous in its domain.

P5) The B-spline curve (1) lies within the union of all convex hulls defined by all subsets of $d+1$ consecutive control points.

P6) On each non-empty sub-interval $[\tau_\ell, \tau_{\ell+1})$, the basis functions of degree $d-1$ can be expressed as combination of the basis functions of degree d, thus, the differentiated B-spline curve may be expressed as:

$$\dot{\mathbf{z}}_i(t) = \dot{\mathbf{P}}^\mathbf{i} \mathbf{B}_{d-1,\boldsymbol{\xi}}(t) = \dot{\mathbf{P}}^\mathbf{i} \mathbf{D}_{d-1,d}^\ell \mathbf{B}_{d,\boldsymbol{\xi}}(t), \ t \in [\tau_\ell, \tau_{\ell+1}), \quad (6)$$

with $\mathbf{D}^{\ell}_{d-1,d}$ computed accordingly. Such a matrix always exist since any polynomial of degree $d-1$, $\dot{\mathbf{z}}_i(t)$ on the sub-interval $[\tau_\ell, \tau_{\ell+1})$, can be described by polynomials of degree d, the B-splines functions $\mathbf{B}_{d,\boldsymbol{\xi}}(t)$, again on the sub-interval $[\tau_\ell, \tau_{\ell+1})$.

Remark 1. Given P5), the end points of the trajectory may be imposed directly through the choice of first and last control points iff the first and last knot elements have multiplicity $d+1$:

$$\boldsymbol{\xi} = \{0 = \tau_0 = \ldots = \tau_d < \tau_{d+1} < \ldots \\ \ldots < \tau_{m-d-1} < \tau_{m-d} = \ldots = \tau_m = T_f\}. \quad (7)$$

For further use, note that we take $n = m - d$ the number of control points.

In [3], a non-uniform knot vector is used to solve the trajectory generation problem: the knot elements themselves are decision variables in the optimization problem, i.e., the trajectory is cut into smaller pieces and continuity between these pieces (and their derivatives) has to be ensured. This is cumbersome and greatly increases the complexity of the problem. On the other hand, taking an uniform knot sequence allows to have an easier formulation and faster computation as most of the calculations can be performed once, prior to the resolution of the optimization problem. Hence, hereinafter, the knot vector is not only clamped (as defined in (7)) but also uniformly spaced, i.e., $\tau_{\ell+1} - \tau_\ell = T_f/(n-d), \forall \ell \in \{d, \ldots, m-d-1\}$. ♦

Lastly, we emphasize the geometric insight of the convex hull property P5) through the illustration in Figure 1. The fourth control point, $\mathbf{P}_3$, is moved along a segment (the dotted line), which influences only locally (over support $[\tau_3, \tau_8)$ the curve's shape (light gray curves) and the union of convex hulls containing them (dashed light gray contours).

The point of practical importance is that, once the parameters of the B-spline basis functions are fixed, the resulting B-spline curve depends only on its control points. Thus, the trajectory planning problem is reduced to finding the optimal coordinates of the control points.

2.2. Knot refinement for B-splines

Introducing a new time instant λ into the original knot vector $\boldsymbol{\xi}$ is called knot insertion. The resulting refined knot vector becomes $\bar{\boldsymbol{\xi}} = \{\tau_0 < \ldots < \tau_j < \lambda < \tau_{j+1} < \ldots < \tau_m\}$ and if the control points corresponding to the refined family of B-splines $\{\bar{B}_{k,d,\bar{\boldsymbol{\xi}}}(t)\}_{k=1,\ldots,\bar{n}}$, induced by $\bar{\boldsymbol{\xi}}$, are chosen as

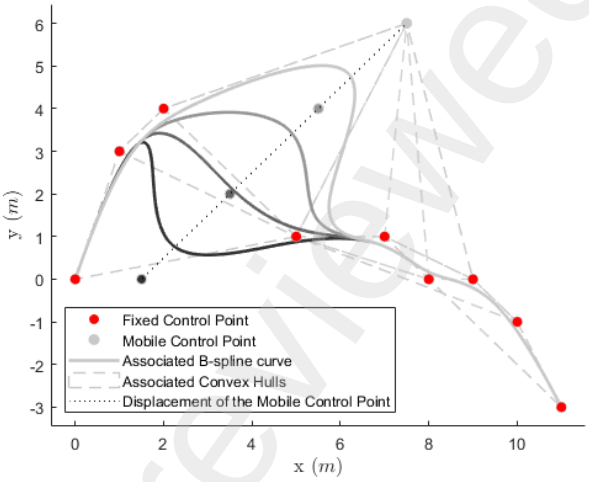


Figure 1: Consequence of the displacement of one control point on the B-spline curve. The convex hulls composed of $d+1$ consecutive control points are bounding the B-spline curve.

$$\mathbf{Q}^i_k = \alpha_k \mathbf{P}^i_k + (1-\alpha_k)\mathbf{P}^i_{k-1} \quad (8)$$

$$\text{with } \alpha_k = \begin{cases} 1, & k \le j-d, \\ \dfrac{\lambda - \tau_k}{\tau_{k+d} - \tau_k}, & j-d+1 \le k \le j, \\ 0, & k \ge j+1, \end{cases}$$

the original B-spline curve (1) composed of n control points may also be expressed in function of $\bar{n} = n+1$ control points:

$$\mathbf{z}_i(t) = \sum_{k=0}^{n-1} \mathbf{P}^i_k B_{k,d,\boldsymbol{\xi}}(t) = \sum_{k=0}^{\bar{n}-1} \mathbf{Q}^i_k \bar{B}_{k,d,\bar{\boldsymbol{\xi}}}(t), \quad \forall t \in [0, T_f]. \quad (9)$$

The advantage of increasing the number of control points is that the union of convex hulls bounding the curve is guaranteed to become tighter [22, Corollary 1] at a reasonable increase in complexity (more control points are considered but they depend linearly on the initial ones, i.e., the number of decision variables in a subsequent optimization problem does not increase).

An illustration of this idea is shown in Figure 2 where a knot refinement (repeated knot insertion) procedure is employed.

2.3. Multicopter model

The quadcopter model has been fully described in [23]. The absolute linear position of the quad-

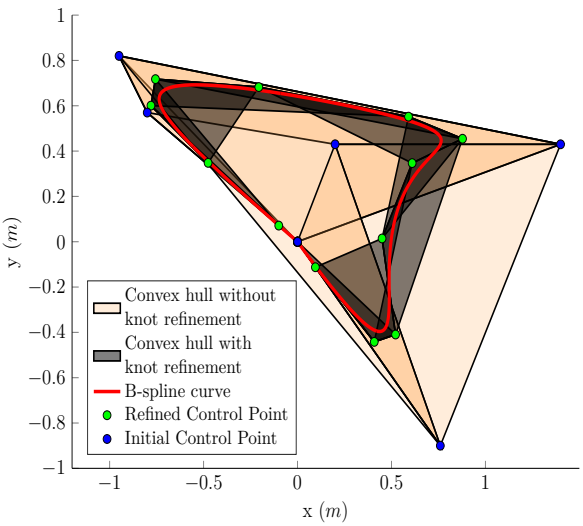


Figure 2: Impact of knot refinement on the the conservativeness of the convex hull property. The same curve (in red) is defined by 11 control points (blue) and 18 control points (green).

copter is defined in the inertial frame x, y, z-axis with $\boldsymbol{\xi} = \begin{bmatrix} x & y & z \end{bmatrix}^\top$. The attitude is defined in the same inertial frame with the three Euler angles $\boldsymbol{\eta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top$. In the body frame, the angular velocities are denoted by $\boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^\top$. The rotation matrix from the body frame to the inertial frame $\mathbf{R}_B^W$ is composed of 3 columns denoted as $\mathbf{x}_B$, $\mathbf{y}_B$ and $\mathbf{z}_B$. The transformation matrix for angular velocities from the inertial frame to the body frame is $\mathbf{N}(\boldsymbol{\eta})$.

$$\mathbf{N}(\boldsymbol{\eta}) = \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix}, \tag{10a}$$

$$\mathbf{R}_B^W = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\theta s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\theta c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}. \tag{10b}$$

where c stands for cos and s stands for sin.
Taking T the normalized thrust, g the gravitational acceleration and $\mathbf{z}_w = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\top$ the world frame's z-axis, the following relations stand:

$$\ddot{\boldsymbol{\xi}} = T\mathbf{z}_B - g\mathbf{z}_w, \tag{11a}$$

$$\dot{\boldsymbol{\eta}} = N^{-1}(\boldsymbol{\eta})\boldsymbol{\omega}. \tag{11b}$$

The model considered in terms of inputs $\mathbf{u} = \begin{bmatrix} T & p & q & r \end{bmatrix}^\top$ and states $\mathbf{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^\top$ is taken from [18] as a nine-state quadcopter model that neglects aerodynamic effects and external disturbances.

2.4. Optimal trajectories generation

The constrained optimization problem for trajectory generation for quadcopter dynamics such as in (11) is usually formulated as follows. Similar constructions appear in the literature [24].

$$\min_{\mathbf{z}(t)} \int_0^{T_f} \mathcal{E}(\mathbf{z}(t))\mathrm{d}t \tag{12a}$$

$$\text{s.t.} \quad \mathbf{z}_i(t) \subset \mathbf{K}_p,\ 0 \le t \le T_f,\ \forall i, \tag{12b}$$

$$v_i(t) \le v_{max},\ 0 \le t \le T_f,\ \forall i, \tag{12c}$$

$$|\phi_i(t)| \le \epsilon,\ |\theta_i(t)| \le \epsilon,\ 0 \le t \le T_f,\ \forall i, \tag{12d}$$

$$T_{min} \le T_i(t) \le T_{max},\ 0 \le t \le T_f,\ \forall i, \tag{12e}$$

$$|p_i(t)| \le \omega_{max},\ |q_i(t)| \le \omega_{max},\ 0 \le t \le T_f, \forall i, \tag{12f}$$

$$\mathbf{p}_i(0) = \mathbf{p}_0^i,\ \mathbf{v}_i(0) = \mathbf{0}_{3\times 1},\ \mathbf{a}_i(0) = \mathbf{0}_{3\times 1},\ \forall i, \tag{12g}$$

$$\mathbf{p}_i(T_f) = \mathbf{p}_f^i,\ \mathbf{v}_i(T_f) = \mathbf{0}_{3\times 1},\ \mathbf{a}_i(T_f) = \mathbf{0}_{3\times 1},\ \forall i, \tag{12h}$$

$$\|\mathbf{z}_1(t_\ell) - \mathbf{p}_\ell^{wp}\|_2 \le d_{wp},\ \ell = 1, \ldots, N_{WP}, \tag{12i}$$

$$\mathbf{z}_j(t) - \mathbf{z}_i(t) = \boldsymbol{\Delta}^{ij,h},\ t_0^h \le t \le t_f^h,\ \forall i, j, h \tag{12j}$$

$$d_{ij}(t) \le \rho,\ \forall (i,j) \in \mathcal{L},\ 0 \le t \le T_f, \tag{12k}$$

$$\mathbf{z}_i(t) \notin \cup Obs,\ 0 \le t \le T_f,\ \forall i, \tag{12l}$$

where:

- (12a) corresponds to a cost function to minimize (trajectory length, the energy spent or the time to achieve a mission);
- (12b) and (12c) describe the position (the trajectories have to stay in a bounded box $\mathbf{K}_p$ representing the indoor environment) and velocity constraints;
- (12d) and (12f) consider the angle and angular velocity constraints;
- (12e) denotes the thrust constraint;
- (12g) and (12h) define initial and terminal restrictions;
- (12i) describes the waypoint constraint and (12l) is the obstacle avoidance constraint;

- (12j) define the formation configuration constraints: the position difference on each axis between agents i and j is $\mathbf{\Delta}^{i,j,h}$, i.e. the position difference corresponding to the h-th formation configuration during its associated time interval $[t_0^h, t_f^h]$;
- (12k) govern communication constraints (the distance between the agents in the pairs (i, j) from $\mathcal{L}$ (all the desired pairs of agents that have to communicate) must be inferior to the communication range ρ).

All the functions (many of which might be nonlinear) appearing in (12) will be expressed in terms of the multicopter trajectory and its derivatives. Taking the trajectory as a B-spline curve (1), a priori fixing the knot vector and the degree of the basis functions and, where needed, using over-approximations such as P5), the continuous-time (and hence, infinite dimensional) optimization problem (12) is recast into a finite-dimensional, sub-optimal optimization problem which depends only on the control points.

Unfortunately, while tools such as Yalmip [25] to pre-process the problem and solvers such as MOSEK [26] to retrieve the solution exist and are quite mature, the underlying issues of nonlinearity in either/both the cost and constraints remain. The question arises whether expensively searching for an optimal solution is worth the hassle in the first place. In practice we would be content with a sub-optimal solution if it can be obtained significantly faster and if additional post-processing can show it is feasible to the original problem.

