Ensuring Quadrotor Safety Through Geofencing with Run Time Assurance

by
Arturo de la Barcena

A thesis submitted to the Faculty of the Department of Mechanical Engineering,
Cullen College of Engineering
in partial fulfillment of the requirements for the degree of

Bachelor of Science

in Mechanical Engineering

Chair of Committee: Dr. Marzia Cescon

Committee Member: Dr. Karolos Grigoriadis

Committee Member: Dr. Christina Chang

University of Houston
May 2023

ABSTRACT

Safety assurance in autonomous safety critical aerospace systems has become an increasingly relevant area of study as mission objectives, hardware, and even human lives become endangered when integrating complex and intelligent control system designs. With this rise of control and mission complexity for autonomous aerospace systems, a balance must be struck between mission objectives and system safety. A recent method of creating this balance has come to be known as online safety assurance techniques or Run Time Assurance (RTA), a control intervention method designed separately from a system's primary controller to assure safety in real time.

The research presented in this thesis analyzes two RTA intervention methods, a switching-based filter known as the Simplex method, and an optimization-based filter known as the ASIF method, in the control of simulated quadcopters to create an impassible safety cube or 'geofence' in each of the quadcopter's reference axes. The safety barriers created for the geofence are formally defined using the nascent topic of Control Barrier Functions (CBFs), independently defined for each flight direction (X, Y, Z), and implemented with Python, the primary language of the simulated environment. This thesis will conclude with a comparison of the two RTA methods and the effectiveness of their implementation on different quadcopter mission objectives.

ACKNOWLEDGEMENTS

I would like to begin by thanking my committee members, Dr. Marzia Cescon, Dr. Karolos Grigoriadis, Dr. Zheng Chen, and Dr. Christina Chang for guiding me through this work and providing all the resources and help necessary to succeed as a student. Thank you all for working and meeting with me every week for the last year and thank you Dr. Cescon for having so much faith in me and my abilities early in my undergraduate career.

I would also like to thank Dr. Kerianne Hobbs and the AFRL Minority Leaders – Research Collaboration Program for providing me with the opportunity to conduct research in such an exciting field and for supporting me through your experience and insight. Thank you Dr. Hobbs and I look forward to our continued work together.

Finally, I would like to thank all of the other graduate and undergraduate students I have worked with in Dr. Cescon's lab for their time, advice, and for providing me their wisdom. Specifically, I would like to thank Mariam Ismail for serving as a student mentor and helping me with all of my questions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. Introduction

The autonomy of aerial vehicles through has become an integral part in carrying out increasingly complex missions from disaster relief through drone surveillance of affected areas to delivery of goods for humanitarian aid [1]. Though the driving action that controls these vehicles may be ideally sound, the stochastic nature of different operating environments and of the vehicle itself necessitates a method of assuring operational safety to prevent damage or complete loss. Designing control systems primarily concerned with safety can, however, lead to overly cautious designs which directly affect the overall performance of the vehicle's task. A method of balancing the system's performance and safety is therefore necessary.

In recent years, a variety of methods collectively known as online safety assurance techniques or Run Time Assurance (RTA), have been proposed to ensure system safety while performing mission objectives with a base controller. These techniques can take multiple forms such as switching to a secondary, predetermined controller when system safety is breached and filtering the original input signal to an optimal input signal to ensure safety at all times. Recently, Control Barrier Function (CBF) based methods have been introduced to ensure the system's safety. Other methods of defining safety have also been explored, notably implicit safe sets which create closed loop trajectories under a backup controller over a given time horizon. This is useful in more complex systems in which a clear safety function may not be possible.

This thesis will explore the use of RTA for a micro-quadcopter platform, namely, Bitcrazie's Crazieflie 2.1 [13], to create a geofence, i.e., a 'safety cube' which the drone

cannot pass through, in all cardinal directions (relative x, y, and z). The outline of this thesis is as follows. Part 2 reviews literature related to the concepts of RTA and CBFs. Part 3 introduces the background information relating to these topics and an overview of the micro-quadcopter's control scheme. Part 4 presents the Python implementation used to test and verify safety assurance. Part 5 explores the RTA implementation on a Crazieflie 2.x in a simulated gym environment by subsystem, ordered in one degree of freedom and two degrees of freedom. The final part draws conclusions from the RTA implementations presented and will discuss potential expansion and future work.

2. Literature Review

This section will explore the currently available literature relating to RTA systems and CBFs. Though this thesis primarily focuses on the aerospace application of these concepts, the literature surveyed in this section fits within the context of safety assurance techniques for safety critical systems.

2.1 Run Time Assurance (RTA)

Known as online safety assurance techniques in some literature, RTA has been used in a variety of applications including control of nonlinear motors [2], autonomous satellites [3], safety of military aircraft [4], and in smaller examples such as in a reaction wheel pendulum [5] and has been implemented in a variety of controller types including reinforcement learning [6]. The need for safety assurance rose out of safety critical robotic operations where damage must be prevented while under normal operation. The control of robotic systems can only guarantee the minimization of error of actions from a given setpoint, not necessarily its safety through the avoidance of other robots or obstacles in its set path.

Collision avoidance is a major point of discussion within the field of aerospace control systems, as well as other robotics fields [7], as several types of air and surface obstacles can pose dangers to high-cost autonomous systems and even human lives. A current example of RTA being used in the aerospace field for collision avoidance is the automatic ground collision avoidance system (Auto-GCAS) which was designed as a backup controller in the event of a pilot losing consciousness or focus while the aircraft is

near the ground [8]. This system has been implemented into the F-16 and F-35 fighter systems and has already saved dozens of lives [4].

This style of collision avoidance has also been implemented in smaller-scale aerospace systems, most notably in other single quadrotor examples where tight space maneuvers through urban environments necessitate safety assurance [9]. A lead and follower safety bubble between two quadcopters in predetermined paths has also been used to assure collision avoidance in both systems [8].

2.2 Control Barrier Functions (CBFs)

The recent interest in autonomous systems has brought safety to the forefront of control system design. In this context, safety can be trained as enforcing the invariance of a set, i.e., the system is not leaving the safety set [11]. Safety conditions which lie at the root of RTA can sometimes give rise to conflicting control objectives and safety assurance. In order to unify a system's objectives with safety assurance, a type of safety condition can be constructed commonly referred to in most literature as a Control Barrier Function (CBF). The functions provide a set of inequality constraints in the control input which can be unified with control functions in a quadratic program (QP) to allow for control objectives and admissible, safe states [10].

These CBFs have been used in optimized safety assurance in the same broad spectrum of safety critical systems as RTA more generally. The construction of CBFs was introduced and used to build a safe system for a reaction wheel pendulum with an LQR controller [5] later explored in further detail in this paper. CBFs were similarly used for other nonlinear models such as bipedal walking robots [10].

CBFs have been used in other literature as a tool in the construction of a type of RTA known as an Active Set Invariance Filter (ASIF) or an optimization filter. In practice, the use of CBFs in this type of RTA filter is the same with a quadratic program being used to optimize the unsafe input control law [11].

3. Background

3.1 Simulated Pybullet Gym Environment

The simulated environment used in this thesis is an open-source Gym-style environment based on Google's Bullet Physics engine [16] using its Python binding, Pybullet, and is one of the first general purpose multi-agent Gym environment for quadcopters [13]. Though it was primarily constructed as a way of studying Reinforcement Learning control, this environment was chosen to simulate the drone described in the previous section because it hosts the ability to construct other types of controllers for the Bitcraze Crazyflie 2.x, its default quadcopter model.

