Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 53-2 (2020) 9276–9281

Real-time Trajectory Generation for Multiple Drones using Bézier Curves ⋆

Bahareh Sabetghadam, Rita Cunha, António Pascoal

Laboratory of Robotics and Engineering Systems (LARSyS), ISR/IST
University of Lisbon
Lisbon, Portugal
(e-mail: bsabetghadam,rita,antonio@isr.ist.utl.pt).

Abstract: Practical applications of drones are expanding into many new areas due to their fast-evolving technology. Looking further into the future, it is very likely that applications will require more than one drone to tackle a specific task, calling for reliable and efficient algorithms that can generate collision-free trajectories for multiple drones, under timing constraints of real-time applications. In this paper, we study a motion planning method based on the Bézier parametrization of spatial paths with a special focus on the less addressed issue in this method, (inefficient) constraint evaluation, that might hinder its use in real-time trajectory generation for multi-drone applications. We take advantage of the Bézier curves properties to obtain a small-scale optimization problem and find a finite set of inequalities that guarantee constraints satisfaction. We also propose a method to lower the conservatism in the resulting set of inequalities without the need to use unnecessary high-degree Bézier curves. Numerical results illustrate the efficacy of the presented method in reducing the computational costs associated with generating collision-free trajectories for multiple drones and re-planning them online with a receding horizon.

Keywords: Trajectory Generation, Real-time, Bezier Curves.

1. INTRODUCTION

In recent years, there has been growing interest in use of drones in many different areas due to the functionality and benefits they offer in terms of mobility, speed, and accessibility. In order to be successfully adopted in emerging application areas, such as transport, construction, and media, the industry should push for highly autonomous drones that, along with several other features, are capable of planning their paths with no (or minimal) supervision. Future applications may also involve a fleet of drones to accomplish a task cooperatively, putting additional strain on the drones' motion planning system. It is therefore imperative to develop algorithms that not only can reliably generate collision-free trajectories but also are efficient enough to satisfy real-time constraints.

The literature on multiple vehicles motion planning is extensive (Goerzen et al. (2010)), yet, the majority of the existing methods tend to be too slow as the number of vehicles increases, or too complicated to be implemented on the computing module onboard drones. One class of methods that efficiently extends beyond the single-vehicle applications includes polynomial-based methods, in which the dimensionality in the problem is reduced by using polynomial parameterization of the paths. However, the result, as will be explained later in this paper, is a semi-infinite optimization problem involving a finite number of variables and an infinite number of constraints. In order to obtain a computationally tractable problem, different approaches have been employed.

(Van Nieuwstadt and Murray (1998)) is one of the first papers that studied the evaluation of inequality constraints in polynomial-based methods. The paper proposes three methods to construct a finite constraint set for linear inequalities, and also a method to approximate nonlinear constraints by linear constraints, which can be done once (off-line) assuming that the constraints do not change in the course of a mission.

Time gridding is one of the approaches that has been widely used for obtaining a standard optimization problem (Mellinger and Kumar (2011)). This method, though straightforward, can not guarantee that the constraints are satisfied for the whole travel time as they are only evaluated on a finite number of points. Using fine discretization can remedy this issue, however, it will increase the number of constraints as well as the computation time.

Special types of polynomial parametric curves, such as Spline (Van Parys and Pipeleers (2017), Mercy et al. (2016)) and Bézier (Choi et al. (2008), (Choe et al., 2015)), have gained popularity in motion planning as they can add an intuitive and geometric interpretation to the design and also provide computational benefits to the problem. Mercy et al. (2017) exploits the 'convex hull' property to convert the infinite set of constraints on a spline curve to a finite set of constraints on its coefficients. The so-called B-spline relaxation can bring about a significant

⋆ This research was supported in part by the MarineUAS project under the Marie Curie Skolodowska grant agreement No 642153, the H2020 EU Marine Robotics Research Infrastructure Network (Project ID 731103)

10.1016/j.ifacol.2020.12.2380

reduction of the computational effort provided that all constraints are expressed in the form of splines. To reduce the conservatism, the paper also suggests representing the curves in a higher dimensional basis by inserting extra knots, which translates into having more constraints.

In Cichella et al. (2018) the De Casteljau's algorithm together with the Gilbert-Johnson-Keerthi distance algorithm are used to ensure constraint satisfaction on the entire Bezier curve. The employed method, as explained in Chen et al. (2009), can find the curve extrema to any desired accuracy, but the computational cost associated with the resulting non-smooth functions is a major drawback to this approach.

