2015 IEEE 54th Annual Conference on Decision and Control (CDC)
December 15-18, 2015. Osaka, Japan

Avoiding Multiple Collisions through Trajectory Replanning using Piecewise Bézier Curves

Syed Bilal Mehdi, Ronald Choe and Naira Hovakimyan

***Abstract*— This paper presents a collision prediction and avoidance algorithm based on piecewise Bézier curves. Upon detection of an obstacle, the algorithm checks for a possible collision and modifies the vehicle's planned trajectory, if needed. Under mild assumptions, the replanned trajectory is guaranteed to avoid the collision and satisfy mission-specific constraints. The bounds on changes in position, velocity and acceleration caused by trajectory replanning are computable a priori. These bounds can be used during the mission-planning phase to guarantee satisfaction of vehicle dynamic constraints by the replanned trajectory. The algorithm presented here can be used to avoid multiple collisions under weaker assumptions as compared to previous results.**

I. INTRODUCTION

Under the NextGen Air Transportation System (NextGen) [1], highly integrated missions are envisioned where groups of multiple capability vehicles operate in the National Airspace System to perform coordinated missions. To this end, Unmanned Aerial Vehicle (UAV) technology needs to be simplified to allow for a single operator to manage multiple UAVs. Furthermore, with human life at risk, safe operation with guarantees is of utmost importance.

Guidance and autopilot systems now exist onboard remotely piloted vehicles for reduction of workload on the operator. Furthermore, algorithms for coordinated mission execution by multiple vehicles can now be found in literature [2-4]. Initial trajectory generation for these missions, however, is still a challenging problem. This task is inherently complicated as these trajectories need to satisfy not only mission-specific constraints but also vehicle dynamic constraints.

In [5] and [6], we provided an *offline* trajectory-generation framework for multi-vehicle time-coordinated missions. In this framework, we alleviate computational load of the problem at hand through the use of Bézier curves for trajectory representation. Bézier curves exhibit several useful properties that allow for fast and efficient verification of the dynamic constraints. Combined with efficient optimization procedures, the framework allows to generate trajectories in a practically feasible way.

The problem of collision avoidance has been explored with widely varying approaches. These include methods based on velocity obstacle [7], artificial potential field [8], collision states [9] and speed adjustment [4], to name a few. In [10], however, we approach the problem of collision avoidance as *online* trajectory replanning through exploitation of geometric properties of Bézier curves. This approach, based on trajectory replanning through use of a small set of avoidance procedures, can also be applied to multi-vehicle missions. In [10], we show that under mild assumptions, the algorithm guarantees collision avoidance along with satisfaction of mission and vehicle dynamic constraints.

Although [10] shows promising results, the applicability of the algorithm is limited due to the fact that it can be applied only once during a mission. This stems from the fact that the algorithm applies only to trajectories that are represented as a single Bézier curve. However, the replanned trajectory is in fact a piecewise Bézier curve and, hence, does not lend to subsequent application of the algorithm.

In this paper we extend the results of [10]. We modify the algorithm to the cases of trajectories represented as piecewise Bézier curves. Thus, the algorithm can be applied to replanned piecewise trajectories, thereby allowing multiple applications. Furthermore, some restrictive assumptions of [10] have been relaxed in this paper.

The paper is organized as follows: we state the mathematical background for the algorithm in Section II. The precise mathematical problem formulation is then provided in Section III, followed by the algorithm in Section IV. Lastly, we present simulation examples in Section V and finish the paper with conclusions and future work in Section VI.

II. BACKGROUND

For trajectories represented as (piecewise) Bézier curves, efficient algorithms exist in the literature that allow for fast computation of parameters such as velocity, acceleration and distance between such trajectories. In what follows, we briefly introduce Bézier curves and a few relevant algorithms.

A. Bézier Curves

Bézier curves [11] are polynomial curves defined over the finite interval $[0, 1]$ by their control points $\bar{\mathbf{r}}_k$ as

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

where n is the degree of the polynomial and $b_k^n(\zeta)$ are the Bernstein polynomials defined as

$$b_k^n(\zeta) = \binom{n}{k}(1-\zeta)^{n-k}\zeta^k, \quad \zeta \in [0, 1]. \tag{2}$$

This work is supported by NASA LaRC.

S. B. Mehdi is with the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign. Email: mehdi1@illinois.edu

R. Choe is with the Aerospace Department, University of Illinois at Urbana-Champaign. Email: choe19@illinois.edu

N. Hovakimyan is with the Faculty of the Department of Mechanical Science and Engineering, University of Illinois at Urbana-Champaign. Email: nhovkim@illinois.edu

B. The de Casteljau Algorithm

The de Casteljau algorithm [11] is used to subdivide a Bézier curve into two independent ones. Specifically, given a Bézier curve and a scalar $\zeta_{\text{div}} \in (0,1)$, the de Casteljau algorithm outputs two Bézier curves $\mathbf{r}_1$ and $\mathbf{r}_2$ such that

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

C. Minimum Distance Calculation

Given two Bézier curves $\mathbf{r}_a(\zeta)$ and $\mathbf{r}_b(\zeta)$ with $\zeta \in [0,1]$, the minimum distance algorithm [12] efficiently calculates

$$\min_{\zeta_a,\zeta_b\in[0,1]} ||\mathbf{r}_a(\zeta_a) - \mathbf{r}_b(\zeta_b)||, \quad \arg\min_{\zeta_a,\zeta_b\in[0,1]} ||\mathbf{r}_a(\zeta_a) - \mathbf{r}_b(\zeta_b)||.$$