The physics is based on a physical system identification performed on the Crazyflie 2.x [13] and independent study done on different physical phenomena including drag, ground effect, downwash, and various other aerodynamic effects [14].

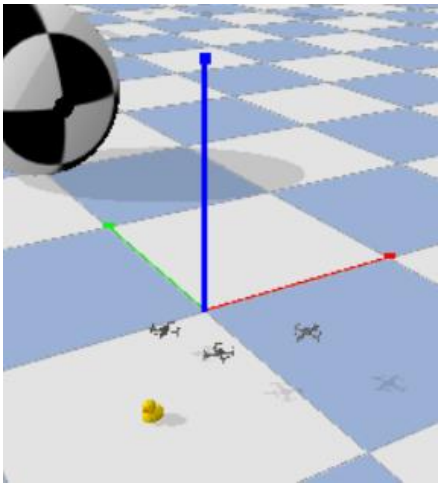

Figure 1: Pybullet Drones Running with 3 Drones

3.2 Dynamic Control System and Quadcopter Representation

A state space representation is a set of equations which explains how a dynamical system evolves over time. This is a convenient tool for a system with multiple degrees of

freedom and dependencies on multiple states. The general form of the state space equation is as follows,

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \boldsymbol{C}(t)\boldsymbol{x}(t) + \boldsymbol{D}(t)\boldsymbol{u}(t) \end{cases} \quad \textbf{(1)}$$

where $x(t) \in \mathbb{R}^n$ denotes the state, $u(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}^n$ are the input and output, respectively, and $A(t), B(t), C(t), D(t)$ are system matrices of appropriate dimensions, while t is the continuous-time index. This representation can be simplified for systems where parameters A, B, C, and D are time invariant and have several non-linearities which can be simplified with a linearization process, resulting in,

$$\begin{cases} \boldsymbol{\Delta\dot{x}}(t) = \boldsymbol{A\Delta x}(t) + \boldsymbol{B\Delta u}(t) \\ \boldsymbol{\Delta y}(t) = \boldsymbol{C\Delta x}(t) + \boldsymbol{D\Delta u}(t) \end{cases} \quad \textbf{(2)}$$

where Δ signifies a vector which has gone through a linearization process. In the case of a nanoquadcopter, a state space representation can be formulated using linearized equations of forces and moments along its six degrees of freedom [12] which results in a 12-element long state vector and a 12x12 state matrix. For the purposes of analysis and CBF construction, this is too large and complex. However, since quadcopters are underactuated platforms, it can be shown that the vertical, lateral, longitudinal, and directional forces act independently from one another and, therefore, the quadcopter dynamics are decoupled into individual subsystems [12] 2ith reference to Fig. 2 representing the reference coordinate frame,

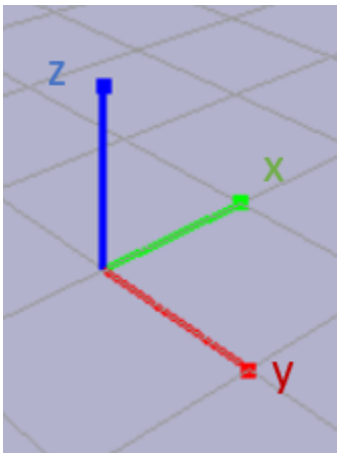

Figure 2: Reference Frame for Drones

in this thesis we will consider the quadcopter motion along the z- and y-axis, the vertical and longitudinal subsystems, respectively.

3.2.1 Vertical Subsystem

The vertical subsystem represents the drone's motion along the z-axis perpendicular to the ground and can be described by,

$$\begin{bmatrix}\Delta\dot{w}\\ \Delta\dot{z}\end{bmatrix} = \begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix}\begin{bmatrix}\Delta w\\ \Delta z\end{bmatrix} + \begin{bmatrix}1/m\\ 0\end{bmatrix}\Delta F_z \quad \textbf{(3)}$$

where w is the velocity in the z direction (vertical relative to the ground), m is the mass of the drone, and ΔFz is the linearized vector form of the thrust in the z direction. This thrust can become control law u_1 for this subsystem, resulting in,

$$\begin{bmatrix}\Delta\dot{w}\\ \Delta\dot{z}\end{bmatrix} = \begin{bmatrix}0 & 0\\ 1 & 0\end{bmatrix}\begin{bmatrix}\Delta w\\ \Delta z\end{bmatrix} + \begin{bmatrix}1/m\\ 0\end{bmatrix}u_1. \quad \textbf{(4)}$$

3.2.2 Longitudinal Subsystem

The longitudinal subsystem describes the quadcopter's East-West motion and it is described by,

$$\begin{bmatrix} \Delta\dot{p} \\ \Delta\dot{\phi} \\ \Delta\dot{v} \\ \Delta\dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta\phi \\ \Delta v \\ \Delta y \end{bmatrix} + \begin{bmatrix} 1/I_{xx} \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta M_x \quad \textbf{(5)}$$

where p is the roll velocity, φ is the roll angle, v is the linear velocity in the y direction, Ixx is the inertia relative to the xx direction, and ΔMx is the moment relative to the x-axis. This moment can, similarly, become a control law, u_2 for this subsystem, resulting in,

$$\begin{bmatrix} \Delta\dot{p} \\ \Delta\dot{\phi} \\ \Delta\dot{v} \\ \Delta\dot{y} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta\phi \\ \Delta v \\ \Delta y \end{bmatrix} + \begin{bmatrix} 1/I_{xx} \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathrm{u}_2. \quad \textbf{(6)}$$

It should be noted that the lateral subsystem, which is responsible for controlling motion in the quadcopter's x-axis body frame, is similar in form to that of the longitudinal subsystem,

$$\begin{bmatrix} \Delta\dot{q} \\ \Delta\dot{\theta} \\ \Delta\dot{u} \\ \Delta\dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta q \\ \Delta\theta \\ \Delta u \\ \Delta x \end{bmatrix} + \begin{bmatrix} 1/I_{yy} \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta M_y. \quad \textbf{(7)}$$

Because of this similarity, the controller and safety assurance measures developed for one subsystem can be equivalently applied to the other. With this understanding, this thesis and particularly Part 5 will therefore primarily focus on the longitudinal subsystem.

3.3 System Control and Architecture

The two primary controllers presented in this work are simple PD and PID controllers constructed for two specific control objectives: (a) a drone lifting off from the ground to a setpoint position; and (b) a drone moving in its relative y axis body frame. Because of the simplicity of the one degree of freedom vertical lift off example, an

integral component is not necessary in its controller. Mathematically, the control signal can be expressed as,

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^{\tau} e(\tau)d(\tau) + T_d\frac{de(t)}{dt}\right). \quad \textbf{(8)}$$

where $e(t)$ is the tracking error, K_p is the proportional coefficient, T_i is the integral time constant (reset time), and T_d is the derivative time constant (rate time). The control scheme for these two systems is different in how they are constructed and are described in the following sections.

