2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-1714-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IROS51168.2021.9636535

Trajectory Optimization For Rendezvous Planning Using Quadratic Bézier Curves

Satyanarayana G. Manyam[1], David W. Casbeer[2], Isaac E. Weintraub[2] and Colin Taylor[3]

***Abstract*— In this paper, we consider a trajectory planning problem where an autonomous vehicle aims to rendezvous with another cooperating vehicle in minimum time. The first vehicle has kinematic constraints, consequently feasible trajectories must have a maximum curvature less than a specified limit. Rendezvous is said to occur at the instant that the two vehicles are collocated with the same heading. We propose a technique to construct a trajectory, composed of piecewise quadratic Bézier curves, that satisfies the vehicle motion constraints and achieves rendezvous in minimum time. The methodology begins by finding safe flight corridors, which are constructed from sequences of triangles using constrained Delaunay triangulation of the feasible space; the triangles define the bounds of Bézier curves. We formulate the necessary constraints for continuity and feasibility as functions of the control points that define the Bézier curves, and the resulting optimization problem is solved using a nonlinear programming solver. The techniques developed were tested using simulated scenarios, and we present the results which highlight the efficacy of the proposed solution approach. Furthermore, the algorithm was implemented and tested in a field test and those results are presented.**

***Index Terms*— path planning, obstacle avoidance, mobile robots, kinematic constraints**

I. Introduction

Trajectory generation in the presence of obstacles for non-holonomic vehicles is an important research topic with several applications in autonomous mobility. The general objective is to determine the trajectory by which a mobile robot departs some initial state, arrives at a desired final state, and minimizes a given cost. In this paper, an autonomous air-vehicle with kinematic constraints aims to rendezvous with a second vehicle in minimum time amid a field of static obstacles. Such rendezvous requirements are necessary in applications where multiple vehicles cooperatively perform data gathering missions, and where two vehicles need to exchange data. The kinematic constraints of the vehicle are due to the minimum turn radius, and therefore the trajectories should satisfy the maximum curvature constraints to be flyable. The two vehicles are cooperating and the planned trajectory of the second vehicle is known to the first vehicle. The rendezvous is defined as two vehicles arriving at a location at the same time with the same heading. In practice, these two vehicles would fly at different altitudes, and a feasible rendezvous only requires the x, y coordinates and heading to be the same. The proposed solution approach is for the controlled vehicle to generate a trajectory using a concatenated set of quadratic Bézier curves (QBC). We assume the vehicles are traveling at constant speed and the objective to rendezvous in minimum time is addressed by optimizing the length of the trajectory. The trajectories are defined by a set of control points, and the constraints and objective are realized as functions of these control points. We formulate this as a nonlinear program with the sets of control points as decision variables, the solution of which gives a trajectory that avoids the obstacles and ensures a smooth path which does not exceed the maximum curvature limit.

Dubins paths [1], [2] are very popular for path planning for vehicles with kinematic constraints in obstacle free environments. Sample based path planning methods, such as RRT [3]–[5], were developed to generate feasible paths in structures environments, and recent advances are proven to be probabilistically complete. Discrete search based methods such as hybrid A* [6] produce quick solutions and are thus viable for online planning. However, these methods are not suitable for the trajectory planning problems with temporal or length constraints. In our previous work [7], we proposed a Dubins based planning to address temporal constraints such as rendezvous problem. Dubins paths are highly discontinuous, and making fine adjustments requires careful attention to the discontinuities in the path length as a function of the path modality [8]. Such methods are not viable for environments with obstacles.

Recently, trajectory generation using splines such as Bézier curves [9]–[13] have gained traction due to their flexibility. In [9], quartic Bézier curves, defined by Bernstein polynomials of degree four, are considered and trajectory generation is posed as a nonlinear program. To address the kinematic constraints, the maximum curvature is computed using a numerical root finding method in each iteration of the nonlinear program. Bézier curves of degree five are used to approximate a clothoids in [11]. In [12], a flight corridor is generated and trajectories are generated using piecewise Bézier curves, where the position and dynamics are bounded using Bézier properties. A similar method with piecewise Bézier curves was used in [13], where the trajectory generation is decoupled into timing variables and path variables. The curvature constraints on the paths due to the kinematics of the vehicles are not addressed in [12], [13].

This work has been supported in part by AFOSR LRIR No. 21RQ-COR084.

[1]Satyanarayana G. Manyam is with Infoscitex corporation, a DCS Company, Dayton, OH, USA msngupta@gmail.com

[2]David W. Casbeer and Isaac E. Weintraub are with the Controls Center of Excellence, Air Force Research Laboratory, WPAFB, OH, USA david.casbeer@afresearchlab.com, isaac.weintraub.1@afresearchlab.com

[3]Colin Taylor is with Parallax Advanced Research, Dayton, OH, USA colin.taylor@parallaxresearch.org

In this paper, we propose a method to construct a trajectory using piecewise QBCs. The use of QBCs is motivated by the fact that there exists a closed form solution to compute the maximum curvature [14] and the length of a QBC. These enable us to explicitly formulate a nonlinear program that can address the maximum curvature constraints and the length constraints on the path to achieve the rendezvous. To deal with obstacle avoidance, we follow [12], [15] and generate a safe flight corridor using constrained Delaunay triangulation (CDT). Using CDT, Kallmann provided an efficient means of computing optimal paths of arbitrary clearance [16]. The technique proposed in this paper involves generating flight corridors as a series of triangles, defining a QBC corresponding to each triangle in the flight corridor, and the obstacle avoidance is addressed by bounding each QBC to lie within the corresponding triangle. Each QBC is a function of its control points, and the nonlinear program minimizes a cost function by simultaneously solving for the control points of all the QBCs. We formulate the constraints of this optimization to satisfy the path continuity, maximum curvature and the constraints on the path length to achieve rendezvous using these control points.

