Trajectory Generation for Drones in Confined Spaces Using an Ellipsoid Model of the Body

Bahareh Sabetghadam, Rita Cunha, *Member, IEEE*, and António Pascoal, *Life Member, IEEE*

***Abstract*—We address the problem of motion planning for aerial drones to allow them to fly through confined spaces and narrow gaps between obstacles. Finding safe and feasible trajectories for steering multiple drones towards some desired positions in tight spaces or guiding them through gaps smaller in width than their diameters is impossible without taking the drones' orientations into account. To incorporate the orientation into the motion planning problem we employ an ellipsoid model of the drone body, and utilize the separating hyperplane theorem for convex sets to derive constraints that guarantee collision avoidance between the ellipsoid-shaped drone and obstacles modeled as ellipsoids, spheres, or polygons. The resulting set of constraints is seamlessly integrated into the motion planning method based on Bézier curve whose properties are exploited for efficient evaluation of the constraints. The efficacy of the proposed method in generating feasible and collision-free trajectories is demonstrated via different simulations involving single and multiple drones.**

***Index Terms*—Autonomous vehicles, Bézier curve, ellipsoids, separating hyperplane, optimization.**

I. Introduction

PRACTICAL applications of drones are expanding into new areas, many of which require the drone to maneuver through unstructured environments with narrow gaps and small spaces between obstacles. Examples of such environments are mountain trails and city streets when filming biking and running events. The complexity of these environments impose a significant challenge to the generation of feasible and collision-free trajectories. In order to avoid the multitude of obstacles in the environment and safely traverse the narrow gaps between them, the drone's attitude angles, together with its dynamic capabilities must be taken into account while generating the trajectories.

Manuscript received March 4, 2021; revised May 7, 2021; accepted May 28, 2021. Date of publication June 11, 2021; date of current version June 30, 2021. This work was supported in part by the Fundao para a Ciência e a Tecnologia and FEDER Funds under Project UIDB/50009/2020 and Project LISBOA-01-0145-FEDER-031411. The work of Bahareh Sabetghadam was supported by the Instituto Superior Técnico through scholarship BL216/2018/IST-ID. Recommended by Senior Editor C. Seatzu. *(Corresponding author: Bahareh Sabetghadam.)*

The authors are with the Laboratory of Robotics and Engineering Systems, ISR/IST, University of Lisbon, 1649-004 Lisbon, Portugal (e-mail: bsabetghadam@isr.ist.utl.pt; rita@isr.ist.utl.pt; antonio@isr.ist.utl.pt).

Digital Object Identifier 10.1109/LCSYS.2021.3088406

Several papers have studied autonomous navigation through narrow gaps. One of the most common ways put forward to enable a drone to fly through a gap and avoid collisions with the gap frame is to directly constrain its attitude angles to specific values at the gap's position [1]–[3]. The angle values are specified using prior knowledge of the gap's pose and position, or estimates of them from detection algorithms with onboard sensing.

In [3] a two-piece trajectory is used to fly a quadrotor through an inclined gap. The first segment of the trajectory enables state estimation via gap detection, while the second steers the quadrotor on the plane that is orthogonal to the gap and passes through its center. Forcing the quadrotor to fly on this plane minimizes the risk of collision with the gap frame, however this might be too conservative and lead to suboptimal trajectories with respect to a given objective function. Moreover, this approach is completely impractical for applications involving multiple drones.

Another way of considering the drone orientation while inspecting for collisions against obstacles is to approximate the drone body as an ellipsoid whose principal axes are aligned with the drone's body frame axes. Modeling the drone body as a sphere, though being simple and commonly assumed in motion planning, is very conservative and fails to validate the trajectories that might be feasible upon considering the vehicle's orientation, see Fig. 1. The ellipsoid model can circumvent this shortcoming yet, as opposed to the sphere model, there are no simple geometric constraints for identifying the collision-free space.

The ellipsoid model has been employed in a number of papers, however, in most of them the collision avoidance constraint is roughly evaluated by checking collisions between the ellipsoid and a sampled set of points from the obstacle. In this letter, we use the separating hyperplane theorem of convex sets to derive constraints for collision avoidance between an ellipsoid-shaped body and an ellipsoid, sphere or, polygon shaped type of obstacle. The obtained set of constraints is integrated into the trajectory generation method described in [4], to guarantee trajectories that are collision-free at every time instant. The simulation results demonstrate the efficacy of the proposed approach for guiding a drone through narrow gaps and flying multiple drones in small spaces.

