2021 International Conference on Unmanned Aircraft Systems (ICUAS)
Athens, Greece. June 15-18, 2021

2021 International Conference on Unmanned Aircraft Systems (ICUAS) | 978-1-6654-1535-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICUAS51884.2021.9476686

Efficient Online Jerk-limited Trajectory Generation for Multicopters Using Barrier Functions

Abdul Hanif Zaini[1], Kun Cao[1] and Lihua Xie[1]

Abstract— **This paper proposes a jerk-limited trajectory generation algorithm using a discrete implementation of the Zeroing Control Barrier Function method. This method achieves greater computational efficiency than previous works as it requires only a single pass in the forward simulation of the triple integrator and does not require a bisection search. A time-optimal trajectory can be generated from any arbitrary initial state with asymmetric constraints on velocity, acceleration and jerk for a target position with any final velocity and zero final acceleration. In addition, we demonstrate a one step ahead trajectory controller combined with an Extended State Observer in simulation of a hexacopter model.**

I. Introduction

The generation and tracking of jerk-limited trajectories is critical for the performance of systems that are incapable of achieving instantaneous acceleration or torque, or might be damaged by such sudden changes. An example of a jerk-limited trajectory is shown in Fig. 3. Examples of such systems include multicopters or multi-rotor helicopters [1], and robot actuators [2]. Multicopters are vehicles that generate thrust for vertical lift via two or more rotors. Its attitude is controlled by varying the relative speed of its rotors and in so doing allow it to move laterally. A common version of this is the quadcopter or quadrotor [3]. Another example is the 6 rotor multicopter or hexacopter in Fig. 1. Multicopters may require smooth trajectories to protect its payload from damage by vibration or sudden accelerations.

Various solutions already exist to produce these jerk-limited trajectories but typically require heavy computation. Methods requiring relatively heavy computation usually involve some optimization method to produce a three-dimensional trajectory with obstacle avoidance. Recent examples include a receding horizon control method in [4], mixed integer quadratic programming to solve for third degree polynomial splines [5], and quadratic programming with n-th degree polynomial trajectories [6].

However, recently in [1], Lai et al. introduced a computationally efficient method for obstacle avoidance and jerk-limited trajectory generation. Their method generates three single-axis jerk-limited trajectories for use within safe flying corridors between way-points. The single-axis trajectory parameters are calculated using kinematic equations and a bisection search of the deceleration start time. A trajectory can be produced from an arbitrary initial state to the desired position and endpoint velocity with zero final acceleration. In addition, the trajectory is time-optimal with asymmetric lower and upper limits on jerk, acceleration. The trajectory time series is then obtained by forward simulating with a triple integrator and the trajectory parameters.

In [7] and [8], Beul and Behnke proposed the use of the analytic solution to a system of 3rd order polynomial equations representing a time-optimal trajectory with arbitrary desired end velocities. While it is shown to be fast in most cases, the authors also reported that the method required tens of thousands of mathematical computations for the scenarios where the resulting trajectory did not have certain constant acceleration or deceleration segments. However, it was also stated that these scenarios are not common and is still solvable in milliseconds on a recent laptop computer.

Another recent example of single-axis jerk-limited trajectory generation involves the use of FIR (Finite Impulse Response) filters to produce a jerk-limited trajectory from an acceleration-limited one [9], [10]. However, the method produces position overshoots which Xia et al. overcomes in [10] by checking and altering acceleration iteratively. Hence, the method requires multiple passes through the trajectory time-series.

A simpler method is reported in [2], which uses and modifies three reference trajectories. However, the method is limited to zero start and end velocities and accelerations.

Main Contributions: The method we propose in this paper is designed to be even more efficient by generating the single-axis jerk-limited time-optimal trajectory in only a single pass. Unlike [1], our method does not require a bisection search for the start of deceleration and just generates the trajectory while forward simulating on the triple integrator model. This is done by using kinematic equations with a discrete implementation of the Zeroing Control Barrier Function (ZCBF) similar to our previous work [11] which is based on [12]. For each time step, only three values of jerk input need to be checked which are the minimum, maximum and zero jerk values. Hence, the resulting computation time is only linear in time horizon or length of trajectory and is not affected by certain scenarios unlike [7] and [8]. Our method is also able to produce trajectories from arbitrary initial values with a non-zero desired velocity and zero acceleration at the target position while enforcing asymmetric constraints on jerk and acceleration. We also demonstrate a single-step ahead trajectory controller combined with an Extended State Observer (ESO) in simulation of a 6 rotor multicopter with drag and wind disturbance.

The research is partially supported by the ST Engineering - NTU Corporate Lab through the NRF corporate lab@university scheme.

[1]The authors are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 `hanif.zaini@ntu.edu.sg`, `ahanifbz@gmail.com` (A.H. Zaini); `KUN001@e.ntu.edu.sg` (K. Cao); `elhxie@ntu.edu.sg` (L. Xie)

Organization: Section II defines the model and objective. Barrier certificates and the discrete treatment are discussed in Section III. The algorithm is presented in Section IV. The one step ahead trajectory controller simulation results are shared in Section V. Finally, some concluding remarks are drawn in Section VI.

II. Problem Definition

A. *Multi-rotor Helicopter Dynamics*

The multi-rotor helicopter such as the one shown in Fig. 1 generates thrust vertically using a combination of multiple fixed-pitch propellers. Its attitude is controlled by varying the relative speed of the rotors. It moves laterally by tilting to provide thrust in the desired direction.

Briefly, the model from [13] is

$$\ddot{\mathbf{p}}_e = -g\mathbf{z} + \tfrac{1}{m}R\mathbf{T}_b \ , \tag{1}$$

$$\dot{R} = RS\left(\boldsymbol{\omega}_b\right) \ , \tag{2}$$

$$\dot{\boldsymbol{\omega}}_b = J_b^{-1}\left(-\boldsymbol{\omega}_b \times J_b\boldsymbol{\omega}_b + \boldsymbol{\tau}_b\right) \ , \tag{3}$$

where subscripts e and b denote Earth and body-attached frames of reference, $\mathbf{p}_e \in \mathbb{R}^3$ and $\boldsymbol{\omega}_b \in \mathbb{R}^3$ are position and angular velocity respectively, g is the gravitational constant, m is the mass, J_b is the inertia matrix, R is the rotation matrix converting from body to Earth frames, $\mathbf{z}$ is the vector $[0\ 0\ 1]^\intercal$, and $\mathbf{T}_b$ and $\boldsymbol{\tau}_b$ are thrust and torque respectively. $S\left(.\right)$ is the skew symmetric matrix operator such that $S\left(\mathbf{a}\right)\mathbf{b} = \mathbf{a}\times\mathbf{b}$.