The paper is constructed as follows: In Section II, the rendezvous problem is described in detail. Next, in Section III, the corridor construction and computation of the trajectory as a series of quadratic Bézier curves is described. In Section IV, an example simulation with experimental results shows the efficacy of the proposed approach for rendezvousing trajectory planning in minimum time. Finally, in Section V, a summary of the paper and concluding remarks are made.

II. Rendezvous Problem

We assume the obstacles in the environment are known *a priori* and can be reasonable modeled by polygons. The initial configuration of the autonomous air-vehicle (AV_1) is given (s_x, s_y, s_θ), where (s_x, s_y) are the position coordinates, and s_θ is the heading measured with respect to a predefined x-axis. The air-vehicle AV_1 is required to rendezvous in some predefined 'feasible-space' with a second, moving air-vehicle AV_2. In this paper, the feasible space is a line-segment that does not intersect with any of the obstacles. Let T_s and T_f represent the ends of this line segment, and AV_2 starts at T_s and travels at constant speed and heading towards the T_f, let this heading direction be t_θ. Let t_s be the time of arrival of the AV_2 at T_s. The autonomous air-vehicle, AV_1, has kinematic constraints, and for the trajectory generated to be flyable, the curvature at any point on the path must be less than the maximum curvature limit, κ_{limit}.

The path planning problem is to find the rendezvous point, R_t, on the line segment $\overline{T_sT_f}$ and the path of minimum length that satisfies the following: (i) The path starts at (s_x, s_y) with heading s_θ and ends at R_t with a heading t_θ. (ii) AV_1 and AV_2 arrive at R_t simultaneously. (iii) The maximum curvature of the path is less then κ_{limit}. (iv) The path is feasible with respect to the obstacles. An illustration of the problem scenario is shown in Figure 1.

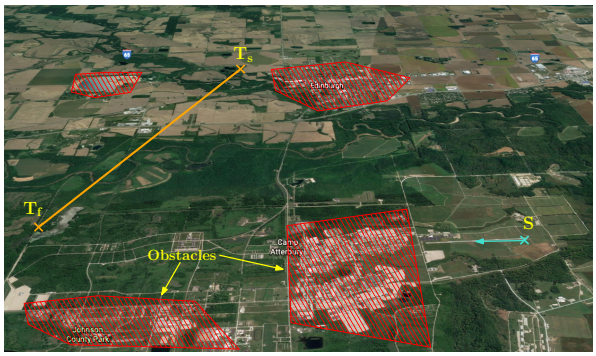


Fig. 1. The problem scenario shows the starting position of AV_1, the rendezvous line segment $\overline{T_sT_f}$ and the obstacles.

III. Technical Approach

In this section, we focus on the path generation that involves a corridor generation and path optimization posed as nonlinear optimization problem. We propose a technique to generate paths using a set of concatenated piecewise QBCs, such that the tangent to the path is continuous everywhere, G^1. This path generation is guided by a predetermined flight corridor, which is a series of triangles where every pair of successive triangles have a common edge. In the following subsections, we present brief overview of QBCs, corridor generation, and path optimization formulated as a nonlinear optimization problem.

A. Quadratic Bézier Curves (QBCs)

Bézier curves are splines that are defined by polynomials over a finite interval in the Bernstein basis. The maximum degree of the polynomials is referred to as the degree of the Bézier curve. Let $P(\tau)$ be a Bézier curve of degree n, and it is given as:

$$P(\tau) = \sum_{k=0}^{n} p_k b_k^n(\tau), \tag{1}$$

where $\tau \in [0, 1]$ is a non-dimensional parameter, and $p_0, p_1, \ldots p_n \in \mathbb{R}^2$ are the control points, and $b_k^n(\tau)$ are the Bernstein polynomials of degree n. Bézier curves have the following properties which makes them well suited for path planning: (i) A Bézier curve always starts at the first control point and ends at the final control point. (ii) The tangent of the curve at these end points aligns with the line passing through the two end control points. (iii) The Bézier curve always lies inside the convex hull of the control points. This property makes them adaptable for path planning in the presence of obstacles, where one can appropriately constrain the control points such that the associated convex hull does not intersect with the obstacles, and construct a feasible path.

In the current problem, the maximum curvature of paths are required to be less than a limit, κ_{limit}, which corresponds to the minimum turn radius of the vehicles. For a QBC, there exists a closed form solution for the maximum curvature and the length of the path. Therefore, we use QBCs for path planning in order to address the curvature constraints and the temporal constraints (manifested as a length constraint).