3.3.1 One Dimension Lift Off

A system in which a drone starts from the ground then moves uniaxially to a certain position above the ground is defined by the vertical subsystem in Eq. (3). This type of system is also known as a double integrator system where z represents the drone's vertical position relative to the ground, which can equally be represented by,

$$\dot{z} = f(z) + g(z)u \quad \textbf{(9)}$$

where $z \in \mathbb{R}$ is bounded along the axis perpendicular to the ground and u is defined by a given control law. The base PD controller used by the simulated gym environment for a uniaxial drone take-off is described by the gain values which were predetermined for the base Crazieflie 2.1 model. Table 1 shows the controller parameters.

Table 1: PD Parameters

Parameters	**Value**
K_v	0.98
K_p	0.49

In order to calculate the input control action, the following equation describing the control law was used,

$$u(t) = a_{target} + K_v * \dot{e}_z(t) + K_p * e_z(t) \quad \textbf{(10)}$$

where a_{target} is the target acceleration as a function of the target position and of the simulated frequency. It should be noted that, for a uniaxial take-off, each propeller must have the same turn rate. From this control law $u(t)$, the RPM of each propeller can be calculated,

$$RPM = \sqrt{\frac{u * m_{drone} + g * m_{drone}}{4 * K_f}} \quad \textbf{(11)}$$

where the mass of the drone and RPM to force coefficient K_f are constants provided by the Crazieflie 2.x object already located within the gym [13].

3.3.2 Lateral Motion

A PID controller is used to define the control actions in the y- and z-directions. The integral component was inserted to this controller so that tracking is introduced as the system representation for this motion is not as accurate, necessitating the need for an error integrator. The equation for the control law, therefore, takes a different form,

$$u(t) = a_{target} + K_v * \dot{e}_y(t) + K_p * e_y(t) + K_i \int_0^{\tau} e_y(t) d\tau. \quad \textbf{(12)}$$

The architecture which defines the motion in the y direction is cascaded to where the output of the rate controller is used as the input for the attitude controller.

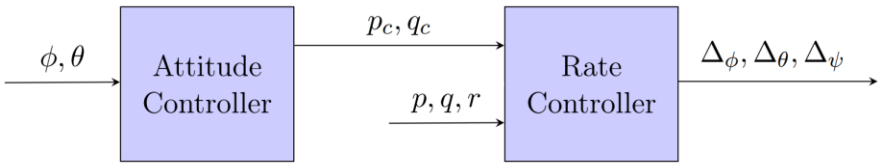

Figure 3: Cascaded Architecture

From this, the quadcopter's desired roll angle can be calculated from the control action in the y direction,

$$\varphi_{des}(t) = -\frac{\ddot{y}(t)}{g}. \tag{13}$$

The angular velocity in the roll direction can be found using this desired roll position and the simulations physics step time. Control action in this angular direction, $\ddot{\varphi}$, can be found using the PID controller described above. Given these PID inputs in the y, z, and φ directions, the thrust control law as described by the vertical state space subsystem can be derived using the linearized model described previously,

$$u_1(t) = m * \left(g * \ddot{z}(t)\right). \tag{14}$$

Similarly, the moment control law can be derived from the longitudinal state space subsystem (5) and be written as,

$$u_2(t) = I_{xx} * \ddot{\varphi}(t). \tag{15}$$

These two control laws can be combined into an array which takes the RPM to force coefficient into account,

$$u(t) = \begin{bmatrix} \frac{u_1(t)}{k_f} \\ \frac{u_2(t)}{k_f * L_{arm}} \\ 0 \end{bmatrix} \tag{16}$$

Where k_f is the RPM to force coefficient and L_{arm} is the quadcopter armlength.

Multiplying this by a (3,3) array, also known as a mixer,

$$u2RPM = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 1 & -1 \\ 2 & -1 & -1 \end{bmatrix} \quad \textbf{(17)}$$

$u(t)$ is converted into a three element RPM array which serves to actuate the drone's four rotors. This array is a tunable parameter and can be changed by the controller designer. Note that two diagonally opposing propellers must have the same turn rate in order to smoothly travel in the y-z plane.

3.4 Run Time Assurance (RTA)

In practice, RTA acts as a third block to the common controller and plant architecture with it being placed between the latter two (Fig. 1) [4] where it functions as a command signal filter to ensure that the signal does not compromise the safety of the system.

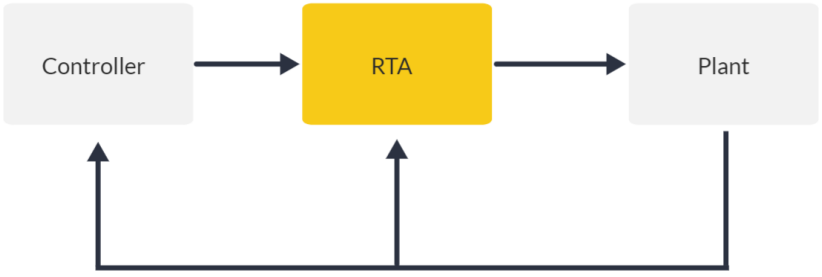

Figure 4: Run Time Assurance Block Diagram

With this type of architecture, the RTA blocks can act independently from the objectives of the primary controller to ensure safety, giving the control system designer the liberty to enable and disable safety parameters. In order to provide safety to a system, safety must first be formally defined.

3.4.1 Safety Equation and Control Barrier Function

The most effective way of defining safety in a dynamical system is by constructing two sets, an allowable set C_A where a state is said to be safe if it remains within the set at all times and a safe set C_s which is defined when an initial state $\boldsymbol{x}(t_0)$ of a dynamical system is within a forward invariant subset of C_A, $C_s \subseteq C_A$. The allowable set must be constrained by a set of safety inequality constraints which can be developed from the system dynamics or, alternatively, defined implicitly using predetermined trajectories under an alternative control law. A set of M inequality constraints φ_i defining the system's safety can be developed such that,

$$\varphi_i(\boldsymbol{x}): \boldsymbol{\mathcal{X}} \to \mathbb{R}, \forall i \in \{1, \dots, M\} \quad \textbf{(18)}$$

which can be used for a system with states within a set of states χ so that the allowable set is defined as,

$$C_A = \{x \in \mathcal{X} \mid \varphi_i(x) \geq 0, \forall i \in \{1, \dots, M\}\}. \quad \textbf{(19)}$$

The inequality constraint can be constructed explicitly from CBFs. The concept of a CBF was introduced in [15] and is most generally defined as a function $h \in C^1(\chi)$ for a system and set of unsafe states which satisfy,

$$x \in \mathcal{X}_u \Longrightarrow h(x) > 0, \quad \textbf{(20)}$$

$$L_g h(x) = 0 \Rightarrow L_f h(x) < 0,$$

$$\{x \in \mathcal{X} \mid h(x) \leq 0\} \neq 0$$

where $L_g$and L_f are Lie derivatives of the system. The function $h(x)$ is called a CBF if there exists an extended κ-class function $\alpha(h(x))$ such that,

$$\sup\left[L_f h(x) + L_g h(x)u + \alpha\big(h(x)\big)\right] \geq 0 \qquad \forall x \in \mathbb{R}^n. \tag{21}$$