III. Problem Formulation

Although our algorithm easily extends to multi-vehicle time-coordinated missions, we constrain ourselves here to only one vehicle for ease of notation. We consider this vehicle to be following its trajectory $\mathbf{p} : [0, t^{\text{f}}] \to \mathbb{R}^2$ during the time interval $t \in [0, t^{\text{f}}]$. Furthermore, we assume $\mathbf{p}(t)$ to be a piecewise Bézier curve with C^2-continuity for smoothness. Specifically, we consider a sequence of $m+1$ time instants in ascending order

$$\lambda = \{\lambda^0 = 0, \lambda^1, \lambda^2, \cdots, \lambda^m = t^{\text{f}}\},$$

and the trajectory defined for $t \in [0, t^{\text{f}}]$ to be represented as a piecewise Bézier curve. Specifically, for each interval $t \in [\lambda^{i-1}, \lambda^i] \subseteq [0, t^{\text{f}}]$, we consider

$$\mathbf{p}(t) = \mathbf{p}^i\left(\frac{t-\lambda^{i-1}}{\lambda^i - \lambda^{i-1}}\right) = \sum_{k=0}^{n} \bar{\mathbf{p}}_k^i b_k^n\left(\frac{t-\lambda^{i-1}}{\lambda^i-\lambda^{i-1}}\right),$$

where $\bar{\mathbf{p}}_k^i$ are control points of the trajectory, $n > 6$ and $i = 1, 2, \cdots, m$.

A few points are in order here. First, note that the complete trajectory $\mathbf{p}(t)$ is not necessarily a single Bézier curve. This is possible, however, such as when $\lambda = \{0, t^{\text{f}}\}$. Second, we can always find a new but equivalent representation of $\mathbf{p}(t)$ by adding a new element $\lambda^{\text{new}} \in (\lambda^{i-1}, \lambda^i)$ to the sequence λ for some $i \in \{1, 2, \cdots, m\}$. This can be performed using the de Casteljau algorithm and subdividing $\mathbf{p}^i$ appropriately.

We consider the vehicle trajectory to satisfy

$$\max_{t\in[0,t^{\text{f}}]} ||\mathbf{v}(t)|| \le v_{d,\max}, \qquad \max_{t\in[0,t^{\text{f}}]} ||\mathbf{a}(t)|| \le a_{d,\max},$$

where $\mathbf{v}(t) = \frac{d\mathbf{p}(t)}{dt}$ and $\mathbf{a}(t) = \frac{d^2\mathbf{p}(t)}{dt^2}$. Here $v_{d,\max}$ and $a_{d,\max}$ are chosen during the trajectory-generation process.

Note that trajectories that satisfy the above mentioned criteria can be found using [5] and [6].

We consider the vehicle to detect an obstacle at $t = t_{\text{c}}$ and predict its trajectory given as $\mathbf{p}_{\text{o}}(t)$. In this scenario, we want the algorithm to satisfy the following:

1. Collision Prediction: We want the algorithm to check if the vehicle and the obstacle are predicted to collide in the future. Mathematically, we want the algorithm to check if

$$||\mathbf{p}(t) - \mathbf{p}_{\text{o}}(t)|| \le d_{\text{safe}},$$

is satisfied for some time $t \in [t_{\text{c}}, t^{\text{f}}]$ and safety distance $d_{\text{safe}} > 0$. Furthermore, we want the algorithm to find a collision reference time $t = t_*$ that satisfies the above inequality.

2. Collision Avoidance: Consider that at $t = t_{\text{c}}$, the vehicle predicts a collision with reference time t_*. Furthermore, let the collision last during the interval $[t_*^\ell, t_*^u]$, i.e.

$$||\mathbf{p}(t) - \mathbf{p}_o(t)|| \le d_{\text{safe}}, \quad t \in [t_*^\ell, t_*^u], \tag{3}$$

$$||\mathbf{p}(t) - \mathbf{p}_o(t)|| > d_{\text{safe}}, \quad t \in [t_{\text{c}}, t_*^\ell) \cup (t_*^u, t^{\text{f}}], \tag{4}$$

where t_*^ℓ and t_*^u represent the beginning and end of the collision window. Then for some positive constants T_1, T_2, R_1 and R_2, if the collision parameters satisfy

$$t_*^\ell - t_{\text{c}} > T_1, \quad t^{\text{f}} - t_*^u > T_2, \tag{5}$$

$$\frac{t_*^u - t_*^\ell}{t_*^\ell - t_{\text{c}}} < R_1, \quad \frac{t_*^u - t_*^\ell}{t^{\text{f}} - t_*^u} < R_2, \tag{6}$$

then under the assumption $\frac{R_1R_2}{R_1+R_2} < \min\left\{\frac{1}{1+R_1}, \frac{1}{1+R_2}\right\}$, we want the algorithm to find a new C^2-continuous trajectory $\mathbf{p}_{\text{new}}(t) : [0, t^{\text{f}}] \to \mathbb{R}^2$ that keeps the position, velocity and acceleration at t_{c} the same

$$\mathbf{p}_{\text{new}}(t_{\text{c}}) = \mathbf{p}(t_{\text{c}}), \quad \mathbf{v}_{\text{new}}(t_{\text{c}}) = \mathbf{v}(t_{\text{c}}), \quad \mathbf{a}_{\text{new}}(t_{\text{c}}) = \mathbf{a}(t_{\text{c}}),$$

