Autonomous Quadrotor Trajectory Planning and Control for In-Flight Aerial Vehicle Capture

Zachary Olkin

July 2, 2020

Abstract

This paper develops a path planning and control architecture for an interceptor air vehicle designed to capture small target UAVs, which are assumed to be at rest or moving slowly. The proposed architecture has three main parts. First, a geometric path planner is developed to determine the trajectory for the quadrotor to travel from its initial location to the rendezvous location. The second component of the architecture is a minimal-time thrust profile generation algorithm. The algorithm represents the quadrotor's acceleration as a B-Spline and uses the convex-hull property of the spline to transform the constraints into functions of the control points so an optimization problem can be formulated to minimize time to capture. Lastly, a low-level controller tracks the orientation and thrust commands. In this architecture, the thrust profile and geometric path are generated independently. The proposed planning and control architecture provide one key advantage over alternative optimal control approaches: by separating the thrust profile generation from the geometric path generation, the minimal time trajectory generation problem has less variables and constraints to calculate, thus allowing on-board calculation of the thrust profile. Results are presented demonstrating aerial capture of targets in simulation.

Contents

1 Introduction

Recently, quadrotors have become a very popular method to accomplish various tasks ranging from first response to surveying land to cinematography. In general, small Unmanned Aerial Vehicles (UAVs) are now easily accessible and because of this, the chance that a person uses a UAV in a manner that is dangerous to the general public or in restricted airspace increases. The focus of this paper is on a new trajectory planning and control architecture to allow an autonomous quadrotor vehicle to intercept and capture another flying vehicle. This will allow for the ability to safely retrieve an undesired UAV from the sky. In this paper, the target UAV is assumed to be stationary or moving slowly. Future works may address a quickly moving target.

Past works have developed methods for path planning, trajectory generation, and quadrotor controllers, but no other papers have used these to develop an architecture for aerial vehicle capture. Unlike other works, this paper will show a different method for independently generating the acceleration profile of the quadrotor and the geometric path in 3D space which will allow for fast computation of the trajectories in real time.

Past works such as [9], [3], and [11] present work on minimum-time trajectory generation. In [3], a method for developing a minimal-time trajectory plan is shown, but the method is not feasible in real time do to the number of variables and constraints in the non-linear programming problem. While [11] considers generating a minimal time trajectory given a set of waypoint and velocities between those waypoints, its control architecture does not provide close tracking of the target with errors of $0.5m - 1m$. [2] shows a method for generating a dynamically feasible path for a controller that uses the differential flatness property, but this method assumes that the total time to complete the path is already known. This work will show a method for following an arbitrary curve in minimal-time through 3D space through real time calculations.

2 Quadrotor Model

The model used for simulation and controller development in this paper uses the rigid body dynamics of the quadrotor, a motor model, induced drag, and rotor flapping. The quadrotor has two reference frames: the body frame, $\mathcal{B}$, and the inertial frame, $\mathcal{A}$. The orthonormal basis of each coordinate system can be written $\{\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}\}$ and $\{\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}\}$ respectively. The orientation of the quadrotor can be completely defined by the rotation matrix $R = \{\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}\} \in SO(3)$ from $\mathcal{A}$ to $\mathcal{B}$ where $SO(3)$ denotes the special orthogonal group.

Let m define the mass of the quadrotor, $g \in \mathcal{A}$ define the acceleration due to gravity, and $J \in \mathbb{R}^{3\times 3}$ represent inertia matrix in the body frame. $F \in \mathcal{B}$ gives the total force acting on the vehicle and $v \in \mathcal{A}$ gives the velocity of the quadrotor in the world frame. $\omega \in \mathcal{B}$ gives the angular velocities about the axis $\{\mathbf{b_1}, \mathbf{b_2}, \mathbf{b_3}\}$ while $\mathbf{M}$ defines the moments about those axis. $\hat{}$ denotes the hat map which creates the lie algebra, $\mathfrak{so}(3)$, from the real valued vector. This is equivalent to generating a skew symmetric matrix from the vector, ω, such that $\hat{\omega} v = \omega \times v$ for any $v \in \mathbb{R}^3$.

The rigid body dynamics of the quadrotor are given in Eqs. (1a) - (1c)

$$m\dot{v} = mg\mathbf{a_3} + RF \tag{1a}$$

$$\dot{R} = R\hat{\omega} \tag{1b}$$

$$J\dot{\omega} = -\omega \times J\omega + \mathbf{M} \tag{1c}$$

The force and moment generated by the ith rotor can be given as a function of the rotor angular velocity, ω_i, by Eq. 2.

$$T_i = k_F \omega_i^2, \; M_i = k_M \omega_i^2 \tag{2}$$

Given the rotor speeds, the total moments and thrust acting on the body is given by Eq. 3. Let L define the length from the center of rotation of the rotor to the center of mass of the quadrotor and T define the force due to the rotors in the direction of $\mathbf{b_3}$. The constants k_F and k_M are defined by the physical properties of the rotor, such as rotor disk area, which will not be investigated in this paper.

$$\begin{bmatrix} T \\ \\ \mathbf{M} \\ \\ \end{bmatrix} = \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & k_F L \\ k_F L & 0 & k_F L & 0 \\ k_M & -k_M & k_M & -k_M \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{3}$$

The simulation used in this paper uses a first order approximation of the motor speeds which drive the rotor speeds. The governing differential equation of the first-order approximation for the motor dynamics is

$$\dot{\Omega}(t) + \tau\Omega(t) = kV \tag{4}$$

where Ω is the rotational velocity of the motor shaft, V is the input voltage, and τ is the time constant of the system (ref).