From [1], the bounds on achievable acceleration due to dynamics (1)-(3) are

$$\|\ddot{\mathbf{p}}_e + g\mathbf{z}\| \le \frac{T_{\text{max}}}{m} \ , \tag{4}$$

$$\ddot{z}_e \ge \ddot{z}_{e,\text{min}} \ge \frac{T_{\text{min}}}{m} - g \ , \tag{5}$$

where T_{max} and T_{min} are the maximum and minimum total thrust available, and z_e is the vertical position.

The smallest upper bound on jerk is

$$\|\dddot{\mathbf{p}}_e\| \le \frac{T_{\text{min}}}{m}\omega_{\text{max}}, \tag{6}$$

where ω_{max} is the maximum rotation rate of the multirotor platform.

Bounds (4)-(6) simply serve to illustrate that the system cannot achieve instantaneous changes in velocity and acceleration. However, for this work we assume that the desired jerk can be achieved instantaneously as the angular velocity response for multi-rotor platforms in practice is significantly faster. We may set smaller bounds on jerk when generating a trajectory.

B. *Triple-Integrator Model*

With consideration for velocity, acceleration and jerk limits, the multirotor helicopter system can be approximated to a constrained triple integrator model for generating a trajectory as it is differentially flat [3]. We consider the single axis triple integrator for this work,

$$\dot{p} = v, \ \dot{v} = a, \ \dot{a} = j, \tag{7}$$

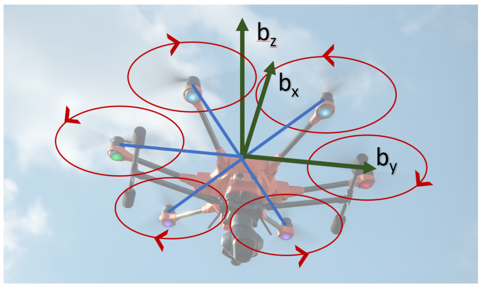

Fig. 1: Body-attached frame denoted by $[b_x,\ b_y,\ b_z]$ and propeller spin directions superimposed on a photo of a 6-rotor multicopter or hexacopter in flight viewed from below.

where p, v, a and j are the position, velocity, acceleration and jerk respectively. We define the states as $x = [p\ v\ a]^\intercal$ and jerk as the input $j \in \mathbb{R}^1$. The system constraints are

$$v_{\text{min}} \le v \le v_{\text{max}} \tag{8}$$

$$a_{\text{min}} \le a \le a_{\text{max}} \tag{9}$$

$$j_{\text{min}} \le j \le j_{\text{max}} \tag{10}$$

where v_{min}, a_{min} and j_{min} are strictly negative constants, and v_{max}, a_{max} and j_{max} are strictly positive constants set by the user with consideration for (4)-(6).

In application, we may apply the single axis trajectory generation to multiple orthogonal axes to produce a smooth 3D motion. An example of this is found in [1] and Section V.

C. *Objective*

The objective of this work is to design a computationally-efficient online time-optimal trajectory generation method for the constrained triple-integrator system (7)-(10). Specifically, the trajectory to be generated is a time series consisting of the required position, velocity, acceleration and jerk at fixed time intervals, T, starting from the current system states, x, up to some desired time horizon. The trajectory is required to bring the constrained system to the desired velocity and zero acceleration simultaneously at the waypoint position.

III. Zeroing Control Barrier Function Background

The proposed method presented in this paper is formulated based on the Zeroing Control Barrier Function (ZCBF). We give a brief review of the ZCBF here based on [12], [14] and our previous work [11].

The ZCBF is a function formulated for generating a set of control inputs that guarantees the forward invariance of a desired set of safe states. In other words, we wish to ensure that the system approaches and stays within this safe set.

Consider the linearly controlled system

$$\dot{x} = f\left(x\right) + g\left(x\right)u, \tag{11}$$

where $x \in \mathbb{R}^n$ is the state, $u \in U \subset \mathbb{R}^m$ is the control input, and f and g are locally Lipschitz continuous.

The set of safe states is defined as $\mathcal{C} \subset \mathbb{R}^n$. The ZCBF candidate, $h : \mathbb{R}^n \to \mathbb{R}$, is then formulated such that

$$\mathcal{C} = \{x \in \mathbb{R}^n \mid h(x) \geq 0\} . \tag{12}$$

Definition III.1 (From [12]). A continuous function $\kappa : (-b, a) \to \mathbb{R}$ for some $a, b > 0$ is an extended class-$\mathcal{K}$ function if it is strictly increasing and $\kappa(0) = 0$.

Definition III.2 (From [12]). Given the set (12) for a smooth function $h : \mathcal{D} \to \mathbb{R}$, where $\mathcal{C} \subseteq \mathcal{D} \subset \mathbb{R}^n$, h is called a ZCBF if there exists an extended class-$\mathcal{K}$ function κ such that

$$\sup_{u \in U} \left\{ \dot{h}(x) + \kappa(h(x)) \right\} \geq 0 \tag{13}$$

for all $x \in \mathcal{D}$, and

$$\dot{h}(x) = \frac{\partial h(x)}{\partial x} (f(x) + g(x) u) \ . \tag{14}$$

The function κ is used to modify the approach speed towards the boundary of set $\mathcal{C}$.

From (13) and (14), the control space that ensures that the states, x, stays in set $\mathcal{C}$ is

$$S(x) = \left\{ u \in U \,\middle|\, \dot{h}(x) + \kappa(h(x)) \geq 0 \right\}, \ x \in \mathcal{D} \ . \tag{15}$$

From theorems in [14] and [12], any Lipschitz continuous controller $u : \mathcal{D} \to \mathbb{R}^m$ where $u \in S(x)$ guarantees $\mathcal{C}$ to be forward invariant and asymptotically stable in $\mathcal{D}$.

In [12], Wang et al. formulated the ZCBF based on the minimum deceleration distance to avoid collision between two robots. The resulting control input constraint is used to bound each robot's nominal controller.

However, for real discrete-time systems such as robots and unmanned multirotor aircraft, the continuous formulation of [12] produces control input bounds that may not be sufficient to ensure safe set invariance at the next time step. Some systems may operate with relatively low control frequencies perhaps due to computational complexity or communications bottleneck. Hence, the maximum speed and acceleration may have to be limited to reduce overshoots and avoid collision.