and avoids the collision

$$||\mathbf{p}_{\text{new}}(t) - \mathbf{p}_o(t)|| > d_{\text{safe}}, \qquad t \in [t_*^\ell, t_*^u],$$

while reaching the desired destination with the desired velocity at the end of the mission

$$\mathbf{p}_{\text{new}}(t^{\text{f}}) = \mathbf{p}(t^{\text{f}}), \quad \mathbf{v}_{\text{new}}(t^{\text{f}}) = \mathbf{v}(t^{\text{f}}),$$

where $\mathbf{v}_{\text{new}}(t) = \frac{d\mathbf{p}_{\text{new}}(t)}{dt}$. Furthermore, we want the difference in the new and the original trajectory to be bounded. That is, we want $\mathbf{p}_{\text{new}}(t)$ to satisfy

$$\max_{t\in[0,t^{\text{f}}]} ||\mathbf{p}_{\text{new}}(t) - \mathbf{p}(t)|| < \Delta_p,$$

$$\max_{t\in[0,t^{\text{f}}]} ||\mathbf{v}_{\text{new}}(t) - \mathbf{v}(t)|| < \Delta_v,$$

$$\max_{t\in[0,t^{\text{f}}]} ||\mathbf{a}_{\text{new}}(t) - \mathbf{a}(t)|| < \Delta_a,$$

where $\mathbf{a}_{\text{new}}(t) = \frac{d^2\mathbf{p}_{\text{new}}(t)}{dt^2}$ and the bounds Δ_p, Δ_v and Δ_a can be computed a priori.

Here, a few words about the assumption specified in (6) are in order. This assumption requires the ratio of collision window to time difference between

1) obstacle detection t_{c} and the beginning of collision window t_*^ℓ, and
2) end of collision window t_*^u and the end of the mission t^{f}

to be smaller than some constants. In other words, it requires the collision window to be only a small portion of the remaining time of the mission $[t_{\text{c}}, t^{\text{f}}]$. Although this restricts the algorithm to be able to handle a subset of collision scenarios, the assumption appears to be reasonable for cases, when the obstacle is not actively pursuing the vehicle and their trajectories overlap only for a short period of time.

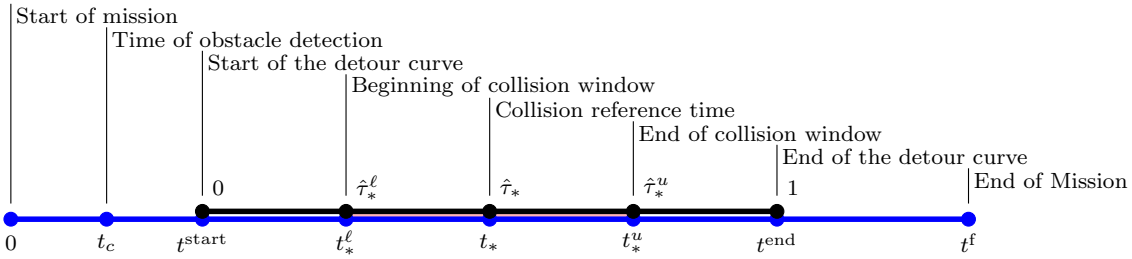

Fig. 1: A timeline showing relevant instants in terms of t and $\hat{\tau}$.

Also note that Equation (6), allows for an arbitrarily large collision window as long as its ratio to $(t_*^\ell - t_{\text{c}})$ and $(t^{\text{f}} - t_*^u)$ is small. This is more relaxed as compared to [10] where the collision window is required to be smaller than a threshold.

IV. Proposed Algorithm

We start this section with the methodology of predicting collisions followed by the trajectory-replanning algorithm.

A. Collision Prediction

The stepwise procedure for collision prediction follows.

Step 1: Express Obstacle's Trajectory using Bézier Curves

For the mission-time interval $t \in [0, t^{\text{f}}]$, we represent the trajectory of the obstacle $\bar{\mathbf{p}}_{o,k}^i$ as a piecewise Bézier curve. Specifically, for each interval $t \in [\lambda^{i-1}, \lambda^i] \subseteq [0, t^{\text{f}}]$, we find the control points $\bar{\mathbf{p}}_{o,k}^i$ for $k = 0, 1, \cdots, n$ such that

$$\mathbf{p}_o(t) = \sum_{k=0}^{n} \bar{\mathbf{p}}_{o,k}^i b_k^n \left(\frac{t - \lambda^{i-1}}{\lambda^i - \lambda^{i-1}} \right), \quad t \in [\lambda^{i-1}, \lambda^i],$$

where $i = 1, 2, \cdots, m$.

Since the right hand side of the above equation is a linear combination of the control points, we can use methods such as least squares to find these control points.

Step 2: Find the Separation Bézier Curves

Now, we can define the separation between the vehicle and the obstacle using a piecewise Bézier curve $\mathbf{d}(t)$ as

$$\mathbf{d}(t) = \mathbf{p}(t) - \mathbf{p}_o(t) = \sum_{k=0}^{n} (\bar{\mathbf{p}}_k^i - \bar{\mathbf{p}}_{o,k}^i) b_k^n(t), \quad t \in [\lambda^{i-1}, \lambda^i],$$

where $i = 1, 2, \cdots, m$.