This definition arises from the condition that $h(x)$ should be greater than 0 at all times including the initial time which gives us a condition to enforce,

$$h(x) = 0 \Rightarrow \dot{h}(x) \geq 0. \tag{22}$$

The time derivative of $h(x)$ given by the above condition can be calculated by,

$$\dot{h}(x) = \frac{dh}{dt} = \frac{dh}{dx} * \frac{dx}{dt} = \frac{dh}{dx}\dot{x} \tag{23}$$

where $\dot{x}$ represents the state equations as given in Eq. (1). Therefore, the time derivative of h(x) can be equally written as,

$$\dot{h}(x) = \frac{dh}{dx}(Ax + Bu) = \frac{dh}{dx}(f(x) + g(x)u) \tag{23.1}$$

where $f(x)$ and $g(x)$ are arbitrary functions of x. Finally, $\dot{h}(x)$ can be calculated for the general case,

$$\dot{h}(x) = \frac{dh}{dx}(f(x) + g(x)u) = L_f h(x) + L_g h(x)u. \tag{23.2}$$

In order to create a smooth controller, this condition can be slightly modified so that it becomes mathematically impossible for $h < 0$ when $\dot{h} \geq 0$ so that the condition to enforce becomes,

$$\dot{h}(x) \geq -\alpha(h(x)) \tag{24}$$

where the continuously increasing nature of the κ-class function α maintains the condition (22). A new safety function $h(x)$ can, therefore, be constructed which then takes the form,

$$\{h(x) = \dot{h}(x) + \alpha(h(x))\} \geq 0 \qquad \forall x \in \mathbb{R}^n \tag{25}$$

or, more generally,

$$\{h(x) = L_f h(x) + L_g h(x) u + \alpha(h(x))\} \geq 0 \qquad \forall x \in \mathbb{R}^n. \tag{25.1}$$

Alternatively, to create the inequality constraint from the dynamics of the system and a CBF, the constraint can be defined online through the generation of backup trajectories. This method is known as an implicit approach to defining safety [8] but will not be further explored in this thesis.

3.4.2 Simplex and ASIF RTA

Now that safety has been defined, it can be used in one of two primary RTA approaches. The first approach, known as a switching filter in some literature, is the simplex approach. This RTA system uses the defined safety to switch between the primary, unsafe control law u and a predetermined safe control law u_{safe}. A simple block diagram is shown below in Fig. 5.

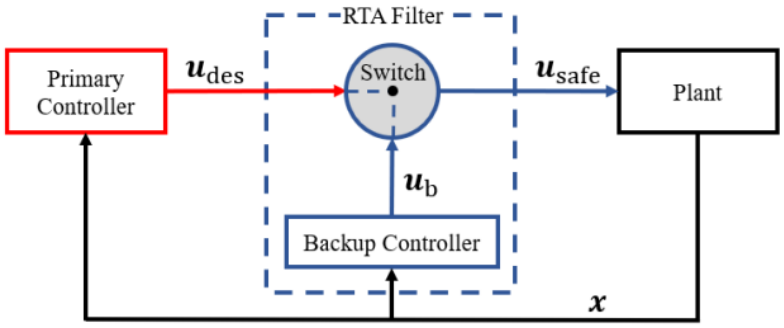

Figure 5: Simplex RTA Block Diagram [8]

Let us illustrate this concept with an example. Consider a double integrator car which has a lead foot controller, i.e., a binary controller which switches between positive

and negative acceleration or thrust and is moving in the positive direction from some negative reference position. There is a wall at $x = 0$ and safety must be assured so that the car does not crash into the wall. The state space equation which describes this system can be written as,

$$\dot{\boldsymbol{x}} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u \tag{26}$$

where state $x_1$represents the car's position and state x_2 represents its velocity. The state and control vector can be represented as $\boldsymbol{x} = [x_1, x_2]^T \in \boldsymbol{\mathcal{X}} = \mathbb{R}^2$ and $\boldsymbol{u} = T \in \boldsymbol{\mathcal{U}} = [-1, 1]$, respectively. The allowable set for this simple system can then be written as,

$$C_A = \{\boldsymbol{x} \in \mathbb{R}^2 \mid x_1 \geq 0\}. \tag{27}$$

Finally, this system's control invariant safety set can be made from a safety equation defined by the system's constant-acceleration kinematics. The unique safety constraint is given by,

$$h(x) = 2x_1 - x_2^2 \tag{27.1}$$

where it is assumed $x_2 > 0$ (meaning it is moving towards the wall). From this, the final safety set can be written as,

$$C_S = \{\boldsymbol{x} \in \mathbb{R}^2 | 2x_1 - x_2^2 \geq 0\}. \tag{27.2}$$

This safety set creates a 'safety envelope' which defines the maximum safe velocity the car can be in at a certain position. A visualization of this envelope is shown in the Fig. 6,

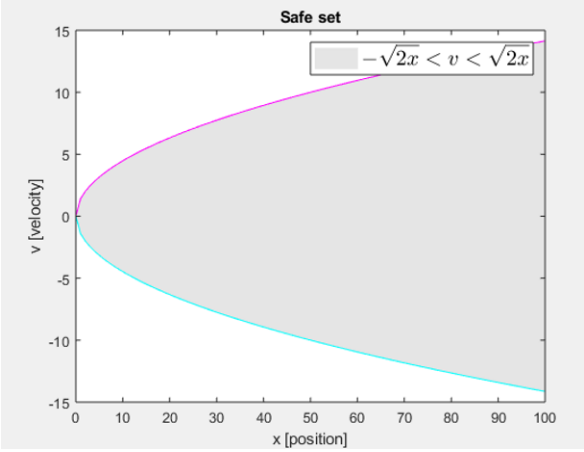

Figure 6: Safety Envelope for Double Integrator Car

If this envelope is violated, then the opposing lead foot backup controller can take over from the primary, similar to a bang-bang control switch, making the car decelerate.

The second RTA approach is an optimization approach which, in some literature, is known as an Active Set Invariance Filter (ASIF). Instead of switching to a backup controller which may deviate greatly from the system's task or setpoint, the optimization approach optimizes the difference between the desired and safe signal through a quadratic program (QP). The constraint for this optimization is defined by the control barrier function as previously discussed,

$$\boldsymbol{u}_{safe}(\boldsymbol{x}) = argmin\ \|u_{des} - u\|^2\ s.t.\ CBF_i(x,u) \geq 0, \forall i\ \in \{1, \dots, M\}. \quad \textbf{(28)}$$

Taking the same double car integrator example as before, a control barrier function can be constructed to be used in an ASIF RTA filter. Using the criteria from equations (18), the constraint for safety can be written in the form,

$$L_f h(x) + L_g h(x) u + \alpha\big(h(x)\big) \geq 0. \quad \textbf{(29)}$$

The first time derivative of the safety constraint h is given by,

$$\dot{h}(x) = -2x_2 - 2x_2 u = -2x_2(1+u) \quad \textbf{(30.1)}$$

which results in the final control barrier function. The CBF can be written as a safety constraint,

$$h(x) = -2x_2(1 + u) + \alpha(-2x_1 - x_2^2). \quad \textbf{(30.2)}$$

4. Methodology

This thesis' primary analysis was conducted using a combination of the Pybullet Gym Environment for all the base simulations and the CVXPY Python package for its convex quadratic programming/optimization solver. In order to verify the validity of the CBF construction using the CVXPY Python package, the method was used in a previously conducted experiment [5] known as an angular pendulum example. In this example, an LQR controller, a type of optimization controller, with known weight matrices is used to control a reaction wheel located at the end of a rotating pendulum where the angle and angular velocity of the pendulum are given by $a, \dot{a}$, respectively and the angular velocity of the reaction wheel is given by $\dot{\theta}$.