II. Related Work

Polynomial-based trajectory generation for differentially flat systems has been studied in multiple papers [5]–[14]. All polynomial-based methods have the advantage of reducing the dimensionality of the underlying infinite-dimensional

optimization problem in motion planning by expressing the flat outputs as polynomial functions fully described with a limited number of coefficients; however, the different approaches of evaluating inequality constraints, including the collision avoidance constraint, can significantly impact their suitability and computational efficiency.

In [6]–[9] collision-free trajectories are generated using polynomials that lie in a safe flight corridor or pass through a sequence of waypoints in space. The obstacle-free space is built using either prior knowledge of the environment or search techniques running onboard, and the resulting inequalities are often evaluated on a finite set of time samples.

In order to avoid problems associated with time gridding, [10] takes advantage of Bézier curve properties to compute and limit (from below) the exact minimum distance of two Bézier curves. The major drawback to this approach is the significant increase in computational cost due to the resulting non-smooth functions. In all the above mentioned papers the collision-free space is obtained using a symmetrical (sphere) model of the drone body, ignoring its real shape. As discussed earlier, this approach invalidates the trajectories whose feasibility rely on the consideration of the body's attitude and thus, is too conservative for motion planning in small spaces.

The authors in [11] built on the spline-based motion planning method in [12], and extended it to include the orientation of a vehicle using a rectangular approximation of its body shape. To avoid colliding with an obstacle the 4 vertices, determined from the vehicle's position and rotation matrix, are enforced to be on one side of a separating hyperplane and the obstacle on the other side.

The work reported in [13] employs an ellipsoid model of the drone body to compute the flight attitude along sequences of motion primitives [14]. A primitive is considered collision-free if the intersection of a subset of the point cloud representing the obstacles with the ellipsoid at some sampled states in time along the primitive is empty.

In this letter, we use the separating hyperplane theorem of convex sets to derive constraints that guarantee collision avoidance for an ellipsoid model at every time instance. The obtained set of constraints can be expressed in terms of Bézier curves, which allows replacing them by a finite number of constraints for efficient computations.

III. Polynomial-Based Motion Planning for Differentially Flat Systems

The motion planning problem addressed in this letter consists of generating a trajectory that guides a quadrotor to a desired final position while minimizing a cost function. The generated trajectory must also satisfy a set of constraints in order to be safe, collision-free, and dynamically feasible. This problem can be cast in the form of the following optimization problem.

$$\begin{aligned} \underset{\mathbf{P}(.)}{\text{minimize}} \quad & J(.) \\ \text{s.t.} \quad & \textit{boundary conditions}, \\ & \textit{dynamic constraints}, \\ & \textit{collision avoidance constraints}, \end{aligned}$$

where $P(.)$ is the trajectory and $J(.)$ is the cost function defined based on the mission goals and objectives. The boundary conditions contain the initial and final values of the quadrotor's position, speed and/or acceleration. The constraints imposed by the quadrotor dynamics must also be considered to guarantee feasible trajectories, and finally the collision avoidance constraint is included to secure safe distance between the quadrotor and obstacles (or other drones) in the environment. The above optimization problem may take in other constraints to ensure that mission-specific requirements are met.

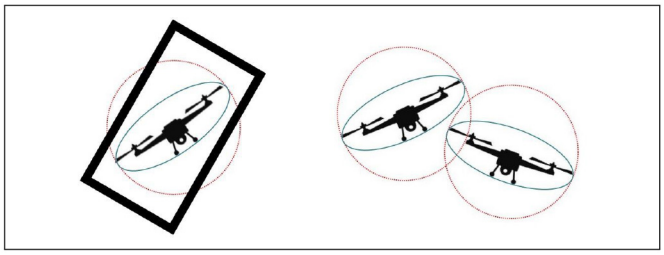

Fig. 1. The conventional method of modeling the drone body as a sphere is not suitable for motion planning in tight spaces, as it invalidates the trajectories that could be feasible upon considering the orientation. The less conservative ellipsoid model allows consideration for the 3D orientation.