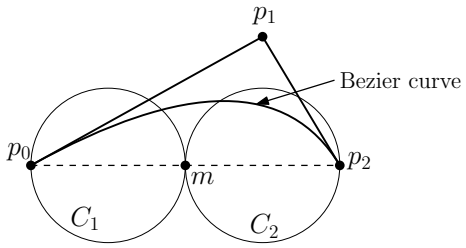


Fig. 2. A quadratic Bézier curve with its control points.

Let p_0, p_1, p_2 be the control points of a QBC, $P(\tau)$, shown in Fig. 2. Let m represent the mid-point of the line joining p_0 and p_2, A_t be the area of the triangle $\triangle p_0, p_1, p_2$. In Fig. 2, C_1 and C_2 are circle with diameters $\overline{p_0 m}$ and $\overline{p_2 m}$ respectively. The closed form solution of the maximum curvature, κ_{max}, of a QBC as a function of the control points is given in [14], and it depends on the position of p_1 as shown below:

$$\kappa_{max} = \begin{cases} \frac{\|p_1 m\|^3}{A_t^2}, \text{ if } p_1 \notin C_1 \cup C_2, \\ \frac{A_t}{\|p_0 p_1\|^3}, \text{ if } p_1 \in C_1, \\ \frac{A_t}{\|p_1 p_2\|^3}, \text{ if } p_1 \in C_2. \end{cases} \tag{2}$$

The length of a QBC can be expressed as analytical function of its control points, and is given below.

$$L_b = \frac{1}{8A^{\frac{3}{2}}}[4A^{\frac{3}{2}}W + 2\sqrt{A}B(W - \sqrt{C}) + (4CA - B^2)\ \log\left\|\frac{2\sqrt{A} + B/\sqrt{A} + 2W}{B/\sqrt{A} + 2\sqrt{C}}\right\|] \tag{3}$$

where $W = \sqrt{A + B + C}$, $A = 4(a_x^2 + a_y^2)$, $B = 4(a_x b_x + a_y b_y)$, $C = b_x^2 + b_y^2$, and $a = p_0 - 2p_1 + p_2$, $b = 2p_1 - 2p_0$.

The idea to define the path as a concatenation of QBCs defined by the control points leverages the existence of closed form solutions for the maximum curvature and the length of a QBC. This property allows the current path planning problem to be posed as nonlinear optimization over the control points. To satisfy the obstacle avoidance constraints, we propose to guide the QBCs using safe flight corridors, that are defined by a series of adjacent triangles. We assign a QBC to each triangle and the QBCs belonging to adjacent triangles are constrained to be G^1 continuous. In the following section, we present the corridor generation algorithm using constrained Delaunay triangulation.

B. Corridor Generation

The overall approach of corridor construction involves three steps. In the first step, we generate the triangulation of the 'feasible-space' using constrained Delaunay triangulation. In the second step, an abstract graph is constructed where each node of the graph represents a triangle, and the nodes corresponding to adjacent triangles are connected by an edge. We construct the corridor using the shortest path between the nodes representing the initial triangle to final triangle.

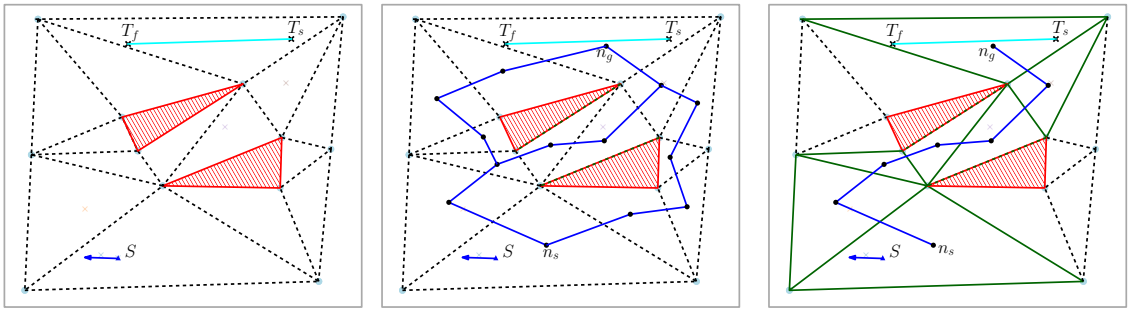
Let O be the set of polygonal obstacles, and let O_v and O_s be the set of vertices and edges of the obstacles, respectively. We use the Delaunay refinement algorithm in [17] to generate the triangulation of the feasible space. This technique generates triangulation with some useful properties such as the following: The union of the triangles is the triangulation domain, and any segment is in the union of the triangulation edges. The Delaunay refinement algorithm aims to bound the smallest angle in the triangles. This avoids generating triangles with large or small angles, and it would be apt for the current path planning problem. One can define the obstacles as 'holes' in the domain and the Delaunay refinement algorithm avoids generating the triangulation inside the obstacles. Let O_h denote the set of 'holes', where each hole is parameterized by a point inside an obstacle. The inputs to the Delaunay refinement algorithm are a *planar straight line graph* (PSLG), which is defined using the vertices and edges of the obstacles, and the set of 'holes', O_h. The vertex set also includes the boundary points, B_d, that defines the whole domain. Let the sequence of triangles given by the triangulation be denoted by S_T. An example of the triangulation generated using this algorithm is shown in Fig. 3(a)

In the second phase, we construct an abstract graph based on the triangulation, S_T, obtained from Delaunay refinement. We construct a graph G_t with vertex set V_t and edge set E_t. A node corresponding to every triangle is added to V_t, and for every pair of adjacent (those that share an edge) triangles, an edge is added to E_t. The construction of the graph is shown in steps 4 - 13 of Algorithm 1. The nodes corresponding to the triangle that contains S is identified as the start node, n_s. The set of nodes that correspond to the triangles that intersect with line segment $\overline{T_s T_f}$ is identified as the goal node set, N_g. For each node $n_g \in N_g$, we list all the paths from n_s to n_g, and add these paths to the set S_{paths}. The set S_{paths} is sorted in ascending order of the path lengths, and for each path in this set a flight corridor is constructed using the triangles corresponding to the sequence of nodes in the path. Algorithm 1 returns the list of the feasible corridors in ascending order of corresponding path lengths in G_t.