2.1 Lumped Parameter Model for Rotor Flapping and Induced Drag

The aerodynamic drag effects due to rotor flapping are included in this paper as they are used in the simulation. As the vehicle moves horizontally through the air, the advancing tip of the rotor will have a higher absolute speed thus giving it more lift. This causes the rotor to spin at an angle no longer parallel to the arm of the quadrotor. For a first order approximation of the effects of rotor flapping, define Eq. 5 as given in [1]. A_{1c} and A_{1s} are positive constants defined by the mechanical properties of the rotor.

$$A_{flap} = \frac{1}{\omega R}\begin{bmatrix} A_{1c} & -A_{1s} & 0 \\ A_{1s} & A_{1c} & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5}$$

The induced drag is modeled as a simple drag force proportional to the vehicle velocity in the $\mathbf{b_1}$ and $\mathbf{b_2}$ directions. Thus the induced drag is given by Eq. 6 where d_x and d_y are the drag coefficients.

$$D = \begin{bmatrix} d_x & 0 & 0 \\ 0 & d_y & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

Both rotor flapping and induced drag can be modeled as a single term by defining

$$E = A_{flap} + D \tag{7}$$

Then one can write the total force on the quadrotor as $F = T\mathbf{b_3} + E(Rv)$.

3 Trajectory Planning and Control Architecture

This trajectory planning and control architecture is split into 3 important parts: generating the geometric path in 3D space, creating the acceleration profile as a function of time, and the low level controllers to track the desired orientation of the quadrotor. The key advantage of this architecture is that by separating the geometric path generation from the thrust profile generation a solution to the minimal-time trajectory generation problem can be solved on-line as compared to [3]. This will allow the quadrotor to arrive at the location of the target as quickly as possible.

3.1 Geometric Path Planning

The path is defined in 3D space by a set of 2 cubic Bezier Curves and a straight line that brings the quadrotor directly on top of the target vehicle. By using Bezier curves, a path that is C^1 continuous can be easily defined. The curve is also guaranteed C^2 continuity at all points except the waypoints. Also, the first derivatives at the waypoint, start, and stop locations are easily controlled. By using Bezier Curves, the dynamic feasability can be controlled (for example, not violating the maximum tilt angle) whereas other methods, like potential field methods, would require post processing to allow the trajectory to be dynamically feasible. For this paper, it is assumed that the quadrotor starts at a location $S = [x_s \; y_s \; z_s]^T \in \mathcal{A}$ such that $z_s < z_{target}$, normally with $z_s = 0$ indicating that the quadrotor is starting on the ground.

The waypoint connecting the two Bezier Curves is defined to be at the same height as the target vehicle, but at a set distance away from the target in the (x, y) plane. By defining a waypoint in this way, we can guarantee no collisions assuming a sufficiently large buffer and that the target is stationary or slowly moving. The angle relative to $y = 0$ on the (x, y) plane where the waypoint lies is defined to be the same as the angle from the $y = 0$ line to the straight line connecting the start and target's location in the (x, y) plane. This allows the quadrotor to always make progress toward the target. In the case that the quadrotor starts directly below the target any angle may be chosen. Now, with a buffer distance, b, and an angle, θ, the location of the waypoint can be written

$$W_1 = [x_{target} + x_{radius} + bcos(\theta),\; y_{target} + y_{radius} + bsin(\theta),\; z_{target}]^T \tag{8}$$

The end location of the second Bezier Curve is defined as a distance, d, directly above the target such that after arriving at this point the vehicle only needs to descend then activate the capture mechanism.

This distance can be calculated to not violate the maximum tilt angle, β, of the quadrotor by solving $d_{min} = tan(\beta)\sqrt{x_{diff}^2 + y_{diff}^2}$ with x_{diff} and y_{diff} equal to the difference in x and y between the waypoint and the target vehicle. So long as $d \geq d_{min}$ the path will not violate the maximum tilt angle of the vehicle.

$$W_2 = [x_{target},\ y_{target},\ z_{target} + d]^T \tag{9}$$

To generate a dynamically feasible path, the curve should be continuous on C^1, including at the waypoints. A Bezier curve is guaranteed to be C^1 continuous, but the control points, $\mathbf{P_i}$, must be specially placed to guarantee C^1 continuity at the waypoints. A cubic Bezier Curve in $\mathbb{R}^3$ is defined as

$$\mathbf{C}(t) = (1-t)^3\mathbf{P_0} + 3(1-t)^2t\mathbf{P_1} + 3(1-t)t^2\mathbf{P_2} + t^3\mathbf{P_3} \quad 0 \leq t \leq 1 \tag{10}$$

where $\mathbf{P_i} \in \mathbb{R}^3$ is the ith control point of the spline in 3D space. To enforce that the spline starts where the quadrotor starts, the 0th control point is set as $\mathbf{P_0} = S$. For the first curve, the end point must be at W_1, so $\mathbf{P_3} = W_1$. The derivatives at the start and end are used to determine the other two control points. Choose the derivative at the start of the path to be $(W_1 - S)/\alpha$ where $\alpha > 0$ is a scaling factor chosen manually. This points the path in the direction of the first waypoint. Then, choose the derivative at then end of the curve as $(W_2 - W_1)/\alpha$. This points the path in the direction of the endpoint while at the first waypoint. The derivative of a cubic Bezier Curve is

$$\mathbf{C}'(t) = 3(1-t)^2(\mathbf{P_1} - \mathbf{P_0}) + 6(1-t)t(\mathbf{P_2} - \mathbf{P_1}) + 3t^2(\mathbf{P_3} - \mathbf{P_2}) \tag{11}$$

Thus $\mathbf{C}'(0) = 3(\mathbf{P_1} - \mathbf{P_0})$ and $\mathbf{C}'(1) = 3(\mathbf{P_3} - \mathbf{P_2})$. Substituting the derivatives and known control points gives Eqs. 12 and 13.

$$\frac{W_1 - S}{\alpha} + \mathbf{P_0} = \mathbf{P_1} \tag{12}$$

$$\frac{W_2 - W_1}{\alpha} + \mathbf{P_3} = \mathbf{P_2} \tag{13}$$

At this point all the control points of the first Bezier curve are known so generating the control points for the second curve is trivial. Use the same method but change the start location, S, to W_1, set the start derivative equal to the derivative at the end of the previous curve, and finally the end derivative of the second curve can be arbitrary since the vehicle comes to a stop at the top of the path. Then re-calculate the control points for the second curve. At this point a spline has been defined that is dynamically feasible and is guaranteed to start, end and pass through the desired waypoint.