In this letter the quadrotor model is described by a nonlinear system of the form

$$\dot{x}(t) = f(x(t), u(t)), \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$ are the state and the input vectors of the model, respectively. The boundary conditions and collision avoidance constraints can be written as equalities and inequalities in the states and inputs of the model, given by

$$\begin{aligned} h(x(t), u(t)) &\leq 0 \quad t \in [0, T], \\ g(x(t), u(t)) &= 0 \quad t \in \{0, T\}. \end{aligned} \tag{2}$$

We single out a particular class of dynamic systems called differentially flat systems, whose states and inputs can be determined from a set of independent variables without integration. More precisely, the nonlinear system (1) is differentially flat [15] if there exists a set of outputs $y \in \mathbb{R}^{n_y}$, equal in number to the inputs $n_y = n_u$, of the form

$$y = \beta(x, u, \dot{u}, \ldots, u^{(p)}), \tag{3}$$

such that all states and inputs of the system can be written as functions of the flat outputs y and a finite number of their derivatives, that is,

$$\begin{aligned} x &= \gamma_x(y, \dot{y}, \ldots, y^{(q)}), \\ u &= \gamma_u(y, \dot{y}, \ldots, y^{(q)}). \end{aligned} \tag{4}$$

Using the differential flatness property, trajectories consistent with (1) can be planned in the space of flat outputs, where the dynamic constraint is trivially satisfied and constraints in (2) are transformed into equality and inequality constraints on the flat output and its derivatives expressed as

$$\begin{aligned} \bar{h}(y, \dot{y}, \ldots, y^{(q)}) &\leq 0 \quad t \in [0, T] \\ \bar{g}(y, \dot{y}, \ldots, y^{(q)}) &= 0 \quad t \in \{0, T\} \end{aligned} \tag{5}$$

The components of the flat output y_j, $j = 1, \ldots, n_y$ are differentially independent [16], and can be parameterized individually by polynomial functions as

$$y_j(t) = \sum_{k=0}^{n_j} a_{jk} \Phi_k(t) \quad j \in \{1, \ldots, n_y\}, \tag{6}$$

where a_{jk} are the coefficients and $\Phi_k(t)$ are the polynomial basis functions. This converts the problem of finding functions in an infinite dimensional space into an approximate one of finding a finite set of coefficients. Consequently, the trajectory generation problem is converted into a semi-infinite optimization problem involving a finite number of variables $a_{jk}, k = 0, \ldots, n_j, j = 1, \ldots, n_y$ and an infinite number of constraints (5).

IV. Bézier Curves

Using special types of polynomial curves, such as those in Bernstein form, to parameterize the trajectory, besides allowing for a geometric interpretation to the design, can significantly reduce the computational costs. In the following we leverage Bézier curve properties to convert the semi-infinite optimization problem into one that is computationally tractable.

A. Definition and Properties

Given a set of $n+1$ control points $\bar{r}_i$, $i = 0, \ldots, n$, the corresponding Bézier curve of degree n is given by

$$r(\tau) = \sum_{i=0}^{n} \bar{r}_i B_{i,n}(\tau). \tag{7}$$

The Bernstein basis polynomials of degree n, $B_{i,n}(\tau)$, are defined as

$$B_{i,n}(\tau) = \binom{n}{i}(1-\tau)^{n-i}(\tau)^i \quad \tau \in [0,1], \tag{8}$$

where $\binom{n}{i}$ denote the binomial coefficients [17]. The Bernstein basis polynomials have several properties including,

- Non-negativity: $B_{i,n}(\tau) \geq 0, \quad \tau \in [0,1]$
- Partition of unity: $\sum_{i=0}^{n} B_{i,n}(\tau) = 1$

The non-negativity property is rather obvious from the definition of the Bernstein polynomials (8), and the partition of unity can be shown by employing Pascal's rule,

$$\sum_{i=0}^{n} \binom{n}{i}(1-\tau)^{n-i}(\tau)^i = ((1-\tau)+\tau)^n = 1. \tag{9}$$

The non-negativity and partition of unity lead to the important *convex hull property* of Bézier curves, that can be expressed as

$$r(\tau) \in CH(\bar{R}), \quad \tau \in [0,1], \tag{10}$$

where $CH(\bar{R})$ is the convex hull defined by the set of control points $\bar{R}$, that is,

$$CH(\bar{R}) = \{a_0\bar{r}_0 + \cdots + a_n\bar{r}_n | a_0 + \cdots + a_n = 1, a_i \geq 0\}. \tag{11}$$

The convex hull property states that the entire Bézier curve is inside a computable region. The convex hull property for a quintic Bézier curve is visualized in Figure 2.