In this paper, we will study the problem of real-time motion planning for multiple vehicles. We will use Bézier curves to parameterize the trajectories, due to their ease of computation and stability. To address the semi-infinite constraints, we exploit the Bézier curve properties to obtain a finite set of constraints that guarantee constraints satisfaction for the whole travel time. Then, we exploit the de Casteljau's algorithm to reduce the conservatism in the obtained set of constraints. With the proposed method in this paper, we can achieve the desired accuracy in the constraint evaluation, while avoiding excessive computations for remote timeslots in a receding-horizon planner. The efficacy of the proposed method is demonstrated by two numerical simulations.

The remainder of this paper is organized as follows: Section 2 explains the polynomial-based motion planning method for differentially flat systems; Section 3 provides a brief introduction to Bézier curves and their properties. Section 4 proposes a method for converting the semi-infinite constraints to a finite set of constraints; Section 5 shows two different simulation examples; and Section 6 discusses conclusions and future work.

2. POLYNOMIAL-BASED MOTION PLANNING FOR DIFFERENTIALLY FLAT SYSTEMS

The motion planning problem that we address in this paper consists of generating a feasible trajectory to steer a vehicle to a desired final position while optimizing a performance index and satisfying a set of constraints. This problem can be formulated as the following optimization problem.

$$\begin{aligned}
&\underset{x(.),u(.)}{\text{minimize}}\ J(x(t),u(t)) && && (1)\\
&\text{subject to } \dot{x}(t)-f(x(t),u(t))=0 && t\in[0,T] && (1.a)\\
&\qquad x(0)=x_0 \quad x(T)=x_f && && (1.b)\\
&\qquad u(0)=u_0 \quad u(T)=u_f && && (1.c)\\
&\qquad h(x(t),u(t))\le 0 && t\in[0,T] && (1.d)
\end{aligned}$$

The function, $J(x(t),u(t))$, evaluates the quality of the trajectory based on the mission requirements and may include different terms to guarantee smoothness and/or energy efficiency of the trajectory. To ensure a feasible trajectory, the vehicle's model is considered as an ODE in (1.a), 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 initial and final conditions on the state and the input of the vehicle

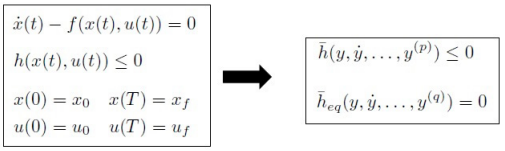


Fig. 1. Converting the constraints in (1) to equivalent constraints in the flat space using (3).

model are considered in (1.b) and (1.c). The problem inequality constraints, such as state and input bounds and obstacle avoidance, are included in $h(x(t),u(t))$, which should hold for the entire travel time $[0,T]$. The optimal solution, $(x^*(t),u^*(t))$, will minimize the objective function (1) and satisfy the set of constraints (1.a)-(1.d).

Polynomial-based methods take advantage of the differential flatness property of a dynamic system to deal with the infinite-dimensional optimization problem (1). The nonlinear system $\dot{x}=f(x,u)$ is called "differentially flat" (LaValle (2006)) if there exists a set of flat outputs $y\in\mathbb{R}^{n_y}$, $n_y=n_u$, of the form

$$y=d(x,u,\dot{u},...,u^{(k)}) \tag{2}$$

such that

$$\begin{aligned}
x&=g(y,\dot{y},...,y^{(p_1)})\\
u&=g'(y,\dot{y},...,y^{(p_2)})
\end{aligned} \tag{3}$$

In other words, the states and the inputs of a differentially flat system can be expressed as functions of the flat outputs and a finite number of their derivatives.

Using this property, trajectories consistent with the dynamics (1.a) can be planned in the space of flat outputs Y. Figure 1 shows the transformation of the original constraints (1.a)-(1.d) in Y, where the dynamic constraint is trivially satisfied and the equality and inequality constraints are expressed as

$$\begin{aligned}
&\bar{h}(y,\dot{y},\dots,y^{(p)})\le 0 && t\in[0,T]\\
&\bar{h}_{eq}(y,\dot{y},\dots,y^{(q)})=0 && t\in\{0,T\}
\end{aligned} \tag{4}$$

The above set of constraints define the feasible region in Y for the new optimization problem. Once the optimal $y^*(t)$ is obtained, the state and the input trajectories can be derived by simple differentiations using (3).