Later, the acceleration profile will need to be "curved" onto the geometric path which will require knowledge of the tangent, normal and binormal vectors as well as the curvature and torsion at each point on the path. This section will discuss the calculations of those values.

Calculating the Frenet-Serret apparatus (tangent, normal, and binormal vectors as well as curvature and torsion) for the curve as function of the control points is not a reasonable approach given each component of the curve is a 3rd order polynomial. Because of this, the values will be calculated numerically. To remove the issue of having two different parameters, one for each curve; each curve can be evaluated from $0 \leq s \leq 1$ at an interval spacing of ds then the two arrays of 3D locations can be concatenated to give a single array of all the locations from the start point to the end point. The entire curve is then called $r(k)$ where $k \in \mathbb{Z}$ is the index into the array.

The values of the tangent vector can be easily computed from the control points as Eq. 14 with $\mathbf{C_i}'(t)$ defined above. Evaluating t with the spacing ds allows each tangent vector to be associated with a k to make $\mathbf{T}(k)$.

$$\mathbf{T}(t) = \frac{\mathbf{C_i}'(t)}{|\mathbf{C_i}'(t)|} \tag{14}$$

The derivative of $\mathbf{T}(k)$ can be numerically calculated as $\mathbf{T}'(k) = (\mathbf{T}(k) - \mathbf{T}(k-1))/ds$ thus allowing the calculation of $\mathbf{N}(k)$ as

$$\mathbf{N}(k) = \frac{\mathbf{T}'(k)}{|\mathbf{T}'(k)|} \tag{15}$$

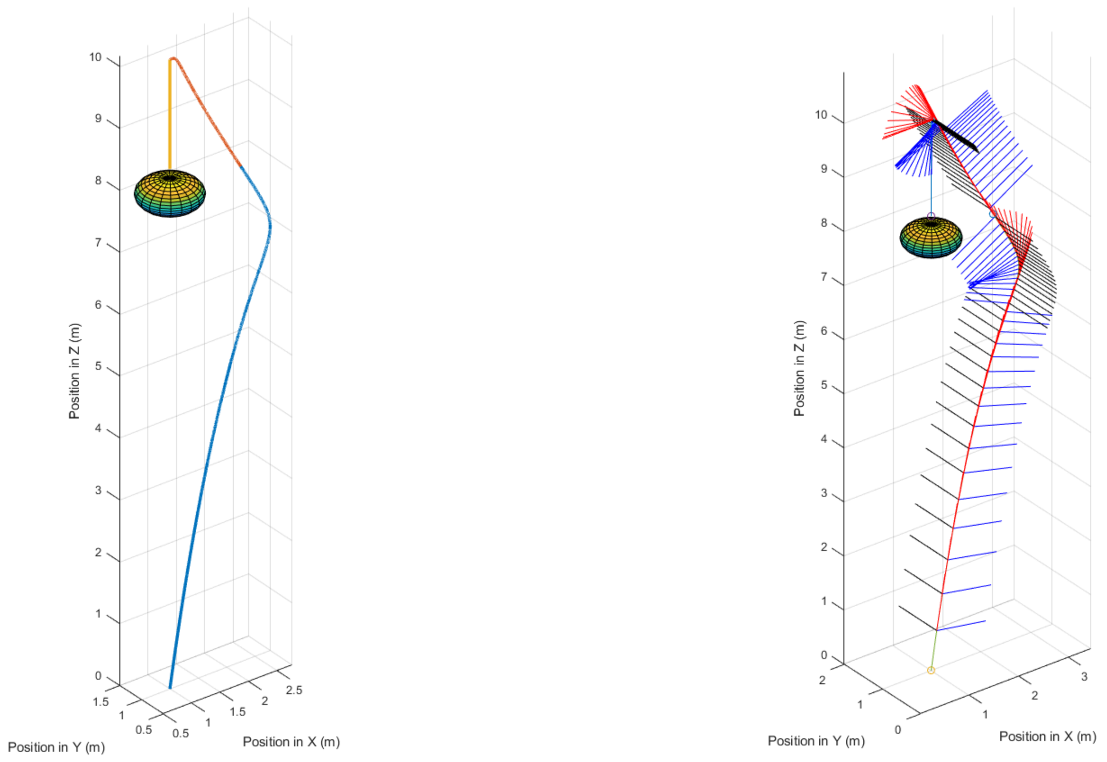


Figure 1: The geometric path generated with and without the tangent, normal and binormal vectors. The ellipsoid at the end of the path represents the target vehicle. The paths are represented by the blue and orange lines in the figure on the left. The tangent vector is red, the normal vector is blue and binormal is black in the figure on the right.

Then for each k we can calculate $\mathbf{B}(k) = \mathbf{T}(k) \times \mathbf{N}(k)$. Curvature, κ is defined as Eq. 16.

$$\kappa(k) = \frac{|\mathbf{T}'(k)|}{|\mathbf{C}'(k)|} \tag{16}$$

Lastly, torsion, τ, is given in Eq. 17.

$$\tau(k) = -\mathbf{N}(k) \cdot \mathbf{B}'(k) \tag{17}$$

This section has discussed the calculation of waypoints, curves and the resulting Frenet-Serret apparatus that accompanies the curve. Now a acceleration profile is needed to allow the quadrotor to follow the spline.

3.2 Acceleration Profile Generation

To calculate the acceleration profile, $a(t)$, the acceleration is separated from the geometric path to make the problem a one dimensional problem with the only constraints being on the maximum velocity, acceleration and jerk of the body. Because of this, the number of constraints and variables is much smaller than that of [3].

One of the goals of the architecture is to allow for the quadrotor to arrive at the endpoint in as quick a time as possible. To achieve this, the acceleration profile is defined as a B-Spline, which means that its integrals and derivatives are also B-Splines. The acceleration considered is the acceleration of the body, but in no particular direction. This means that the total acceleration will move the body the required arc length in minimal time, but the low level controller will need to "bend" the profile to the geometric path. Do to this fact, though, the optimization problem now only has body dynamics constraints and considers only one dimension of travel making it quicker to compute than [3].