B. De Casteljau's Algorithm

The de Casteljau's algorithm is a recursive method to evaluate a point on a Bézier curve corresponding to a parameter value $\tau \in [0,1]$ using only its control points [17]. Initializing with $\bar{r}_i^{(0)} = \bar{r}_i$, at each iteration j a set of $n-j+1$ points is calculated by linear interpolations of the points in the previous iteration, according to

$$\bar{r}_i^{(j)} = (1-\tau_0)\bar{r}_i^{(j-1)} + \tau_0\bar{r}_{i+1}^{(j-1)} \quad \begin{matrix} i = 0, \ldots, n-j \\ j = 1, \ldots, n \end{matrix} \tag{12}$$

The single point obtained at $j = n$ is the point on the curve corresponding to τ_0, that is,

$$\bar{r}_0^{(n)} = r(\tau_0). \tag{13}$$

This algorithm also provides a subdivision scheme for Bézier curves. The two sets of $\{\bar{r}_0^{(k)}, k = 0, \ldots, n\}$ and $\{\bar{r}_{n-k}^{(k)}, k = 0, \ldots, n\}$ form the control points for two Bézier curves of degree n that divide the original curve $r(\tau)$ at τ_0. Figure 3 shows a three-level subdivision of a cubic Bézier curve for $\tau = 0.5$. It can be proven that the control polygon generated by recursive subdivision converges to the original Bézier curve.

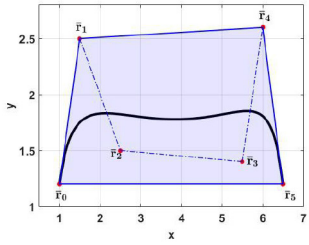

Fig. 2. A quintic Bézier curve contained within the convex hull defined by its 6 control points. The control points are shown in red circles and control polyline in dashed line segments.

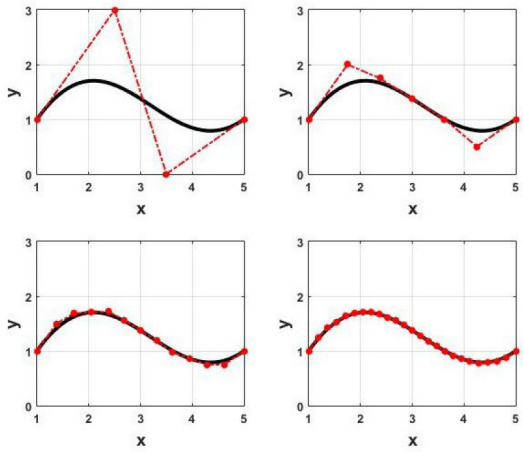

Fig. 3. A cubic Bézier curve (top left) is subdivided into two Bézier curves of the same degree (top right) using the de Casteljau's algorithm. The control polygon generated by recursive subdivision converges to the original Bézier curve.

C. Evaluating Inequalities in Bézier Form

Following the Bézier curve properties mentioned above, any inequality constraint of the form $\bar{h}(\tau) \leq 0, \tau \in [0,1]$, where $\bar{h}(\tau)$ is a Bézier curve, can be satisfied by imposing constraints on its control points. More Specifically, if $\bar{h}(\tau)$ is defined as

$$\bar{h}(\tau) = \sum_{i=0}^{n_h} \bar{h}_i B_{i,n_h}(\tau), \tag{14}$$

then the inequality constraint can be replaced by a finite set of constraints on $\bar{h}_i$,

$$\bar{h}_i \leq 0 \quad for \quad i = 0, \ldots, n_h. \tag{15}$$

The above set of constraints ensures that $\bar{h}(\tau)$ satisfies the inequality constraint over the entire interval $[0, 1]$. However, Ineqs. (15) might be conservative due to the existing gap between the control points and the actual curve. Recursive subdivision of $\bar{h}(\tau)$ using the de Casteljau's algorithm can resolve this issue by refining the representation with closer control points to the curve. This will result in an increased number of inequality constraints, yet the optimization variables continue to be the same [4].

V. Trajectory Generation Using Bézier Curves

In this Section, we introduce the system dynamics and inspect the relation between the orientation and the flat outputs of our choice. Then, we derive constraints that guarantee collision avoidance between two ellipsoid-shaped drones and show that they can be parameterized as Bézier curves.

A. Quadrotor Model

In this letter, the quadrotor equations of motion are described by

$$m\ddot{\mathbf{p}} = -TR\mathbf{e}_3 + mg\mathbf{e}_3, \tag{16}$$

where $R \equiv {}^I_B R \in SO(3)$ is the rotation matrix from body frame $\{B\}$ to inertial frame $\{I\}$, m is the quadrotor mass, $g = 9.8 \frac{m}{s^2}$ is the gravitational acceleration, and $\mathbf{e}_3 = [0\ 0\ 1]^T$. The first term on the right hand side of (16) is the net thrust in the $-\mathbf{z}_B$ direction and the second term is gravity in the $\mathbf{z}_W$ direction.