Step 3: Use Minimum Distance Algorithm for Bézier Curves

Given the separation Bézier curves, we can use the minimum distance algorithm to find

$$\min_{t \in [\lambda^{i-1}, \lambda^i]} ||\mathbf{d}(t)||, \quad \arg\min_{t \in [\lambda^{i-1}, \lambda^i]} ||\mathbf{d}(t)|| \tag{7}$$

for all subintervals $[\lambda^{i-1}, \lambda^i]$ and $i = 1, 2, \cdots, m$. Thus, we can find

$$d_{\text{min}} = \min_{t \in [0, t^{\text{f}}]} ||\mathbf{d}(t)||, \quad \arg\min_{t \in [0, t^{\text{f}}]} ||\mathbf{d}(t)|| \tag{8}$$

If $d_{\text{min}} \leq d_{\text{safe}}$, the algorithm predicts a collision with a reference time $t_* = \arg\min_{t \in [0, t^{\text{f}}]} ||\mathbf{d}(t)||$.

B. Trajectory Replanning

Consider that at time $t = t_{\text{c}}$, the vehicle predicts a collision with reference time $t = t_* > t_{\text{c}}$. In this scenario, the algorithm adds a detour to $\mathbf{p}(t)$ in order to find a new trajectory $\mathbf{p}_{\text{new}}(t)$. Steps to calculate the detour and thus determine $\mathbf{p}_{\text{new}}(t)$ are described as follows.

Step 1: Determine the Beginning and End of Detour

Note that a detour that lasts for a long period of time should lead to a slow avoidance maneuver thus requiring smaller velocity and acceleration changes. This is generally desirable. However, for a fixed number of control points defining the detour, a longer lasting detour would provide a coarse control over its shape possibly leading to bad detours.

Therefore, as a first step, we determine an appropriate beginning and end of the detour as

$$\begin{bmatrix} t^{\text{start}} \\ t^{\text{end}} \end{bmatrix} = \begin{cases} \begin{bmatrix} t_{\text{c}}, & \frac{t_* - t_{\text{c}} + \hat{\tau}_{\text{ds}}^\ell t_{\text{c}}}{\hat{\tau}_{\text{ds}}^\ell} \end{bmatrix}^\top, & \text{if } \frac{t_* - t_{\text{c}}}{t^{\text{f}} - t_{\text{c}}} < \hat{\tau}_{\text{ds}}^\ell, \\ \begin{bmatrix} \frac{t_* - \hat{\tau}_{\text{ds}}^u t^{\text{f}}}{1 - \hat{\tau}_{\text{ds}}^u}, & t^{\text{f}} \end{bmatrix}^\top, & \text{if } \frac{t_* - t_{\text{c}}}{t^{\text{f}} - t_{\text{c}}} > \hat{\tau}_{\text{ds}}^u, \\ \begin{bmatrix} t_{\text{c}}, & t^{\text{f}} \end{bmatrix}^\top, & \text{otherwise,} \end{cases} \tag{9}$$

where t^{start} and t^{end} represent the start and end time for the detour and $0 < \hat{\tau}_{\text{ds}}^\ell \leq \hat{\tau}_{\text{ds}}^u < 1$ are design parameters.

We also define the mapping $g : [t^{\text{start}}, t^{\text{end}}] \to [0, 1]$ that maps t to a normalized time variable $\hat{\tau}$:

$$g(t) = \frac{t - t^{\text{start}}}{t^{\text{end}} - t^{\text{start}}}. \tag{10}$$

With this mapping, we can write time instants related to the collision in terms of $\hat{\tau}$ as

$$\hat{\tau}_*^\ell = g(t_*^\ell), \quad \hat{\tau}_* = g(t_*), \quad \hat{\tau}_*^u = g(t_*^u),$$

where it can be shown that $\hat{\tau}_* \in [\hat{\tau}_{\text{ds}}^\ell, \hat{\tau}_{\text{ds}}^u]$.

For illustration, we provide Figure 1. Note that the figure shows $t^{\text{start}} \neq t_{\text{c}}$ and $t^{\text{end}} \neq t^{\text{f}}$. This is not an accurate representation as the algorithm sets $t^{\text{start}} = t_{\text{c}}$ and/or $t^{\text{end}} = t^{\text{f}}$ (See (9)). Also note that the figure shows $t_*^\ell > t^{\text{start}}$ and $t^{\text{end}} > t_*^u$. This can be proved to be true as long as the assumptions specified in Section III hold.

Step 2: Determine the Time Profile of the Detour Magnitude

As we will later see, the detour curve will have a constant direction, however, its magnitude will vary with time. Next, we calculate the magnitude profile of the detour.

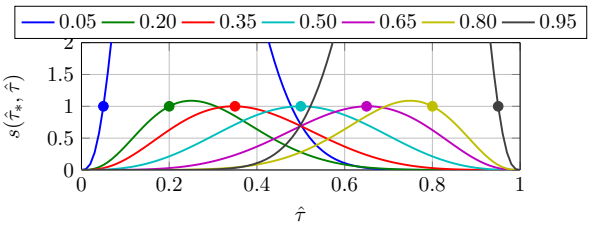

Fig. 2: Magnitude profile $s(\hat{\tau}_*, \hat{\tau})$ for different values of $\hat{\tau}_*$ and $n = 15$. Similar profiles are obtained for other values of n. The value of $\hat{\tau}_*$ is shown as a solid circle for every curve.