As given in [3], B-Splines can be defined by

- A set of $(n+1)$ control points. These take on a similar role in B-Splines and Bezier Curves: they describe the location and derivatives of the spline.
- A polynomial of degree k
- A set of $(m+1)$ knots which define where the spline switches from one polynomial to another polynomial.

Like a Bezier spline, a B-Spline is defined by a set of polynomials, but a B-Spline defines the entire spline as one curve whereas the Bezier spline requires multiple curves and the stitching between the curves is manual. There are many different types of B-Splines, but this paper will only be using non-uniform, clamped B-Splines. Being non-uniform means that the spacing between the knots can be different between each knot. Clamping means that the start and end of the B-Spline are guaranteed to be at the first and last control points. This is achieved through increasing the multiplicity of the knots at both ends of the spline to be of multiplicity $k+1$.

For this application, the knots are times throughout the path. The control points define a convex hull around the spline. A proof of this fact can be found in [12]. Since the control points define a convex hull, the spline can never exceed the values of the control points, and thus the optimization problem can now be solved in terms of the control points and knots which together completely define the curve and its constraints.

A B-Spline is defined recursively by Eqs. 18a, 18b, and 18c. τ is the vector of knots, and t is the time. See [12] for more details.

$$B_{i,0}(t) = \begin{cases} 1 & \text{if} \quad \tau_i \le t \le \tau_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{18a}$$

$$B_{i,k}(t) = \omega_{i,k}(t) B_{i,k-1}(t) + (1 - \omega_{i+1,k}) B_{i+1,k-1}(t) \tag{18b}$$

where

$$\omega_{i,k}(t) = \begin{cases} \frac{t-\tau_i}{\tau_{i+k}-\tau} & \text{if} \quad \tau_{i+k} > \tau_i \\ 0 & \text{otherwise} \end{cases} \tag{18c}$$

To create a spline effected by the control points, we simply re-write the spline as

$$\mathcal{C} = \sum_{i=0}^{n} B_{i,k} p_i \tag{19}$$

where p_i is the ith control point.

From the geometric path, the total arc length of the path is known, so the final control point (the value of which the position is clamped to) is defined to be the total arc length of the path. Since the quadrotor always starts at the beginning of the path, the first control point is always set to 0. Now the position spline is clamped to start at $s = 0$ and end at $s = s_{final}$. This leaves the selection of all the control points and knot spacing. To write the constraints as a function of the control points and knots, the derivative of the B-Spline with respect to time must be taken. Eq. 20 gives the equation for the control points of the first derivative of a B-Spline (more information can be found in [12]).

$$p_i' = \frac{k}{\tau_{i+k+1} - \tau_{i+1}} (p_{i+1} - p_i) \tag{20}$$

To determine the minimum time acceleration profile to achieve this, a optimization problem with a linear cost function and non-linear constraints is solved. Define $\mathbf{x}$ as a vector of the knot spacing (difference between consecutive knots) and the control points, then solve

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & D(\mathbf{x}) \\ \text{subject to} \quad & f_{derivatives}(\mathbf{x}) \leq 0 \end{aligned} \tag{21}$$

with

$$D(\mathbf{x}) = \sum_{j=0}^{n-k} \Delta\tau_j \tag{22}$$

This optimization problem attempts to minimize the sum of the knot spacing which in turn minimizes the time. Given the B-Spline is clamped to start at 0 and s_{final}, the path must go the entire distance, and the non-linear constraint function, $f_{derivatives}(\mathbf{x})$, constrains the velocity, accelerations and jerks to be dynamically feasible. Let l be the lth derivative of the position spline and $d_{i,l}$ be the maximum value for the lth derivative. Then the constraint equation is written

$$\forall(i),\ l \in [0,3],\ f_{derivatives}(\mathbf{x}) = \left(p_i^{(l)}\right)^2 - d_{i,l}^2 \tag{23}$$

Lastly, the start and end velocities and accelerations are constrained to 0. Assuming the maximum snap is very high, there is no need to constrain the start and end jerk.

Solving this gives the set of control points and knot spacing (from which the knots can be recovered) for the position spline. Using Eq. 20 one can construct the subsequent derivatives. The output of a spline with limits: $v \leq 3m/s$, $a \leq 5m/s^2$, $j \leq 11m/s^3$ and total arc length of $8.37m$ is shown in Fig. 2.

For the purposes of this report, there is no need to adjust the yaw of the quadrotor. Thus the yaw, $\psi(t)$, is set to 0 for all time.

3.3 Low Level Controllers

Given the geometric path and the acceleration profile, the quadrotor now needs to be controlled to follow the path in minimal time. The control architecture consists of a thrust controller which attempts to track the desired acceleration, position and velocity and also determines the desired orientation, as well as an orientation controller to track the desired rotation.

Defining $\mathbf{e_r}$ as the cross track position error, $\mathbf{e_v}$ as the velocity error, $\mathbf{e_a}$ as the acceleration error, $\mathbf{a_{des}}$ as the desired acceleration, and $\mathbf{F}_{ext}$ as an approximation of the induced drag and rotor flapping forces, the commanded force can be written as

$$\mathbf{F}_{com} = k_r \mathbf{e}_r + k_v \mathbf{e}_v + k_a m \mathbf{e}_a + m \mathbf{a}_{des} - mg\mathbf{a_3} + \mathbf{F}_{ext} \tag{24}$$