An approach that allows to handle efficiently the constraints and objective functions even when nonlinear, is population algorithms like Particle Swarm Optimization (PSO), [27, 28, 29] or Differential Evolution (DE), [30]. Both of them are minimizing a global objective function that takes into account the constraints as penalty terms. The general idea of these population algorithms is to compute many trajectories (candidate solutions), compute their associated cost and update them iteratively until a threshold (number of iteration, convergence to a local minimum) is achieved, at which point, the best candidate is selected as the solution. Beyond the variations of a particular method, the basic idea is to penalize constraint violation such as to 'push' the solution towards (at least) a local minimum. The disadvantage, as with all constraint relaxation methods is that there is no guarantee that the end result is feasible. Still, the solution can be checked in a post processing step and the entire procedure can be repeated for a different tuning of the parameters. Since computation time strongly depends on population size and choices made in the constraint description, the user has much leeway in deciding what is the best strategy in obtaining a 'good-enough' solution. Lastly, it should be stated that it is surprisingly easy to arrive to an optimization which is prohibitively-difficult to solve by classical methods and that heuristic methods such as PSO or DE may prove to be the only option.

3. Connectivity maintenance problem

The motion planning scheme for a formation composed of a leader and its followers is twofold:

i) First, an optimal trajectory for the leader is generated while respecting the dynamical, space, waypoint, the initial and final constraints of (12) using the differential evolution algorithm.

   Once the leader's trajectory is obtained, we carry a knot refinement procedure to increase the degrees of liberty for the trajectories of the followers. Note, that the leader's trajectory remains unchanged, it is the followers that have to adapt.

ii) Second, for the followers, we keep the above constraints (except the waypoint constraint, since their movement is relative to the aforecomputed leader's trajectory) while, additionally, satisfying formation configuration and communication constraints.

Both problems of the motion planning mechanism are using a differential evolution algorithm, as mentioned in Section 2.

3.1. Trajectory planning for the leader

The objective function is given as a sum of weighted penalty terms, each of them expressed in function of variable $\mathbf{X} = \begin{bmatrix}\mathbf{P}_0^1 & \mathbf{P}_1^1 & \dots & \mathbf{P}_{n-1}^1\end{bmatrix}$ which gathers the control points (subsequently denoted by upper script '1') defining the leader's trajectory as in (1):

$$F^{\mathrm{L}}(\mathbf{X}) = \sum_{q=0}^{7} \gamma_q F_q(\mathbf{X}), \tag{13}$$

where $F_0(\mathbf{X})$ penalizes the input effort, $F_1(\mathbf{X})$ the position, $F_2(\mathbf{X})$ the velocity, $F_3(\mathbf{X})$ the roll and pitch angles, $F_4(\mathbf{X})$ the thrust, $F_5(\mathbf{X})$ the angular velocity, $F_6(\mathbf{X})$ the collision with the obstacles, $F_7(\mathbf{X})$ the distances between the waypoints and the position of the leader at the waypoints times and with $\gamma_0, \ldots, \gamma_7$ the weights associated to each of these penalties.

We detail now the expression of each of these penalties corresponding to the constraints (12b)–(12i). As mentioned earlier, we are exploiting B-spline constructions and their properties to arrive at expressions which involve the control points. The reader is invited to take a look to [18] for the proofs to obtain constraints on position, velocity, angle, thrust, angular velocity and waypoint passing. We will obtain less conservative expressions for the angular and angular velocity constraints (proofs in Appendices 6.1 and 6.2) and convert all the constraints in penalty terms that will penalize a trajectory if the constraint is not satisfied.

There are multiple ways in which the penalty may be enforced. Hereinafter we propose to define σ: $\mathbb{R}^n \to \mathbb{R}_+$ such that

$$\sigma(\mathbf{x}) = \sum_{i=1}^{n} \max(0, x_i). \tag{14}$$

Noting that any of the considered constraints may be put into form $h(\mathbf{x}) \leq 0$, the ramp function $\sigma(h(\mathbf{x}))$ returns zero whenever the constraint is verified and the 1-norm of the constraint violation components (those terms $h_i(x_i) \geq 0$), when not.

3.1.1. Minimizing the input effort

The objective (12a) of the optimization problem is to minimize the input effort which corresponds to the trajectory's snap (its fourth-order derivative). Using P3), the derivative property of the B-spline functions, the trajectory's snap may be expressed in terms of control points:

$$\begin{aligned} F_0(\mathbf{X}) &= \min_{\mathbf{z}_1(t)} \int_0^{T_f} \left\| \mathbf{z}_1^{(4)}(t) \right\|_2^2 \mathrm{d}t \\ &= \int_0^{T_f} \left\| \sum_{k=0}^{n-5} (\mathbf{X}\mathbf{M}_{d,d-4})_k B_{k,d-4,\boldsymbol{\xi}}(t) \right\|_2^2 \mathrm{d}t. \end{aligned} \tag{15}$$

Remark 2. Through standard manipulations [21], (15) may be brought into a quadratic form where only control points appear. Furthermore, in the particular case of B-splines of degree 4, the snap function becomes a step function of amplitude $(\mathbf{X}\mathbf{M}_{d,d-4})_k$ and constant time step $T_f/(n-d)$. Thus, the cost associated to the snap of the trajectory becomes in this case simply

$$F_0(\mathbf{X}) = \frac{T_f}{n-d} \sum_{k=0}^{n-1} \|(\mathbf{X}\mathbf{M}_{d,d-4})_k\|_2^2.$$

♦

3.1.2. Space constraints penalization

As we are working in an indoor environment, the space is limited and restricted to a box $\mathbf{K}_p$ with lower and upper bounds on each axis, as per (12b). The convexity property of the B-spline curves implies that restricting the control points to be inside this box guarantees that the trajectory will be inside this box. Thus, the position penalty is expressed as

$$F_1(\mathbf{X}) = \sum_{k=0}^{n-1} \sigma\left(\mathbf{x}_{\min} - \mathbf{P}_k^1\right) + \sigma\left(\mathbf{P}_k^1 - \mathbf{x}_{\max}\right). \tag{16}$$

3.1.3. Velocity constraints penalization

For the velocity penalty, the total velocity has to be lower or equal to the maximum velocity v_{max}, as per (12c). The same reasoning leads to the following expression using the derivative control points. We remind that the respective derivative control points are obtained by multiplying the initial control points by the matrix appearing in property P3): $\dot{\mathbf{P}}_k^1 = (\mathbf{X}\mathbf{M}_{d,d-1})_k$, $\ddot{\mathbf{P}}_k^1 = (\mathbf{X}\mathbf{M}_{d,d-2})_k$, $\dddot{\mathbf{P}}_k^1 = (\mathbf{X}\mathbf{M}_{d,d-3})_k$. Thus,

$$F_2(\mathbf{X}) = \sum_{k=0}^{n-2} \sigma\left(\|\dot{\mathbf{P}}_k^1\|_2 - v_{\max}\right). \tag{17}$$

3.1.4. Roll and pitch constraints penalization

The roll and pitch angles should have a modulus inferior to ϵ, as per (12d). The associated penalty term is defined as (see the sufficient conditions (50), derived in Appendix 6.1):

$$F_3(\mathbf{X}) = \sum_{k=0}^{n-3} \sum_{i=0}^{n-3} \sigma\left(G(\epsilon, i, k, \mathbf{P}^1)\right) \tag{18}$$

with

$$\begin{aligned} G(\epsilon, i, k, \mathbf{P}) = \cot^2 \epsilon\, \ddot{\mathbf{P}}_i^\top \ddot{\mathbf{P}}_k - \left(1 + \cot^2 \epsilon\right) \ddot{\mathbf{P}}_i^\top \mathbf{z}_w \mathbf{z}_w^\top \ddot{\mathbf{P}}_k \\ - 2g \mathbf{z}_w^\top \ddot{\mathbf{P}}_k - g^2. \end{aligned} \tag{19}$$

3.1.5. Thrust constraints penalization

The thrust has to stay between its bounds $T_{\min}$ and $T_{\max}$, as per (12e). The expression of the associated penalty is:

$$F_4(\mathbf{X}) = \sum_{k=0}^{n-3} \sigma\left(\|\ddot{\mathbf{P}}_k^1 + g\mathbf{z}_w\|_2 - T_{\max}\right) + \sigma\left(-{}^z\ddot{P}_k^1 - g + T_{\min}\right), \quad (20)$$

as a reminder, $\mathbf{z}_w = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^\top$ is the world frame's z-axis.

3.1.6. Angular velocity constraints penalization

The angular velocities' magnitude should be bounded by $\omega_{\max}$, as in (56). The constraints are expressed using the second and third derivatives of the control points (see the sufficient conditions (61), derived in Appendix 6.2):

$$F_5(\mathbf{X}) = \sum_{\ell=1}^{n-d} \sum_{k=\ell+d-2}^{\ell+d} \sum_{i=\ell+d-2}^{\ell+d} \sigma\left(S(\ell,i,k,\mathbf{P}^1)\right) \quad (21)$$

where

$$S(\ell,i,k,\mathbf{P}) = (\dddot{\mathbf{P}}\mathbf{D}^\ell_{d-3,d-2})_i^\top (\dddot{\mathbf{P}}\mathbf{D}^\ell_{d-3,d-2})_k - \omega_{\max}^2 \left(\ddot{\mathbf{P}}_i + \mathbf{z}_w g\right)^\top \left(\ddot{\mathbf{P}}_k + \mathbf{z}_w g\right). \quad (22)$$

3.1.7. Initial and final conditions

The initial and final conditions (12g) and (12h) are formulated for the position ($\mathbf{p}_0$ and $\mathbf{p}_f$), velocity ($\mathbf{v}_0$ and $\mathbf{v}_f$) and acceleration ($\mathbf{a}_0$ and $\mathbf{a}_f$). With Remark 1, the initial position and final position give the values of the first and last control points:

$$\mathbf{P}_0^1 = \mathbf{p}_0 \quad (23a)$$

$$\mathbf{P}_{n-1}^1 = \mathbf{p}_f \quad (23b)$$

The same reasoning applies to the initial velocity $\dot{\mathbf{z}}_\mathbf{1}(t=0) = \dot{\mathbf{P}}_0^1 = \mathbf{v}_0$, where noting the dependence between $\dot{\mathbf{P}}_0^1$ and $\mathbf{P}^i$, we have

$$\frac{d}{\tau_{d+1}}(\mathbf{P}_1^1 - \mathbf{P}_0^1) = \mathbf{v}_0 \quad (24)$$

which allows to express

$$\mathbf{P}_1^1 = \mathbf{P}_0^1 + \frac{\tau_{d+1}}{d}\mathbf{v}_0 = \mathbf{p}_0 + \frac{\tau_{d+1}}{d}\mathbf{v}_0. \quad (25)$$

A similar calculus starting from $\dot{\mathbf{z}}_\mathbf{1}(t=T_f) = \dot{\mathbf{P}}_{n-2}^1 = \mathbf{v}_f$ leads to

$$\mathbf{P}_{n-2}^1 = \mathbf{P}_{n-1}^1 - \frac{T_f - \tau_{n-1}}{d}\mathbf{v}_f = \mathbf{p}_f - \frac{T_f - \tau_{n-1}}{d}\mathbf{v}_f. \quad (26)$$

The same principle applied twice for the initial $\ddot{\mathbf{z}}_\mathbf{1}(t=0) = \ddot{\mathbf{P}}_0^1 = \mathbf{a}_0$ and final $\ddot{\mathbf{z}}_\mathbf{1}(t=T_f) = \ddot{\mathbf{P}}_{n-3}^1 = \mathbf{a}_f$ acceleration is applied and results in:

$$\frac{d-1}{\tau_{d+1}}\left(\frac{d(\mathbf{P}_2^1 - \mathbf{P}_1^1)}{\tau_{d+2}} - \frac{d(\mathbf{P}_1^1 - \mathbf{P}_0^1)}{\tau_{d+1}}\right) = \mathbf{a}_0, \quad (27a)$$

$$\frac{d-1}{T_f - \tau_{n-1}}\left(\frac{d(\mathbf{P}_{n-1}^1 - \mathbf{P}_{n-2}^1)}{T_f - \tau_{n-1}} - \frac{d(\mathbf{P}_{n-2}^1 - \mathbf{P}_{n-3}^1)}{T_f - \tau_{n-2}}\right) = \mathbf{a}_f, \quad (27b)$$

which, after further manipulation give the expressions of control points:

$$\mathbf{P}_2^1 = \mathbf{p}_0 + \frac{\tau_{d+1} + \tau_{d+2}}{d}\mathbf{v}_0 + \frac{\tau_{d+2}\tau_{d+1}}{d(d-1)}\mathbf{a}_0, \quad (28a)$$

$$\mathbf{P}_{n-3}^1 = \mathbf{p}_f - \frac{2T_f - \tau_{n-2} - \tau_{n-1}}{d}\mathbf{v}_f + \frac{(T_f - \tau_{n-2})(T_f - \tau_{n-1})}{d(d-1)}\mathbf{a}_f. \quad (28b)$$

Note that (23)–(28) are not simply penalties to be added in (13). It is meaningless to compute trajectories that are not respecting the initial and final conditions. These boundary constraints have to be satisfied by all the particles at all iterations. Hence, the algorithm takes random particles except the first and last three control points of each particle which are imposed directly by (23), (25), (26), (28). The other control points (the remaining $n-6$ of them) are then optimized by the algorithm.

3.1.8. Obstacle collision penalization

The obstacle avoidance constraint (12l) is implemented to protect the multicopters against collisions. This requires to account for both the obstacles' shape, for tracking errors and for a safety region around the multicopter[1]. All these elements may be brought back to the original setting of generating the trajectory of an adimensional agent by simply enlarging the initial obstacles with the necessary offsets.