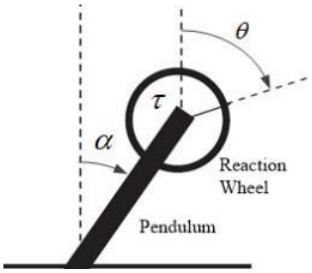

Figure 7: Reaction Wheel Pendulum Drawing [5]

The state space equation for the system was given as,

$$\begin{bmatrix} \dot{\alpha} \\ \ddot{\alpha} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 66 & 0 & 1 \\ -66 & 0 & -6 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\alpha} \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ -70 \\ 385 \end{bmatrix} u. \quad \textbf{(31)}$$

The safety for this system was defined at $\alpha_{max} = 0.087$ and a simple CBF was constructed using this maximum angle,

$$h(x) = (\alpha_{max}^2 - \alpha^2 - c_2\dot{\alpha}^2). \quad \textbf{(32)}$$

Though the safety parameter that is being tracked is α, the inclusion of $\dot{\alpha}$ is necessary since the control input u is solely dependent on it and h(x) must impact u in some way. It should be noted, however, that the effect of the angular velocity in the CBF can be reduced by making the parameter c_2 comparatively small. The input to the system was dependent on a reference to the pendulum angular displacement and was tracked by the controller. The graph from the python implementation without the CBF is presented in Fig. 8.

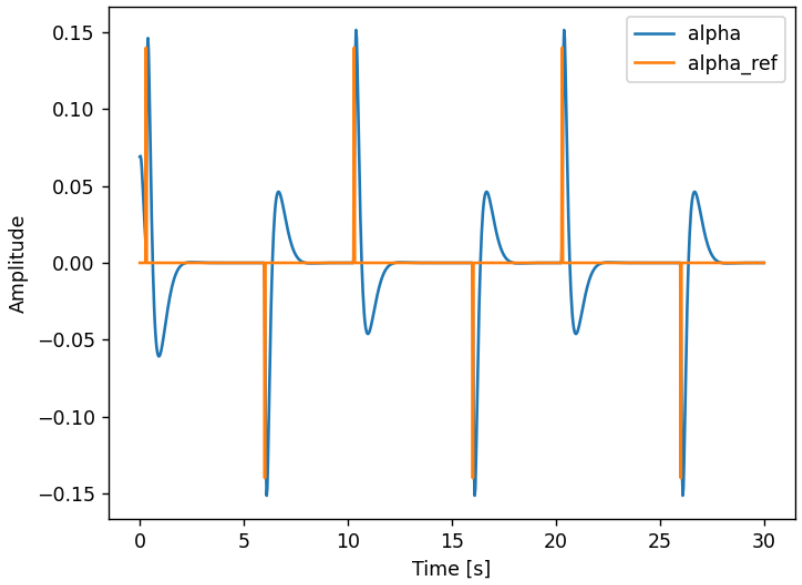

Figure 8: Reaction Wheel Pendulum Example without RTA

As can be seen in the figure, the α state follows the given reference signal with the given LQR controller but goes above the safe amplitude defined at $\alpha_{max} = 0.087$, therefore necessitating the need for a CBF to be implemented into this system. The original control law u can be optimized using a QP to solve a minimization problem defined by,

$$u_{safe} = \min\left(u_{ref} - u_{safe}\right)^2 \mid \frac{dh}{dx}[Ax + Bu] \geq -\beta h(x) \quad \textbf{(33)}$$

where β is a tunable parameter. Fig. 9 shows the results of this simulation.

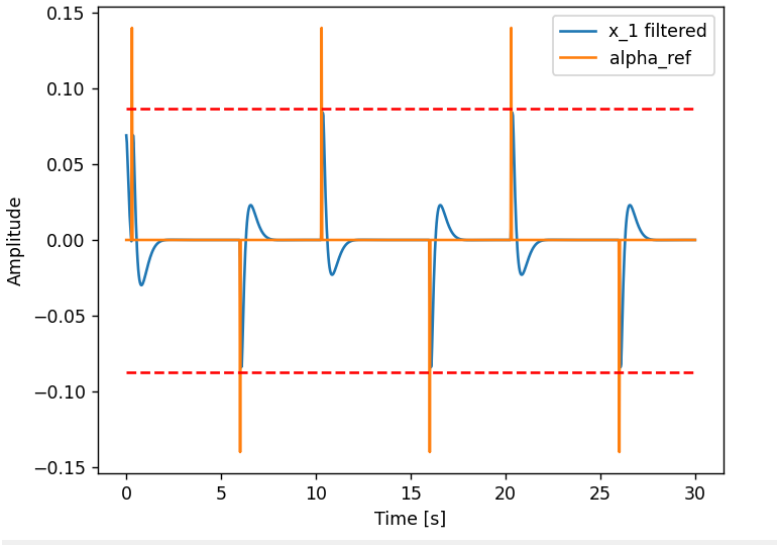

Figure 9: Reaction Wheel Pendulum Example with RTA

Where the red dashed lines represent the safety bounds of this system. As can be seen from this figure, safety is assured at all times without necessarily having to include a backup controller.

5. Results

5.1 Simulating Safety for 1-D Uniaxial Take-off

This example was studied within the simulated environment with graphs showing the relevant position and velocity presented to demonstrate the effectiveness of the safety assurance used. The difference between the Simplex and ASIF approaches will be analyzed from the data presented in these graphs.

5.1.1 Simplex Approach

Using the PD controller explained in the background section, a setpoint of $1\ [m]$ above the ground was set. For a simplex approach, a function which defines the safe states of the system can be defined by setting a 'ceiling' value which, to test the effectiveness of the simplex RTA, should be set below the setpoint. A backup control law must also be defined. For this example, a ceiling at $0.5\ [m]$ above the ground and a constant backup control law of $u = -1$ (where the control law is defined as the thrust or action force) was set. Since the backup control law provides a constant acceleration, the safety constraint can be defined using a simple kinematic equation of constant acceleration,

$$w_f^2 - w_0^2 = 2a_z(z_f - z_0) \tag{34}$$

which can be reordered according to the ceiling value to arrive at,

$$z_f = z_0 - \frac{1}{2}w_0^2 + \frac{c}{2}, \qquad z_f \geq 0, \qquad h(z) = z_f \tag{35}$$

where c is a parameter which defines the ceiling value and $h(z)$ is the safety function. Defining a safety function in this fashion ensures that the drone does not exceed the set ceiling value and does not move with a high enough velocity at any given position to

crash into the ceiling even with the backup controller active, effectively similar to the double integrator car example. The control invariant safety constraint can therefore be written as,

$$C_S = \left\{ z \in \mathbb{R}^2 \mid z_0 - \frac{1}{2} w_0^2 + \frac{c}{2} \geq 0 \right\}. \tag{36}$$

With a defined safety constraint, a Simplex RTA filter can be implemented by an if statement included in the base controller which will switch to the backup control law if $h(x) < 0$ is violated. The following figures show the simulation running without RTA and with Simplex RTA with $h(x)$ as previously defined.