The main steps involved in the corridor construction are shown in Figs. 3(a) - 3(c). The figures show the initial position of AV_1, S, the start and end of the rendezvous line segment T_s and T_f, and the obstacles, which are represented as red polygons. The triangulation generated by the Delaunay refinement algorithm are shown as black dashed lines in Fig. 3(a). The abstract graph G_t is superimposed on the triangles, and it is shown with black nodes and blue edges in Fig. 3(b). A corridor is constructed using the path of shortest length in S_{paths}; it is shown as green triangles in Fig. 3(c).

C. Path Optimization

In this section, we present the nonlinear programming formulation for path optimization. The path consists of set of QBCs that are concatenated with G^1 continuity, *i.e.* the path is continuous and has continuous tangent everywhere. The output of the corridor construction presented in Section III-



(a) Triangulation generated using the Delaunay refinement algorithm.

(b) Abstract graph (G_t) constructed using the triangulation.

(c) Corridor constructed based on the shortest path on G_t.

Fig. 3. Illustration of the steps involved in the corridor construction.

Algorithm 1 Corridor construction for the path planning

function SAFECORRIDORS(O_v, O_s, O_h, B_d)
  $O_v \leftarrow O_v \cup B_d$
  $S_T \leftarrow$ DELAUNAYREFINEMENT(O_v, O_s, O_h)
  $V_t, E_t \leftarrow \{\}$
  for $\Delta_u \in S_T$ **do**
    $S_{adj} \leftarrow$ GETADJACENTTRIANGLES(Δ_u)
    if NODE(Δ_u) $\notin V_t$ **then**
      $V_t \leftarrow V_t \cup$ CREATENODE(Δ_u)
    for $\Delta_v \in S_{adj}$ **do**
      if NODE(Δ_v) $\notin V_t$ **then**
        $V_t \leftarrow V_t \cup$ CREATENODE(Δ_v)
      if $(\Delta_u, \Delta_v) \notin E_t$ **then**
        $E_t \leftarrow E_t \cup$ CREATEEDGE(Δ_v) ▷ the weight of the edges are set to the distance between centroids of the triangles
  $G_t \leftarrow$ CREATEGRAPH(V_t, E_t)
  $n_s \leftarrow$ GETSTARTNODE()
  $N_g \leftarrow$ GETGOALNODESET()
  $S_{paths} \leftarrow \{\}$
  for $n_g \in N_g$ **do**
    $S_{paths} \leftarrow S_{paths} \cup$ ALLTPATHS(G_t, n_s, n_g)
  $S_{paths} \leftarrow$ SORT(S_{paths})
  $S_{corrs} \leftarrow \{\}$
  for $s_p \in S_{path}$ **do**
    $S_{corrs} \leftarrow S_{corrs} \cup$ GETCORRIDOR(s_p)
  return S_{corrs}

B is given as a sequence of triangles, and let this be $\Delta = \{\triangle_1, \ldots, \triangle_m\}$, where m is the total number of triangles in the corridor. The path construction involves m QBCs corresponding to the m triangles in Δ. An illustration of this is shown in Fig. 4, where the black x's in each triangle are the control points of the QBC. Let the control points of k^{th} QBC be $\mathbf{p^k} := \{p_0^k, p_1^k, p_2^k\}$, that define the curve as in (1). The idea is to define the path using m sets of these control points, $\mathbf{p^k}$'s, and formulate the objective and the constraints such that they could be solved using a nonlinear program solver.

The Bézier curve always starts at the first control point and ends at the final control point. Therefore, the first control point of the first Bézier curve, p_0^1, should be the starting position of the AV_1, (s_x, s_y). The initial heading of AV_1 is given as s_θ, and therefore the second control point p_1^1 should lie along the ray starting from (s_x, s_y) with a heading s_θ. The path needs to end on the line segment $\overline{T_s T_f}$ with a heading aligned in the direction $T_s \rightarrow T_f$. Because of this, the last control point of the last QBC, p_2^m is constrained to lie on the $\overline{T_s T_f}$, and p_1^m is constrained to lie along a ray starting at p_2^m with heading $t_\theta + \pi$.

