Onboard Safety Guarantees for Racing Drones: High-Speed Geofencing With Control Barrier Functions

Andrew Singletary, *Graduate Student Member, IEEE*, Aiden Swann, Yuxiao Chen, and Aaron D. Ames, *Fellow, IEEE*

***Abstract*—This letter details the theory and implementation behind practically ensuring safety of remotely piloted racing drones. We demonstrate robust and practical safety guarantees on a 7" racing drone at speeds exceeding 100 km/h, utilizing only online computations on a 10 g micro-controller. To achieve this goal, we utilize the framework of control barrier functions (CBFs) which give guaranteed safety encoded as forward set invariance. To make this methodology practically applicable, we present an implicitly defined CBF which leverages backup controllers to enable gradient-free evaluations that ensure safety. The method applied to hardware results in smooth, minimally conservative alterations of the pilots' desired inputs, enabling them to push the limits of their drone without fear of crashing. Moreover, the method works in conjunction with the preexisting flight controller, resulting in unaltered flight when there are no nearby safety risks. Additional benefits include safety and stability of the drone when losing line-of-sight or in the event of radio failure.**

***Index Terms*—Robot safety, aerial systems: mechanics and control.**

I. Introduction

As HOBBY drones become more and more capable, the interest in drone racing continues to increase. Transparency Market Research predicts the drone racing industry to reach a valuation of $786 m by 2027. And while progress is being made in autonomous racing drones [1], [2], it is still an area dominated by humans [3]. The goal of this paper is to study racing drones in the theoretic context of achieving safe flight through minimal pilot interventions. Importantly, we demonstrate this practically in a realistic scenario: high speed drone flight.

Safety of small aerial vehicles is a heavily researched area. These works generally focus on safely planning trajectories rather than intervening along a desired trajectory. In this setting, [4] accounts for the low computational ability of drones, as well as the slow updates of mapping software, in their design of a planner for quick flight in unknown environments using motion primitives. While this and similar planners [5] have demonstrated results in unknown environments, they have not been demonstrated at the high speeds seen in drone racing. This is true for nearly all vision planners [6], [7], as the localization and mapping algorithms simply cannot keep up with speeds that human operators are capable of. Moreover, for known environments, reinforcement learning has been utilized to plan highly dynamic trajectories at speeds exceeding 60 km/h [8], but attempting to track these trajectories on hardware results in large tracking errors. This points to the difficulty of adapting existing strategies to ensure safety with human operators in the loop.

While most drone racing research focuses on autonomy [9], this work departs from this paradigm with the goal of giving the human operators as much freedom and control authority as possible subject to safety constraints. In particular, we seek to guarantee safety of the drone in known environments (with realistic sensing and actuation constraints) through minimal operator control intervention. This can be viewed as a "safety filter" that allows the pilot to aggressively operate the drone in a safe fashion—even at high speeds and when performing aggressive maneuvers.

While the concept of minimally invasive, shared control systems is less studied than its fully autonomous counterpart, several approaches exist in the literature. In [10], the authors propose a sampling-based MPC approach that generates many possible safe trajectories at each time-step, and chooses the one closest to the user's desired input subject to safety conditions. Another MPC-based approach is demonstrated in [11], which uses learning to minimize conservatism, but neither of these approaches are able to run in real-time on a microcontroller. In the context of geofencing, [12] presents an MPC-based approach, but it lacks the guaranteed feasibility of [10]. In [13], an Explicit Reference Governor (ERG) scheme modifies the derivative of the applied inputs subject to safety constraints utilizing Lyapunov functions. While this approach is optimization-free and could be implemented online, it is difficult to find the required upper-bound of the Lyapunov function that guarantees constraint satisfaction.

Control barrier functions [14], have recently been proven to provide an effective means of enforcing safety on a wide variety of robotic systems [15], (including drones [16]). However,

Manuscript received September 9, 2021; accepted December 27, 2021. Date of publication January 25, 2022; date of current version February 2, 2022. This letter was recommended for publication by Editor Jun Zhang upon evaluation of the Associate Editor and reviewers' comments. This work was supported by in part by AeroVironment and in part by NSF CPS under Award #1932091. *(Corresponding author: Andrew Singletary.)*

The authors are with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: asinglet@caltech.edu; aswann@caltech.edu; chenyx@umich.edu; ames@caltech.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2022.3144777, provided by the authors.

Digital Object Identifier 10.1109/LRA.2022.3144777

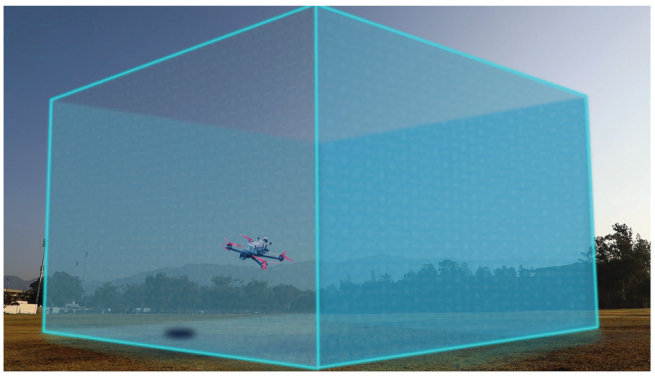


Fig. 1. An illustration of geofencing—the pilot can move freely in the boxed region but cannot leave this volume—which is enforced during high speed flight (upwards of 100 km/h) through the methodology presented in this paper.

state-of-the-art barrier function implementations are not particularly well-suited for high-speed drone flight as they typically rely on simplified models that become invalid at high speeds, or have no guarantees of feasibility. The backup-set approach can be used to enforce safety of quadrotors on the full-order dynamics [17], but requires more onboard computational power than is available on small racing drones.