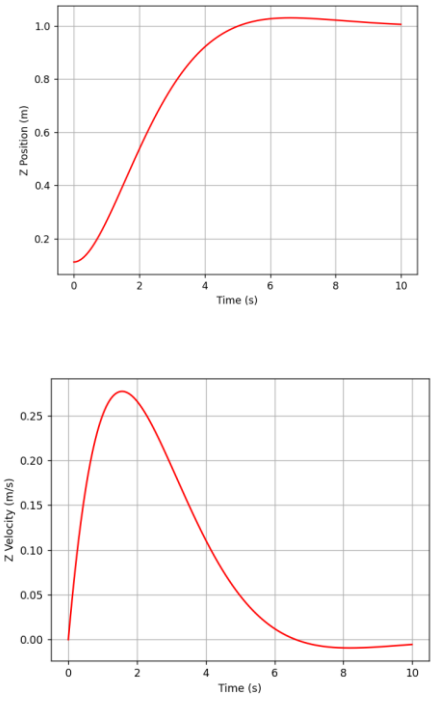

Figure 10: 1-D Uniaxial Takeoff without RTA

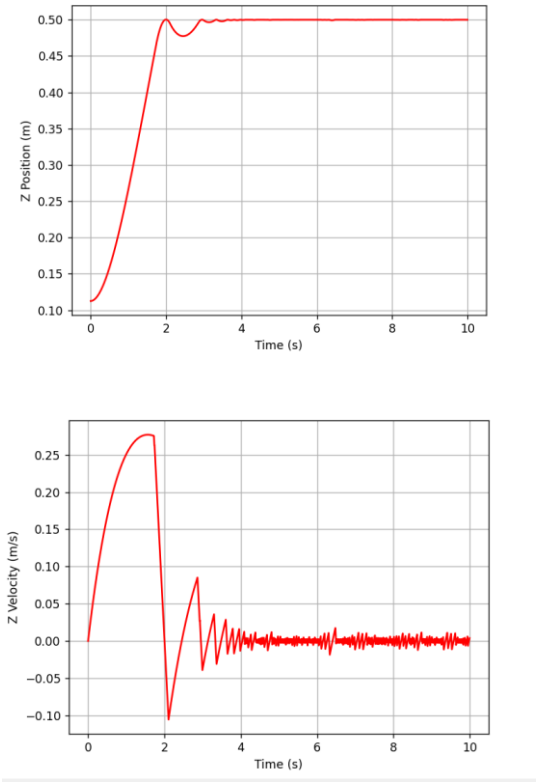

Figure 11: 1-D Uniaxial Takeoff with Simplex RTA

As can be seen from Fig. 11, the Simplex RTA implementation successfully forces the drone to settle just below $z = 0.5\ [m]$, but with some steady state error in the position and chattering in the velocity. This is a highly intrusive type of behavior which can be reduced using optimization methods.

5.1.2 ASIF Approach

From the definition of a CBF as previously introduced, an optimized filter can be introduced to the uniaxial model with a strengthening function $\alpha(h(x))$ being a simple linear equation for this example (Note that this function can be tuned by the control designer). Using state-space representation variables, the new $h(x)$ for a ceiling set at $c = 0.5\ [m]$ can be written as,

$$h(x) = -2x_2(1+u) + (-2 * x_1 - x_2^2 + 1) \quad \textbf{(37)}$$

where x_1 and x_2 are defined as,

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} z \\ w \end{bmatrix} \quad \textbf{(38)}$$

and the control law u controls the drone's acceleration or thrust. Implemented into the simulation, this new safety function results in a smoother climb to the ceiling with less chatter in the decelerating velocity as can be seen in Fig. 12.

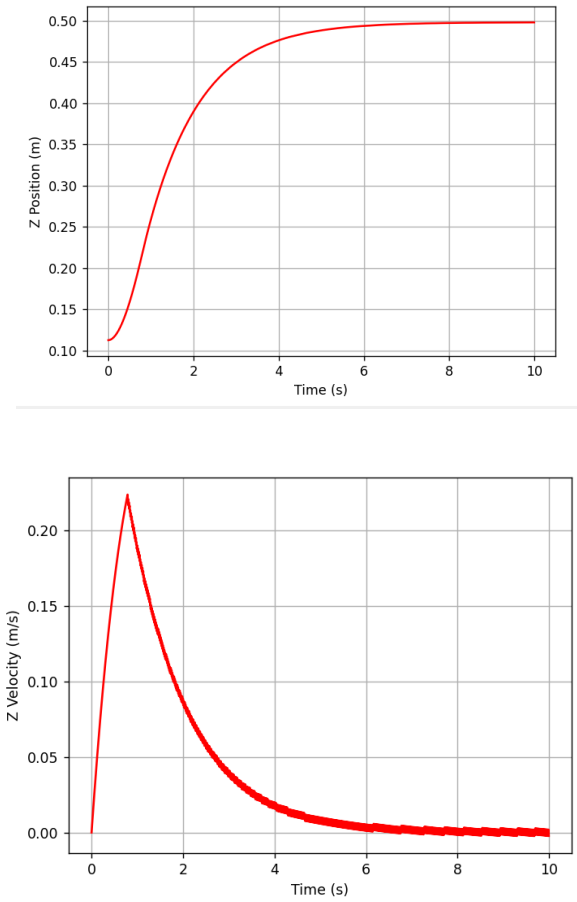

Figure 12: 1-D Uniaxial Takeoff Example with Optimization RTA

Quadratic Programming (QP) may be used in this scenario, however, because of the simplicity of the backup controller, an if statement similar to the simplex approach was used in conjunction with the CBF.

5.2 Simulating Safety for Longitudinal Motion

In this problem, a drone starts from an arbitrary position above the ground ($z = 0.15\ [m]$ in this case) and is set to travel to the point $y = 1\ [m]$ from $y = 0\ [m]$. With this setpoint and the PID controller previously introduced Eq. (12), a simulated trajectory can be formed which results in Fig 13. Note that the red line represents the safety limit at $y = 0.8\ [m]$.