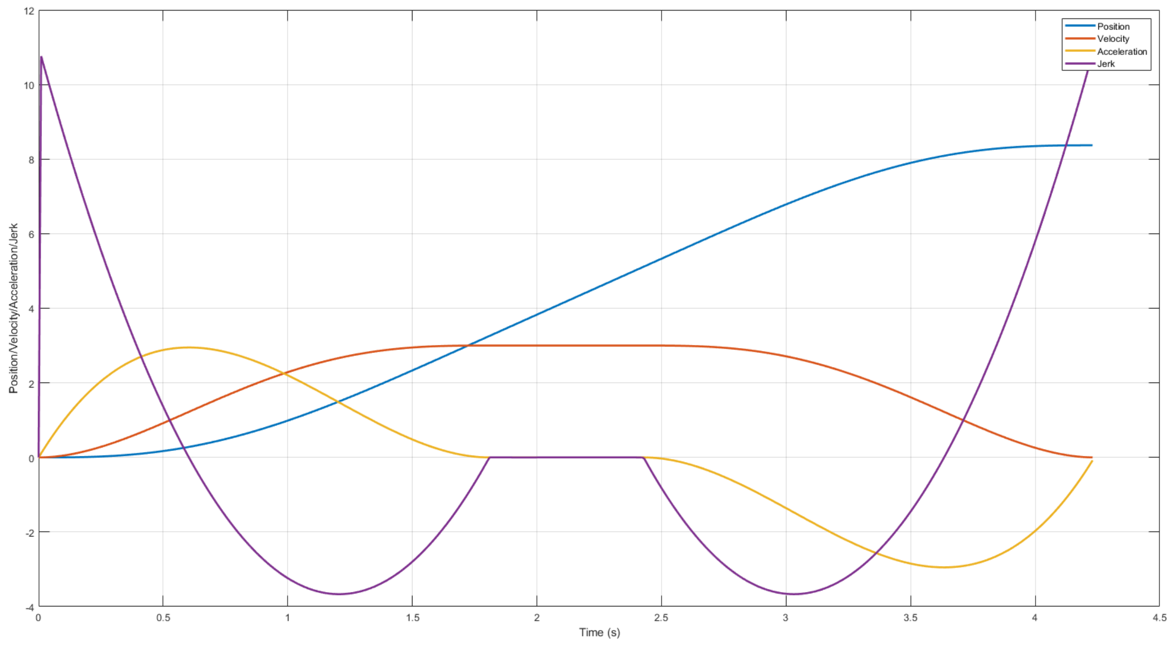


Figure 2: A B-Spline generated to traverse a path. The start and end of the position, velocity and acceleration and constrained. The end of the jerk is not constrained because the assumption that the jerk can change almost instantly is made.

k_r, k_v, and k_a can be written as diagonal 3×3 matrices to provide different gains for each direction. The $\mathbf{F}_{ext}$ is needed as an attempt to counteract the drag forces. In this architecture, the acceleration is time based and generated without knowledge of the vehicle's direction of travel which means at the time of the acceleration profile generation, the drag forces are unknown and thus cannot be accounted for. If this algorithm is run without attempting to counteract the drag force, then the quadrotor never reaches its final destination because whenever it applies force horizontally there is a drag force, and since the acceleration lookup is time based, rather than position based, if the quadrotor does not hit its target accelerations then it cannot compensate for it later.

$\mathbf{e_r}$ is calculated as the cross track error. A k is selected from within a sphere of radius L around the current position of the quadrotor which gives the index of the reference position is known and denoted as γ. The cross track error is computed as shown in Eq. 25 from [6].

$$\mathbf{e}_r = ((r(\gamma) - P) \cdot \mathbf{N}(\gamma))\mathbf{N}(\gamma) + ((r(\gamma) - P) \cdot \mathbf{B}(\gamma))\mathbf{B}(\gamma) \tag{25}$$

The velocity error is based on the current velocity of the quadrotor and the speed generated from the acceleration profile at any time, t. Let k_{min} give the index into the path of the location in the path closest to the quadrotor. The velocity should always be in the direction of the tangent to the path, giving

$$\mathbf{v}_{des} = v(t)\mathbf{T}(k_{min}) \tag{26}$$

and a velocity error

$$\mathbf{e}_v = \mathbf{v}_{des} - v \tag{27}$$

where v is the current velocity of the quadrotor. A similar approach is used when calculating the acceleration error, except that the acceleration is not only in the direction of the tangent. We can derive the acceleration as follows

$$\mathbf{a}_{des} = \mathbf{v}'_{des} = a(t)\mathbf{T}(k_{min}) + v(t)\mathbf{T}'(k_{min}) \tag{28a}$$

simplifying,

$$\mathbf{a}_{des} = a(t)\mathbf{T}(k_{min}) + \kappa(k_{min})v^2(t)\mathbf{N}(k_{min}) \tag{28b}$$

Where $a(t)$ and $v(t)$ come from the acceleration profile. Then $\mathbf{e}_a$ is calculated in the same manner as Eq. 27.

$$\mathbf{e}_a = \mathbf{a}_{des} - a \tag{29}$$

$\mathbf{F}_{ext}$ is an approximation of the rotor flapping and induced drag force on the quadrotor. It is best approximated by gathering empirical results, but given the time constraints of this report, results are only shown in simulation, so a model is used to approximate the external forces as shown in section 2.1.

Now that the commanded force vector is known, a controller on the orientation is developed. The controller takes the form of Eq. 30. This is an orientation controller that acts in $SO(3)$ such as the controller from [7]. For a thorough treatment on controllers in $SO(3)$ see [4].

$$\mathbf{M} = k_R\mathbf{e}_R + k_\omega\mathbf{e}_\omega \tag{30}$$

The commanded force defines a desired body coordinate frame given as

$$\mathbf{b}_{3,des} = \frac{\mathbf{F}_{com}}{\|\mathbf{F}_{com}\|} \tag{31}$$

Since the force from the thrust can only be applied in the $\mathbf{b}_3$ direction, $\mathbf{b}_{3,des}$ is now the desired orientation of the quadrotor. To get the entire orthonormal basis, we must introduce an intermediate frame. By Euler's Rotation Theorem, it is understood that a single rotation can be decomposed into two separate rotations with an intermediate frame. So an intermediate frame, $\mathcal{Z}$, is defined with axis $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3$ such that the only difference between $\mathcal{A}$ and $\mathcal{Z}$ is a rotation about $\mathbf{a_3}$ by $\psi(t)$. Therefore one can define $\mathbf{z}_1 = [cos(\psi(t)),\ sin(\psi(t)),\ 0]^T$. To generate $\mathbf{b}_{2,des}$ the fact that $\mathbf{b}_{2,des}$ is always perpendicular to $\mathbf{b}_{3,des}$ and $\mathbf{z}_1$ is used (see fig. (ref)). $\mathbf{b}_{2,des}$ can be computed with Eq. 32.