In our previous work [11], we presented a discretized modification to bound a holonomic robot within some fixed distance from a desired nominal path. The state at next time step is used to determine the permissible control space. This next state is obtained using the system model and the range of available control inputs.

If there exists a ZCBF candidate, $h(x)$, as described in Definition 2, the required control space is simply

$$S_d(x) = \{u \in U \mid h(x(t+T)) \geq 0\} \ , \tag{16}$$

where t and T are the current sampling time and interval respectively. Note that Definition 2 requires that $h(x)$ is smooth and zero at the boundary of set $\mathcal{C}$. This requirement is no longer necessary in the discrete case as we just require that $h(x(t+T)) \geq 0$. However, defining it according to Definition 2 simplifies the search for the control space boundary by providing direction.

The continuous version in (15) allowed for initial states to be such that $h(x) < 0$ as long as $S(x)$ in (15) is not empty. The discretized version in (16) is stricter where $h(x(t+T)) \geq 0$ must be reachable within one time step. This is a problem when external disturbances can push the system states out of the safe set $\mathcal{C}$.

In [11], we obtain conservative bounds on S_d by searching the space U in discrete intervals. If S_d is empty, the $u \in U$ that maximizes $h(x(t+T))$ is used.

In summary, the use of the ZCBF, $h(x)$, is to map the system's possible states to a scalar value that indicates the system is in the desired set if $h(x) \geq 0$. The smoothness requirement of $h(x)$ and the existence of class-$\mathcal{K}$ function (Definitions 1 and 2) enables the design of a Lipschitz continuous controller that satisfies (15). The discretized version for discrete systems in (16) does not need these requirements but they are useful for simplifying control space search. However, both cases can become infeasible due to unmodeled disturbances, i.e., $S(x)$ or $S_d(x)$ are empty.

In the sequel, we adapt the ZCBF and the discretized approach of (16) into an optimization problem to generate a jerk-limited trajectory in one dimension. Instead of generating a control set that guarantees forward invariance, we use the ZCBF to drive the system to the boundary of the safe set $\mathcal{C}$ regardless of initial state.

IV. Jerk-limited Barrier-based Trajectory Generation

A. Optimization with ZCBF

In our application, we formulate the ZCBF such that the boundary of set $\mathcal{C}$, where $h(x) = 0$, is the objective. As (7) is a single input system we simply require that the ZCBF be also monotonic in control input $j \in [j_{\min},\ j_{\max}]$.

The ZCBF here is inspired by [12] which was developed on in our previous work in [11]. In both works, the ZCBF chosen represents the distance of the robot from some boundary subtracted by the minimum required distance to decelerate to zero velocity relative to the boundary. In [12], only maximum acceleration is used to calculate the deceleration distance. In [11], we consider the maximum jerk as well but the acceleration is non-zero after reducing velocity to zero. In the current application, we require the acceleration to be zero as it reaches the desired velocity at the waypoint position.

The control input at each time step to generate the trajectory is obtained by solving

$$j_{\text{input}} = \underset{j \in [j_{\min},\ j_{\max}]}{\operatorname{argmin}} \ (|h(x(t+T))| + d_{\text{acc}} + d_{\text{vel}}) \ , \tag{17}$$

where

- $h(x(t+T))$ is the ZCBF function at next time step and is strictly decreasing in $j \in [j_{\min},\ j_{\max}]$ (defined in Section IV-B), and
- d_{acc} and d_{vel} are offset values encoding soft constraints on acceleration and velocity respectively (defined in Section IV-C).

Solving (17), is simplified by searching only 3 values of jerk input which are $j_{\min}$, 0 and $j_{\max}$. Only these three values of jerk input are needed to generate a time-optimal trajectory in addition to maximizing acceleration in either direction where possible [15].

B. Discrete ZCBF Design

The design objective for the discrete ZCBF candidate, $h_{\mathrm{T}} = h(x(t+T))$, is to encode safe combinations of states, x, and input, j, for the constrained system (7) such that if the system approaches $d_0 = 0$ and $h_{\mathrm{T}} \geq 0$, then $v_0 \to v \leq v_d$ and $a_0 \to a \leq 0$. d_0, v_0, a_0 and v_d are defined as follows.

$$d_0 = |p_{\text{desired}} - p(t)| \ , \tag{18}$$

$$v_0 = \begin{cases} v(t) & \text{if } p_{\text{desired}} - p(t) \geq 0 \ , \\ -v(t) & \text{otherwise} \ , \end{cases} \tag{19}$$

$$a_0 = \begin{cases} a(t) & \text{if } p_{\text{desired}} - p(t) \geq 0 \ , \\ -a(t) & \text{otherwise} \ , \end{cases} \tag{20}$$

$$v_d = \begin{cases} v_{\text{desired}} & \text{if } p_{\text{desired}} - p(t) \geq 0 \ , \\ -v_{\text{desired}} & \text{otherwise} \ , \end{cases} \tag{21}$$

where p_{desired} and v_{desired} are the desired waypoint position and velocity respectively. We simplify the problem by considering the positive direction to be towards p_{desired}.

Algorithm 1 is used to determine h_{T}. Subscript T denotes the value at next time step where the time interval is of length T. $j \in \{j_{\min},\ 0,\ j_{\max}\}$ is the jerk value to be checked. a_0, v_0, d_0 and v_d are from (18)-(21). $a_{\min}$, $j_{\min}$ and $j_{\max}$ are constrains set by the user in (9) and (10).

Algorithm 1's function is summarized in the two following propositions.

Proposition IV.1. *Using Algorithm 1, h_T is strictly decreasing in $j \in [j_{min},\ j_{max}]$ with all other inputs constant.*

Proof. In lines 1-2 of Algorithm 1, with other inputs constant, a_{T} and v_{T} are strictly increasing in j. Consequently, in lines 4-9, $t_{a_{\mathrm{T}}\to 0}$ and $v_{a_{\mathrm{T}}\to 0}$ are also strictly increasing in j.

For $v_{a_{\mathrm{T}}\to 0} > v_d$, the total deceleration distance, $s_1 + s_2 + s_3$, increases with a_T, v_T, $v_{a_{\mathrm{T}}\to 0}$ and $t_{a_{\mathrm{T}}\to 0}$. Thus $\max(s_1 + s_2 + s_3,\ 0)$ is non-decreasing in j.