Next, the convexity property P5) is exploited: the trajectory is designed to stay inside the union of convex polytopes defined by the curve's control points. Since the property may be conservative we may use the knot refinement to tighten the convex

[1]While the trajectory returned by (13) assumes pointwise dimensions for the multicopter, in reality its dimensions cannot always be ignored.

hulls as shown in Figure 2. To reduce the computational effort, the penalty for violating the constraint is defined here as the count of non empty intersections between the convex hulls (noted $C_j^H(\mathbf{X})$) and the enlarged polyhedral obstacles (noted Obs_i)

$$F_6(\mathbf{X}) = \sum_{i=1}^{N_{Obs}} \sum_{j=1}^{\tilde{n}-d} \delta(i, j, \mathbf{X}), \tag{29}$$

where $\delta(i, j, \mathbf{X}) = \begin{cases} 1, & C_j^H(\mathbf{X}) \cap Obs_i \neq \emptyset \\ 0, & otherwise. \end{cases}$

Remark 3. Knot refinement allows to tighten the convex hulls in order to approximate less conservatively the trajectory. [31] studies the distance between a B-spline curve and its control polygon. In particular, the distance is decreasing when using knot refinement and converging to 0 when the number of control points goes to ∞. However, knot refinement comes at a price: increasing the number of control points will increase the number of convex hulls, so the computational time will be higher because more intersections between polytopes and convex hulls will have to be checked. ♦

3.1.9. Waypoints conditions

The waypoint criteria imposes that the leader is, at a specified time, near a specified position (the waypoint), as per (12i). The constraint can be either an equality or it may be relaxed to a small neighborhood centered around the waypoint (e.g., a small sphere of radius d_{wp}). Consequently, the expression taking into account all the waypoints becomes:

$$F_7(\mathbf{X}) = \sum_{l=1}^{N_{WP}} \sigma \left(\left\| \mathbf{p}_\ell^{wp} - \sum_{k=0}^{n-1} \mathbf{P}_k^1 B_{k,d,\boldsymbol{\xi}}(t_\ell) \right\|_2 - d_{wp} \right). \tag{30}$$

3.2. Trajectories planning for the followers

While not strictly required, hereafter we assume that the followers have the same number of control points as the leader, subsequent to its knot refinement procedure, i.e., $\tilde{n}$ control points. Since the trajectory (as defined by the control points) of the leader has been already computed and remains unchanged henceforth, it is natural to define the control points of the followers relatively to those of the leader, thus, for the j-th follower we define the control point offsets:

$$\boldsymbol{\Delta}_k^{1j} = \mathbf{P}_k^j - \mathbf{P}_k^1, \quad \forall k = 0, \ldots, \tilde{n} - 1. \tag{31}$$

Here, $\mathbf{P}_k^j$ is the k-th absolute control point of agent j and $\boldsymbol{\Delta}_k^{1j}$ is its corresponding k-th relative (to the k-th leader's control points) control point.

As the degree, knot vector and basis functions are fixed, the particles representing the followers' trajectories gather all the offsets $\mathbf{Y} := \left\{ \boldsymbol{\Delta}_k^{1j} \right\}_{j>1, k=0,\ldots,\tilde{n}-1}$ to solve a new differential evolution algorithm.
The control points of the leader after knot refinement are an input of the optimization function which allows to pass from the relative to the absolute position of the control points as needed.

The objective function is given as a sum of weighted penalty terms:

$$F^{\mathrm{F}}(\mathbf{Y}) = \sum_{q=0}^{8} \gamma_q F_q(\mathbf{Y}), \tag{32}$$

where $F_0(\mathbf{Y}), \ldots, F_6(\mathbf{Y})$ are defined as in the list below (13), but replacing n by $\tilde{n}$ and the numerical values and absolute control points corresponding to the other agents. $F_7(\mathbf{Y})$ penalizes the formation tracking error and $F_8(\mathbf{Y})$ corresponds to the communication loss penalization. $\gamma_0, \ldots, \gamma_8$ are the weights associated to each of these penalties. All the particles are satisfying the initial and final conditions at each iterations.

3.2.1. Formation tracking penalization

The objective (12j) is to minimize the formation tracking error. The formation to be attained is a priori known but may change along the trajectory. It is defined in terms of relative positions between the leader and its followers, i.e., position displacement, as in [32].

Remark 4. The leader's behavior is arguably simpler to model (fewer penalty terms) and hence less control points (n) suffice in providing a reasonable trajectory whereas the followers have to avoid collisions but communicate between themselves and maintain and switch between formations, thus requiring $\tilde{n} > n$ control points. Knot refinement allows to mesh together these elements: the leader's trajectory is reformulated with $\tilde{n}$ control points depending linearly on the original n control points (hence, no increase in complexity) which allows to express the followers' trajectories through displacements (31) and gives the additional flexibility of working with $\tilde{n}$ control points. ♦

Lemma 1. Let notation

$$\boldsymbol{\Delta}_s^{ij} = \boldsymbol{\Delta}_k^{1j} - \boldsymbol{\Delta}_k^{1i}, \quad \forall k = s-d, \ldots, s, \tag{33}$$

denote a constant control point displacement between a pair of followers $i \neq j$ for $d+1$ consecutive control points (those indexing from $s-d$ to s).

Then, the interdistance $d_{ij}(t)$ between these followers remains constant over the sub-interval $t \in [\tau_s, \tau_{s+1})$.

Proof. Since any two followers $i \neq j$ share the same knot vector ζ, we may define their interdistance over some sub-interval $t \in [\tau_s, \tau_{s+1})$ as

$$\begin{aligned} d_{ij}(t) &= \|\mathbf{z}_j(t) - \mathbf{z}_i(t)\|_2 \\ &= \left\| \sum_{k=s-d}^{s} \mathbf{P}_k^j B_{k,d,\boldsymbol{\xi}}(t) - \sum_{k=s-d}^{s} \mathbf{P}_k^i B_{k,d,\boldsymbol{\xi}}(t) \right\|_2 \\ &= \left\| \sum_{k=s-d}^{s} (\mathbf{P}_k^j - \mathbf{P}_k^i) B_{k,d,\boldsymbol{\xi}}(t) \right\|_2 \\ &= \left\| \sum_{k=s-d}^{s} \left(\boldsymbol{\Delta}_k^{1j} - \boldsymbol{\Delta}_k^{1i} \right) B_{k,d,\boldsymbol{\xi}}(t) \right\|_2, \end{aligned} \tag{34}$$

where we used the local support property P1) and (31). Introducing (33) in (34) leads, through property P2), to

$$\begin{aligned} d_{ij}(t) = \left\| \sum_{k=s-d}^{s} \boldsymbol{\Delta}_s^{ij} B_{k,d,\boldsymbol{\xi}}(t) \right\|_2 &= \left\| \boldsymbol{\Delta}_s^{ij} \sum_{k=s-d}^{s} B_{k,d,\boldsymbol{\xi}}(t) \right\|_2 \\ &= \left\| \boldsymbol{\Delta}_s^{ij} \right\|_2, \end{aligned} \tag{35}$$

thus proving that $d_{ij}(t)$ remains constant for sub-interval $t \in [\tau_s, \tau_{s+1})$. □

Extending Lemma 1 to more agents provides sufficient conditions for ensuring formation stability. The idea is to penalize the error between the real relative positions of the control points $(\boldsymbol{\Delta}_k^{1j} - \boldsymbol{\Delta}_k^{1i})$ and the desired relative position corresponding to the formation $(\boldsymbol{\Delta}_s^{ij})$. Once again the knot refinement provided more control points to impose locally the formation. The expression of the formation's penalty is:

$$F_7(\mathbf{Y}) = \sum_{i>j}^{\tilde{n}-1} \sum_{s=d}^{} \sum_{k=s-d}^{s} \left\| \boldsymbol{\Delta}_s^{ij} - \left(\boldsymbol{\Delta}_k^{1j} - \boldsymbol{\Delta}_k^{1i} \right) \right\|_2, \tag{36}$$

which penalizes for all pairs of followers, for each list of $d+1$ consecutive control points, the variation from the desired inter-dosition.

Remark 5. Whenever the formation changes, terms $\boldsymbol{\Delta}_s^{ij}$ which force its tracking as per (36) have to change accordingly (from $\boldsymbol{\Delta}_s^{ij,1}$ to $\boldsymbol{\Delta}_s^{ij,2}$). Via the local support property P1), we have that if the change occurs at a time t_{switch}, e.g., from $\boldsymbol{\Delta}_s^{ij,1}$ to $\boldsymbol{\Delta}_s^{ij,2}$, it follows that

$$\boldsymbol{\Delta}_\ell^{ij} \leftarrow \boldsymbol{\Delta}_\ell^{ij,1}, \ \forall \ell \leq s-d-1 \text{ and } \boldsymbol{\Delta}_\ell^{ij} \leftarrow \boldsymbol{\Delta}_\ell^{ij,2}, \ \forall \ell \geq s+1, \tag{37}$$

where $s = \arg\max_\ell \tau_\ell \leq t_{\text{switch}}$. Taking the first inequality we observe that the last term active for the first formation has index $s-d-1$ which means that the last B-spline function active in the interdistance description (34) is $B_{s-d-1,d,\boldsymbol{\xi}}(t)$ whose support covers sub-interval $[\tau_{s-d-1}, \tau_s)$, thus, earlier than the switch time t_{switch}. A similar argument may be carried for the second inequality where we note that the 'earliest' a Bspline curve becomes active is at $\tau_{s+1} > t_{\text{switch}}$. Conversely, indices $s-d \leq \ell \leq s$ denote an intermediary period in which the control points should ensure the transition from the first to the second formation. One possible idea is to have a linear interpolation for $s-d \leq \ell \leq s$:

$$\boldsymbol{\Delta}_\ell^{ij} \leftarrow \frac{(s+1)-\ell}{d+2} \boldsymbol{\Delta}_\ell^{ij,1} + \frac{\ell-(s-d-1)}{d+2} \boldsymbol{\Delta}_\ell^{ij,2}. \tag{38}$$

♦

3.2.2. Communication loss penalization

The communication loss, defined as per (12k) is penalized by a sufficient formulation which penalizes the interdistance (34) between those followers which have to communicate, i.e., the pairs $(i,j) \in \mathcal{L}$.

Recalling (34), applying the triangle inequality and making use of the positiveness property P2) leads to an upper bound for the interdistance between any pair of followers over the sub-interval $[\tau_s, \tau_{s+1})$:

$$\begin{aligned} d_{ij}(t) &= \left\| \sum_{k=s-d}^{s} \left(\boldsymbol{\Delta}_k^{1j} - \boldsymbol{\Delta}_k^{1i} \right) B_{k,d,\boldsymbol{\xi}}(t) \right\|_2 \\ &\leq \sum_{k=s-d}^{s} \left\| \left(\boldsymbol{\Delta}_k^{1j} - \boldsymbol{\Delta}_k^{1i} \right) \right\|_2 B_{k,d,\boldsymbol{\xi}}(t). \end{aligned} \tag{39}$$

Since B-spline basis functions partition the unity we arrive at:

$$d_{ij}(t) \leq \max_{k=s-d,\ldots,s} \left\| \left(\boldsymbol{\Delta}_k^{1j} - \boldsymbol{\Delta}_k^{1i} \right) \right\|_2. \tag{40}$$

Extending for all the knot's sub-intervals we arrive at a sufficient condition for the communication con-

straint ($d_{ij}(t) \leq \rho$):

$$\max_{k=0,\ldots,\tilde{n}-1} \left\| \left(\boldsymbol{\Delta}_k^{1j} - \boldsymbol{\Delta}_k^{1i} \right) \right\|_2 \leq \rho \tag{41}$$

which is then penalized through objective component

$$F_8(\mathbf{Y}) = \sum_{(i,j)\in\mathcal{L}} \sigma \left(\max_{k=0,\ldots,\tilde{n}-1} \left\| \left(\boldsymbol{\Delta}_k^{1j} - \boldsymbol{\Delta}_k^{1i} \right) \right\|_2 - \rho \right). \tag{42}$$

3.3. Implementation procedure

Planning the trajectories of a hierarchical leader-follower formation is detailed in Algorithm 1.

The algorithm can be decomposed in three main steps:

i) Use differential evolution algorithm to obtain the trajectory of the leader (the one passing through the waypoints) defined by the n control points in $\mathbf{X}_{best}$ minimizing the fitness function F^{L} (in (13)) (from line 1 to 14).

ii) Apply knot refinement (in (9)) to increase the number of control points defining the trajectories to add degrees of liberty for switching the formation. (line 15).

iii) Use differential evolution algorithm (in (32)) to obtain the control points defining the trajectories of the followers switching formation and keeping the communication range (from line 16 to 29).

4. Simulations, experiments and comparisons

With respect to our preliminary results presented in [19], where we addressed the optimal trajectory generation problem for a multicopter using Particle Swarm Optimization (PSO), this study extends our findings. Initially, we propose a comparative scenario employing both PSO and Differential Evolution (DE) based on B-splines to demonstrate the computational advantages of DE. Subsequently, we exclusively employ DE in additional scenarios, which highlight the benefits of our formation optimization problem, incorporating obstacle avoidance and inter-agent communication constraints. To summarize, we will begin by presenting the numerical values employed in the simulations and experiments. This will be followed by the enumeration and description of all scenarios. Subsequently, we will conduct a comprehensive analysis, comparison, and discussion of the results.