The quadrotor dynamics (16) with the four inputs is differentially flat [8]. The position $\mathbf{p} \in \mathbb{R}^3$ can be defined as the flat output parameterized as the Bézier curve

$$\mathbf{p}(\tau) = \sum_{k=0}^{n} \mathbf{p}_k B_{k,n}(\tau), \tag{17}$$

where $\mathbf{p}_k \in \mathbb{R}^3$ are the control points and $\tau \in [0, 1]$ is defined as

$$\tau = \frac{t}{T} \tag{18}$$

The linear velocity, $\mathbf{v}$, and linear acceleration, $\mathbf{a}$, can also be expressed as parametric Bézier curves by the first and second derivative of the flat output with respect to time.

The thrust T and rotation matrix R can also be expressed as functions of the flat output and its derivatives, that is,

$$T = m\|\ddot{\mathbf{p}} - g\mathbf{e}_3\|. \tag{19}$$

Assuming the rotation matrix to be parameterized by the Z-Y-X Euler angles $\boldsymbol{\lambda} = [\phi, \theta, \psi]^T$,

$$R = R_z(\psi)R_y(\theta)R_x(\phi), \tag{20}$$

then $R = [\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B]$ can be written as

$$\mathbf{z}_B = \frac{\mathbf{g}\mathbf{e_3} - \ddot{\mathbf{p}}}{\|\mathbf{g}\mathbf{e_3} - \ddot{\mathbf{p}}\|} \quad \mathbf{x}_B = \frac{\mathbf{r} \times \mathbf{z}_B}{\|\mathbf{r} \times \mathbf{z}_B\|}, \quad \mathbf{y}_B = \mathbf{z}_B \times \mathbf{x}_B \tag{21}$$

where $\mathbf{r}$ is defined as

$$\mathbf{r} = [-\sin\psi, \cos\psi, 0]^T \tag{22}$$

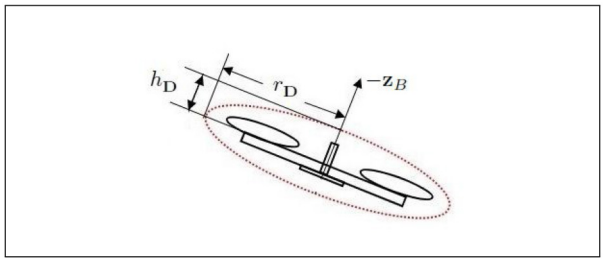

Fig. 4. The drone body is approximated by an ellipsoid that is aligned with the axes of the body frame. The length of the principal semi-axis is determined from the drone frame size.

and ψ is the yaw angle. The yaw angle is considered as the fourth component of the flat output, which can therefore be controlled independently without affecting the path planning. Eq. (21) declares that the orientation can be fully determined from the second derivative of $\mathbf{p}(\tau)$ and the yaw angle.

B. Collision Avoidance Constraint

Here, the drone body is approximated by an ellipsoid E centered at $c = \mathbf{p}(t)$ with its principal axes aligned in the direction of the body frame axes, given by

$$E \equiv \{x | (x - c)^T LL^T (x - c) \leq 1\} \tag{23}$$

where $L = R\Lambda^{\frac{1}{2}}$ and Λ is a 3×3 diagonal matrix with diagonal elements equal to $\frac{1}{r_D^2}$, $\frac{1}{r_D^2}$, and $\frac{1}{h_D^2}$.

A separating hyperplane for two ellipsoids E_1 and E_2 is a hyperplane that has E_1 on one side of it and E_2 on the other side, that is the hyperplane H defined as

$$H \equiv \{x | a^T x - b = 0\} \tag{24}$$

is a separating hyperplane for E_1 and E_2 if

$$a^T x - b \leq 0 \quad \forall x \in E_1 \tag{25}$$
$$a^T x - b > 0 \quad \forall x \in E_2 \tag{26}$$