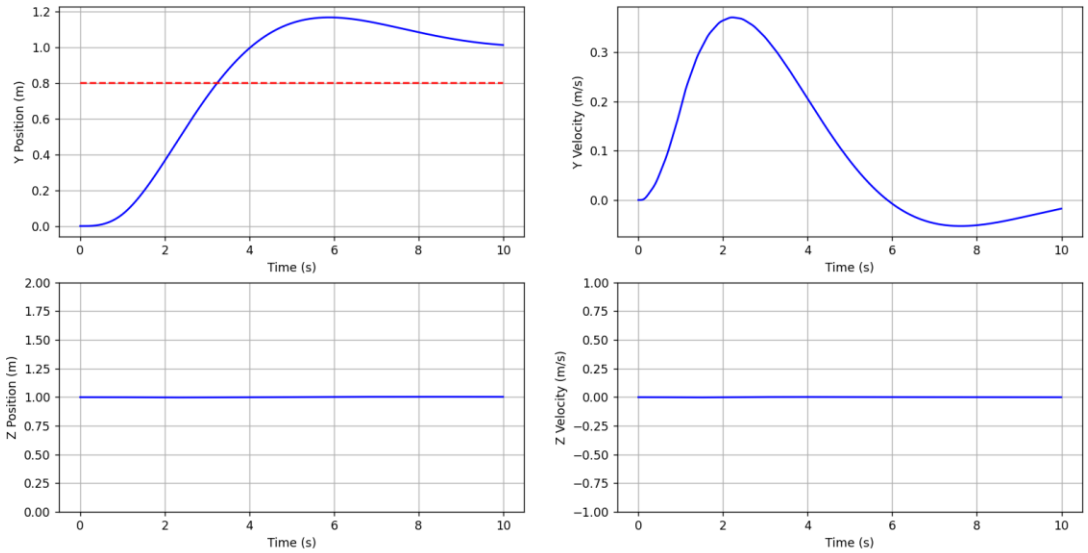

Figure 13: Longitudinal Motion Without RTA

A wall, or geofence, at a particular y-position, similar in concept to the ceiling in the one degree of freedom example, was set at $y = 0.8\ [m]$. Since this problem is higher dimensional with a 4x4 state matrix defining the roll and y positions and velocities, in order to develop a safety equation, the full definition of the control barrier function, Eq. (25.1), must be used. The state matrix to develop this CBF is given by the longitudinal subsystem Eq. (5). Unlike the uniaxial example, a safety equation cannot be derived from the system's kinematics. An alternative method of developing a CBF can be adopted from the reaction wheel pendulum example where the square of the maximum safety

parameter can be subtracted by the square of that parameter's current state and, if necessary, by a second parameter which is control dependent. The simplest equation that meets these criteria can be written as,

$$h(y) = y_{max}^2 - c_1 p^2 - y^2 \tag{39}$$

where the constant c_1 can be used to minimize the effect of the roll velocity p's effect on the safety equation. For the purposes of creating the CBF, the partial derivative of $h(y)$ relative to its states should be taken. The partial derivative for each state can be written in a vector,

$$\frac{\partial h}{\partial \boldsymbol{y}} = \begin{bmatrix} -2\mathrm{c}_1 p\dot{p} \\ 0 \\ 0 \\ -2\mathrm{y}\dot{y} \end{bmatrix}. \tag{40}$$

From this, the final CBF to be enforced can be created,

$$\frac{\partial h}{\partial \boldsymbol{y}} \left(\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta \phi \\ \Delta v \\ \Delta y \end{bmatrix} + \begin{bmatrix} 1/I_{xx} \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathrm{u}_2 \right) + \alpha h(y) \geq 0 \tag{41}$$

which results in the control invariant safety constraint set,

$$\mathrm{C}_S = \left\{ \boldsymbol{y} \in \mathbb{R}^2 \mid \frac{\partial h}{\partial \boldsymbol{y}} \left(\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta \phi \\ \Delta v \\ \Delta y \end{bmatrix} + \begin{bmatrix} 1/I_{xx} \\ 0 \\ 0 \\ 0 \end{bmatrix} \mathrm{u}_2 \right) + \alpha h(y) \ \geq 0 \right\} \tag{42}$$

The strengthening function in this case can be a constant that is multiplied by the safety equation h(y). This constant is a tunable parameter which can be changed by the controller. Implementing this CBF into a QP similar to that introduced in the

methodology section (using CVXPY) results in the graphs found in Fig. 14.

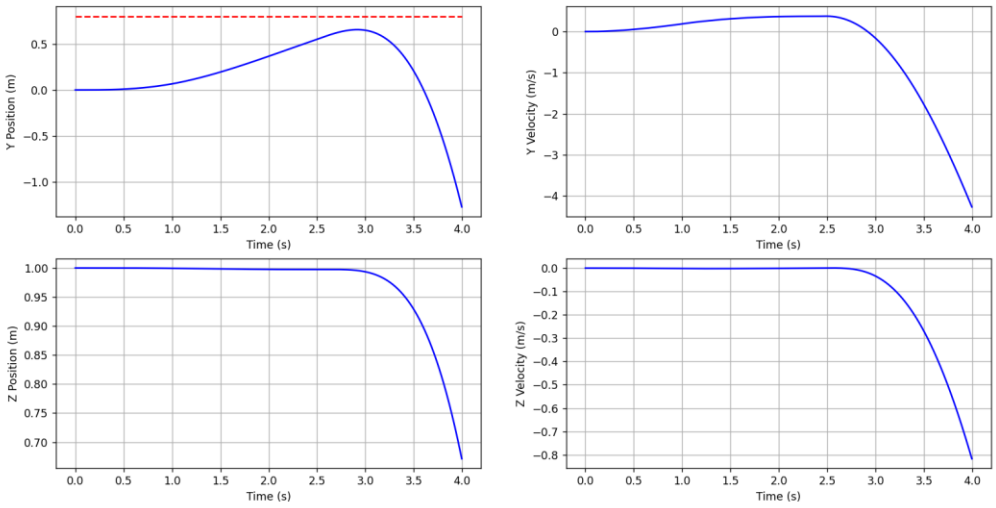

Figure 14: Longitudinal Motion With RTA

Note that the drone successfully avoids the wall at y = 0.8 [m]. After encountering the wall, however, the drone moves away from it with increasing velocity in the opposite direction while falling towards the ground as can be seen in the decaying z position. To better visualize this error, the RPM values and the safety function h(x) were plotted over the simulation time in Fig. 15 and 16, respectively.

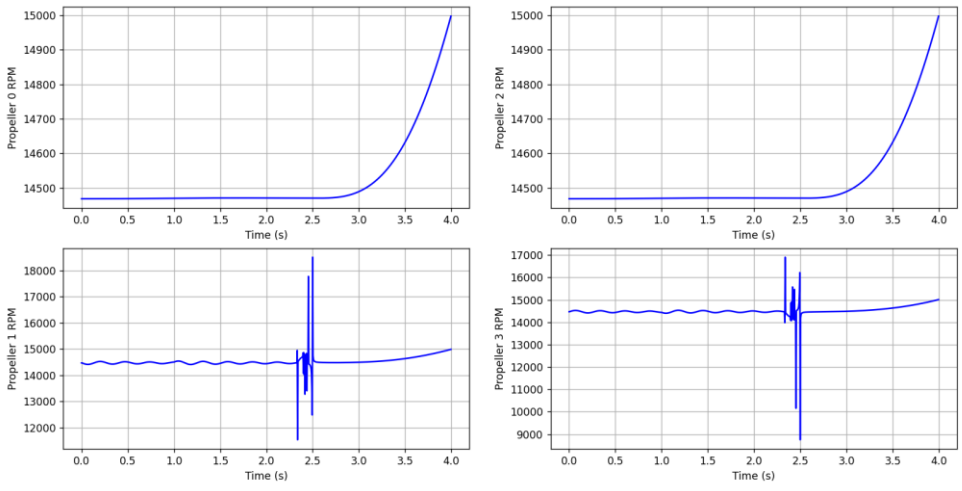

Figure 15: RPM Values With RTA

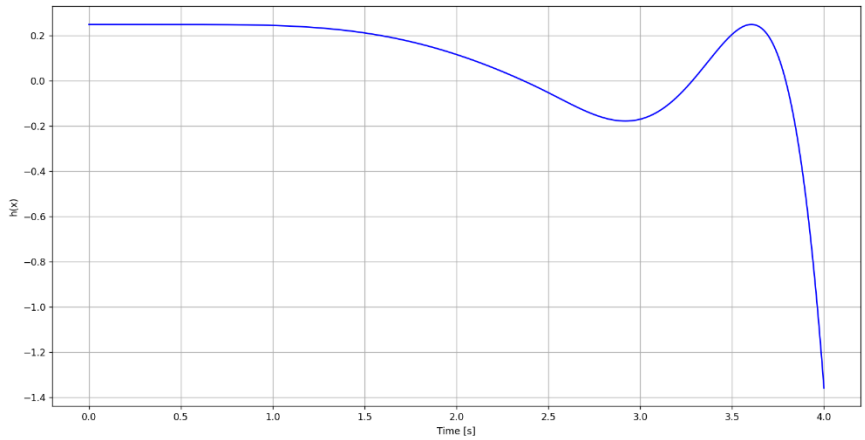

Figure 16: h(x) Over Time

The RPM values from propellers 0 and 2 plotted in Fig. 15 increase uncontrollably after the barrier is encountered at the three second mark. This increase corresponds to a change in the convexity of the safety function h(x) within the time domain.