For $v_{a_{\mathrm{T}}\to 0} \leq v_d$ and $a_{\mathrm{T}} > 0$, s_1 similarly increases with a_T, v_T and $t_{a_{\mathrm{T}}\to 0}$. Considering only positive deceleration distances, $\max(s_1 + s_2 + s_3,\ 0)$ is smaller than in the case $v_{a_{\mathrm{T}}\to 0} > v_d$ with $s_2 = 0$ and $s_3 = 0$ and decreased jerk, j. If instead $a_{\mathrm{T}} \leq 0$, $s_1+s_2+s_3 = 0$ and therefore non-decreasing in j. Thus $\max(s_1 + s_2 + s_3,\ 0)$ is also non-decreasing in j here.

Hence, h_{T} is strictly decreasing in j because d_{T} is also strictly decreasing while $\max(s_1 + s_2 + s_3,\ 0)$ is non-decreasing. ■

Remark 1. This strictly decreasing property of h_{T} ensures that it is clear that increasing acceleration using positive j drives h_{T} to 0 or more negative. Intuitively, increasing acceleration also drives $d_0 \to 0$. In the next proposition, we also show how this and maintaining $h_{\mathrm{T}} \geq 0$ achieves the ZCBF design objective.

Algorithm 1: Calculate h_{T}

```
Input  : $j$, $a_0$, $v_0$, $d_0$, $v_d$, $a_{\min}$, $j_{\min}$, $j_{\max}$ and $T$
Output: $h_{\mathrm{T}}$
1  $a_{\mathrm{T}} = a_0 + jT$
2  $v_{\mathrm{T}} = v_0 + a_0 T + \frac{1}{2} j T^2$
3  $d_{\mathrm{T}} = d_0 - \left(v_0 T + \frac{1}{2} a_0 T^2 + \frac{1}{6} j T^3\right)$
4  if $a_T > 0$ then
5      $t_{a_{\mathrm{T}}\to 0} = -a_{\mathrm{T}} / j_{\min}$
6      $v_{a_{\mathrm{T}}\to 0} = v_{\mathrm{T}} + a_{\mathrm{T}} t_{a_{\mathrm{T}}\to 0} + \frac{1}{2} j_{\min} {t_{a_{\mathrm{T}}\to 0}}^2$
7  else
8      $t_{a_{\mathrm{T}}\to 0} = -a_{\mathrm{T}} / j_{\max}$
9      $v_{a_{\mathrm{T}}\to 0} = v_{\mathrm{T}} + a_{\mathrm{T}} t_{a_{\mathrm{T}}\to 0} + \frac{1}{2} j_{\max} {t_{a_{\mathrm{T}}\to 0}}^2$
10 end if
11 if $v_{a_T\to 0} > v_d$ then
12     $a_1 = \max(a_{\mathrm{T}},\ a_{\min})$
13     if $a_T > 0$ then
14         $a_- = -\sqrt{2\left(v_{a_{\mathrm{T}}\to 0} - v_d\right) / \left(\frac{1}{j_{\max}} - \frac{1}{j_{\min}}\right)}$
15     else
16         $a_- = -\sqrt{\left(2\left(v_T - v_d\right) - \frac{{a_1}^2}{j_{\min}}\right) / \left(\frac{1}{j_{\max}} - \frac{1}{j_{\min}}\right)}$
17     end if
18     $a_3 = \max(a_-,\ a_{\min})$
19     $t_3 = -a_3 / j_{\max}$
20     $t_1 = (a_3 - a_1) / j_{\min}$
21     $v_2 = v_{\mathrm{T}} + a_1 t_1 + \frac{1}{2} j_{\min} {t_1}^2$
22     $v_3 = v_d - a_3 t_3 - \frac{1}{2} j_{\max} {t_3}^2$
23     $t_2 = \max((v_3 - v_2)/a_{\min},\ 0)$
24     $s_1 = v_{\mathrm{T}} t_1 + \frac{1}{2} a_1 {t_1}^2 + \frac{1}{6} j_{\min} {t_1}^3$
25     $s_2 = v_2 t_2 + \frac{1}{2} a_{\min} {t_2}^2$
26     $s_3 = v_3 t_3 + \frac{1}{2} a_3 {t_3}^2 + \frac{1}{6} j_{\max} {t_3}^3$
27 else
28     $s_2 = 0$
29     $s_3 = 0$
30     if $a_T > 0$ then
31         $s_1 = v_{\mathrm{T}} t_{a_{\mathrm{T}}\to 0} + \frac{1}{2} a_{\mathrm{T}} {t_{a_{\mathrm{T}}\to 0}}^2 + \frac{1}{6} j_{\min} {t_{a_{\mathrm{T}}\to 0}}^3$
32     else
33         $s_1 = 0$
34     end if
35 end if
36 return $h_T = d_T - \max(s_1 + s_2 + s_3,\ 0)$
```

Remark 2. Negative distances are unnecessary as the algorithm is for restricting movement towards the waypoint. Moreover, it will add to the complexity with additional cases to ensure $\max(s_1 + s_2 + s_3,\ 0)$ is non-decreasing.

Proposition IV.2. *Consider system* (7) *with input $j \in [j_{min},\ j_{max}]$ and no external disturbances. Using Algorithm 1, if the current states, x, are such that an input $j \in [j_{min},\ j_{max}]$ exists that produces $h_T \geq 0$, then an input $j \in [j_{min},\ j_{max}]$ also exists such that $h_T \geq 0$ for the next time step. Then, if*

j is chosen at each time step satisfying $h_T \geq 0$, then it is guaranteed that if $d_0 \to 0$,

1) $v_0 \to v \leq v_d$ *and*
2) $a_0 \to a \leq 0$.

Proof. For system (7), lines 1-3 of Algorithm 1 perfectly predicts remaining distance, d_{T}, to p_{desired} as well as the acceleration and velocity at the next time step.

In addition, $s_1 + s_2 + s_3$ in Algorithm 1 in general is the minimum distance needed to reduce the predicted velocity, v_{T}, to the desired velocity (condition 1) with zero final acceleration (condition 2) using $j \in \{j_{\min},\ 0,\ j_{\max}\}$ and $a_{\min} \leq a \leq a_{\max}$.

Specifically, if $v_{a_{\mathrm{T}} \to 0} > v_d$ (velocity after reducing absolute acceleration to zero is greater than desired), v_d would be reached by reducing acceleration from predicted a_{T} to $a_{\min}$ at jerk, $j_{\min}$ and then raising it back to 0 with $j_{\max}$. The required negative acceleration and duration applied to reduce $v_{a_{\mathrm{T}} \to 0}$ to v_d is calculated in lines 12-26. This deceleration sequence is shown in Fig. 2.