Algorithm 1: Solving a motion planning problem for a team of multicopters satisfying a given scenario.

Input: list of parameters in (13) and (32) $n, \tilde{n}, \omega^{\mathrm{L}}, c_r^{\mathrm{L}}, n_{part}^{\mathrm{L}}, n_{iter}^{\mathrm{L}}, \omega^{\mathrm{F}}, c_r^{\mathrm{F}}, n_{part}^{\mathrm{F}}, n_{iter}^{\mathrm{F}}$;
Output: $\mathbf{X}_{best}$ and $\mathbf{Y}_{best}$ minimizing the fitness functions F^{L} (in (13)) and F^{F} (in (32));

1 Create n_{part}^{L} random matrices of n control points respecting the initial and final conditions: $\mathbf{X}_1, ..., \mathbf{X}_{n_{part}^{\mathrm{L}}}$
2 for i=1:n_{iter}^{L} do
3 for k=1:n_{part}^{L} do
4 Choose randomly 3 particles $\mathbf{X}_a, \mathbf{X}_b, \mathbf{X}_c$;
5 $\mathbf{X}_d \leftarrow \mathbf{X}_a + \omega^{\mathrm{L}}(\mathbf{X}_b - \mathbf{X}_c)$;
6 $\mathbf{X}_k^* \leftarrow f(\mathbf{X}_k, \mathbf{X}_d, c_r^{\mathrm{L}})$;
7 Compute $F^{\mathrm{L}}(\mathbf{X}_k^*)$ in (13);
8 if $F^{\mathrm{L}}(\mathbf{X}_k^*) < F^{\mathrm{L}}(\mathbf{X}_k)$ do
9 $\mathbf{X}_k \leftarrow \mathbf{X}_k^*$;
10 Store $F^{\mathrm{L}}(\mathbf{X}_k^*)$;
11 end
12 end
13 end
14 Find $\mathbf{X}_{best} = \arg\min_{\mathbf{X}_k} F^{\mathrm{L}}(\mathbf{X}_k) \quad \forall k \in \{1, ..., n_{part}^{\mathrm{L}}\}$
15 $\tilde{\mathbf{X}} \leftarrow \mathit{KnotRefinement}(\mathbf{X}_{best}, \tilde{n})$ (9)
16 Create n_{part}^{F} random matrices of $\tilde{n}$ control points per follower respecting the initial and final conditions: $\mathbf{Y}_1, ..., \mathbf{Y}_{n_{part}^{\mathrm{F}}}$
17 for i=1:n_{iter}^{F} do
18 for k=1:n_{part}^{F} do
19 Choose randomly 3 particles $\mathbf{Y}_a, \mathbf{Y}_b, \mathbf{Y}_c$;
20 $\mathbf{Y}_d \leftarrow \mathbf{Y}_a + \omega^{\mathrm{F}}(\mathbf{Y}_b - \mathbf{Y}_c)$;
21 $\mathbf{Y}_k^* \leftarrow f(\mathbf{Y}_k, \mathbf{Y}_d, c_r^{\mathrm{F}})$;
22 Compute $F^{\mathrm{F}}(\mathbf{Y}_k^*)$ in (32);
23 if $F^{\mathrm{F}}(\mathbf{Y}_k^*) < F^{\mathrm{F}}(\mathbf{Y}_k)$ do
24 $\mathbf{Y}_k \leftarrow \mathbf{Y}_k^*$;
25 Store $F^{\mathrm{F}}(\mathbf{Y}_k^*)$;
26 end
27 end
28 end
29 Find $\mathbf{Y}_{best} = \arg\min_{\mathbf{Y}_k} F^{\mathrm{F}}(\mathbf{Y}_k) \quad \forall k \in \{1, ..., n_{part}^{\mathrm{F}}\}$

Table 1 briefly enumerates the scenarios further detailed in this section.

The numerical parameters used and the results obtained for the trajectory planning of each sce-

Table 1: Explanation of the scenarios.

	Scenario	Multicopters	Bounds	Waypoints	Obstacles
1	PSO-spline vs DE-spline	1	Tightened	3	No
2	DE-spline vs MOSEK [18]	1	Tightened	8	No
3	Algorithm 1	4	Tightened	3	No
4	Experimentation	4	Standard	3	Yes

nario are listed in Table 2 and later in Table 6.

The B-splines used hereinafter have degree $d = 4$ and the simulation time is always set at $T_f = 30$s. The gravitational acceleration g is 9.81 m/s^2. The positions of all the multicopters must stay inside the box defined by $\mathbf{x}_{\min} = \begin{bmatrix}-1.5 & -1 & 0\end{bmatrix}^\top$ and $\mathbf{x}_{\max} = \begin{bmatrix}1.5 & 1 & 1.5\end{bmatrix}^\top$.

The way-points through which the trajectories outputted by the various scenarios have to pass are gathered in Tables 3 and 4.

The penalty weights used throughout the scenarios are enumerated in Table 5.

The code implementing Algorithm 1 may be run at https://github.com/marguetv/CEP.git and was run in Matlab 2023a using its codegen option to make a C-MEX variant of the Gilbert-Johnson-Keerthi function [33] testing polyhedron intersections and CasADi [34] to generate the B-splines basis functions.

4.1. Scenario 1: Comparison between Particle Swarm Optimization (PSO-spline) and Differential Evolution (DE-spline) with B-splines algorithm

The simulation compares two algorithms implementing evolutionary optimization over identical parameters and test conditions. We take the same number of particles n_{part}, iterations n_{iter} and weights γ_i for the objective function detailed in Table 5 and provide those parameters specific to each algorithm: for the DE-spline, the differential weight F is set to 0.15 and the crossover probability c_r is set to 0.7, and for the PSO-spline, the damping weight is set to 1 and the learning coefficients are set to $c_1 = 1.2$ for exploration and $c_2 = 1.5$ for exploitation. All these values were adjusted to optimize the performance of the algorithms.

The test is defined as one multicopter passing through the neighborhoods (defined as spheres of radii $d_{wp} = 5cm$) of the first three waypoints enumerated in Table 3 at pre-specified times and landing back to the origin at $T_f = 30s$, while respecting the dynamical constraints detailed in this paper.

Due to the stochastic nature of the algorithms, the trajectories may differ between successive runs. Consequently, we run each algorithm 100 times and average the outputs of interest (e.g., computational time, trajectory cost, maximal distance between waypoints and the leader's position). The results corresponding to one of the 100 simulations are shown in Figures 3 and 4.

The two algorithms output trajectories that pass near the waypoints (all the 100 simulations for both algorithm output trajectories passing at less than 5cm of all the waypoints at the specified times) (Figure 3) and respect the dynamical bounds (Figure 4). However, we found the DE-spline algorithm to be, by small but measurable percentages, superior to the PSO-spline algorithm. For example, the computation time is slightly faster (1.23s with a standard deviation of 0.06s for DE-spline versus 1.31s with a standard deviation of 0.04s for PSO-spline) and the value of the objective function is slightly lower for DE-spline (9.0×10^{-3} with a standard deviation of 2.4×10^{-3} versus 1.19×10^{-2} with a standard deviation of 4.1×10^{-3} for PSO-spline).

Considering also that the trajectory, velocity and thrust are smoother for DE-spline (fewer and less pronounced 'spikes' in higher order derivatives) we will use hereinafter the DE-spline algorithm to run the next scenarios.

4.2. Scenario 2: Applying our DE-spline approach to the Example 1 from [18]

Again, a single multicopter is flying but this time it passes near each of the eight waypoints from Table 3. These values duplicate the scenario presented in [18], except in the number of control points, decreased from 41 to 20, and in the spline degree, decreased from 5 to 4. These modifications lead to a faster convergence time and still provide a feasible trajectory.

The resulted trajectory (and associated waypoints) for one simulation are shown in Figure 5 and all the dynamical constraints are verified as shown in Figure 6. As before, we have run 100 instances

Table 2: Numerical values of the bounds.

Bounds	$v_{\max}$ [m/s]	$T_{\min}$ [m/s^2]	$T_{\max}$ [m/s^2]	ϵ [°]	$\omega_{\max}$ [°/s]
Tightened	0.5	9.7	9.9	1.75	1.5
Standard	1	0	10.5	7	30

Table 3: Numerical values of the waypoints for the scenarios 1 and 2.

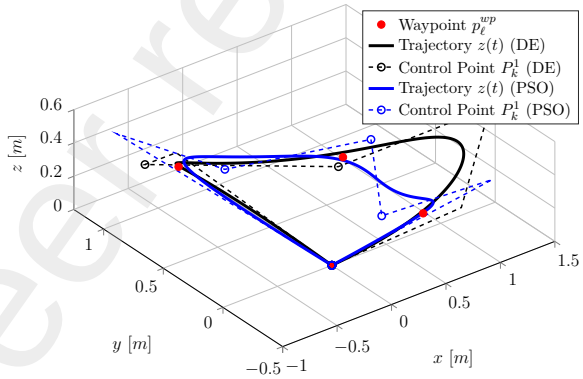
ℓ	1	2	3	4	5	6	7	8
$\mathbf{p}_\ell^{wp}$	-0.75	0.65	0.40	-0.15	0.65	-0.50	-0.60	0.25
	0.60	0.50	-0.40	0.25	-0.65	0.50	-0.60	0.25
	0.50	0.25	0.40	0.25	0.25	0.75	0.50	0.25
t_ℓ[s]	7.8	15.3	24	4.5	12.6	18	21	27

Table 4: Numerical values of the waypoints for the scenarios 3 and 4.

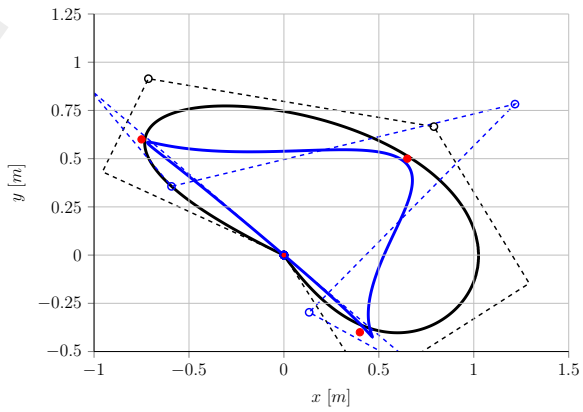
ℓ	1	2	3
$\mathbf{p}_\ell^{wp}$	-1.0	0.2	0.5
	0.6	0.5	-0.5
	0.6	0.8	0.4
t_ℓ[s]	13	17	24

and averaged the results to obtain the mean computational time and mean penalty cost in Table 6. We note that the mean of the largest neighborhood around a way-point has a radius of 19.6cm.

The way-point radii may be decreased by either increasing the number of particles and iterations (more degrees of freedom) or by increasing the way-point criteria's weight associated to (30). Neither is attractive: the first directly leads to higher computational time and the second gives less importance to the other constraints. Furthermore, although the degree and the number of control points are fewer in our implementation, the velocity, thrust, angle and angular velocity profiles are very similar (see our Fig. 6 and 7 compared to Fig. 4 and Fig. 6 from [18]). This indicates that our approach is converging towards the solution produced by the solver utilized in the alternative path planning method. Moreover, our method achieves this convergence with fewer control points, offering the flexibility to incorporate additional constraints of any type, including non-convex constraints for obstacle avoidance.



(a) 3D view



(b) top view ($x-y$ plane)

Figure 3: Trajectory generated with the parameters in Table 3. We show the trajectories (continuous) and the control points (connected circles) obtained with DE-spline (black) and PSO-spline (blue) along with the waypoints (red spheres) (Scenario 1).

4.3. Scenario 3: Formation switching with 4 multicopters

In this scenario a team of four agents (one leader and three followers) switch from a "square" to a "line" formation, and back again to the "square" formation to land at their initial positions. The leader has to pass near the waypoints listed in Ta-

Table 5: Numerical values of the weights for the cost functions.

	Scenario	γ_0	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
F^{L}	1,2,3,4	1	1	4×10^4	40	8×10^4	5×10^3	10^6	5×10^4	–
F^{F}	3	1	1	4×10^4	40	8×10^4	5×10^6	10^3	10^6	–
F^{F}	4	1	1	4×10^4	40	8×10^4	5×10^3	10^3	10^6	10^6

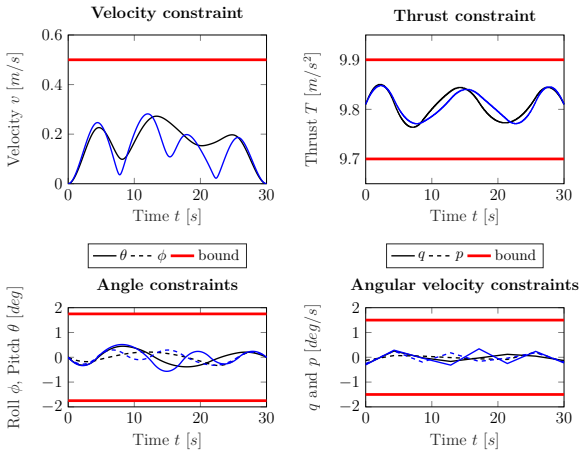


Figure 4: Constraints satisfaction for the obtained trajectories obtained with DE-spline (black) and PSO-spline (blue) (Scenario 1).