We define the magnitude profile of the detour over the normalized time variable $\hat{\tau} \in [0, 1]$ as a Bézier curve. Specifically, the profile is given as

$$s(\hat{\tau}_*, \hat{\tau}) = \sum_{k=0}^{n} \bar{s}_k(\hat{\tau}_*) b_k^n(\hat{\tau}), \qquad \hat{\tau} \in [0, 1], \tag{11}$$

where we set control points $\bar{s}_k(\hat{\tau}_*)$ for $k = \{0, 1, \cdots, n\}$ as

$$\bar{s}_k(\hat{\tau}_*) = \begin{cases} \frac{b_k^n(\hat{\tau}_*)}{\sum_{j=3}^{n-3}(b_j^n(\hat{\tau}_*))^2}, & 3 \leq k \leq n-3, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

We now briefly describe the choice for this profile.

First, as we will later see, defining the profile as a Bézier curve ensures that the new trajectory is a piecewise Bézier curve allowing for multiple application of the algorithm.

Second, we set the first and the last three control points as zero. This ensures that the position, velocity and acceleration change at the beginning and end of the detour is zero (see [11]) guaranteeing C^2-continuity of the new trajectory.

Finally, the expression for the rest of the control points is found as a least squares solution to the following equation

$$s(\hat{\tau}_*, \hat{\tau} = \hat{\tau}_*) = \begin{bmatrix} b_0^n(\hat{\tau}_*) & b_1^n(\hat{\tau}_*) & \cdots & b_n^n(\hat{\tau}_*) \end{bmatrix} \begin{bmatrix} \mathbf{0}_{3\times 1} \\ \bar{s}_3^n \\ \cdots \\ \bar{s}_{n-3}^n \\ \mathbf{0}_{3\times 1} \end{bmatrix} = 1,$$

where $\mathbf{0}_{3\times 1}$ is a (3×1) column vector of zero entries. Thus, the magnitude of the detour is set as 1 at the collision reference time $\hat{\tau}_*$. Furthermore, the 2-norm of the vector containing the control points is minimized. Note that other norms (such as the infinity norm) could have been used here. However, the choice ensures a closed-form solution (namely (12)), allowing for real-time application of the algorithm.

Furthermore, as we show in Figure 2, this choice of control points provides a nice Guassian-like shape of the profile. The exception to this observation are the cases when $\hat{\tau}_*$ is close to 0 or 1. This, however, is not a problem since $\hat{\tau}_* \in [\hat{\tau}_{\text{ds}}^{\ell}, \hat{\tau}_{\text{ds}}^{u}]$, where $\hat{\tau}_{\text{ds}}^{\ell}$ and $\hat{\tau}_{\text{ds}}^{u}$ are design parameters that can be chosen to be, for example, close to 0.5.

Step 3: Scale Detour with an Appropriate Vector

In this step, we finalize the detour as

$$\Delta_K(\hat{\tau}_*, t) = K s(\hat{\tau}_*, g(t)) \mathbf{u}, \quad t \in [t^{\text{start}}, t^{\text{end}}], \tag{13}$$

where $\mathbf{u} = \frac{\mathbf{d}(t_*)}{||\mathbf{d}(t_*)||}$, and $K \in \mathbb{R}^+$ is a scaling factor that satisfies

$$\min_{t \in [t_*^{\ell}, t_*^{u}]} ||\Delta_K(\hat{\tau}_*, t) + \mathbf{d}(t))|| > d_{\text{safe}}. \tag{14}$$

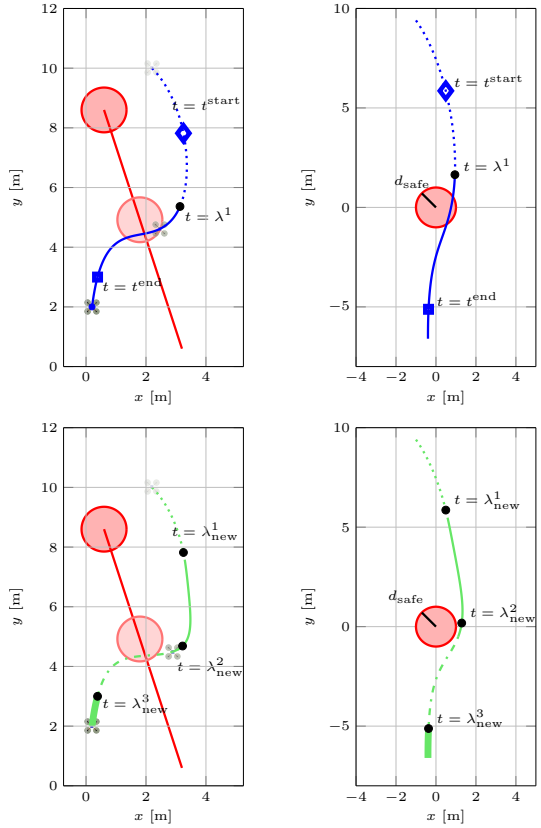

Fig. 3: Illustration of the algorithm: Top left panel shows the original trajectory (blue) along with that of the obstacle (red). The separation curve is shown in the top right panel. The bottom left panel shows the replanned trajectory (green) with corresponding separation curve in bottom right panel (red). The diamond, square and thick black dots indicate t^{start}, t^{end} and λ's, respectively. Every unique line type indicates a single Bézier curve.