However, if $v_{a_{\mathrm{T}} \to 0} < v_d$, condition 1 is met but condition 2 is checked. If predicted acceleration $a_T > 0$, then line 31 determines minimum deceleration distance for condition 2.

Since the deceleration distance is calculated using $j \in \{j_{\min},\ 0,\ j_{\max}\}$, there always exists an input $j \in [j_{\min},\ j_{\max}]$ such that h_T is unchanged from the previous time step to the current, e.g., select $j = j_{\min}$ when $t_1 \neq 0$ to match the required deceleration sequence. Moreover, we established in Prop. IV.1 that h_{T} is strictly decreasing in $j \in [j_{\min},\ j_{\max}]$ for the current state, x. Hence, we can not only ensure that $h_{\mathrm{T}} \geq 0$ for subsequent steps, but also reduce it if a more positive $j \in [j_{\min},\ j_{\max}]$ is feasible to drive $d_0 \to 0$. Thus $h_{\mathrm{T}} \geq 0$ at the current time step indicates that conditions 1 and 2 are reachable as $d_0 \to 0$.

■

Remark 3. Selecting j to reduce h_{T} while maintaining $h_{\mathrm{T}} \geq 0$ automatically drives $d_0 \to 0$. h_{T} is simply the buffer distance before deceleration to desired velocity is required. When the buffer is at or close to zero, the encoded deceleration sequence is followed if $h_{\mathrm{T}} \approx 0$ is maintained.

Remark 4. It is possible for $v_0 < v_d$, $a_0 < 0$ or both when $h_{\mathrm{T}} = 0$ at $d_0 = 0$. This is due to insufficient distance to accelerate to v_d, increase acceleration from negative to 0 or both.

C. Soft Constraints on Acceleration and Velocity

It should be noted that Algorithm 1 does not impose explicit limits on acceleration and velocity. As the algorithm runs at discrete fixed time intervals, T, and the range of jerk inputs, j, searched is also discrete, it is not always possible for the system to reach the maximum acceleration exactly as required by the deceleration sequence. This can result in some situations where there is no feasible input.

Instead, we propose the use of soft constraints implemented in (17) as follows.

$$d_{\text{vel}} = \alpha_{\text{v}} \max\left(v_{a_{\mathrm{T}} \to 0} - v_{\max}, 0\right)\ , \tag{22}$$

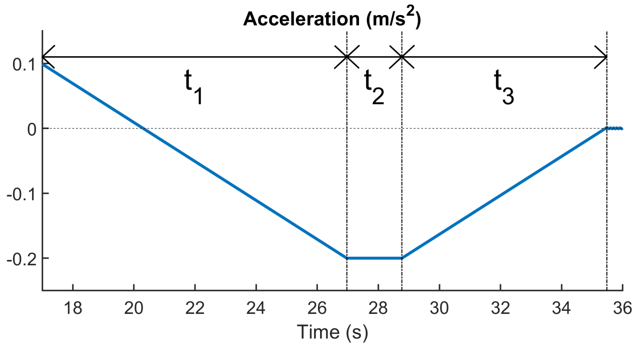

Fig. 2: Deceleration sequence divided into three sections: section 1 with time t_1 where acceleration is reduced from the current value to $a_{\min}$ at jerk $j_{\min}$, section 2 with time t_2 where acceleration is maintained at $a_{\min}$, and section 3 with time t_3 where acceleration is raised from $a_{\min}$ to zero at jerk $j_{\max}$. t_1, t_2 and t_3 are used to obtain respectively the deceleration distances s_1, s_2 and s_3 in Algorithm 1.

$$d_{\text{acc}} = \alpha_{\text{a}} \max\left(\max\left(a_T - a_{\max}, a_{\min} - a_T\right), 0\right)\ , \tag{23}$$

where

- a_{T} and $v_{a_{\mathrm{T}} \to 0}$ are respectively obtained from line 1 and lines 4-9 of Algorithm 1,
- $a_{\max}$, $a_{\min}$ and $v_{\max}$ are the motion constraints from (8)-(9) defined by the user, and
- α_{v} and α_{a} are non-negative constant parameters.

In (22), $v_{a_{\mathrm{T}} \to 0}$ is used to account for continuous acceleration. If $v_{a_{\mathrm{T}} \to 0} - v_{\mathrm{m}} > 0$, jerk values that increase acceleration are penalized. When $v_{a_{\mathrm{T}} \to 0} - v_{\mathrm{m}} \leq 0$, no penalty is needed as the system is automatically driven to increase acceleration again using Algorithm 1 and (17). Comparing v_{T} instead with v_{m} will excessively reduce velocity below the maximum due to constrained jerk. Consequently, negative velocity need not be constrained.

It is recommended that the user set $\alpha_{\text{a}} > \alpha_{\text{v}}$ preferably by an order of magnitude or more to prioritize acceleration constraint first over velocity.

D. Trajectory Generation Examples

In this section, we demonstrate the trajectory generation by forward simulating with a triple integrator. Specifically, jerk values are obtained for the current time step using (17) and Algorithm 1, and then used to calculate the position, velocity and acceleration at the next time step using kinematic equations. The time intervals between steps is constant.

In Fig. 3, we generate a simple trajectory at 400Hz with zero velocity and acceleration at start and end for a waypoint that is 40m away. This trajectory also exhibits the use of asymmetric limits on both acceleration and jerk. The acceleration and velocity are appropriately constrained by the terms (22) and (23) within the bounds denoted by black dotted lines. The parameters used are $\alpha_{\text{v}} = 1\text{e}5$ and $\alpha_{\text{a}} = 1\text{e}6$.