The components of the flat output $y(t)$ are differentially independent, i.e. there is no differential relation of the form $R(y,\dot{y},...,y^{(r)})=0$ (Rigatos (2015)). Polynomial-based methods employ this property to individually describe the evolution of each flat output with a polynomial function

$$y_j(t)=\sum_{k=0}^{n_j} a_{jk}\Phi_k(t) \quad j\in\{1,\dots,n_y\} \tag{5}$$

where a_{jk} are the coefficients and $\Phi_k(t)$ are the polynomial basis functions. Using the parameterization in (5), the trajectory generation problem (1) is converted into a semi-infinite optimization problem with a finite number of variables and an infinite number of constraints.

$$\underset{\substack{a_{jk}\\ k=1,\dots,n_j\\ j=1,\dots,n_y}}{\text{minimize}} \; J(y(t), \dot{y}(t), \dots, y^{(p)}(t)) \tag{6}$$

$$\text{s.t.} \quad \bar{h}(t, \bar{a}) \leq 0 \qquad t \in [0, T] \tag{6.a}$$

$$\bar{h}_{eq}(t_i, \bar{a}) = 0 \qquad t_i \in \{0, T\} \tag{6.b}$$

where $\bar{a}$ is the vector of variables including all coefficients a_{jk}. In the following, we use Bernstein basis polynomials and Bézier curves to parameterize the output, and exploit their properties to deal with the inequality constraints (6.a).

3. BÉZIER CURVES

A Bézier curve $r(\tau)$ is a parametric polynomial that is expressed in terms of Bernstein basis functions $B_{i,n}(\tau)$ and a set of control points $\bar{r}_i$ as

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

where $\bar{r}_i \in \mathbb{R}^2$ for planar curves and $\bar{r}_i \in \mathbb{R}^3$ for spatial curves. The Bernstein polynomial of degree n is defined over the interval $[0, 1]$ as

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

where $\binom{n}{i} = \frac{n!}{i!(n-i)!}$. The Bernstein polynomials form a partition of unity, i.e. the sum of $B_{i,n}$ over all i is equal to 1 for any $\tau \in [0, 1]$.

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

Figure (2) shows the Bernstein polynomials of degree 4. The Bernstein basis together with the control points determine the shape of a Bézier curve. Figure (3) shows a Bézier curve of degree 4 which is coincident with its control points at the two ends. A Bézier curve always starts and ends at its first and last control points respectively.

$$r(0) = \bar{r}_0 \quad r(1) = \bar{r}_n \tag{10}$$

Moreover, a Bézier curve lies within the convex hull defined by its control points. This property states that the entire curve, except for the two endpoints, will be inside a computable region.

Bézier curves have several properties and algorithms that can be extremely useful in trajectory generation applications. We provide two of the most important ones below.

De Casteljau's Algorithm:

Using the definition in (7) to evaluate a point on a Bézier curve can cause numerical instability for high degree curves. The de Casteljau's algorithm has established an alternative method to find $r(\tau)$ using only the control points $\bar{r}_i$.

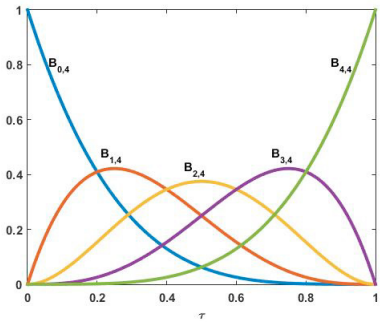


Fig. 2. Bernstein basis functions of degree 4.

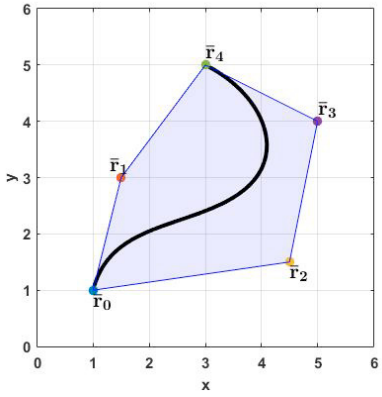


Fig. 3. A Bézier curve of degree 4 contained within the convex hull defined by its 5 control points.