Ideally, we want K to be the smallest value that satisfies the above equation. However, for real-time application, we only find a sub-optimal value of K that still satisfies (14). It can be shown that an a priori computable value $K = K^u$ is guaranteed to satisfy (14). Thus, we set $K_0 = K^u$ and sample values $K_k \in (0, K^u]$ for $k = 1, 2 \cdots, M$ to find K as

$$K = \min_{k=1,\cdots,M} K_k,$$

$$\text{subject to} \quad \min_{t \in [t^{\text{start}}, t^{\text{end}}]} ||\Delta_K(\hat{\tau}_*, t) + \mathbf{d}(t)|| > d_{\text{safe}}.$$

Note that a larger domain for t is used in the above constraint as compared to (14). Although, this makes the procedure conservative, it allows the algorithm to work without the knowledge of t_*^{ℓ} and t_*^{u} which can be hard to determine in real-time.

Step 4: Add Detour to the Original Trajectory

In this final step, we add the detour to the original trajectory to get $\mathbf{p}_{\text{new}}(t)$. For this, we first add t^{start} and t^{end} to the sequence λ to create the new sequence λ_{new} that has

$m+3$ elements:

$$\lambda_{\text{new}} = \{\lambda^0_{\text{new}} = 0, \lambda^1_{\text{new}}, \lambda^2_{\text{new}}, \cdots, \lambda^j_{\text{new}} = t^{\text{start}}, \cdots, \lambda^l_{\text{new}} = t^{\text{end}}, \cdots, \lambda^{m+2}_{\text{new}} = t^{\text{f}}\}.$$

We then write the original trajectory as

$$\mathbf{p}(t) = \mathbf{p}^i\left(\frac{t-\lambda^{i-1}_{\text{new}}}{\lambda^i_{\text{new}}-\lambda^{i-1}_{\text{new}}}\right), \quad t \in [\lambda^{i-1}_{\text{new}}, \lambda^i_{\text{new}}],$$

for $i = \{1, 2, \cdots, m+2\}$. Similarly, we write the detour as a function of time $\Delta_K(t)$ as

$$\Delta_K(t) = \Delta^i_K\left(\hat{\tau}_*, \frac{t-\lambda^{i-1}_{\text{new}}}{\lambda^i_{\text{new}}-\lambda^{i-1}_{\text{new}}}\right), \quad t \in [\lambda^{i-1}_{\text{new}}, \lambda^i_{\text{new}}],$$

for $i = \{j+1, j+2\cdots, l\}$, where $\Delta^i_K(\hat{\tau}_*, \cdot)$ are Bézier curves. Then the new trajectory can be written as

$$\mathbf{p}_{\text{new}}(t) = \begin{cases} \mathbf{p}^q\left(\frac{t-\lambda^{q-1}_{\text{new}}}{\lambda^q_{\text{new}}-\lambda^{q-1}_{\text{new}}}\right) + \Delta^q_K\left(\hat{\tau}_*, \frac{t-\lambda^{q-1}_{\text{new}}}{\lambda^q_{\text{new}}-\lambda^{q-1}_{\text{new}}}\right), & t \in [\lambda^{q-1}_{\text{new}}, \lambda^q_{\text{new}}] \\ \mathbf{p}^r\left(\frac{t-\lambda^{r-1}_{\text{new}}}{\lambda^r_{\text{new}}-\lambda^{r-1}_{\text{new}}}\right), & t \in [\lambda^{r-1}_{\text{new}}, \lambda^r_{\text{new}}] \end{cases} \tag{15}$$

where q and r are given as $q = \{j+1, j+2, \cdots, l\}$ and $r \in \{1, \cdots, j, l+1, \cdots, m+2\}$.

An illustration of the algorithm is shown in Figure 3.

Theorem 1. *Consider that at mission time $t = t_c$, the vehicle predicts a collision with reference time t_* such that $||\mathbf{p}(t) - \mathbf{p}_o(t)|| \leq d_{safe}$ for $t \in [t^\ell_*, t^u_*]$ and*

$$t^\ell_* - t_c > T_1, \quad t^f - t^u_* > T_2,$$

$$\frac{t^u_* - t^\ell_*}{t^\ell_* - t_c} < R_1, \quad \frac{t^u_* - t^\ell_*}{t^f - t^u_*} < R_2.$$

Then the trajectory found through (9), (11), (12), (13), (14) *and* (15) *with design parameters $\hat{\tau}^\ell_{ds}$ and $\hat{\tau}^u_{ds}$, that satisfy $\hat{\tau}^\ell_{ds} \in \left(\frac{R_1R_2}{R_1+R_2}, \frac{1}{1+R_1}\right)$ and $\hat{\tau}^u_{ds} \in \left(\frac{R_2}{1+R_2}, 1-\frac{R_1R_2}{R_1+R_2}\right)$, will keep position, velocity and acceleration at current mission time t_c unchanged*

$$\mathbf{p}_{new}(t_c) = \mathbf{p}(t_c), \quad \mathbf{v}_{new}(t_c) = \mathbf{v}(t_c), \quad \mathbf{a}_{new}(t_c) = \mathbf{a}(t_c),$$

satisfy the boundary conditions

$$\mathbf{p}_{new}(t^f) = \mathbf{p}(t^f), \quad \mathbf{v}_{new}(t^f) = \mathbf{v}(t^f),$$

and avert the collision, as

$$||\mathbf{p}_{new}(t) - \mathbf{p}_o(t)|| > d_{safe}, \qquad \forall t \in [t^\ell_*, t^u_*],$$

while ensuring a bounded difference in the position, velocity and acceleration

$$\max_{t\in[0,t^f]} ||\mathbf{p}_{new}(t) - \mathbf{p}(t)|| < \Delta_p,$$
$$\max_{t\in[0,t^f]} ||\mathbf{v}_{new}(t) - \mathbf{v}(t)|| < \Delta_v,$$
$$\max_{t\in[0,t^f]} ||\mathbf{a}_{new}(t) - \mathbf{a}(t)|| < \Delta_a,$$

where the bounds Δ_p, Δ_v and Δ_a can be computed a priori.