To simplify the formulation of constraints, we use a change of coordinates that define the control points of the QBCs. The path consists of m Bézier curves that are concatenated with G^1 continuity. The set $\{p_1^2, \ldots p_1^{m-1}\}$ are the mid-control points of the QBCs corresponding to the triangles 2 to $m-1$. Let δ be the variable that defines distance between the final control point on $\overline{T_s T_f}$ and T_s. Let α_1 be the variable that defines the distance between first control point and second control point of the first QBC, and let α_m be the variable that defines distance between p_1^m and p_2^m. To satisfy the G^1 continuity, each knot point between two successive QBCs is constrained to lie on straight line connecting the middle control points of two successive QBCs. For example, consider two successive QBCs, indexed j and $j+1$. For continuity, the last control point of QBC$_j$, p_2^j, and the first control point of QBC$_{j+1}$, p_0^{j+1} have to be the same, and p_2^j or p_0^{j+1} should lie on the straight line connecting p_1^j and p_1^{j+1}. We define the set of variables γ_j, $j = 1 \ldots m-1$, such that knot points are given as $p_2^j = (1-\gamma_j)p_1^j + \gamma_j p_1^{j+1}$. The bounds $0 \leq \gamma_j \leq 1$ constrain the knot point p_2^j to lie between p_1^j and p_1^{j+1}. Let $\boldsymbol{x}$ represent the set of all the variables defined above. Note that the p_1^i's are Cartesian coordinates of a point, and therefore consists of two variables representing the x and y coordinates. With a little abuse of notation, we define $\boldsymbol{x}$ as the following:

$$\boldsymbol{x} := \{\alpha_1, \alpha_m, \delta, \gamma_1 \ldots \gamma_{m-1}, p_1^2, \ldots p_1^{m-1}\} \tag{4}$$

The control points that define the m QBCs are functions of $\boldsymbol{x}$, and in the following subsections, we define the formulations of the constraints that are imposed and the objective of the nonlinear program as functions of the control points.

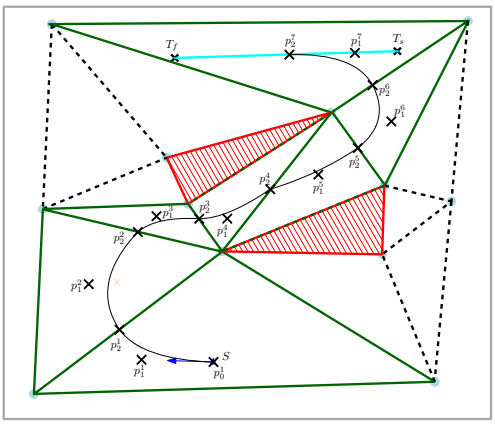


Fig. 4. A QBC for each triangle along the corridor and the corresponding control points are shown. These control points (p_i^j's) are the variables in the nonlinear program and the path is constrained and optimized over these variables.

In the following sections, for brevity, we do not explicitly show the control points as functions of $\boldsymbol{x}$.

1) Initial and Final Configurations: The control point p_0^1 is set to the starting position (s_x, s_y), and the definition of the variables δ, α_1 and α_m implicitly satisfies the initial and final position and heading constraints.

2) Continuity: By the definition of the variables, γ_j's, the knot points always lie on straight line connecting the previous and next control points. This implicitly satisfies the G^1 continuity.

3) Obstacle Avoidance: The main premise for the corridor generation as a sequence of triangles is to formulate the obstacle avoidance constraints. The sequence of triangles, S_{corr}, does not intersect with any of the obstacles. We define each QBC so that the first and third control point lies on edges of a triangle and the mid-control point lies inside the triangle. Therefore, due to the convex hull property of the Bézier curves, it is sufficient to constrain the control points of each QBC to the triangles in S_{corr}, and the resulting path satisfies the obstacle avoidance constraints. Let $\{u^j, v^j, w^j\}$ be the vertices of triangle j in S_{corr}. The mid-control point, p_1^j, of the corresponding QBC is constrained to lie inside the triangle $\triangle u^j v^j w^j$. Without loss of generality, we assume the vertices $\{u^j, v^j, w^j\}$ are in counter-clockwise sequence, the following constraints ensures the mid-control points to lie inside the corresponding triangles. The subscripts x and y represent the corresponding coordinates of the points.

$$(p_{1x}^j - u_x^j)(v_y^j - u_y^j) - (p_{1y}^j - u_y^j)(v_x^j - u_x^j) < 0 \quad (5)$$

$$(p_{1x}^j - v_x^j)(w_y^j - v_y^j) - (p_{1y}^j - v_y^j)(w_x^j - v_x^j) < 0 \quad (6)$$

$$(p_{1x}^j - w_x^j)(u_y^j - w_y^j) - (p_{1y}^j - w_y^j)(u_x^j - w_x^j) < 0 \quad (7)$$

$$\forall j = 2, \dots m-1.$$

Let p_1^j and p_1^{j+1} be the mid-control points of two QBCs corresponding to successive triangles in S_{corr}, and let v_1^j and v_2^j be the vertices of the common edge of the two triangles. The knot point is given as $p_2^j = (1-\gamma_j)p_1^j + \gamma_j p_1^{j+1}$, and it is constrained to lie on the line segment connecting v_1^j and v_2^j using the following constraints.