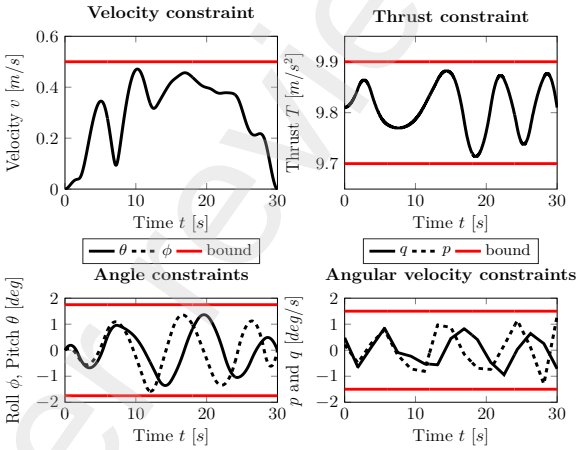


Figure 6: Constraints satisfaction for the obtained trajectory (Scenario 2).

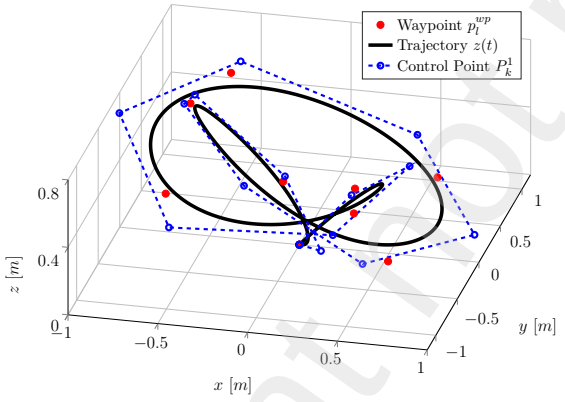


Figure 5: Trajectories position generated with the parameters in Table 3. We show the position trajectory (black) along with the waypoints (red spheres) and the control points (dashed connected, blue circles) (Scenario 2).

ble 4 at the specified times. The three followers track it because they have to keep the inter-multicopter position corresponding to the formation. In addition, the range (less than $\rho = 0.75$ m) between communicating agents also has to be respected. We assume the pairs in communication are (1, 2), (2, 3) and (3, 4). The distance in the formations between the agents is 50cm for the line formation and also for the square side of the square formation. All the agents need to respect the dynamical constraints. The trajectories of the leader and the followers are obtained using Algorithm 1. The trajectories obtained are illustrated in Figures 7 and 8. Each trajectory is represented by a color and we can clearly see the formation switching along the simulation (it happens at times $t_{\mathrm{switch},1} = 5s$ and $t_{\mathrm{switch},2} = 20s$).

The waypoints are also close to the trajectory of the leader: all the 100 simulations output trajectories passing at less than 5cm of all the waypoints at the specified times. We see in Figure 9 that all the dynamical constraints stay inside the bounds.

Moreover, the formation switching and formation tracking behavior can be observed in Figure 10. That is, the distances between the multicopters remain almost constant, as they should be, whenever the trajectories are far from the switching times and agent inter-distances d_{12}, d_{23} and d_{34} are always less than $\rho = 0.75$m, i.e., the communication constraint is satisfied.

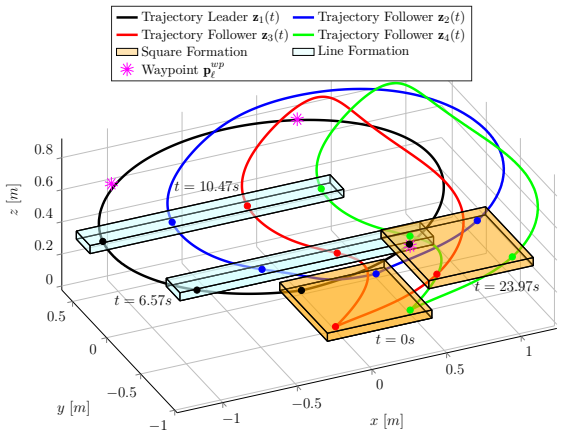


Figure 7: Trajectories generated with the parameters in Table 3. We show the leader's trajectory (black) passing near the waypoints (magenta stars) and the followers' trajectories (blue, red, green) switching their formation. If not redefined in the legends, the color associated to each agent will be kept in all the remaining figures in particular for the constraints satisfaction profiles (Scenario 3).

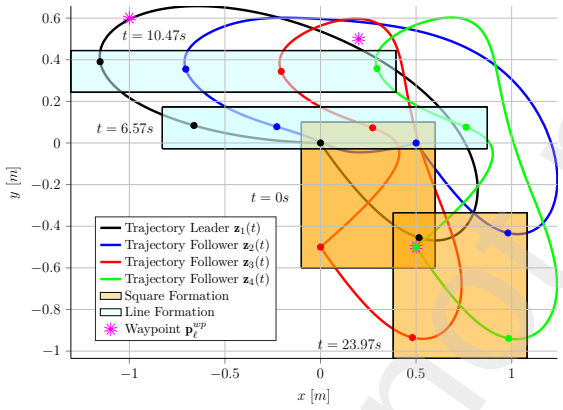


Figure 8: Top view of Fig. 7 (Scenario 3).

4.4. Scenario 4: Obstacle avoidance for 4 multicopters switching their formation

To the previous scenario, with changed physical bounds (we take standard bounds for multicopter experiments), as delineated in Table 2, we add two obstacles that have to be avoided, as per (29). To make the problem non-trivial, the obstacles block the line of sight between successive waypoints. The locations and dimensions of the inflated[2] obstacles appear in Figures 11 and 12.

[2]To the original obstacle shape, we add the dimension of the drone and the tracking error bounds taken from [18].

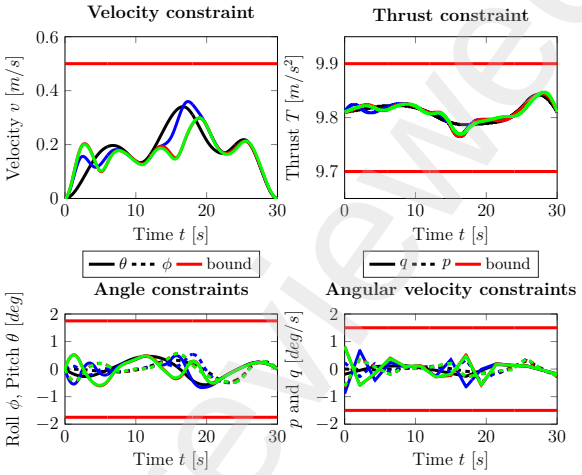


Figure 9: Constraints satisfaction for the obtained trajectories (Scenario 3).

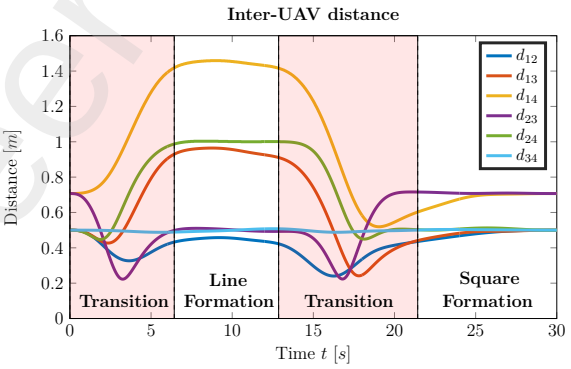


Figure 10: Distance between the agents. As a reminder, penalties concerning the communication constraint are only taken into-account for d_{12}, d_{23} and d_{34} (Scenario 3).

The goal is then to generate a trajectory for the leader that avoids collision with the obstacles and satisfies all the dynamical constraints while being as close as possible from the waypoints at the specified times. Also the other agents have to avoid collision, satisfy the dynamical and communication constraints and then be as close as possible from the desired formation.

The simulation results are shown from Figures 11 to 14. We clearly see that the bounds are respected for the constraints on velocity, thrust, angle and angular velocity, the multicopters are switching their formation while respecting the communication constraint.

Moreover, the leader is passing as close as possible from the waypoints (the mean, over 100 simulations, of the maximum distance between the way-

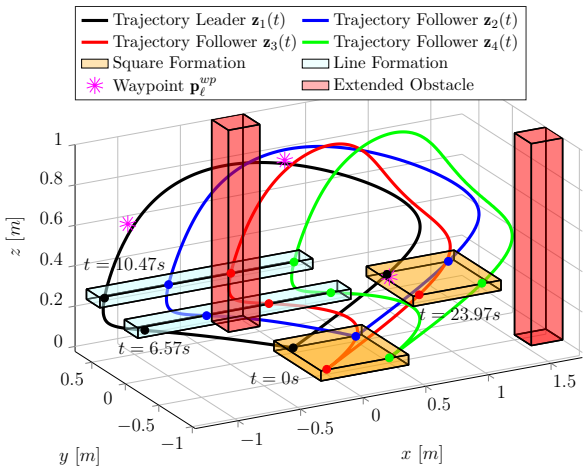


Figure 11: Trajectories generated for Scenario 4. The leader's trajectory (black) passes near the waypoints (magenta stars), the followers' trajectories (blue, red, green) switch their formation and avoid the obstacles (Scenario 4).

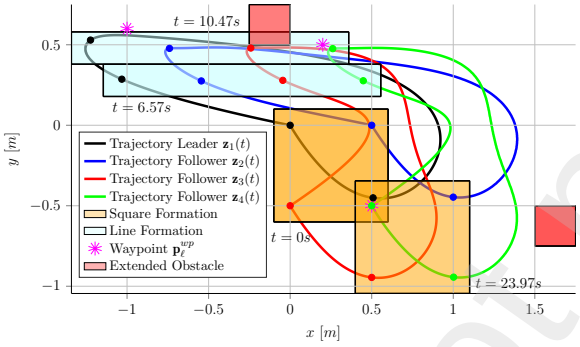


Figure 12: Top view of Fig. 11 (Scenario 4).

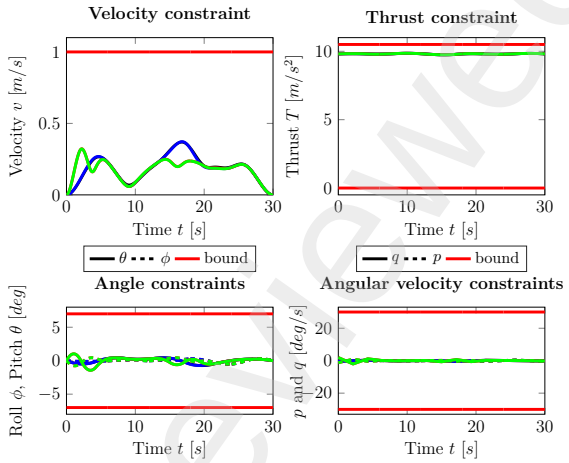


Figure 13: Constraints satisfaction for the obtained trajectories (Scenario 4).

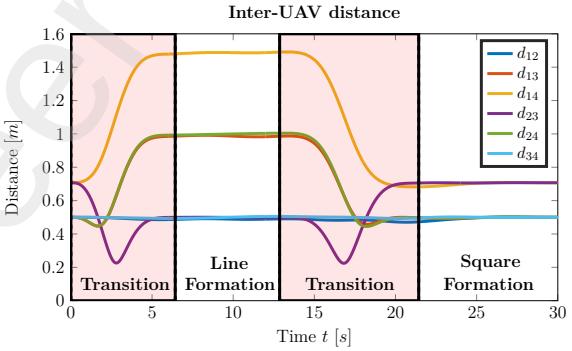


Figure 14: Distance between the agents. As a reminder, penalties concerning the communication constraint are only taken into-account for d_{12}, d_{23} and d_{34} (Scenario 4).

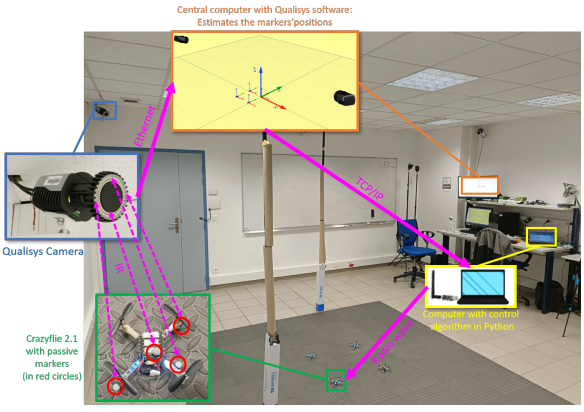


Figure 15: Esisarium platform for algorithm testing.

points and the position of the leader at the associated time is 7.6cm) while avoiding collision with the obstacles. The communication constraint is satisfied as the curves of Figure 14 representing the distances between multicopter 1 and multicopter 2, between multicopter 2 and multicopter 3 and between multicopter 3 and multicopter 4 are all always inferior to the communication range. All the interdistances are kept almost constant outside of the transition zones, meaning that the formation is correctly tracked.

4.5. Experimental validation

The experiments are conducted with four nanodrones in our Esisarium platform at LCIS laboratory presented in Figure 15. The dimensions of the box covered by the maximum number of cameras are $3 \times 2 \times 2$m. To fully utilize this volume, an additional set of waypoints for scenarios 3 and 4, out-

Table 6: Simulations parameters and computational analysis associated (mean over 100 simulations). The a) and b) variants that appears for Scenarios 3 and 4 show the parameters considered for the leader and, respectively, for the followers.