Given a specific value of τ, the algorithm starts by dividing each polyline between $\bar{r}_i$ and $\bar{r}_{i+1}$, $i = 0, \dots, n-1$, in a ratio of $\tau : 1 - \tau$ to obtain n new points indicated by $\bar{r}_i^{(1)}$. Repeating the subdivision n times yields a single point $\bar{r}_0^{(n)}$ which can be shown to be the point on the curve corresponding to τ. For a Bézier curve of degree n with $n+1$ control points and $\tau = \tau_0$, the above procedure can be expressed as the following recursive formula:

$$\begin{aligned} &\bar{r}_i^{(0)} = \bar{r}_i && i = 0, \dots, n \\ &\bar{r}_i^{(j)} = (1 - \tau_0)\bar{r}_i^{(j-1)} + \tau_0 \bar{r}_{i+1}^{(j-1)} && j = 1, \dots, n \\ & && i = 0, \dots, n-j \\ &\bar{r}_0^{(n)} = r(\tau_0) \end{aligned} \tag{11}$$

The de Casteljau's algorithm also provides a method for subdividing a Bézier curve into two Bézier curves of the same degree at point τ_0, with the two sets of $n+1$ control points for the pieces $[0, \tau_0]$ and $[\tau_0, 1]$ defined as

$$\begin{aligned} &[0, \tau_0] : && \bar{r}_0^{(0)}, \bar{r}_0^{(1)}, \dots, \bar{r}_0^{(n)} \\ &[\tau_0, 1] : && \bar{r}_n^{(0)}, \bar{r}_{(n-1)}^{(1)}, \dots, \bar{r}_0^{(n)} \end{aligned} \tag{12}$$

Figure (4) illustrates the application of the De Casteljau's algorithm for dividing a cubic Bezier curve into two segments at different points.

Degree Elevation:

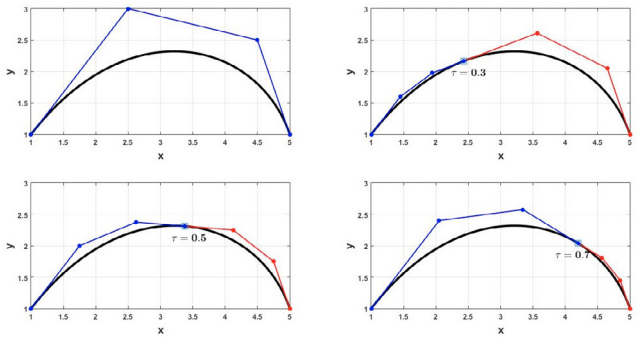


Fig. 4. A cubic Bézier curve is divided into two Bézier curves of the same order, at $\tau = 0.3, 0.5, 0.7$, using the de Casteljau's algorithm.

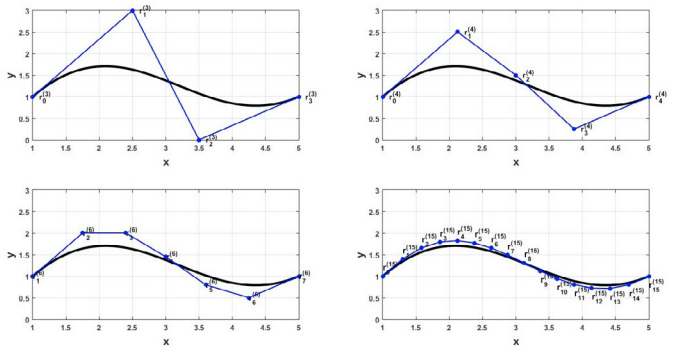


Fig. 5. A cubic Bézier curve is expressed in terms of higher degree basis functions, $n = 4, 6, 15$, using the degree elevation algorithm.

The Degree Elevation algorithm is extremely useful for applications involving two or more Bézier curves with different degrees. It allows increasing the degree of a Bézier curve , i.e. changing the basis, without changing its shape. The Bernstein basis polynomials of degree n, $B_{i,n}$ can be expressed in terms of the Bernstein polynomials of degree $n+1$:

$$B_{i,n}(\tau) = (1 - \frac{i}{n+1})B_{i,n+1}(\tau) + \frac{i+1}{n+1}B_{i+1,n+1} \quad i = 0, 1, \ldots, n \quad (13)$$

More generally, $B_{i,n}$ can be written in terms of basis functions of degree $n + r$ as

$$B_{i,n}(\tau) = \sum_{j=i}^{i+r} \frac{\binom{n}{i}\binom{r}{j-i}}{\binom{n+r}{j}} B_{j,n+r}(\tau) \quad (14)$$