$$(v_{2y}^j - v_{1y}^j)(p_{2x}^j - v_{1x}^j) - (v_{2x}^j - v_{1x}^j)(p_{2y}^j - v_{1y}^j) = 0 \quad (8)$$

$$\min(v_{1x}^j, v_{2x}^j) \le p_{2x}^j \le \max(v_{1x}^j, v_{2x}^j) \quad (9)$$

$$\forall j = 1, \dots, m-1.$$

4) Maximum Curvature: The maximum curvature of QBC$_j$, $\kappa_{max}(p_0^j, p_1^j, p_2^j)$, is given in (2), and the following set of constraints ensures the curvature constraints are satisfied everywhere along the path,

$$\kappa_{max}(p_0^j, p_1^j, p_2^j) \le \kappa_{limit}, \forall j = 1, \dots, m. \quad (10)$$

5) Rendezvous: For the rendezvous, the two air vehicles AV_1 and AV_2 should arrive at a point on $\overline{T_s T_f}$ at the same time. We assume the air vehicles are traveling at equal and constant speed. This assumption allows the temporal constraints to be posed as as length constraints on the path. The length of each QBC, $L_(p_0^j, p_1^j, p_2^j)$ is given by (3), and the sum of lengths of all the QBCs gives the total length of the path. Let ν_t be the constant speed of the air vehicles; recall t_s is the time of arrival of AV_2 at T_s, and δ is the distance between final control points and T_s. The rendezvous constraints are formulated as follows:

$$\sum_{j=1}^{m} L_(p_0^j, p_1^j, p_2^j) = (t_s + \delta)\nu_t. \quad (11)$$

6) Optimization Problem: The objective of the path optimization is to minimize the time of rendezvous, and since we assume vehicles are traveling at constant speed, this is equivalent to minimizing total length. Due to the equality constraints in (11), it is sufficient to minimize the variable δ. Therefore the nonlinear optimization problem can be posed as the following:

$$\text{minimize } \delta \quad (12)$$

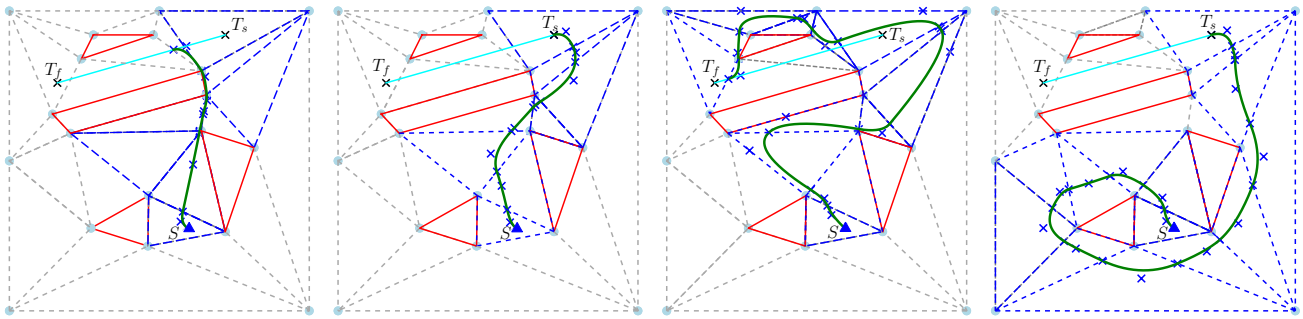
$$\text{subject to: (5)-(7), (8)-(9), (10) and (11)} \quad (13)$$

D. Starting Guess

We solve the above nonlinear optimization problem using gradient based search techniques which are highly dependent on the starting guess provided to find quality solutions. Here, we briefly describe how the starting guess for each of the variables is chosen. The variable δ that defines the last control point of the last QBC is set to be $0.5 * |\overline{T_s T_f}|$. The variables α_1 and α_2 are chosen randomly such that the control points p_1^1 and p_1^m lies inside the first and last triangles respectively. For every QBC except the first and last curves, the second control point, p_1^j, is chosen to be at the centroid of the corresponding triangle $\triangle_j$. The variables γ_i's are chosen such that the control points defined by them lie on the common edges between successive triangles.

IV. Simulation Results

We tested the rendezvous path planning by solving the nonlinear optimization problem in (12)-(13). We generate the set of feasible flight corridors using Algorithm 1; this set of corridors are in ascending order of corresponding



(a) $t_s = 6000$ (b) $t_s = 10000$ (c) $t_s = 20000$ (d) $t_s = 25000$

Fig. 5. Rendezvous paths for scenarios with different values of t_s

path lengths in G_t. Starting with the flight corridor corresponding to the shortest path, we iteratively solve the nonlinear program with the next shortest corridor until a solution is found. The nonlinear program is solved using the sequential least squares quadratic programming (SLSQP) algorithm from the 'NLopt' library [18]. We use the the automatic differentiation package of Julia [19] to compute the gradients of the objective and the constraints that are required in SLSQP algorithm. The code was written in Julia [20] and was run on MacBookPro with 16GB RAM and i7 processor.

The trajectories generated for a scenario with five obstacles and four different arrival times, t_s, of AV_2 at T_s are shown in Fig. 5(a) - Fig. 5(d). The initial position of AV_1 is $(4000, -2000)$ and initial heading direction is 2.6 radians with respect to x-axis. The ends of the rendezvous line segment T_s and T_f are set as $(5000, 7000)$ and $(-2000, 5000)$. The maximum curvature limit of the path is set to be 0.002, which corresponds to a minimum turn radius of 500. The values of t_s corresponding to the four scenarios are 6000, 10000, 20000 and 25000 respectively. The trajectories computed reflect the increasing values of t_s, where the algorithm finds a longer flight corridor for higher t_s. In the second scenario, t_s is sufficiently high and the rendezvous occurs at the starting point. This is as expected as the algorithm aims for a rendezvous as early as possible and hence returns a path to the starting position of the rendezvous line segment. In the third and fourth scenarios, the values of t_s are even larger, and the algorithm finds the feasible trajectory using a longer flight corridor to satisfy the rendezvous constraints.