Simulation	n_{iter}	n_{part}	n	Penalty	Time $[s]$
Scenario 1 (DE-spline)	100	100	11	9.0×10^{-3}	1.23
Scenario 1 (PSO-spline)	100	100	11	1.2×10^{-2}	1.31
Scenario 2 (DE-spline)	200	500	20	1.9×10^{4}	26.9
Scenario 3a (DE-spline)	100	200	11	8.3×10^{-3}	2.45
Scenario 3b (DE-spline)	300	300	18	9.4×10^{6}	39.2
Scenario 4a (DE-spline)	100	200	11	1.6×10^{3}	10.3
Scenario 4b (DE-spline)	300	300	18	1.5×10^{6}	143

lined in Table 4, was established. Their positions are measured using passive markers on each drone: the cameras emit infrared (IR) pulses of light whose reflections on a marker are then gathered by the cameras and used to triangulate the markers' position and, after further post-processing, the object's pose (its position and attitude).

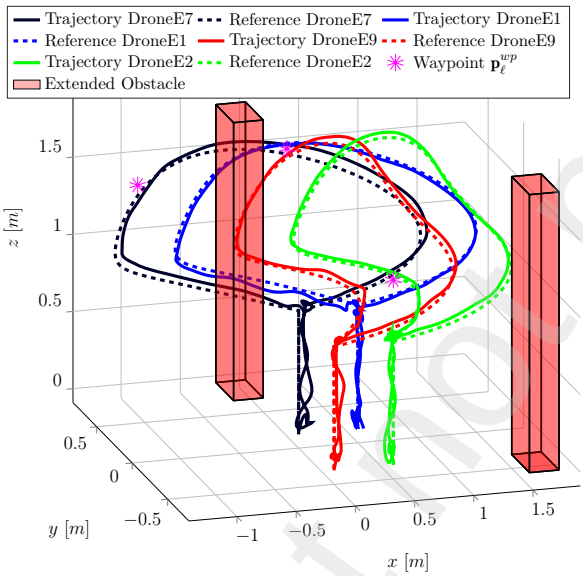


Figure 16: Trajectories planned (in dashed) and tracked (in solid) with the parameters from Scenario 4, where the leader's trajectory (black) passes near the waypoints (magenta stars), the followers' trajectories (blue, red, green) switch their formation and avoid the obstacles (Scenario 4).

We emphasize that this paper primarily addresses trajectory planning rather than trajectory following. While various methods exist for tracking trajectories, such as Model Predictive Control as in [35, 36], Sliding Mode Control as in [37], or Disturbance Observer-based Control as in [38, 39] we maintain consistency with [18], by implementing the same feedback trajectory tracking mecha-

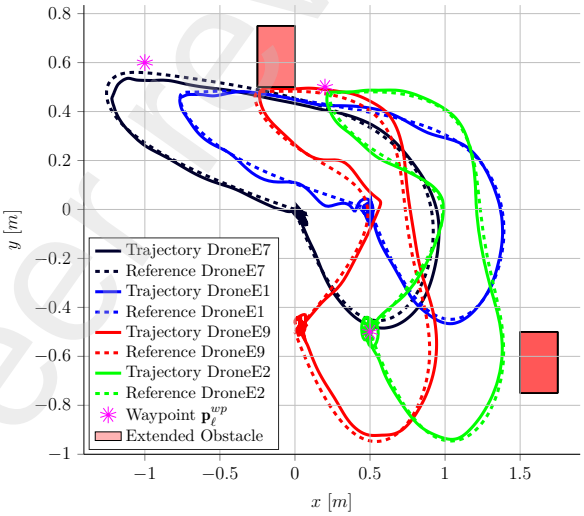


Figure 17: Top view of Fig. 16 (Scenario 4).

nism. Specifically, we utilize a Linear Quadratic Regulator (LQR) controller and a Control Barrier Function-Quadratic Program (CBF-QP) with identical parameters ($\delta = 0.1$, $a_1 = 6$ and $a_2 = 8$), to ensure comparability. Hence, using the measurements data, a QP based on CBF is solved online to track the trajectories. CBFs are used as a safety filter to guarantee boundedness between the real trajectory and the nominal safe trajectory that we obtained with our algorithm. The results provided by CBF-QP are added to the ones coming from LQR that are localized at the ground station and used to compute the new inputs of the multicopters, knowing the reference trajectory and the real-time trajectory. The inputs are then sent to the multicopters using the antenna on the computer and the receiver onboard. Every 100 ms, the new reference point of the trajectory is updated.

For completeness, we recall here the optimization problem that is solved online (for trajectory track-

ing) using the solver OSQP in Python in [18] and in our experiments:

$$\min_{\boldsymbol{\mu}} \quad \|\Psi(\pi(\mathbf{z})) - \boldsymbol{\mu}\|_2^2 \tag{43a}$$

$$\text{s.t.} \left|\mu_x - \ddot{x}^{ref} - a_1(\dot{x}^{ref} - \dot{x}) - a_2(x^{ref} - x)\right| \leq a_2\delta, \tag{43b}$$

$$\left|\mu_y - \ddot{y}^{ref} - a_1(\dot{y}^{ref} - \dot{y}) - a_2(y^{ref} - y)\right| \leq a_2\delta, \tag{43c}$$

$$\left|\mu_z - \ddot{z}^{ref} - a_1(\dot{z}^{ref} - \dot{z}) - a_2(z^{ref} - z)\right| \leq a_2\delta, \tag{43d}$$

where $\boldsymbol{\mu} = [\mu_x \quad \mu_y \quad \mu_z]^\top$ and $\Psi(\pi(\mathbf{z}))$ yields the nominal virtual inputs with $\pi(\mathbf{z})$ a LQR controller with parameters $Q = \text{diag}(500I_3, 250I_3)$ and $R = 10I_3$, taking the state vector as the positions and velocities on each axis and the input vector as the accelerations on each axis.

The objective of the experiment is to track the 4 references obtained with the algorithm presented in this paper. The video of the experiment is available at https://tinyurl.com/CEPmarguetVID. In experiments, we observe some ground effects appearing during the take off and landing part. To circumvent the challenge of modeling these inherently nonlinear interactions and to ensure conservative bounds, the trajectories generated using our approach are as follows:

- they commence after a seven-second take-off phase, reaching an altitude of 80 cm, followed by a one-second hovering phase at the same location.
- they are tracked with an offset of 80 cm to mitigate ground-induced perturbations.
- they include a preceding one-second hovering phase and a subsequent five-second landing phase.

In total, the experimental flights will last 44 seconds (the actual 30 seconds of the simulated trajectories and the ancillary 14 seconds for take-off, hovering and landing). The multicopters 1, 2, 3 and 4 are associated respectively to the rigid bodies called DroneE7, DroneE1, DroneE9 and DroneE2 and will be plotted respectively in black, blue, red and green. In the Figures 18 to 23 illustrating the experimental results, the values are shown during the entire test flights, but the first 8 and last 6 seconds are represented in a blue box as they are not the focus of the experiments. We clearly see in Figures 16 and 17 that there is no collision with the obstacles. Figure 18 confirms that the multicopters are satying in the allowed volume and Figure 20 shows that the velocity bounds are respected for all the multicopters. The decomposition on each axis is shown in Figure 19. Figures 21 and 22 show that the thrust, roll angle, pitch angle and their derivatives stayed inside the bounds for all the multicopters. The center of DroneE7, the leader, is always in a sphere of radius inferior to 10cm centered in the waypoints at the specified time.

Figure 23 confirms that the communication constraint is satisfied as the inter-multicopter distance always stays inferior to the communication range. Moreover, the inter-multicopter distance is not varying excessively during the formation-tracking stage.

Figure 24 shows the tracking performance obtained via the "LQR with CBFQP" controller, used in all four multicopters: all the curves stay positive thus guaranteeing that the tracking error is inferior to δ on each axis. In other words, all the trajectories obtained are included in the square shaped corridor of thickness 2δ centered in the reference trajectories. However, during the additional parts of the trajectories (take-off and landing), the constraint is not satisfied on the z-axis due to the aforementioned ground effect nonlinearities.

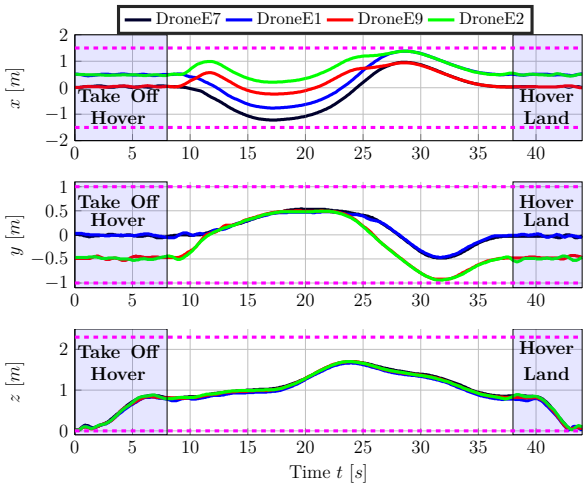


Figure 18: Trajectories measured on each axis for each multicopter (Scenario 4).

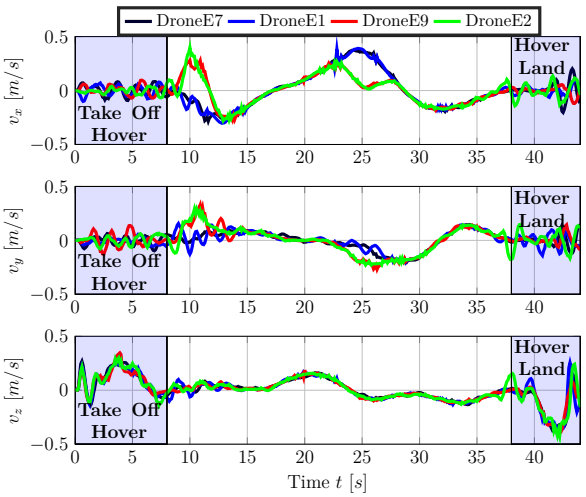


Figure 19: Velocities on each axis (Scenario 4).

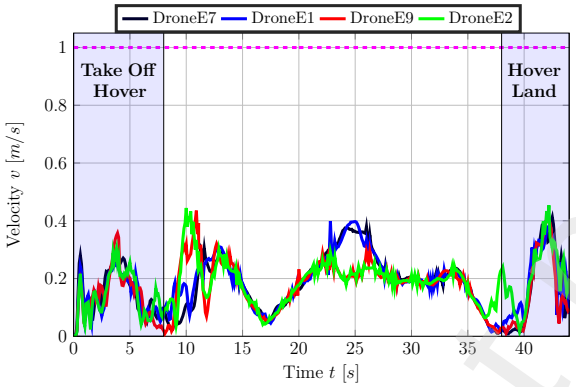


Figure 20: Velocities for each multicopter (Scenario 4).

5. Conclusion

This paper introduces an offline motion planning algorithm designed to address the non-convex optimization problem associated with generating trajectories for multicopters. The algorithm employs B-spline parametrizations with additional properties, supplemented by a knot refinement step to incorporate various constraints. These constraints encompass bounds on positions, velocities, angles, angular velocities, thrusts, obstacle avoidance, waypoint passage, communication maintenance, and formation switching. The proposed algorithm enables the adjustment of weight parameters to prioritize specific constraints over others, as well as the tuning of the number of particles and iterations to balance the trade-off between optimality and computation time. Unlike existing solvers in the literature, this

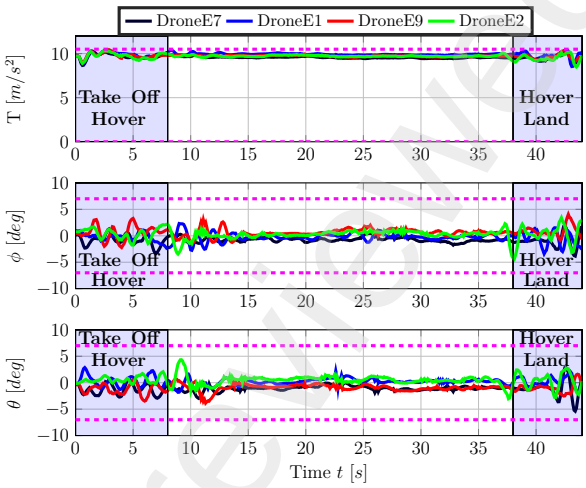


Figure 21: Thrust T, Roll ϕ and Pitch θ angles measured for each multicopter (Scenario 4).

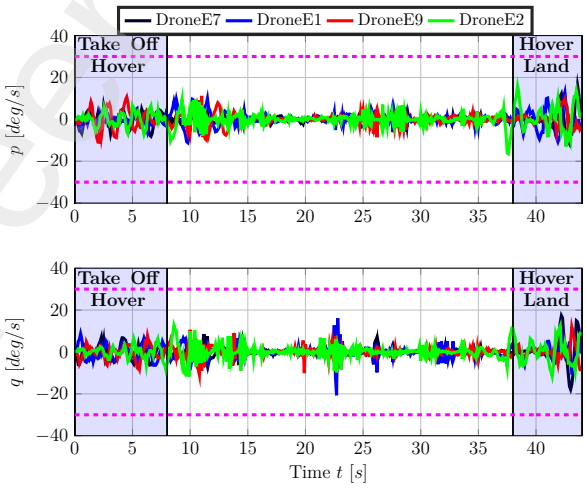


Figure 22: Measured angular velocities (Scenario 4).