In Fig. 4, we generate the same trajectory as Fig. 3 but at 40Hz instead of 400Hz. Fig. 4, shows a slight overshoot

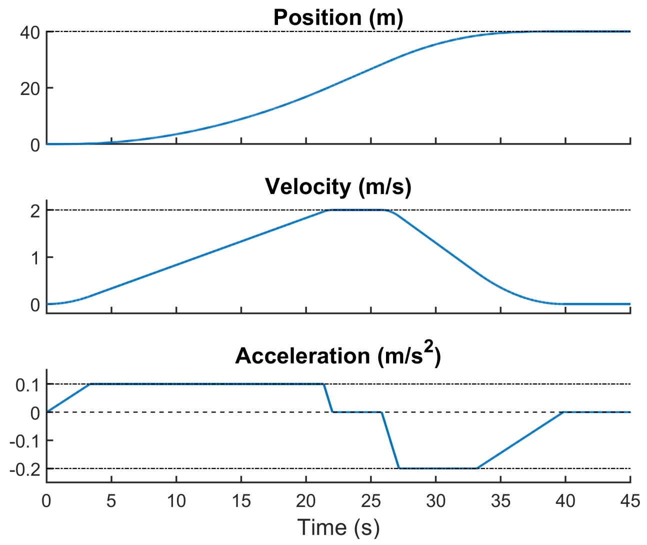

Fig. 3: Jerk-limited time-optimal trajectory generated using the proposed algorithm at 400Hz in blue solid lines. Target position is at $40m$ and final velocity is $0ms^{-1}$. The velocity limit here is $2ms^{-1}$. Asymmetric constraints on acceleration and jerk are used here: $0.1ms^{-2}$ maximum and $-0.2ms^{-2}$ minimum accelerations, and $0.03ms^{-3}$ maximum and $-0.15ms^{-3}$ minimum jerks.

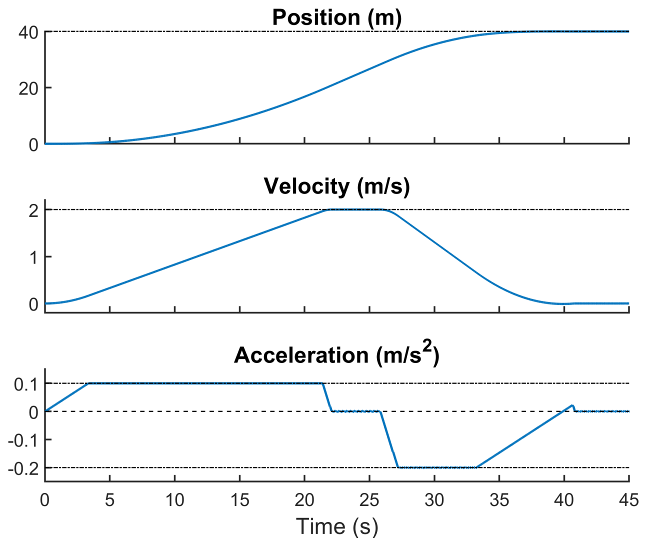

Fig. 4: Jerk-limited time-optimal trajectory generated using the proposed algorithm at 40Hz in blue solid lines. Target position is at $40m$ and final velocity is $0ms^{-1}$. The velocity limit here is $2ms^{-1}$. Asymmetric constraints on acceleration and jerk are used here: $0.1ms^{-2}$ maximum and $-0.2ms^{-2}$ minimum accelerations, and $0.03ms^{-3}$ maximum and $-0.15ms^{-3}$ minimum jerks. A slight overshoot in the acceleration is visible when position approaches $40m$.

in the acceleration at 40s. This acceleration overshoot is due to the system compensating for a slight undershoot of the position near 40m which is not visible in the figure. This overshoot also occurs because of the discrete values of jerk and time interval used to calculate the trajectory. This can be mitigated by using smaller intervals or by checking additional jerk values between j_{min} and j_{max}.

In Fig. 5, a trajectory is generated with an initial non-zero velocity of $-0.5ms^{-1}$ and final velocity of $1ms^{-1}$. This demonstrates how the algorithm sufficiently accounts for the deceleration distance to reduce it to $1ms^{-1}$ or less before reaching the $40m$ target position. The algorithm is also able to generate from any initial velocity. However, this is already evident as the next trajectory point is determined based on the current at each time step.

A trajectory with negative final velocity can also be generated. However, the system will overshoot the target position and return with the negative final velocity at the target.

Fig. 6 shows how (22) drives the velocity to within the imposed $2ms^{-1}$ limit without overshooting by ensuring sufficient time to raise the negative acceleration back to zero at the limit. The use of soft constraints drives the system to reduce the acceleration and velocity if they initially or currently exceed the imposed limits.

However, it should be noted that as this is a discrete implementation, the resulting trajectory position, velocity and acceleration values only approximates the exact values from the time-optimal trajectory encoded in Algorithm 1. The solution improves with smaller time intervals as highlighted with Fig. 3 and 4.

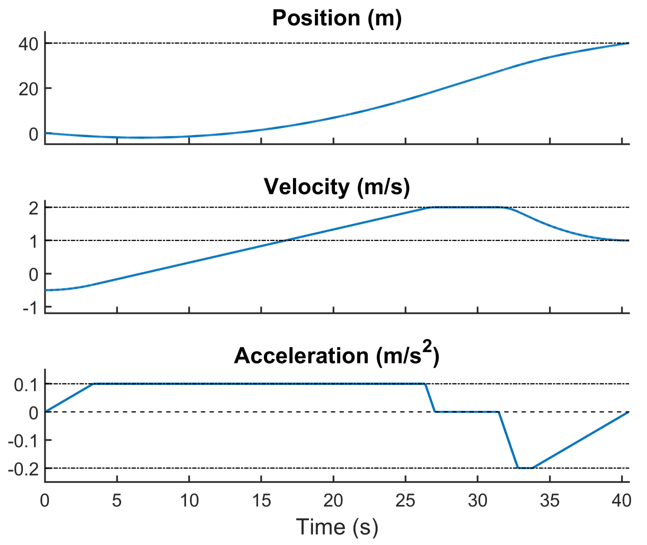

Fig. 5: Jerk-limited time-optimal trajectory generated using the proposed algorithm at 400Hz in blue solid lines. Initial velocity is $-0.5ms^{-1}$. Target position is at $40m$ and final velocity is $1ms^{-1}$. The velocity limit here is $2ms^{-1}$. Asymmetric constraints on acceleration and jerk are used here: $0.1ms^{-2}$ maximum and $-0.2ms^{-2}$ minimum accelerations, and $0.03ms^{-3}$ maximum and $-0.15ms^{-3}$ minimum jerks.