Remark 1. *If $\frac{R_1R_2}{R_1+R_2} < \min\left\{\frac{1}{1+R_1}, \frac{1}{1+R_2}\right\}$, then there exist design parameters $\hat{\tau}^\ell_{ds}$ and $\hat{\tau}^u_{ds}$ that satisfy $\hat{\tau}^\ell_{ds} \in \left(\frac{R_1R_2}{R_1+R_2}, \frac{1}{1+R_1}\right)$ and $\hat{\tau}^u_{ds} \in \left(\frac{R_2}{1+R_2}, 1-\frac{R_1R_2}{R_1+R_2}\right)$.*

Remark 2. *As we will later show, collision avoidance and satisfaction of dynamic constraints can be guaranteed if the assumptions on the collision scenario are met. However, if these assumptions are not guaranteed, the algorithm can still be used albeit with an additional last step for verification of collision avoidance and dynamic constraints.*

V. Simulations

In this section, we demonstrate the algorithm using a typical mission scenario for a multirotor. We consider two collisions such that the assumptions on the collision scenario are satisfied only for one. However, we show that the algorithm avoids both the collisions and satisfies all the constraints. This includes vehicle dynamic constraints, here characterized by maximum velocity and acceleration.

A. Mission Scenario and Assumptions

We consider a multirotor that can achieve a maximum speed and acceleration of $v_{\max} = 5\,\text{m/s}$ and $a_{\max} = 10\,\text{m/s}^2$, respectively, and can detect obstacles up to $25\,\text{m}$ away. Here, we consider other multirotors (also with maximum speed of $v_{\max} = 5\,\text{m/s}$) as obstacles. We assume that these obstacles do not actively pursue the vehicle and that a center-to-center distance of $d_{\text{safe}} = 1\,\text{m}$ is sufficient for avoiding collisions.

With an obstacle detection range of $25\,\text{m}$ and a maximum speed of $v_{\max} = 5\,\text{m/s}$ for both the vehicle and the obstacles, the time difference between obstacle detection and collision should at least be [1] $T_1 = 2.5\,\text{s}$. Here, we also assume $T_2 = T_1 = 2.5\,\text{s}$. Finally, without any active pursuit, we assume collision windows to be small and set $R_1 = R_2 = 0.25$.

B. Trajectory Generation

We set the design parameters as $\hat{\tau}^\ell_{ds} = \hat{\tau}^u_{ds} = 0.5$ and get the following bounds for a collision-avoidance maneuver:

$$\Delta_p = 2.52\,\text{m}, \ \Delta_v = 1.80\,\text{m/s}, \ \Delta_a = 2.93\,\text{m/s}^2.$$

Given these bounds, we generate the initial trajectory with

$$v_{d,\max} = 3.2\,\text{m/s}, \ a_{d,\max} = 7.07\,\text{m/s}^2,$$

and obtain a trajectory with $t^{\text{f}} = 10\,\text{s}$. The values of $v_{d,\max}$ and $a_{d,\max}$ set above ensure that the dynamic constraints of the vehicle are met even when the velocity and acceleration are varied up to Δ_v and Δ_a for collision avoidance.

C. Simulation Results

We simulate the obstacle to initially fly with a constant velocity of $3.2\,\text{m/s}$, on a collision course with the vehicle such that[2] $\hat{\tau}^\ell_* = 3.77\,\text{s}$ and $\hat{\tau}^u_* = 4.22\,\text{s}$. Since the obstacles can be detected $25\,\text{m}$ away, all assumptions will be satisfied

[1] This is calculated for the case when the obstacle and the vehicle are travelling directly towards each other.

[2] Note that the algorithm does not need $\hat{\tau}^\ell_*$ and $\hat{\tau}^u_*$. However, for analysis, we can use these parameters to confirm satisfaction of assumptions.

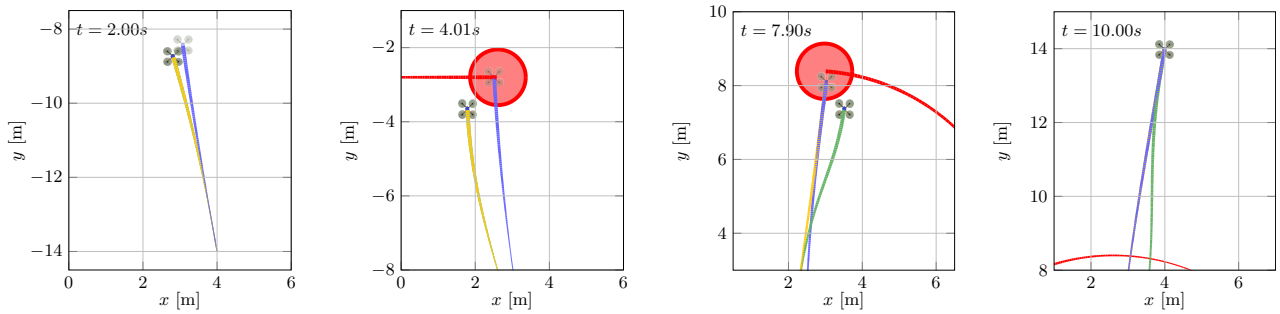

Fig. 4: Simulation shown at different times with $\mathbf{p}(t)$, $\mathbf{p}_{\text{new},1}(t)$ and $\mathbf{p}_{\text{new},2}(t)$ shown as blue, yellow and green trails, respectively. The obstacle's trajectory is shown as a red trail where the obstacle itself is shown as a red circle.