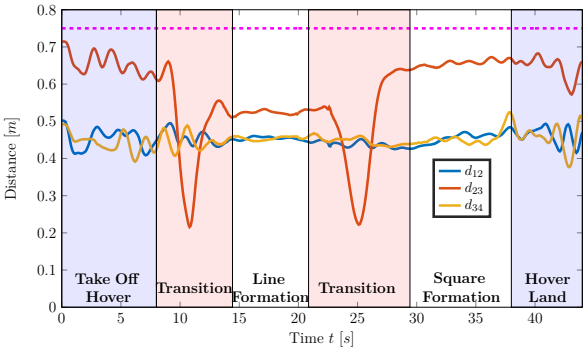


Figure 23: Inter-multicopter distance measured for each communication link that has to be kept (Scenario 4).

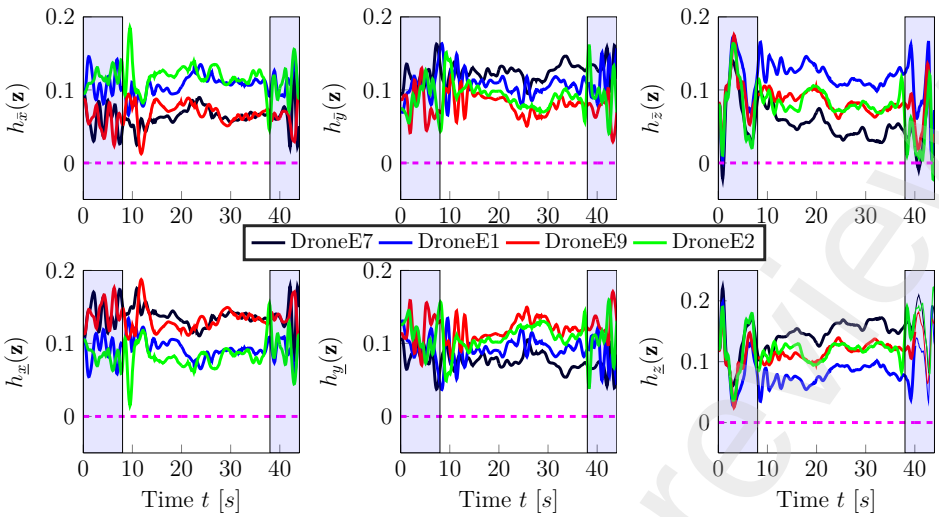


Figure 24: Tracking performance on each axis for each multicopter (Scenario 4).

method imposes no restrictions on constraint formulation and provides complete visibility into the optimization process. Additionally, experiments involving multiple nanodrones were conducted to validate the effectiveness of the designed trajectories.

Future work will explore the utilization of the Schoenberg quasi-interpolant to streamline the complexity of the B-spline functions, as demonstrated in a previous study focusing on fixed-wing aircraft applications [40]. Another avenue for research involves enhancing computational efficiency to enable online implementation of the algorithm. For instance, this could entail decomposing the trajectory optimization problem into smaller subproblems, facilitating quicker resolution.

6. Appendix

6.1. Non-conservative computation of angular bounds

The angular constraints

$$\|\mathbf{A}_\epsilon \ddot{\mathbf{r}}(t)\|_2 \leq \mathbf{z}_w^\top \ddot{\mathbf{r}}(t) + g, \quad \forall t \in [0, T_f], \tag{44}$$

are expressed in terms of control points in [18] as

$$\left\|\mathbf{A}_\epsilon \ddot{\mathbf{P}}_k\right\|_2 \leq \mathbf{z}_w^\top \ddot{\mathbf{P}}_k + g \quad \forall k = 0, \ldots, n-3, \tag{45}$$

where[3] $\mathbf{A}_\epsilon = \mathrm{diag}(\cot\epsilon, \cot\epsilon, 0)$ and g is the gravitational acceleration.

[3] cot is the notation for the cotangent

A less conservative expression is obtained if we replace in (44) the term $\ddot{\mathbf{r}}(t)$ with its corresponding B-spline curve representation $\forall t \in [0, T_f]$,

$$\left\|\mathbf{A}_\epsilon \sum_{k=0}^{n-3} \ddot{\mathbf{P}}_k B_{k,d-2,\boldsymbol{\xi}}(t)\right\|_2 \leq \mathbf{z}_w^\top \sum_{k=0}^{n-3} \ddot{\mathbf{P}}_k B_{k,d-2,\boldsymbol{\xi}}(t) + g, \tag{46}$$

which, after squaring and regrouping the lhs and rhs terms, leads to

$$\begin{aligned}
&\cot^2\epsilon \sum_{k=0}^{n-3}\sum_{i=0}^{n-3} \ddot{\mathbf{P}}_i^\top \ddot{\mathbf{P}}_k B_{k,d-2,\boldsymbol{\xi}}(t) B_{i,d-2,\boldsymbol{\xi}}(t) \\
&\leq (1+\cot^2\epsilon) \sum_{k=0}^{n-3}\sum_{i=0}^{n-3} \ddot{\mathbf{P}}_i^\top \mathbf{z}_w \mathbf{z}_w^\top \ddot{\mathbf{P}}_k B_{k,d-2,\boldsymbol{\xi}}(t) B_{i,d-2,\boldsymbol{\xi}}(t) \\
&\quad + 2g\mathbf{z}_w^\top \sum_{k=0}^{n-3} \ddot{\mathbf{P}}_k B_{k,d-2,\boldsymbol{\xi}}(t) + g^2, \quad \forall t \in [0, T_f].
\end{aligned} \tag{47}$$

Next, using the unit partitioning property of the B-splines functions we may group the rhs of (47) to arrive at

$$\begin{aligned}
&\cot^2\epsilon \sum_{k=0}^{n-3}\sum_{i=0}^{n-3} \ddot{\mathbf{P}}_i^\top \ddot{\mathbf{P}}_k B_{k,d-2,\boldsymbol{\xi}}(t) B_{i,d-2,\boldsymbol{\xi}}(t) \\
&\leq \sum_{k=0}^{n-3}\sum_{i=0}^{n-3} \left[(1+\cot^2\epsilon)\ddot{\mathbf{P}}_i^\top \mathbf{z}_w \mathbf{z}_w^\top \ddot{\mathbf{P}}_k + 2g\mathbf{z}_w^\top \ddot{\mathbf{P}}_k + g^2\right] \\
&\qquad .B_{k,d-2,\boldsymbol{\xi}}(t) B_{i,d-2,\boldsymbol{\xi}}(t),
\end{aligned} \tag{48}$$

which, with notation (19), may be written compactly as

$$\sum_{k=0}^{n-3}\sum_{i=0}^{n-3} G(\epsilon,i,k,\mathbf{P})B_{k,d-2,\boldsymbol{\xi}}(t)B_{i,d-2,\boldsymbol{\xi}}(t) \le 0, \forall t\in[0,T_f]. \tag{49}$$

Since the B-spline basis functions are positive (as per Property P2)), we may now provide sufficient conditions for (44) to hold, both

i) globally (over the entire horizon $[0,T_f]$):

$$\begin{aligned} &G(\epsilon,i,k,\mathbf{P}) \le 0, \quad \forall k=0,\dots,n-3\\ &\text{and } i=\max(0,k+d-2),\dots,\min(n-3,k+d); \end{aligned} \tag{50}$$

ii) or, locally, for some knot sub-interval $[\tau_s,\tau_{s+1})$:

$$\begin{aligned} &\forall k, i=\max(0,s+d-2),\dots,\min(n-3,s+d),\\ &G(\epsilon,i,k,\mathbf{P}) \le 0. \end{aligned} \tag{51}$$

For both (50) and (51) we made use of Property P1) to assess for each pair of indices (i,k) whether the product $B_{k,d-2,\boldsymbol{\xi}}(t)B_{i,d-2,\boldsymbol{\xi}}(t)$ is empty or not, first on the entire interval $[0,T_f]$, and second for the knot sub-interval $[\tau_s,\tau_{s+1})$.

Lastly, recalling (19), we have that $G(\epsilon,k,k,\mathbf{P})$

$$\begin{aligned} &= \cot^2\epsilon\,\ddot{\mathbf{P}}_k^\top\ddot{\mathbf{P}}_k - \left(\cot^2\epsilon+1\right)\ddot{\mathbf{P}}_k^\top\mathbf{z}_w\mathbf{z}_w^\top\ddot{\mathbf{P}}_k - 2g\mathbf{z}_w^\top\ddot{\mathbf{P}}_k - g^2,\\ &= \ddot{\mathbf{P}}_k^\top\mathbf{A}_\epsilon^\top\mathbf{A}_\epsilon\ddot{\mathbf{P}}_k - \left(\ddot{\mathbf{P}}_k^\top\mathbf{z}_w\mathbf{z}_w^\top\ddot{\mathbf{P}}_k + 2g\mathbf{z}_w^\top\ddot{\mathbf{P}}_k + g^2\right),\\ &= \|\mathbf{A}_\epsilon\ddot{\mathbf{P}}_k\|_2^2 - \left(\mathbf{z}_w^\top\ddot{\mathbf{P}}_k + g\right)^2, \end{aligned}$$

which shows that (50) is equivalent with (45) whenever $i=k$. This shows that (50) provides a less conservative bounding of (44) since, in addition to the conditions enumerated in (45), we augment the constraint set with the cases for which $i\neq k$.

6.2. Non-conservative computation of angular velocity bounds

The angular velocity constraints are expressed in terms of control points in [18] as

$$\mathbf{z}_w^\top\ddot{\mathbf{P}}_k \ge \zeta_{\ell-d+1} - g, \quad \forall k=\ell-d+2,\dots,\ell, \tag{52a}$$

$$\left\|\dddot{\mathbf{P}}_k\right\|_2 \le \zeta_{\ell-d+1}\omega_{\max}, \quad \forall k=\ell-d+3,\dots,\ell. \tag{52b}$$

with $\boldsymbol{\zeta}=(\zeta_1,\dots,\zeta_{n-d+1})^\top$ a vector whose entries are positive constants.

A less conservative expression is obtained by first recalling the initial inequality from [18]

$$\|\mathbf{h}_\omega\|_2 \le \frac{\|\dddot{\mathbf{r}}(t)\|_2}{T(t)} = \frac{\|\dddot{\mathbf{r}}(t)\|_2}{\|\ddot{\mathbf{r}}(t)+\mathbf{z}_w g\|_2} \le \omega_{\max}, \tag{53}$$

where, since all terms are positive, we may put it into form

$$\|\dddot{\mathbf{r}}(t)\|_2^2 \le \omega_{\max}^2\,\|\ddot{\mathbf{r}}(t)+\mathbf{z}_w g\|_2^2\,. \tag{54}$$

Replacing with the B-spline curves associated with $\ddot{\mathbf{r}}(t)$, $\dddot{\mathbf{r}}(t)$ we arrive at

$$\left\|\dddot{\mathbf{P}}\mathbf{B}_{d-3,\boldsymbol{\xi}}(t)\right\|_2^2 \le \omega_{\max}^2\left\|\ddot{\mathbf{P}}\mathbf{B}_{d-2,\boldsymbol{\xi}}(t)+\mathbf{z}_w g\right\|_2^2. \tag{55}$$

Further applying the partition of unity property P2), we have

$$\left\|\dddot{\mathbf{P}}\mathbf{B}_{d-3,\boldsymbol{\xi}}(t)\right\|_2^2 \le \omega_{\max}^2\left\|\left(\ddot{\mathbf{P}}+\mathbf{z}_w g\mathbb{1}_{1\times(n-1)}\right)\mathbf{B}_{d-2,\boldsymbol{\xi}}(t)\right\|_2^2. \tag{56}$$

Since the lhs and lrs of (56) have different orders of the B-splines ($d-3$ and, respectively, $d-2$), we analyze the inequality over each knot vector sub-interval and make use of Property P6) which allows to express any $d-3$ - order B-spline curve in terms of $d-2$ - order B-spline functions:

$$\dddot{\mathbf{P}}\mathbf{B}_{d-3,\boldsymbol{\xi}}(t) = \dddot{\mathbf{P}}\mathbf{D}_{d-3,d-2}^{\ell}\mathbf{B}_{d-2,\boldsymbol{\xi}}(t). \tag{57}$$

Thus, on each knot sub-interval ℓ, the following relation holds

$$\begin{aligned} &\left\|\dddot{\mathbf{P}}\mathbf{D}_{d-3,d-2}^{\ell}\mathbf{B}_{d-2,\boldsymbol{\xi}}(t)\right\|_2^2\\ &\qquad\le \omega_{\max}^2\left\|\left(\ddot{\mathbf{P}}+\mathbf{z}_w g\mathbb{1}_{1\times(n-1)}\right)\mathbf{B}_{d-2,\boldsymbol{\xi}}(t)\right\|_2^2. \end{aligned} \tag{58}$$