Using (13) and (14), we can obtain the corresponding set of control points that converts a Bézier curve to a higher degree curve. The following formula gives the new $n + 2$ control points for the case of increasing the degree by one:

$$\bar{r}_i^{n+1} = \frac{i}{n+1}\bar{r}_{i-1}^{n} + (1 - \frac{i}{n+1})\bar{r}_i^{n} \quad i = 0, \ldots, n+1 \quad (15)$$

It can be implied from (15) that each polyline will exactly contain one new control point. Figure (5) shows a cubic Bézier curve expressed in terms of basis functions of degree $4, 6$, and 15 with the new set of control points obtained using (15).

4. TRAJECTORY GENERATION USING BÉZIER CURVES

In this Section, we explain our approach to deal with the semi-infinite constraints in (6) using Bézier parameterization of trajectories.

The example we study below is a motion planning problem for multiple drones, where collision-free trajectories are required to guide the drones from their initial positions to final positions. The problem consists of boundary conditions, such as initial position, speed, and acceleration of the drones as well as their desired final state. It also includes other constraints, such as speed and acceleration bounds and inter-vehicle collision avoidance, expressed in terms of inequalities.

In the following, we adopt a simplified model for each of the drones (quadrotors) given by a double integrator of the form

$$\begin{aligned} \dot{\mathbf{p}}_Q &= \mathbf{v}_Q \\ \dot{\mathbf{v}}_Q &= \mathbf{a}_Q, \end{aligned} \quad (16)$$

where $\mathbf{v}_Q = [v_x \ v_y \ v_z]^T \in \mathbb{R}^3$ is the linear velocity and $\mathbf{a}_Q = [a_x \ a_y \ a_z]^T \in \mathbb{R}^3$ is the linear acceleration. If we consider the flat output to be defined as

$$\mathbf{y} = \mathbf{p}_Q \quad (17)$$

then $\mathbf{y}(t)$ can be parameterized with a n-degree spatial Bézier curve as

$$\mathbf{y}(\tau) = \sum_{k=0}^{n} y_k B_{k,n}(\tau) \quad (18)$$

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

$$\tau = \frac{t}{T} \quad (19)$$

The linear speed, $\mathbf{v}_Q$, and linear acceleration, $\mathbf{a}_Q$, can also be expressed as parametric Bézier curves of degree $n-1$ and $n-2$, respectively, by simply differentiating the flat output $\mathbf{y}$.

$$\begin{aligned} \mathbf{v}_Q(\tau) &= \sum_{k=0}^{n-1} v_k B_{k,n}(\tau) \\ \mathbf{a}_Q(\tau) &= \sum_{k=0}^{n-2} a_k B_{k,n}(\tau) \end{aligned} \quad (20)$$

where the control points v_k and a_k are obtained as

$$v_k = \frac{n(y_{k+1} - y_k)}{T} \qquad k = 0, \dots, n-1$$
$$a_k = \frac{n(n-1)(y_{k+2} - 2y_{k+1} + y_k)}{T^2} \qquad k = 0, \dots, n-2 \tag{21}$$

Considering (18) and (20), the initial and final conditions on the position, speed and, acceleration of the vehicle will determine the control points y_0, y_1, y_2 and y_{n-2}, y_{n-1}, y_n.

The problem we consider includes inequality constraints such as speed and acceleration bounds

$$\underline{\mathbf{v}}_Q \leq \mathbf{v}_Q(\tau) \leq \bar{\mathbf{v}}_Q \quad \tau \in [0,1]$$
$$\underline{\mathbf{a}}_Q \leq \mathbf{a}_Q(\tau) \leq \bar{\mathbf{a}}_Q \tag{22}$$

and inter-vehicle collision avoidance

$$\left\| \mathbf{p}_Q^{[i]}(\tau) - \mathbf{p}_Q^{[j]}(\tau) \right\|_2^2 \geq R^2 \quad \tau \in [0,1] \tag{23}$$

which must be satisfied for all $\tau \in [0,1]$. Similar constraints to (23) can be added to enforce vehicle-obstacle collision avoidance with R being the safe distance to be kept between the vehicles and obstacles.

The above inequality constraints can be rewritten in terms of Bézier curves using (18) and (20). In the following, we present an efficient method for dealing with the infinite set of constraints in (22) and (23).