$$\mathbf{b}_{2,des} = \frac{\mathbf{b}_{3,des} \times \mathbf{z}_1}{\|\mathbf{b}_{3,des} \times \mathbf{z}_1\|} \tag{32}$$

Finally, $\mathbf{b}_{1,des}$ must be perpendicular to both $\mathbf{b}_{3,des}$ and $\mathbf{b}_{2,des}$ which logically leads to the equation

$$\mathbf{b}_{1,des} = \mathbf{b}_{2,des} \times \mathbf{b}_{3,des} \tag{33}$$

The desired rotation matrix is defined as $R_{des} = [\mathbf{b}_{1,des},\ \mathbf{b}_{2,des},\ \mathbf{b}_{3,des}]$. The rotation error function is Eq. 34 where $\vee$ denotes the vee map which recovers the vector from the lie algebra $\mathfrak{so}(3)$.

$$\mathbf{e}_R = \frac{1}{2}\Big(R_{des}^T R - R^T R_{des}\Big)^\vee \tag{34}$$

Eq. 30 requires an angular velocity error, $\mathbf{e}_\omega$, so a desired angular velocity needs to be computed. A formulation for computing the desired angular velocities from [2] is used. This utilizes the differential flatness property of the quadrotor to compute the angular velocity from the commanded force and desired jerk. To calculate the jerk, one will need the derivative of Eq. 28b. Eq. 35 give this derivative and thus shows the equation for jerk with the k_{min} on the normal, tangent, and binormal vectors omitted along with the k_{min} on the curvature and torsion omitted. A detailed derivation of kinematics in 3D space can be seen in [5].

$$j_{des}(t) = (j(t) - \kappa^2 v^3(t))\mathbf{T} + (3\kappa v(t)a(t) + \dot{\kappa}v^2(t))\mathbf{N} + \kappa\tau v^3(t)\mathbf{B} \tag{35}$$

From [2], the desired angular velocities are

$$\omega_{des} = \begin{bmatrix} -\mathbf{h}_\omega \cdot \mathbf{b_2} \\ -\mathbf{h}_\omega \cdot \mathbf{b_1} \\ \dot{\psi}(t)\mathbf{b_3} \cdot \mathbf{a_3} \end{bmatrix} \tag{36}$$

with $\mathbf{h}_\omega$ as defined in Eq. 37.

$$\mathbf{h}_\omega = \frac{m}{\mathbf{F}_{com}}(j_{des}(t) - (\mathbf{a_3} \cdot j_{des}(t)\mathbf{a_3}) \tag{37}$$

To generate the angular velocity error, the angular velocity needs to be compared to the actual angular velocity.

$$\mathbf{e}_\omega = \omega - R^T R_{des}\omega_{des} \tag{38}$$

With this error, the orientation controller can be used to calculated the moments, **M**, that are used to determine target rotational velocities for the rotors on the quadrotor by inverting the matrix in Eq. 3.

A motor controller using proportional and feed forward terms are used in this work. Details are not presented here because the specific implementation of the motor controller is not important for the architecture to function properly.

At this point, a geometric path and acceleration profile have been developed along with methods to follow the independently generated parts. Next, the results of this development will be shown in simulation.

4 Results

The simulation used in this paper was custom developed and written in Matlab using the equations presented in section 1.

The path was generated using the following parameters:

- Start point: $(2, 1, 0)$
- Location of the target vehicle: $(1, 1, 8)$
- Buffer around the vehicle: $0.75m$
- Maximum tilt angle: $\frac{\pi}{10}$

The B-Spline was generated with the following parameters:

- Polynomial degree: $k = 5$
- 18 knots, with multiplicity 6 at either end of the spline
- $n + 1 = 12$ control points

The simulation was run with various controller parameters depending on the maximum speed that the profile was limited to. All simulations were run with the controller operating at $100hz$. Fig. 3 shows the path in dashed orange and the actual path the quadrotor took when the B-Spline was limited to $1m/s$ in magenta. Fig. 4 shows the position error of the quadrotor from the closest location on the path. One can see that the error is generally less than 2cm although it occasionally grows to be near 5cm when operating at less than $1m/s$. The section of the path that brings the quadrotor directly above the vehicle is not followed as that was not discussed in this paper. To follow it though, another acceleration profile could be generated and followed.

Fig. 5 shows the location of the quadrotor while following the path with a maximum velocity at $2m/s$. One can see in Fig. 6 that the error is not only larger than when traveling at the $1m/s$ maximum, but also the controller shows a noticeable amount of oscillation. This is most likely due to the aggressive tuning of k_R such that the vehicle can follow the curve just below the first waypoint. If k_R is decreased then the quadrotor will not follow the path and rather shoot off into the sky. If the parameter is much larger then the oscillations will grow to become unstable. Increasing k_ω to reduce the oscillations will slow the tracking and limit the ability to follow the curve.

Attempting to travel faster than $2m/s$ is not dynamically feasible around a curve such as the one generated by the path. Trials with higher maximum velocities were not controllable around such a curve and resulted in the quadrotor becoming unstable.

Fig. 7 shows how the accelerations were tracked in all 3 dimensions for the $1m/s$ path. Although there is slight oscillation in the actual acceleration, the effect is not see very much in the position error because the mass of the quadrotor acts like a dampener on the oscillations.

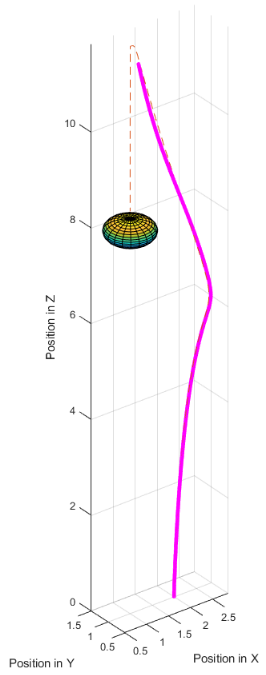


Figure 3: A demonstration of the entire process with the maximum velocity limit set at 1m/s. The dashed orange line is the desired trajectory while the magenta line is the actual path of the quadrotor.