The ellipsoid E_1 can be equivalently represented as

$$E_1 \equiv \{x | y^T y \leq 1, y \equiv L_1^T (x - c_1)\} \tag{27}$$

With the transformation $y \equiv L_1^T (x - c_1)$, E_1 is transformed into a unit ball at the origin and H into a hyperplane H', defined by

$$H' \equiv \{y | a_1^T y - b_1 = 0\} \tag{28}$$

where $a_1 = L_1^{-1} a$, and $b_1 = b - a^T c_1$. If the inequality

$$|b - a^T c_1| \geq \|a^T L_1^{-T}\| \tag{29}$$

holds, then the unit ball and H' do not intersect and as a consequence E_1 and H do not intersect in the original space. Accordingly, the satisfaction of the following set of inequalities ensures that H is a separating hyperplane for E_1 and E_2:

$$\begin{aligned} &a(\tau)^T \mathbf{p}_1(\tau) - b(\tau) < 0 \\ &a(\tau)^T \mathbf{p}_2(\tau) - b(\tau) > 0 \\ &|b(\tau) - a(\tau)^T \mathbf{p}_i(\tau)| \geq \|a(\tau)^T L_i^{-T}(\tau)\|, \quad i = 1, 2. \end{aligned} \tag{30}$$

The normal vector $a(\tau)$ and the offset $b(\tau)$ are parameterized as Bézier curves to describe the time evolution of

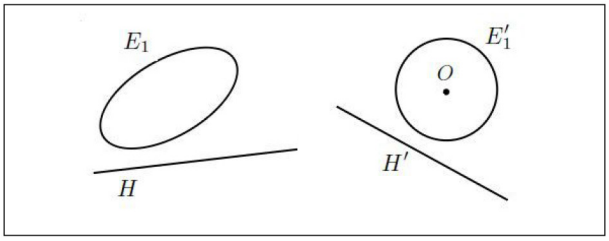

Fig. 5. 2D sketch of an ellipsoid E_1 and a hyperplane H in the original space (left) and the corresponding sketch in the transformed space (right) in which E_1 is a unit ball at the origin.

the separating hyperplane. Having a, b and $\mathbf{p}_i, i = 1, 2$ parameterized as Bézier curves and noting that

$$L_i^{-T} L_i^{-1} = R_i \Big(r_{\mathbf{D}}^2 I - (r_{\mathbf{D}}^2 - h_{\mathbf{D}}^2) \mathbf{e_3} \mathbf{e_3}^T \Big) R_i^T, \tag{31}$$

$\|a^T(\tau) L_i^{-T}(\tau)\|^2$ can be expressed as a Bézier curve according to

$$\begin{aligned} \|a(\tau)^T L_i^{-T}(\tau)\|^2 = r_{\mathbf{D}}^2 \big(a(\tau)^T a(\tau)\big) \\ - (r_{\mathbf{D}}^2 - h_{\mathbf{D}}^2) \big(a(\tau)^T \mathbf{z}_B(\tau)\big)^2. \end{aligned} \tag{32}$$

where $\mathbf{z}_{B,i}$ is fully obtained from $\ddot{\mathbf{p}}_i$ as stated in (21). Therefore, the inequalities in (30) can be effortlessly written as Bézier curves and evaluated with the approach described in Section IV-C. Note that the constraint for collision avoidance between an ellipsoid and a polygon can be easily derived in the same fashion as (30).

VI. Simulation Results

In this section, the efficacy of the proposed method for generating feasible and collision-free trajectories are assessed through different simulations involving one or two drones navigating narrow gaps. Here, the drones are assumed to have a hub-to-hub length of 360 mm, a height of 220 mm, and a propeller length of 230 mm. Therefore, the drone body is approximated by an ellipsoid with $r_D = 295$ mm and $h_D = 110$ mm. The cost function in all the simulations below is given by

$$J = \int_0^1 \left\| \mathbf{p}^{(4)}(\tau) \right\|^2 d\tau + \rho T, \tag{33}$$

where T is the total flight time, and ρ is a scaling coefficient. A smaller ρ results in longer flight times. The lower and upper bounds on the first, second, third, and fourth derivatives of the curve are $7\frac{m}{s}$, $10\frac{m}{s^2}$, $50\frac{m}{s^3}$, and $60\frac{m}{s^4}$ respectively.