In addition, as observed with the overshoot in Fig. (22), the discrete implementation can lead to chattering about the target waypoint. This may be mitigated by implementing

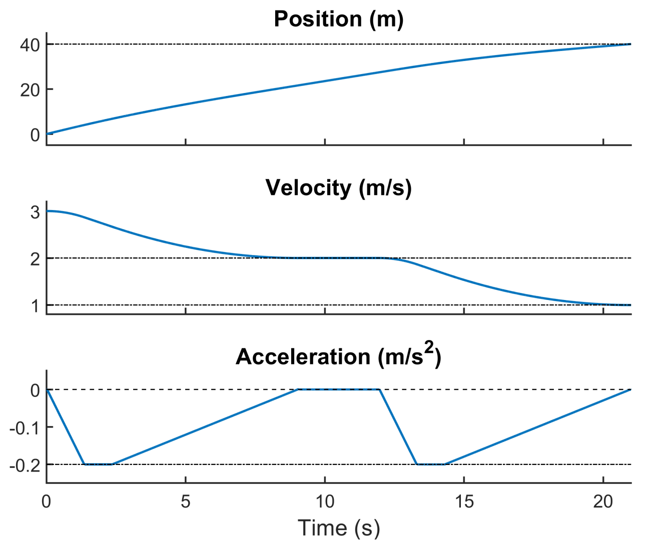

Fig. 6: Jerk-limited time-optimal trajectory generated using the proposed algorithm at 400Hz in blue solid lines. Initial velocity is $3ms^{-1}$. Target position is at $40m$ and final velocity is $1ms^{-1}$. The velocity limit here is $2ms^{-1}$. Asymmetric constraints on acceleration and jerk are used here: $0.1ms^{-2}$ maximum and $-0.2ms^{-2}$ minimum accelerations, and $0.03ms^{-3}$ maximum and $-0.15ms^{-3}$ minimum jerks.

thresholds on distance, velocity and acceleration about the waypoint. Within the threshold, the trajectory position, velocity and acceleration may be set to the desired final waypoint values.

E. Computational Requirement Comparison

For a single trajectory point, our proposed method only requires the solving of (17) in combination with Algorithm 1 for 3 jerk values, $j \in \{j_{\text{min}},\ 0,\ j_{\text{max}}\}$. Specifically, the mathematical computation is simple and fixed for each trajectory point.

The rest of the trajectory is generated by repeating this process with the forward simulation of the triple integrator model similar to Lai et al.'s work in [1]. Thus, the computational complexity is $O(n)$ where n is the number of time intervals desired or until the target waypoint is reached.

In comparison, Lai et al.'s work uses a bisection search with $O(\log(n))$ complexity to determine the start of the deceleration sequence. Typically, this bisection search would only be required at the start of the trajectory generation process to determine its parameters. Thus the performance is expected to be similar assuming the time interval length used for both methods is similar. However, if frequent readjustments and regeneration are required, then this bisection search is conducted each time. Our method is more efficient in comparison.

Additionally, unlike Beul and Behnke's work in [7] and [8], our method is invariant computationally under different scenarios. Specifically, our method is able to handle scenarios where no constant acceleration and deceleration is required in the trajectory without requiring additional computation.

V. Simulation Implementation

In addition to trajectory generation, our proposed method of (17) and Algorithm 1 may also be used as a one step ahead trajectory controller when combined with an Extended State Observer (ESO) to fly a multicopter from one way-point to the next. We use a 2nd order ESO based on [16] in our simulations. We test this controller with a control frequency of 40Hz in simulation of a hexacopter model subject to drag and wind turbulence disturbance.

(17) and Algorithm 1 in this implementation is used to obtain the required jerk for the current time step, which is then used to determine the acceleration input for the hexacopter's attitude control. The acceleration input before subtracting for disturbance is simply

$$a_{\text{d}}(t) = j_{\text{input}}(t)T \ , \tag{24}$$

where T is the control time interval.

The attitude setpoint for common flight controllers such as the one described in [13] is determined based on command accelerations without compensating for disturbances. Specifically, the attitude is obtained by solving for the rotation matrix, R, or its components in (1).

The initial states used in Algorithm 1 prior to obtaining $j_{\text{input}}(t)$ at each time step is

$$x(t) = [p(t),\ v(t),\ a_{\text{d}}(t-T)]^{\intercal} \ . \tag{25}$$

We use the desired acceleration at the previous time step, $a_{\text{d}}(t-T)$, to determine the desired acceleration for the current. This is to avoid the typically noisy acceleration signal of the multicopter's inertial measurement unit.

Finally, the disturbance estimated, $a_{\text{ESO}}(t)$, by the ESO is subtracted from $a_{\text{d}}(t)$ to obtain

$$a_{\text{cmd}}(t) = a_{\text{d}}(t) - a_{\text{ESO}}(t) \ . \tag{26}$$

The 2nd order ESO takes as input the previous position and the previous acceleration command, $a_{\text{cmd}}(t-T)$.

The acceleration command, $a_{\text{cmd}}(t)$, of this one step ahead trajectory controller implementation is obtained independently for two axes. The main axis runs parallel to the direction from the previous way-point to the next way-point. The second axes is orthogonal to the first and its direction is determined by the multicopter's position deviation from the line joining the way-points. This second axis controls the multicopter to move via a smooth trajectory back to the line if it is disturbed by wind. In this work, we demonstrate this controller on the XY-plane only. The total command is the sum of this two axes.

The simulation scenario is a square way-point path flight of a hexacopter subject to a 5m/s wind in the positive X-axis direction with some turbulence generated using the Dryden wind turbulence model [17]. The simulation is done in MATLAB Simulink based on [13]. The results are shown in Figures 7-9.

In Fig. 7, the X and Y axes positions are shown in blue with the desired way-point positions in dotted red lines.

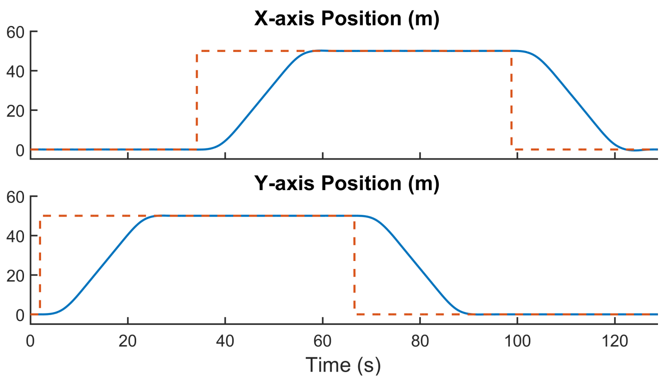

Fig. 7: X and Y positions of the hexacopter square way-point flight simulation. The blue line denotes the position and the red dotted line denotes the way-point target position.

Smooth trajectories are observed with relatively negligible overshoots even with the wind.

In Fig. 8, the X and Y velocities similarly show smooth trajectories with velocity successfully constrained to less than 3m/s, which is the limit set for this simulation.