6. Discussion and Conclusion

As can be seen in the simpler one degree of freedom vertical example, a switching filter developed with CBF, though not reliant on quadratic programming, results in a smoother climb to the safety limit which can be explained by when the RTA switches to the backup controller. In the simplex approach, the RTA only reacts when the drone reaches the safety limit, forcing a switch between the primary and secondary controller until it is settled at the safety limit whereas the ASIF approach creates a visible actuation transition (as can be seen in the velocity graph) at around one second. It can therefore be concluded that an ASIF approach is more desirable when creating a safety controller.

Developing a safety constraint for the longitudinal subsystem necessitates a different approach as the safety parameter, the y-position, is underactuated, meaning that the controller does not directly control the parameter, rather, it only controls the roll. Because of this, the method in which the CBF is constructed is different to that in the one degree of freedom example in that it must be developed independent from kinematic equations while ensuring that the control law still appears in the function. Through developing a set of safety constraints in this fashion, safety can be ensured so that the drone does not pass the established geofence limit through the implementation of a quadratic program to create an alternative, safe control law. Because of the similarities between the longitudinal and lateral subsystems, the fashion in which safety is developed in one can equally be applied to the other. A geofence can therefore be created in each of the cardinal directions with respect to the drone.

Though the drone avoided the safety limit in the longitudinal case, the trajectory that is formed after safety is assured forces the drone to move backwards away from the safety limit indefinitely while causing the drone to dip in altitude. The primary PID controller attempts to reverse this dipping effect by actuating an increase in the RPM values for the propellers most directly controlling vertical thrust, propellers 0 and 2, but fails to actuate in time to eventually avoid a crash. Though assuring safety, this motion moves the drone further away from the target, jeopardizing the initial objective. Additional work in ensuring mission objectives while also ensuring safety is required.

6.1 Flight Test Recommendations

The RTA introduced in this thesis can be readily applied to physical testing with the Crazyflie 2.x platform using the Lighthouse positioning system [17] for state tracking. The additional communication between the drone's flight computer and the controller will not cause significant delay if a Simplex-type RTA method is utilized, however, the additional time required for the QP to solve for the filtered control signal may cause significant communication delay. An alternative to creating safety assurance with a CBF without necessarily having to utilize a QP would be an explicit method of switching between primary and secondary controller types as used in the uniaxial takeoff example. This would effectively take the form of an if statement where,

$$If\ \dot{h}(x)\left(Ax + Bu_{primary}\right) \geq -\alpha h(x)$$

$$then\ u_{safe} = u_{primary}$$

$$If\ \dot{h}(x)\left(Ax + Bu_{primary}\right) < -\alpha h(x)$$

$$then\ u_{safe} = \frac{-\alpha h(x) - \dot{h}(x)Ax}{\dot{h}B}.$$

Creating an RTA filter in this fashion ensures safety in a similar manner as the QP would while reducing computational resources and most essentially time.

References

[1] R. Hewett, S. Puangpontip, “On Controlling Drones for Disaster Relief,” Procedia Computer Science, Volume 207, Pages 3703-3712, 2022

[2] H. M. Sweatland, A. Islay, and W. E. Dixon, “Higher-Order Control Barrier Function for Constraining Position in Motorized Rehabilitative Cycling,” 2021 61st IEEE Conference on Decision and Control (CDC), 2022

[3] K. Dunlap, M. Hibbard, M. Mote, and K. L. Hobbs, “Comparing Run Time Assurance Approaches for Safe Spacecraft Docking,” arXiv:2110.00447v1 [eess.SY], 2021

[4] K. L. Hobbs, M. L. Mote, M. C. L. Abate, S. D. Coogan, and E. M. Feron, “Run Time Assurance for Safety-Critical Systems,” IEEE Control Systems Magazine, Vol 43. No. 2 p. 28-65, 2023

[5] C. I. Chinelato, G. P. Das Neves, and B. A. Angelico, “Safe Control of a Reaction Wheel Pendulum Using Control Barrier Function,” IEEE DOI 10.1109/ACCESS.2020.3018713, 2020

[6] N. Hamilton, K. Dunlap, T. T. Johnson, K. L. Hobbs, “Ablation Study of How Run Time Assurance Impacts the Training and Performance of Reinforcement Learning

Agents," arXiv:2207.04117v1 [cs.LG], 2022

[7] X. Yi, A. Chakarvarthy, and Z. Chen, "Cooperative Collision Avoidance Control of Servo/IPMC Driven Robotic Fish With Back-Relaxation Effect," IEEE Robotics and Automation Letters, Vol. 6, No. 2, 2021

[8] K. Dunlap, "Run Time Assurance for Intelligent Aerospace Control Systems," M.S. Thesis submitted to the University of Cincinnati, 2022

[9] K. Zhou, "Safety Control in Quadrotors Based on Control Barrier Function," M.S. Thesis submitted to Eindhoven University of Technology, 2021

[10] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K, Sreenath, and P. Tabuada, "Control Barrier Functions: Theory and Applications," arXiv:1903.11100v1 [cs.SY], 2019

[11] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control Barrier Function Based Quadratic Programs for Safety Critical Systems," IEEE Transactions and Automatic Control, Vol. 62, No. 8, 2017

[12] C. Luis., J. Le Ny, "Design of a Trajectory Tracking Controller for a Nanoquadcopter," Mobile Robotics and Autonomous Systems Laboratory,

Polytechnique Montreal, 2016

[13] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Scheollig, “Learning to Fly – a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control,” arXiv:2103.02142v3 [cs.RO], 2021

[14] J. Forster, “System Identification of the Crazyflie 2.0 Nano Quadcopter,” B.S. Thesis submitted to the Institute for Dynamic Systems and Control Swiss Federal Institute of Technology (ETH) Zurich, 2015

[15] P. Wieland and F. Allgöwer, “Constructive safety using control barrier functions,” IFAC Proc. Volumes, vol. 40, no. 12, pp. 462-467, 2007

[16] Himanshu, K. Harikumar, and J. V. Pushpangathan, “Waypoint Navigation of Quadrotor using Deep Reinforcement Learning,” IFAC PapersOnLine 55-22, pgs. 281-286, 2022

[17] A. Taffanel, B. Rousselot, J. Danielsson, K. McGuire, K. Richardsson, M. Eliasson, T. Antonsson, and W. Hönig, “Lighthouse Positioning System: Dataset, Accuracy, and Precision for UAV Research,” arXiv:2104.11523v1 [cs.RO], 2021