Via the local support property P1), (58) simplifies to involve only the non-zero (on the ℓ-th sub-interval basis functions):

$$\begin{aligned} &\left\|\sum_{k=\ell+d-2}^{\ell+d}(\dddot{\mathbf{P}}\mathbf{D}_{d-3,d-2}^{\ell})_k B_{k,d-2,\boldsymbol{\xi}}(t)\right\|_2^2\\ &\qquad\le \omega_{\max}^2\left\|\sum_{k=\ell+d-2}^{\ell+d}\left(\ddot{\mathbf{P}}_k+\mathbf{z}_w g\right)B_{k,d-2,\boldsymbol{\xi}}\right\|_2^2. \end{aligned} \tag{59}$$

With notation (22), (59) becomes

$$\sum_{k=\ell+d-2}^{\ell+d}\sum_{i=\ell+d-2}^{\ell+d} S(\ell,i,k,\mathbf{P})B_{i,d-2,\boldsymbol{\xi}}^\top(t)B_{k,d-2,\boldsymbol{\xi}}(t) \le 0. \tag{60}$$

Since the B-spline basis functions are positive (as per Property P2)), a sufficient condition for (53) to hold on the sub-interval $[\tau_\ell, \tau_{\ell+1})$ is that

$$S(\ell, i, k, \mathbf{P}) \leq 0, \quad \forall i, k = \ell + d - 2, \ldots, \ell + d. \quad (61)$$

Lastly, recalling (22), we have that

$$S(\ell, k, k, \mathbf{P}) = (\dddot{\mathbf{P}}\mathbf{D}^{\ell}_{d-3,d-2})_k^{\top}(\dddot{\mathbf{P}}\mathbf{D}^{\ell}_{d-3,d-2})_k - \omega_{\max}^2\left(\ddot{\mathbf{P}}_k + \mathbf{z}_w g\right)^{\top}\left(\ddot{\mathbf{P}}_k + \mathbf{z}_w g\right),$$

which, under assumption (61) and after the application of the square root, may be written as

$$\frac{\|(\dddot{\mathbf{P}}\mathbf{D}^{\ell}_{d-3,d-2})_k\|_2}{\|\ddot{\mathbf{P}}_k + \mathbf{z}_w g\|_2} \leq \omega_{\max},$$

which means that there exists some ζ such that

$$\|(\dddot{\mathbf{P}}\mathbf{D}^{\ell}_{d-3,d-2})_k\|_2 \leq \bar{\zeta}\omega_{\max}, \quad \|\ddot{\mathbf{P}}_k + \mathbf{z}_w g\|_2 \geq \bar{\zeta}.$$

Since $(\dddot{\mathbf{P}}\mathbf{D}^{\ell}_{d-3,d-2})_k = \sum_j \dddot{\mathbf{P}}_j \mathbf{D}^{\ell}_{d-3,d-2}(j,k)$ it follows that $\|(\dddot{\mathbf{P}}\mathbf{D}^{\ell}_{d-3,d-2})_k\|_2 \leq \sum_j \|\dddot{\mathbf{P}}_j\|_2 \cdot |\mathbf{D}^{\ell}_{d-3,d-2}(j,k)|$ holds. Thus, whenever conditions (52) hold, we may take $\bar{\zeta}$ which verifies simultaneously

$$\sum_j \left(\max_{j=\ell-d+3,\ldots,\ell} \zeta_{\ell-d+1} \cdot |\mathbf{D}^{\ell}_{d-3,d-2}(j,k)| \right) \leq \bar{\zeta},$$

$$\bar{\zeta} \geq \max_{j=\ell-d+3,\ldots,\ell} \zeta_{\ell-d+1}.$$

This shows that (52) is equivalent with (60) whenever $i = k$ (after further manipulations, as discussed in [18, Remark 2]). Hence, (60) provides a less conservative bounding of (53) since, in addition to the conditions enumerated in (52), we augment the constraint set with the cases for which $i \neq k$. Not least, it should be mentioned that our implementation avoids the use of ancillary constants ζ_ℓ, as done in (52).

7. Acknowledgements

This work has been partially supported by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissements d'avenir, La Région Auvergne-Rhône-Alpes, Pack Ambition Recherche 2021 - PlanMAV, RECPLA-MALCIR, Ambition Internationale 2023, Horizon-TA C7H-REG24A10, France, and by a grant from the National Program for Research of the National Association of Technical Universities - GNAC ARUT 2023; Project ID: 207, UNSTPB, Romania.

References

[1] X. Li, Y. Zhao, J. Zhang, Y. Dong, A hybrid pso algorithm based flight path optimization for multiple agricultural uavs, in: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), 2016, pp. 691–697. doi:10.1109/ICTAI.2016.0110.

[2] B. Bethke, M. Valenti, J. P. How, Experimental demonstration of uav task assignment with integrated health monitoring, IEEE Robotics automation magazine march (2010).

[3] G. Rousseau, C. Stoica Maniu, S. Tebbani, M. Babel, N. Martin, Minimum-time b-spline trajectories with corridor constraints. application to cinematographic quadrotor flight plans, Control Engineering Practice 89 (2019) 190–203. doi:https://doi.org/10.1016/j.conengprac.2019.05.022.

[4] P. Chen, F. Ouyang, Y. Zhang, Y. Lan, Preliminary evaluation of spraying quality of multi-unmanned aerial vehicle (uav) close formation spraying, Agriculture 12 (2022) 1149.

[5] J. Fu, G. Wen, X. Yu, Z.-G. Wu, Distributed formation navigation of constrained second-order multiagent systems with collision avoidance and connectivity maintenance, IEEE Transactions on Cybernetics 52 (2020) 2149–2162.

[6] S. Vargas, H. M. Becerra, J.-B. Hayet, Mpc-based distributed formation control of multiple quadcopters with obstacle avoidance and connectivity maintenance, Control Engineering Practice 121 (2022) 105054.

[7] A. Caregnato-Neto, M. R. Maximo, R. J. Afonso, A line of sight constraint based on intermediary points for connectivity maintenance of multiagent systems using mixed-integer programming, European Journal of Control 68 (2022) 100671.

[8] L. Zhu, C. Ma, J. Li, Y. Lu, Q. Yang, Connectivity-maintenance uav formation control in complex environment, Drones 7 (2023) 229.

[9] B. Zhou, F. Gao, L. Wang, C. Liu, S. Shen, Robust and efficient quadrotor trajectory generation for fast autonomous flight, IEEE Robotics and Automation Letters 4 (2019) 3529–3536. doi:10.1109/LRA.2019.2927938.

[10] Z. Shiller, Off-line and on-line trajectory planning, Motion and Operation Planning of Robotic Systems: Background and Practical Approaches (2015) 29–62.

[11] M. T. R. Khan, M. Muhammad Saad, Y. Ru, J. Seo, D. Kim, Aspects of unmanned aerial vehicles path planning: Overview and applications, International Journal of Communication Systems 34 (2021) e4827.

[12] A. Pekarovskiy, T. Nierhoff, S. Hirche, M. Buss, Dynamically consistent online adaptation of fast motions for robotic manipulators, IEEE Transactions on Robotics 34 (2018) 166–182. doi:10.1109/TRO.2017.2765666.

[13] B. Sabetghadam, R. Cunha, A. Pascoal, Real-time trajectory generation for multiple drones using bézier curves, IFAC-PapersOnLine 53 (2020) 9276–9281.

[14] T. Lyche, C. Manni, H. Speleers, Foundations of spline theory: B-splines, spline approximation, and hierarchical refinement, in: Splines and PDEs: From Approximation Theory to Numerical Linear Algebra, Springer, 2018, pp. 1–76.

[15] L. Wang, Y. Guo, Speed adaptive robot trajectory generation based on derivative property of b-spline curve,

IEEE Robotics and Automation Letters 8 (2023) 1905–1911. doi:10.1109/LRA.2023.3241812.

[16] R. T. Rodrigues, N. Tsiogkas, A. Pascoal, A. P. Aguiar, Online range-based slam using b-spline surfaces, IEEE Robotics and Automation Letters 6 (2021) 1958–1965. doi:10.1109/LRA.2021.3060672.

[17] N. T. Nguyen, P. T. Gangavarapu, N. F. Kompe, G. Schildbach, F. Ernst, Navigation with polytopes: A toolbox for optimal path planning with polytope maps and b-spline curves, Sensors 23 (2023). doi:10.3390/s23073532.

[18] V. Freire, X. Xu, Flatness-based quadcopter trajectory planning and tracking with continuous-time safety guarantees, IEEE Transactions on Control Systems Technology 31 (2023) 2319–2334. doi:10.1109/TCST.2023.3250954.

[19] V. Marguet, C. K. Dinh, I. Prodan, F. Stoican, Constrained PSO-splines trajectory generation for an indoor nanodrone, in: 2024 International Conference on Unmanned Aircraft Systems, ICUAS '24, Chania Crète, Greece, 2024. URL: https://hal.science/hal-04544687.

[20] S. S. Mansouri, C. Kanellakis, E. Fresk, D. Kominiak, G. Nikolakopoulos, Cooperative coverage path planning for visual inspection, Control Engineering Practice 74 (2018) 118–131.

[21] F. Stoican, I. Prodan, D. Popescu, L. Ichim, Constrained trajectory generation for uav systems using a b-spline parametrization, in: 2017 25th Mediterranean Conference on Control and Automation (MED), 2017, pp. 613–618. doi:10.1109/MED.2017.7984185.

[22] F. Stoican, A. Postolache, I. Prodan, Nurbs-based trajectory design for motion planning in a multi-obstacle environment, in: 2021 European Control Conference (ECC), 2021, pp. 2014–2019. doi:10.23919/ECC54610.2021.9654974.

[23] T. Luukkonen, Modelling and control of quadcopter, Independent research project in applied mathematics, Espoo 22 (2011).

[24] N. T. Nguyen, I. Prodan, L. Lefèvre, Flat trajectory design and tracking with saturation guarantees: a nanodrone application, International Journal of Control 93 (2020) 1266–1279.

[25] J. Löfberg, Yalmip : A toolbox for modeling and optimization in matlab, in: In Proceedings of the CACSD Conference, Taipei, Taiwan, 2004.

[26] M. ApS, The MOSEK optimization toolbox for MATLAB manual. Version 9.0., 2019. URL: http://docs.mosek.com/9.0/toolbox/index.html.

[27] B. Salamat, A. M. Tonello, Stochastic trajectory generation using particle swarm optimization for quadrotor unmanned aerial vehicles (uavs), Aerospace 4 (2017) 27.

[28] B. P. Duisterhof, S. Li, J. Burgués, V. J. Reddi, G. C. de Croon, Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 9099–9106.

[29] N. Naidja, S. Font, M. Revilloud, G. Sandou, An interactive game theory-pso based comprehensive framework for autonomous vehicle decision making and trajectory planning, in: IFAC World Congress-22nd WC 2023, 2023.

[30] X. Yu, C. Li, J. Zhou, A constrained differential evolution algorithm to solve uav path planning in disaster scenarios, Knowledge-Based Systems 204 (2020) 106209.

[31] D. Lutterkort, J. Peters, Tight linear bounds on the distance between a spline and its b-spline control polygon (1999).

[32] O. Mechali, L. Xu, X. Xie, J. Iqbal, Theory and practice for autonomous formation flight of quadrotors via distributed robust sliding mode control protocol with fixed-time stability guarantee, Control Engineering Practice 123 (2022) 105150. doi:https://doi.org/10.1016/j.conengprac.2022.105150.

[33] M. Sheen, Fast 3d collision detection – gjk algorithm, GitHub project, 2023. URL: https://github.com/mws262/MATLAB-GJK-Collision-Detection.

[34] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, M. Diehl, CasADi – A software framework for nonlinear optimization and optimal control, Mathematical Programming Computation 11 (2019) 1–36. doi:10.1007/s12532-018-0139-4.

[35] I. Prodan, S. Olaru, R. Bencatel, J. Borges de Sousa, C. Stoica, S.-I. Niculescu, Receding horizon flight control for trajectory tracking of autonomous aerial vehicles, Control Engineering Practice 21 (2013) 1334–1349. doi:https://doi.org/10.1016/j.conengprac.2013.05.010.

[36] H.-T. Do, I. Prodan, Indoor experimental validation of mpc-based trajectory tracking for a quadcopter via a flat mapping approach, in: 2023 European Control Conference (ECC), 2023, pp. 1–6. doi:10.23919/ECC57647.2023.10178260.

[37] R. Falcón, H. Ríos, A. Dzul, Comparative analysis of continuous sliding-modes control strategies for quad-rotor robust tracking, Control Engineering Practice 90 (2019) 241–256. doi:https://doi.org/10.1016/j.conengprac.2019.06.013.

[38] A. Castillo, R. Sanz, P. Garcia, W. Qiu, H. Wang, C. Xu, Disturbance observer-based quadrotor attitude tracking control for aggressive maneuvers, Control Engineering Practice 82 (2019) 14–23. doi:https://doi.org/10.1016/j.conengprac.2018.09.016.

[39] K. Guo, J. Jia, X. Yu, L. Guo, L. Xie, Multiple observers based anti-disturbance control for a quadrotor uav against payload and wind disturbances, Control Engineering Practice 102 (2020) 104560. doi:https://doi.org/10.1016/j.conengprac.2020.104560.

[40] V. Marguet, F. Stoican, I. Prodan, On the application of the schoenberg quasi-interpolant for complexity reduction in trajectory generation, in: 2023 European Control Conference (ECC), 2023, pp. 1–6. doi:10.23919/ECC57647.2023.10178175.