In the first example, we consider a single drone traversing a 120 × 48cm gap with $\phi_{gap} = 45°$ (Fig. 6). Flying through such a gap while maintaining a safe distance away from the frame edges would not be possible without proper adjustments of pitch and roll angles. In order to avoid collisions with the gap frame, the trajectory shown with the solid line is generated using separating hyperplanes between the ellipsoid and edges, without directly imposing constraints on attitude angles. Fig. 6 compares this trajectory to the one obtained from the frequently used approach that minimizes the collision risk by aligning the drone's orientation with that of the gap as flying past its center [3]. Observing the input profile corresponding to each trajectory (Fig. 7), we can confirm that the proposed

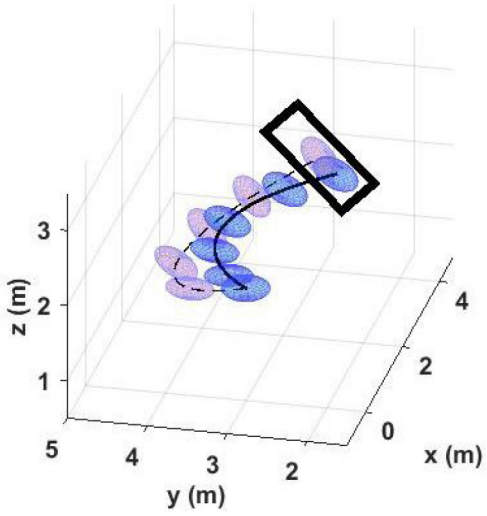

Fig. 6. Trajectories guiding a drone through a narrow gap inclined at 45° with respect to the horizontal plane. The ellipsoids show the flight attitude at different time samples along the trajectories. The objective function values for the solid line and dashed line are 1.0071×10^3 and 2.7777×10^3 respectively.

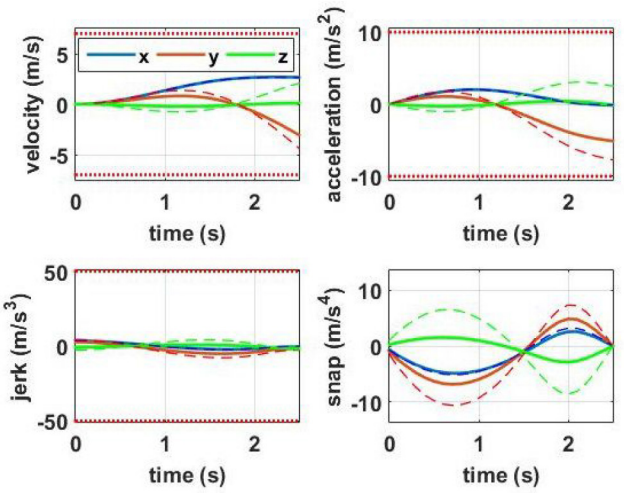

Fig. 7. The velocity, acceleration, jerk, and snap along the x-y-z axis for the generated trajectories in Fig. 6. The dotted lines show the lower and upper bounds.

method outperforms the other one in the sense that it yields a less conservative trajectory, offering a reduced value of the cost function. The trajectory in Fig. 6 is generated using a spatial Bézier curve of degree $n = 6$. The inequalities are evaluated as explained in Section IV-C with control points obtained after two-level subdivisions to secure non-conservative sets of constraints. The optimal solution is computed in 306 ms on a desktop computer with 2.60 GHz i7-4510U CPU and 6.00 GB RAM, using the FORCESPro NLP solver [18].

In the next example, we consider two drones switching their positions. Figure 8 compares the trajectories generated with an ellipsoid model to those generated with the widely used sphere model of the drone body. Clearly, the sphere model takes no notice of the drone's shape and orientation, and imposes a conservative bound on the minimum distance between two drones. This results in an extended arc-length as shown in Fig. 8. The

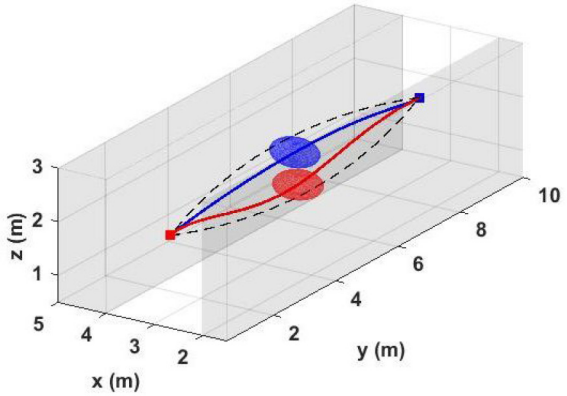

Fig. 8. Comparing collision-free trajectories, for two drones switching positions, generated with an ellipsoid model of the drone body (solid lines) and a sphere model (dashed lines).