4.1 Efficient Evaluation of Inequality Constraints

Here, we assume that all inequality constraints can be expressed in terms of Bézier curves as

$$\mathbf{h}(\tau) = \sum_{i=0}^{n_h} \bar{h}_i B_{i,n_h}(\tau) \tag{24}$$

Considering the partition of unity property (9), $h(\tau) \leq 0$ can be converted into the following finite set of constraints on the control points:

$$\bar{h}_i \leq 0 \quad for \quad i = 0, \dots, n_h \tag{25}$$

The above set of constraints can be conservative due to the existing gap between the control points and the actual curve. As explained in the previous section, degree elevation and de Casteljau's algorithm set up different ways of finding new control points that are closer to the curve.

The degree elevation algorithm gives new control points for the curve expressed in higher dimensional basis polynomials, and the de Casteljau's algorithm provides the control points for different segments of the curve expressed as bézier curves of the same degree. In both algorithms, the control points approaches the curve as we keep repeating the process.

Both approaches can be utilized to reduce the conservatism in (25), by finding closer control points to the curve without increasing the number of variables, yet, while the former provides control points that are distributed along $h(\tau)$, the latter allows generating more new control points on a specific part of the curve. This can be extremely useful when solving the problem online in a receding horizon manner as excessive computations for remote horizon can be avoided.

5. SIMULATION RESULTS

In this section, the efficacy of the method described above is evaluated through two different examples. The model and constraints in the two examples are as described in Section 4, and the objective function is a measure of the trajectories' smoothness defined as

$$J = \sum_{j=0}^{N_v} \int_0^1 \left\| \frac{d^k \mathbf{y}^{[j]}(\tau)}{d\tau^k} \right\|^2 d\tau \tag{26}$$

where N_v is the number of drones. The computation times mentioned below are all obtained on a desktop computer with 2.60 GHz i7-4510U CPU and 6.00 GB RAM.

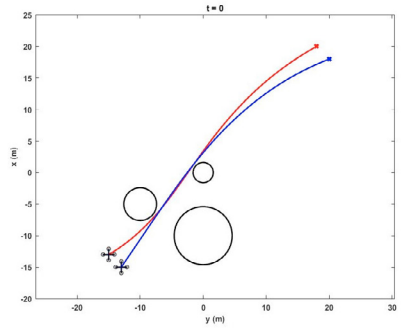
In the first example, we consider two drones flying in an environment with static and moving obstacles. The goal is to generate smooth trajectories that guide the drones to their desired final positions, while avoiding the obstacles. Also, a minimum distance of $2m$ must be kept between the two drones during the entire flight time. In order to compensate for the uncertainties in the obstacles' positions, the optimization problem (6) is solved in a receding horizon manner taking into account the most recent measurements of positions.

Figure (6) shows the trajectories generated at different time instances. At $t = 0$ the whole trajectory, from the drone's initial position to its final position, is generated using planar Bézier curve of degree 6. At $t = 5.6s$ and $t = 6.8s$, the trajectories are re-planned to avoid collision with the new obstacles detected along the horizon. When re-planning the trajectories, it is necessary to consider the continuity conditions of the curve and its derivatives to ensure that the two segments of the path are joined smoothly.

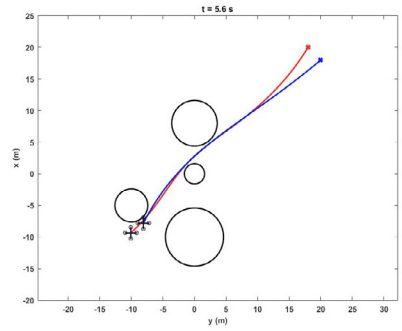
The method described in section 4.1 is utilized for dealing with the inequality constraints. With the new control points obtained by the de Castejau's algorithm, the trajectories of the two drones are generated by solving an optimization problem with a small number of variables and constraints. The average computation time for solving this problem is 146 ms, which is much faster than using the GJK-based method with an average computation time of 1547 ms, without compromising on the performance of the trajectories.

In the second example, we consider generating collision-free trajectories for five drones. The trajectories should be generated such that the drones arrive at their final positions at the same time, with the desired speed and acceleration.