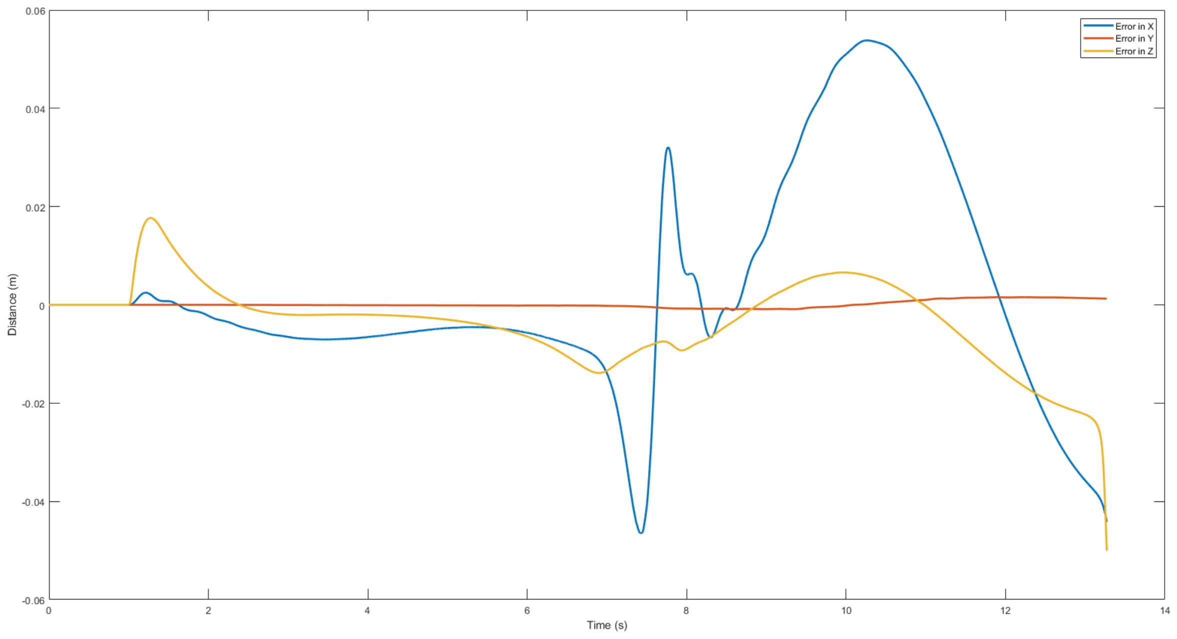


Figure 4: Position error over time in X, Y, and Z as the quadrotor follows a path with maximum speed set to 1m/s.

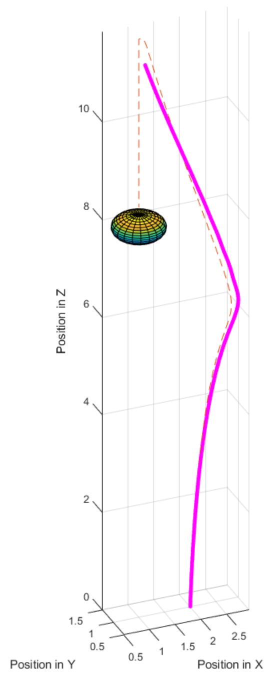


Figure 5: A demonstration of the entire process with the maximum velocity limit set at 2m/s. The dashed orange line is the desired trajectory while the magenta line is the actual path of the quadrotor.

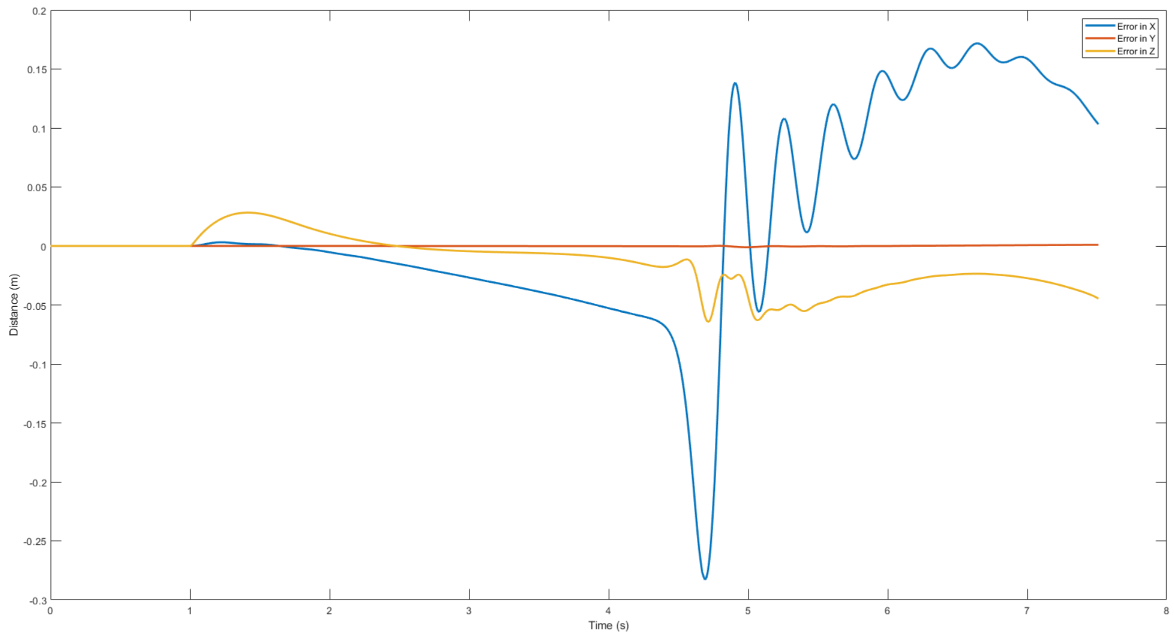
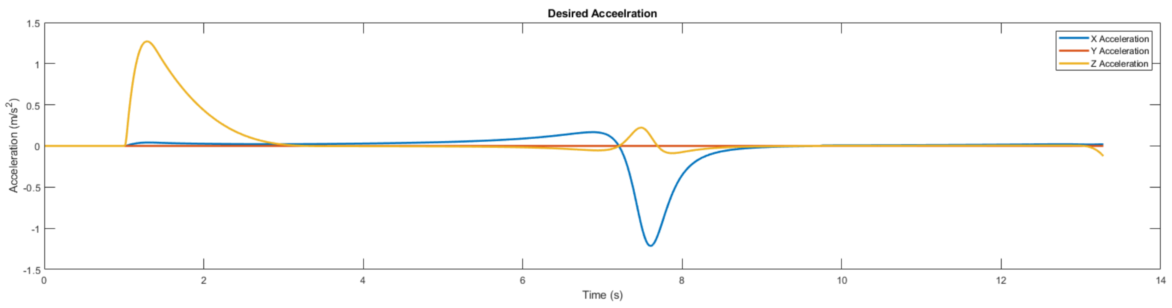


Figure 6: Position error over time in X, Y, and Z as the quadrotor follows a path with maximum speed set to 2m/s.