Lastly, in Fig. 9, the values for a_d, a_{cmd} and the system's actual acceleration, are respectively plotted in blue, yellow and red. This important figure shows how the system's actual acceleration is able to closely track the desired acceleration, a_{d}, obtained via our proposed one step ahead trajectory generation method. This is achieved by subtracting the disturbance acceleration, a_{ESO}, estimated by the ESO. The resulting command a_{cmd} deviates from the desired acceleration a_d produced by the trajectory controller to compensate for the disturbance without requiring any alteration to the attitude control. This compensation works to match the actual acceleration as closely as possible to the desired one. The result is a smooth jerk-limited nearly time-optimal motion of the system without requiring partial or full generation of the entire trajectory prior to tracking it.

VI. Conclusion

In this paper, we proposed an efficient online jerk-limited barrier-based trajectory generation method that only requires a single pass. Specifically, the trajectory is directly obtained via forward simulation of the triple integrator without requiring a bisection search and trajectory parameters a priori. The Zeroing Control Barrier Function was applied in a discrete manner to encode the deceleration sequence. Only the minimum, maximum and zero jerk values need to be checked at each time interval. The trajectory generation capability was demonstrated with varying interval frequency, non-zero start and end velocities, and velocities greater than the velocity limit. In addition, we demonstrated a one step ahead trajectory controller in combination with an Extended State Observer in simulation of a hexacopter model subjected to drag and wind disturbances. For future work, we may consider the application of our proposed method to path planning through obstacles such as the safe flying corridor described in [1]. Application to three-axes flight may also be considered.

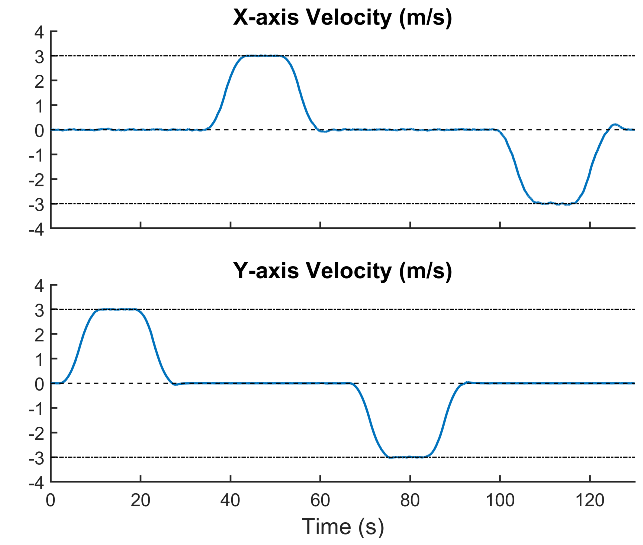

Fig. 8: X and Y velocities of the hexacopter square way-point flight simulation. The blue lines denote the velocities. The upper and lower black dotted lines denote respectively the imposed upper and lower limits on the velocity.

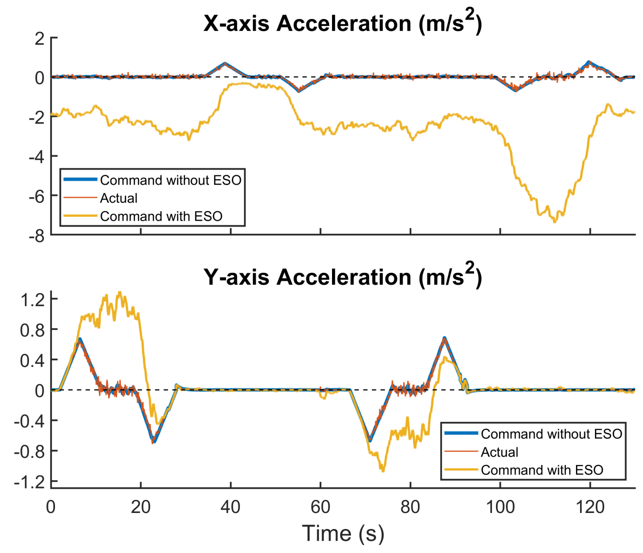

Fig. 9: X and Y accelerations of the hexacopter square way-point flight simulation. Command without ESO in blue is the desired acceleration, a_{d}, actual acceleration is shown in red, and the command with ESO in yellow is a_{cmd} from (26).

References

[1] S. Lai, M. Lan, Y. Li, and B. M. Chen, "Safe navigation of quadrotors with jerk limited trajectory," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 1, pp. 107–119, 2019.

[2] P. Byeong-Ju, L. Hong-Jun, O. Kwang-Kyo, , and M. Chae-Joo, "Jerk-limited time-optimal reference trajectory generation for robot actuators," *The International Journal of Fuzzy Logic and Intelligent Systems*, vol. 17, no. 4, pp. 264–271, 2017.

[3] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 2520–2525.

[4] S. Lai, K. Wang, H. Qin, J. Q. Cui, and B. M. Chen, "A robust online path planning approach in cluttered environments for micro rotorcraft drones," *Control Theory and Technology*, vol. 14, no. 1, pp. 83–96, 2016.

[5] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1934–1940.

[6] L. Xi, Z. Peng, and L. Jiao, "Trajectory generation for quadrotor while tracking a moving target in cluttered environment," in *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 6792–6797.

[7] M. Beul and S. Behnke, "Analytical time-optimal trajectory generation and control for multirotors," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2016, pp. 87–96.

[8] ——, "Fast full state trajectory generation for multirotors," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 408–416.

[9] P. Besset, R. Bearée, and O. Gibaru, "Fir filter-based online jerk-controlled trajectory generation," in *2016 IEEE International Conference on Industrial Technology (ICIT)*, 2016, pp. 84–89.

[10] G. Xia, X. Xia, B. Zhao, C. Sun, P. Wang, and Y. yang, "A method of online trajectory generation based on fir filter," in *2019 Chinese Automation Congress (CAC)*, 2019, pp. 919–924.

[11] A. H. Zaini and L. Xie, "Jerk-limited holonomic robot motion planning using barrier functions," in *2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 2019, pp. 410–415.

[12] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[13] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[14] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[15] R. Haschke, E. Weitnauer, and H. Ritter, "On-line planning of time-optimal, jerk-limited trajectories," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 3248–3253.

[16] M. Ran, J. Li, and L. Xie, "A new extended state observer for uncertain nonlinear systems," submitted.

[17] Dryden wind turbulence model. Mathworks. [Online]. Available: https://www.mathworks.com/help/aeroblks/drydenwindturbulencemodelcontinuous.html