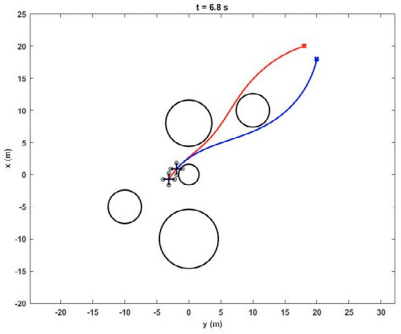
Figure (7) shows the trajectories for 5 drones generated such that they reach their final points in minimum time while avoiding inter-vehicle collisions. In this example the trajectories are parameterized with spatial Bézier curve of degree 8. Using the method described in this paper, the computation time for solving this problem is 647 ms. This



(a) $t = 0$



(b) $t = 5.6s$



(c) $t = 6.8s$

Fig. 6. Trajectories for two drones re-planned at different time instances to deal with the uncertain environment and avoid collision with obstacles.

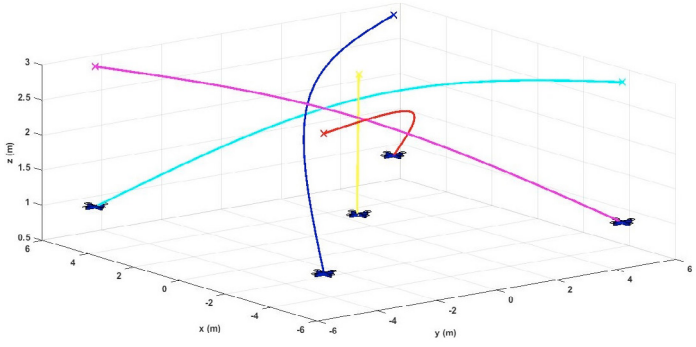


Fig. 7. Collision-free trajectories for 5 drones generated with Bézier curves of degree 8.

method does not only guarantee constraint satisfaction for the entire flight time but also provides a 10X faster computation time compared to the time gridding method with a sampling period of $1s$.

6. CONCLUSIONS

In this paper we considered the problem of generating collision-free trajectories for multiple drones using a reliable and efficient algorithm that can meet the timing constraints of real-time applications. We parameterized the trajectories using Bézier curves and exploited their properties to obtain a small set of constraints on the control points. In order to reduce the conservatism of the constraints, we utilized the de Casteljau's algorithm to find additional control points without increasing the number of optimization variables or using unnecessary high degree basis functions. The obtained results demonstrate the ability of the proposed method to speed up the solution time of generating and re-planning trajectories. Validating the proposed approach with real experiments in a dynamic environment is left for future work.

REFERENCES

Chen, X.D., Chen, L., Wang, Y., Xu, G., Yong, J.H., and Paul, J.C. (2009). Computing the minimum distance between two bézier curves. *Journal of Computational and Applied Mathematics*, 229(1), 294–301.

Choe, R., Puig, J., Cichella, V., Xargay, E., and Hovakimyan, N. (2015). Trajectory generation using spatial pythagorean hodograph bézier curves. In *AIAA Guidance, Navigation, and Control Conference*, 0597.

Choi, J.w., Curry, R., and Elkaim, G. (2008). Path planning based on bézier curve for autonomous ground vehicles. In *Advances in Electrical and Electronics Engineering-IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, 158–166. IEEE.

Cichella, V., Kaminer, I., Walton, C., Hovakimyan, N., and Pascoal, A. (2018). Bernstein approximation of optimal control problems. *arXiv preprint arXiv:1812.06132*.

Goerzen, C., Kong, Z., and Mettler, B. (2010). A survey of motion planning algorithms from the perspective of autonomous uav guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4), 65.

LaValle, S.M. (2006). Planning algorithms.

Mellinger, D. and Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, 2520–2525. IEEE.

Mercy, T., Van Loock, W., and Pipeleers, G. (2016). Real-time motion planning in the presence of moving obstacles. In *Control Conference (ECC), 2016 European*, 1586–1591. IEEE.

Mercy, T., Van Parys, R., and Pipeleers, G. (2017). Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology*.

Rigatos, G.G. (2015). *Nonlinear control and filtering using differential flatness approaches: applications to electromechanical systems*, volume 25. Springer.

Van Nieuwstadt, M.J. and Murray, R.M. (1998). Real-time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 8(11), 995–1020.

Van Parys, R. and Pipeleers, G. (2017). Spline-based motion planning in an obstructed 3d environment. *IFAC-PapersOnLine*, 50(1), 8668–8673.