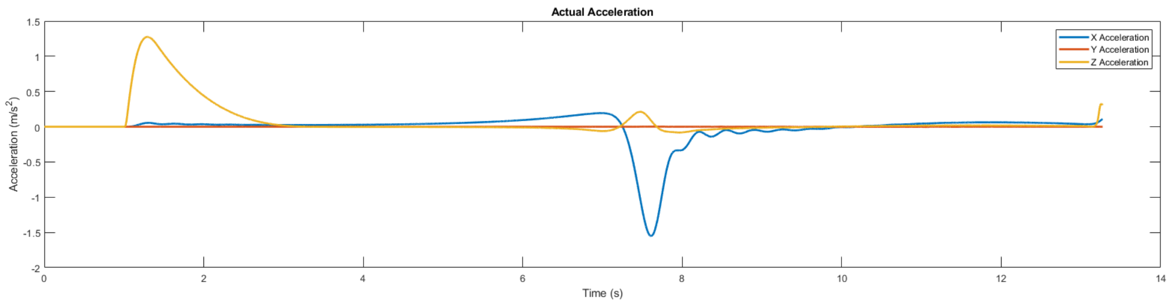


Figure 7: The desired acceleration in all 3 directions and the actual acceleration in all 3 directions when traveling at a limit of $1m/s$

5 Conclusion

This paper presented a new way to generate time-optimal acceleration profiles for quadrotor vehicles such that a target vehicle can be captured. The new method utilizes a geometric path generation algorithm using cubic Bezier Curves. Using the total arc length of the path, a non-linear optimization problem can be formulated by representing the acceleration at a B-Spline and writing the constraints using the control points and knots of the B-Spline. Finally, the acceleration profile is adapted to the geometric path using the Frenet-Serret apparatus of the path. Then the targets are tracked using a thrust and orientation controller.

The architecture is run on a simulation accounting for the rigid body dynamics, rotor flapping, induced drag and a first order motor model. It is shown that the architecture provides reasonable tracking performance of less than 5cm at slower speeds and less than 20cm at higher speeds. With this architecture, the quadrotor can travel about $12.25m$ in $7.5s$. This shows a significant speed performance over methods such as minimum snap trajectory generation as used in [2].

Yet this is a preliminary work. The next step for improvement would be to impose velocity and acceleration constraints based on the curvature of the path to enable speeding up and slowing down over the course of the path. Attempting to do this with the B-Spline optimization method may prevent it from running on real-time systems, so a different algorithm may need to be used.

Accounting for the rotor drag and induced drag properly in real life may prove quite challenging. Do to this fact, another improvement would be to parameterize the path in terms of its own arc length, then assign a velocity, acceleration and jerk to each point on the path. This would allow the acceleration profile to be a function of position rather than time and in turn would eliminate the need for the external force compensation.

With the above listed improvements, this architecture should be usable in the real world to capture target vehicles.

References

[1] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicles," IEEE Robotics and Automation Letters, Aug. 2012, doi: 10.1109/MRA.2012.2206474.

[2] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, May 2011, pp. 2520–2525, doi: 10.1109/ICRA.2011.5980409.

[3] G. Rousseau, C. Stoica Maniu, S. Tebbani, M. Babel, and N. Martin, "Minimum-time B-spline trajectories with corridor constraints. Application to cinematographic quadrotor flight plans," Control Engineering Practice, vol. 89, pp. 190–203, Aug. 2019, doi: 10.1016/j.conengprac.2019.05.022.

[4] Yun Yu, Shuo Yang, Mingxi Wang, Cheng Li, and Zexiang Li, "High performance full attitude control of a quadrotor on SO(3)," in 2015 IEEE International Conference on Robotics and Automation (ICRA), May 2015, pp. 1698–1703, doi: 10.1109/ICRA.2015.7139416.

[5] A. Havens, "Curvature, Natural Frames, and Acceleration For Plane and Space Curves," p. 25.

[6] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," The International Journal of Robotics Research, vol. 31, no. 5, pp. 664–674, Apr. 2012, doi: 10.1177/0278364911434236.

[7] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in 49th IEEE Conference on Decision and Control (CDC), Atlanta, GA, Dec. 2010, pp. 5420–5425, doi: 10.1109/CDC.2010.5717652.

[8] B. Rubi, B. Morcego, and R. Perez, "A Survey of Path Following Control Strategies for UAVs Focused on Quadrotors," Journal of Intelligent and Robotic Systems, 2019, doi: 10.1007/s10846-019-01085-z.

[9] S. Spedicato and G. Notarstefano, "Minimum-time trajectory generation for quadrotors in constrained environments," arXiv:1706.06478 [math], Jun. 2017, Accessed: Jul. 01, 2020. [Online]. Available: http://arxiv.org/abs/1706.06478.

[10] T. Kunz and M. Stilman, "Time-Optimal Trajectory Generation for Path Following with Bounded Acceleration and Velocity,"

[11] G. Hoffmann, S. Waslander, and C. Tomlin, "Quadrotor Helicopter Trajectory Tracking Control," presented at the AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii, Aug. 2008, doi: 10.2514/6.2008-7410.

[12] De Boor, C., 1978. A practical guide to splines. volume 27 of Mathematics of Computation.