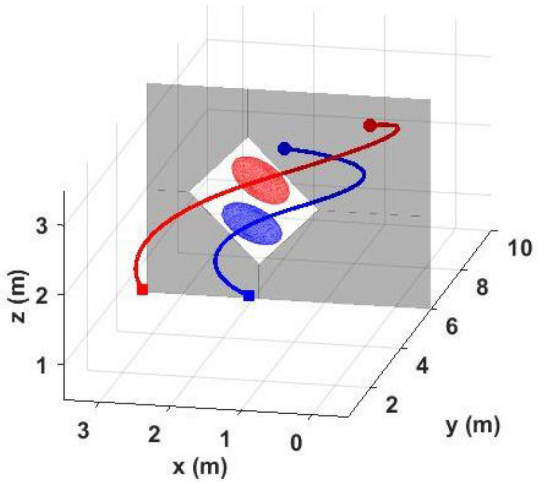

Fig. 9. Collision-free trajectories for steering two drones through a small hole in a wall towards their desired position on the other side. The initial and final positions are shown with squares and circles respectively.

main disadvantage, however, arises when dealing with confined spaces where modeling the drone body with a sphere yields an infeasible problem. This is better shown in the next example (Fig. 9) where two drones are flying through a small hole in a wall to reach their given desired positions on the other side of the wall. The set of constraints in (30) with a separating hyperplane, rotating and translating over time, is incorporated into the optimization problem to guarantee collision avoidance between the two ellipsoids for the entire flight time. The average computation time for obtaining the optimal trajectories in Fig. 9, parameterized as Bézier curves of degree $n = 8$, is 673 ms.

VII. Conclusion

In this letter we addressed the motion planning problem for drones flying in small spaces, where finding feasible trajectories is impossible unless the flight attitude angles are taken into account. In order to include the attitude angles in the trajectory generation problem, we modeled the drone body with an ellipsoid and utilized the separating hyperplane theorem to derive collision avoidance constraints between two ellipsoids. Satisfying the resulting set of constraints enables the planner to generate trajectories that can safely navigate a drone through narrow gaps, or maneuver multiple drones in a tight space without collisions. The seamless integration of the constraints into our proposed approach to efficient evaluation of inequalities in terms of Bézier curves allows generating feasible and collision-free trajectories while meeting timing constraints in real-time applications.

References

[1] T. Hirata and M. Kumon, "Optimal path planning method with attitude constraints for quadrotor helicopters," in *Proc. Int. Conf. Adv. Mechatronic Syst.*, Aug. 2014, pp. 377–381.

[2] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and IMU," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 404–411, Apr. 2017.

[3] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza, "Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 5774–5781.

[4] B. Sabetghadam, R. Cunha, and A. Pascoal, "Real-time trajectory generation for multiple drones using Bézier Curves," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9276–9281, 2020.

[5] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *Int. J. Robust Nonlinear Control*, vol. 8, no. 11, pp. 995–1020, Sep. 1998.

[6] S. Liu *et al.*, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1688–1695, Jul. 2017.

[7] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1476–1483.

[8] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 2520–2525.

[9] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. Robot. Res.*, 2016, pp. 649–666.

[10] V. Cichella *et al.*, "Safe coordinated maneuvering of teams of multirotor unmanned aerial vehicles: A cooperative control framework for multivehicle, time-critical missions," *IEEE Control Syst. Mag.*, vol. 36, no. 4, pp. 59–82, Aug. 2016.

[11] T. Mercy, R. Van Parys, and G. Pipeleers, "Spline-based motion planning for autonomous guided vehicles in a dynamic environment," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 6, pp. 2182–2189, Nov. 2018.

[12] T. Mercy, W. Van Loock, and G. Pipeleers , "Real-time motion planning in the presence of moving obstacles," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2016, pp. 1586–1591.

[13] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE(3)," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2439–2446, Jul. 2018.

[14] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 2872–2879.

[15] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, May 2006.

[16] G. G. Rigatos, *Nonlinear Control and Filtering Using Differential Flatness Approaches: Applications to Electromechanical Systems*, vol. 25. Cham, Switzerland: Springer, 2015.

[17] R. T. Farouki, "The Bernstein polynomial basis: A centennial retrospective," *Comput. Aided Geometr. Design*, vol. 29, no. 6, pp. 379–419, Aug. 2012.

[18] A. Domahidi and J. Jerez. (2019). *Embotech AG: Forces Professional*. [Online]. Available: https://embotech.com/FORCES-Pro