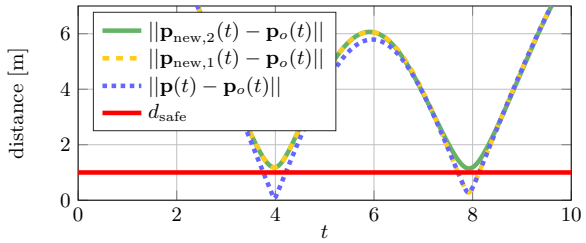

Fig. 5: Distance between the obstacle and the vehicle for the three trajectories $\mathbf{p}(t)$, $\mathbf{p}_{\text{new},1}(t)$ and $\mathbf{p}_{\text{new},2}(t)$.

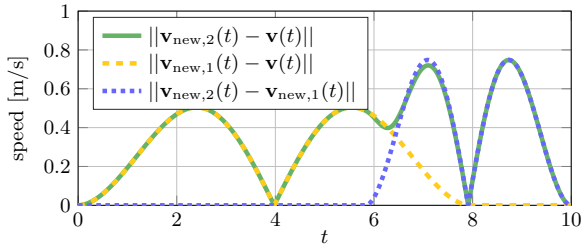

Fig. 6: Speed differences between the three trajectories.

for this collision. In this scenario, the algorithm generates the new trajectory $\mathbf{p}_{\text{new},1}(t)$ and avoids the collision as shown in Figure 4 for $t = 4.01\,\text{s}$. The distance $||\mathbf{p}_{\text{new},1}(t) - \mathbf{p}_o(t)||$ is shown in Figure 5, whereas, the variation in speed caused by the maneuver is shown in Figure 6.

To show multiple applications of the algorithm, we now consider the obstacle to abruptly change its trajectory and fly in a circular path, on a collision course such that $\hat{\tau}_*^{\ell} = 7.72\,\text{s}$ and $\hat{\tau}_*^{u} = 8.15\,\text{s}$. Note that the assumption $t^{\text{f}} - t_*^{u} \geq T_2$ is not satisfied here. However, the new trajectory generated by the algorithm $\mathbf{p}_{\text{new},2}(t)$ still avoids the collision without violating any constraints. Figure 4 shows the vehicle avoiding this collision at $t = 7.90\,\text{s}$. The distance $||\mathbf{p}_{\text{new},2}(t) - \mathbf{p}_o(t)||$ is shown in Figure 5, whereas, the variations in speed caused by the maneuver are shown in Figure 6[3].

VI. Conclusion

In this paper, we present a collision avoidance algorithm that can be applied to piecewise Bézier curves to avoid multiple collision. Upon detection of an obstacle, the algorithm checks for a possible collision in the future. If a collision is predicted, the algorithm modifies the trajectory such that the collision is avoided and mission specific-constraints are satisfied. Under mild assumptions, the algorithm guarantees bounded position, velocity and acceleration changes during the avoidance maneuvers. By using these bounds during trajectory generation, we can guarantee collision avoidance with satisfaction of vehicle dynamic constraints.

In future, we plan to extend this algorithm to more general scenarios. One such extension is for collisions that don't satisfy assumption specified in (6). Another key extension is towards a three-dimensional setting. Lastly, the algorithm needs to be analyzed for vehicles that impose more stringent dynamic constraints, such as maximum path curvature.

[3]A video for this simulation is also available online at https://youtu.be/ALZQcEK8f_Q

References

[1] "Concept of operations for the next generation air transportation system," Joint Planning and Development Office, Version 3.2, September 2010, available at http://www.dtic.mil/dtic/tr/fulltext/u2/a535795.pdf.

[2] R. Ritz and R. D'Andrea, "Carrying a flexible payload with multiple flying vehicles," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 3465–3471.

[3] S. Gupte, P. I. T. Mohandas, and J. M. Conrad, "A survey of quadrotor unmanned aerial vehicles," in *2012 Proceedings of IEEE Southeastcon conference*, 2012, pp. 1–6.

[4] V. Cichella, R. Choe, S. B. Mehdi, E. Xargay, N. Hovakimyan, V. Dobrokhodov, and I. Kaminer, "Trajectory generation and collision avoidance for safe operation of cooperating UAVs," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2014, AIAA 2014-0972.

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

[6] R. Choe, J. Puig-Navarro, V. Cichella, E. Xargay, and N. Hovakimyan, "Trajectory generation using spatial Pythagorean Hodograph Bézier curves," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2015, AIAA 2015-0597.

[7] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE International Conference on Robotics and Automation*, vol. 2, March 1985, pp. 500–505.

[9] T. Fraichard and H. Asama, "Inevitable collision states. A step towards safer robots?" in *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2003, pp. 388–393.

[10] S. B. Mehdi, R. Choe, V. Cichella, and N. Hovakimyan, "Collision avoidance through path replanning using Bézier curves," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2015, AIAA 2015-0598.

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

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