We have also run the algorithm for hundred instances where the positions of the obstacles and the domain end points are as shown in Fig. 5(a) - Fig. 5(d). For each instance, the initial position and heading of the AV_1 are generated randomly from a uniform distribution. The values of t_s are also generated randomly in the interval $[l_{min}, l_{min} + 5000]$, where l_{min} is the shortest feasible path from initial configuration to any point on $\overline{T_s T_f}$ without rendezvous constraints. We use two different triangulation algorithms to generate the flight corridors; the first one is where the triangles produced are forced to satisfy the Delaunay property, and in the second one a Constrained Delaunay Triangulation (CDT) was used where the triangles were restricted to have a minimum internal angle of 10 degrees. The computational results with these two triangulations are shown in Table I. The CDT performed better with respect to the percentage of successful instances, mean cost of the objective and mean computation times. Please note that the nonlinear program could not find a solution for some instances; due to the random generation of the instances, it is possible that those instances may not have any feasible solution. The computation times reported are sufficiently fast for online trajectory generation.

TABLE I

AVERAGE COMPUTATION TIMES AND COST FOR 100 INSTANCES

Triangulation	Percent Success	Mean cost	Mean Comp. time
Delaunay	78 %	12538	1.52 (secs)
CDT	84 %	11506	1.02 (secs)

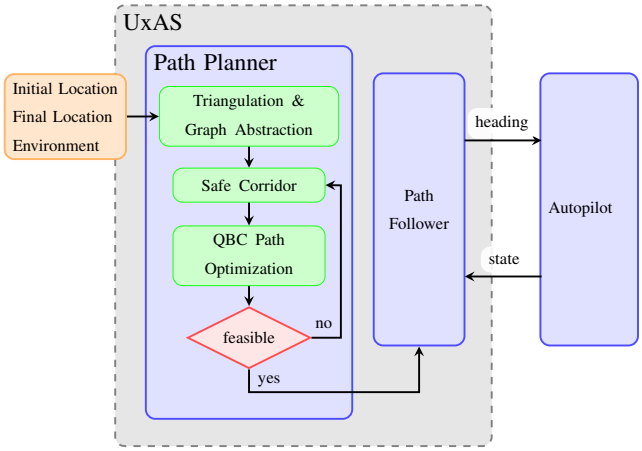


Fig. 6. Graphical depiction of the flight test implementation.

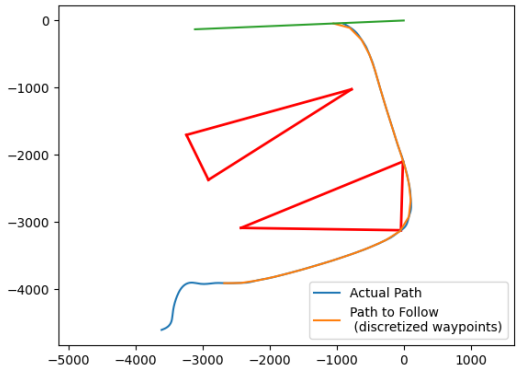
A. Flight Test

The algorithm described in this paper was tested experimentally in a flight test on a Martin UAV Bat-4. This UAV was equipped with a Piccolo Autopilot and an NVIDIA Jetson TX2, which is running a copy of UxAS [21], [22]. UxAS is an extendable set of modular services aimed toward cooperative UAV mission management to handle the follow-

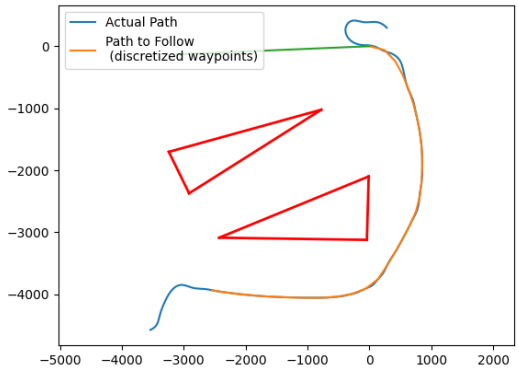
ing high level capabilities of sUAS systems: task planning, task assignment, task execution, path planning, and sensor management.

For this flight test the system functionality can be described as follows. Initially, the aircraft is commanded to loiter to the south west of the start location. The path planner initiates the plan based on a pre-defined start location. This path is discretized into straight line segments and sent to UxAS to follow. UxAS uses the air vehicle state information that is broadcast from the autopilot to compute heading commands to follow the path [23], [24]. The trajectory generated by the algorithm presented may not be curvature continuous, however the path following error for such paths is insignificant for small UAVs.

The flight test occurred at Camp Atterbury Indiana in November of 2020, and the scenario is depicted in Figures 7(a) and 7(b). The figures depict a local tangent plane with units of meters above Camp Atterbury. The obstacles are in red and the path for the (synthetic) AV_2 is green. The orange path is the discretized path computed using the same algorithm and code as describe above and the blue path is the telemetry of the UAV. For this initial test, the start location for the planning was not computed based on the position of AV_1; instead, AV_1 was commanded to loiter south-west of the start location. For this reason, you see the blue path progress to the orange path to follow. From the flight test, we were able to verify that the algorithm is light-enough to be able to run on an on-board ARM processor.



(a)



(b)

Fig. 7. Flight test results with smaller t_s

V. Conclusion

We presented a path planning problem for a rendezvous application that involves two air-vehicles with kinematic motion constraints. The application requires two air vehicles to rendezvous on a predefined line segment. The environment has obstacles and the trajectory generated is subject to obstacle avoidance and curvature constraints. To address the temporal constraints due to rendezvous, we impose length constraints on the path. We propose a methodology that involves generating paths as a concatenation of piecewise quadratic Bézier curves that are G^1. A flight corridor, constructed as a sequence of triangles, is generated using Delaunay refinement algorithm and the curves are bound to the corridor to avoid the obstacles. The existence of closed form solution for maximum curvature and the length of QBCs enables us to pose the problem as a nonlinear optimization problem. This algorithm finds feasible paths that are locally optimal and the proposed methodology is viable for on-board computation. This is corroborated using simulations and experiments. The proposed method can be easily extended to the scenario where the paths for the two vehicles needed to be generated simultaneously. The nonlinear program solvers are dependent on good starting solutions and it is a potential drawback that needs to be addressed in the future work. Also, the path planning assumes no disturbance due to wind, considering wind disturbances is another direction of future work.

References

[1] L. E. Dubins, "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, vol. 79, no. 3, p. 497, jul 1957.

[2] X.-N. Bui, J.-D. Boissonnat, P. Soueres, and J.-P. Laumond, "Shortest path synthesis for dubins non-holonomic robot," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1994, pp. 2–7.

[3] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[4] J. L. Blanco, M. Bellone, and A. Gimenez-Fernandez, "TP-space RRT–kinematic path planning of non-holonomic any-shape vehicles," *International Journal of Advanced Robotic Systems*, vol. 12, no. 5, p. 55, 2015.

[5] D. Ghosh, G. Nandakumar, K. Narayanan, V. Honkote, and S. Sharma, "Kinematic constraints based Bi-directional RRT (KB-RRT) with parameterized trajectories for robot path planning in cluttered environment," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8627–8633.

[6] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[7] S. G. Manyam, D. Casbeer, A. Von Moll, and Z. Fuchs, "Optimal dubins paths to intercept a moving target on a circle," in *American Control Conference (ACC)*, 2019, pp. 828–834.

[8] S. G. Manyam, S. Rathinam, D. Casbeer, and E. Garcia, "Tightly Bounding the Shortest Dubins Paths Through a Sequence of Points,"

Journal of Intelligent and Robotic Systems: Theory and Applications, vol. 88, no. 2-4, pp. 495–511, 2017.

[9] C. Chen, Y. He, C. Bu, J. Han, and X. Zhang, "Quartic Bézier Curve Based Trajectory Generation for Autonomous Vehicles with Curvature and Velocity Constraints," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 6108–6113.

[10] J. Chen, T. Liu, and S. Shen, "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1476–1483.

[11] Y. Chen, Y. Cai, J. Zheng, and D. Thalmann, "Accurate and Efficient Approximation of Clothoids using Bézier Curves for Path Planning," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1242–1247, 2017.

[12] F. Gao, W. Wu, Y. Lin, and S. Shen, "Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 344–351.

[13] W. Sun, G. Tang, and K. Hauser, "Fast uav trajectory optimization using bilevel optimization with analytical gradients," in *American Control Conference (ACC)*, 2020, pp. 82–87.

[14] H. Deddi, H. Everett, and S. Lazard, "Interpolation with Curvature Constraints," INRIA Lorraine, Villers-Lès-Nancy, France, Research Report RR-4064, 2006. [Online]. Available: https://hal.inria.fr/inria-00072572

[15] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.

[16] M. Kallmann, "Shortest Paths with Arbitrary Clearance from Navigation Meshes," in *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Madrid, Spain, Jul. 2010, pp. 159–168.

[17] J. R. Shewchuk, "Delaunay refinement algorithms for triangular mesh generation," *Computational Geometry*, vol. 22, no. 1-3, pp. 21–74, 2002.

[18] S. G. Johnson, "The nlopt nonlinear-optimization package," *URL https://arxiv. org/abs/1607.07892*.

[19] J. Revels, M. Lubin, and T. Papamarkou, "Forward-mode automatic differentiation in Julia," *arXiv:1607.07892 [cs.MS]*, 2016. [Online]. Available: https://arxiv.org/abs/1607.07892

[20] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM Review*, vol. 59, no. 1, pp. 65–98, 2017.

[21] S. Rasmussen, D. Kingston, and L. Humphrey, "A Brief Introduction to Unmanned Systems Autonomy Services (UxAS)," in *International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018.

[22] "OpenUxAS." [Online]. Available: https://github.com/afrl-rq/OpenUxAS

[23] D. Nelson, D. Barber, T. McLain, and R. Beard, "Vector Field Path Following for Miniature Air Vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 519–529, Jun. 2007.

[24] R. Beard and T. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.