The goal of this paper is to provide formal guarantees of safe flight via minimal operator intervention for high speed drones, utilizing only onboard sensing and computation. To achieve this goal, we leverage the *implicit control barrier functions* framed in the context of backup controllers. The concept of using implicit control barrier functions to guarantee safety in a more computationally efficient manner, without the use of derivatives, was first introduced in [18]. While the method presented in this paper is based on our previous work [18], we have made significant improvements to enhance its performance and make it more applicable to hardware implementation. The previous algorithm had a tendency to get stuck near the boundary of the safe set if model-mismatch or large disturbances were present, and thus it was tested only in simulation. The first contribution of this paper is a more refined safety filter that avoids issues near the set boundaries and interfaces with any off-the-shelf flight controller, dramatically extending the practicality of the method while still retaining the formal safety guarantees. The second contribution is our extensive, real-world testing of this safety filter. To this end, *we experimentally realize control barrier functions to enforce geofencing (cf. Fig. 1) at speeds of 100 km/h.*

II. Preliminaries

A. Safe Flight and Set Invariance

To prevent crashing and guarantee safety, our goal is to ensure that the system's state $x(t)$ stays in a predefined safe set S, such as the box shown in Fig. 1 typically seen in geofencing. Before formalizing this notion of safety, we must first introduce some notation and definitions.

We consider a nonlinear control-affine dynamic system:

$$\dot{x} = f(x) + g(x)u, \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in U$ is the control input, to be chosen from an admissible input set $U \subseteq \mathbb{R}^m$, while $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ describe the dynamics. Given a controller $u = \rho(x)$ and an initial condition $x(t_0) = x_0 \in \mathbb{R}^n$, the solution to this system is given by the flow map $x(t) = \Phi_\rho(x_0, t), t \geq t_0$, i.e., Φ_ρ takes the initial state and time of flow and maps to the future state of the closed-loop system under ρ.

The goal of forcing the system to remain in S at all time can be achieved through the concept of set invariance.

Definition 1: A set S is called *invariant* if the system state stays in the set indefinitely, i.e. $\forall t \geq t_0,\ x(t) \in S$. S is *control invariant* if there exists a control law $k : \mathbb{R}^n \to U$ mapping x to the admissible input set U such that the system is invariant under k, i.e. $\forall t \geq 0, \forall x_0 \in S,\ \Phi_k(x_0, t) \in S$.

B. Control Barrier Functions

Given an invariant set S, a control barrier function can be formulated and used to enforce set invariance at each time-step.

Definition 2 ([14]): Let $S \subset \mathbb{R}^n$ be the set defined by a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$:

$$\begin{aligned} S &= \{x \in \mathbb{R}^n : h(x) \geq 0\}, \\ \partial S &= \{x \in \mathbb{R}^n : h(x) = 0\}, \\ \mathrm{Int}(S) &= \{x \in \mathbb{R}^n : h(x) > 0\}. \end{aligned}$$

Then h is a *control barrier function (CBF)* if $\frac{\partial h}{\partial x} \neq 0$ for all $x \in \partial S$ and there exists an *extended class K function* ([14, Definition 2]) α such that for all $x \in S$, $\exists u$ s.t.

$$\underbrace{\frac{\partial h}{\partial x} f(x) + \frac{\partial h}{\partial x} g(x) u}_{\dot{h}(x,u)} \geq -\alpha(h(x)). \tag{2}$$

By choosing an input u such that (2) holds, the invariance of S is guaranteed [14]. Therefore, given an input from the flight operator, or another preexisting controller, $u_{\text{des}}(x, t)$, we can pick the closest input $u^*(x, t)$ to $u_{\text{des}}(x, t)$ that guarantees safety by solving the following quadratic program (QP):

$$\begin{aligned} u^*(x,t) = \underset{u\in U}{\operatorname{argmin}} & \|u - u_{\text{des}}(x,t)\|^2 \\ \text{s.t.} & \frac{\partial h}{\partial x} f(x) + \frac{\partial h}{\partial x} g(x) u \geq -\alpha(h(x)). \end{aligned} \tag{3}$$

As mentioned in the introduction, CBFs work very well for maintaining set invariance. However, they are often restricted in their usage due to the difficulty in constructing an invariant set S. Several methods for computing such sets exist [19], [20], but almost all methods for nonlinear systems suffer heavily from the curse of dimensionality and are inapplicable to drones without model simplification.

C. Backup Controllers to Construct Invariant Sets

The backup set approach [21] was recently proposed as a remedy to the difficulty of obtaining control invariant sets. This approach assumes the knowledge of a very small backup set S_B that is invariant under some backup controller $\pi(x)$. This can be, for example, a set describing the hovering maneuver for the drone and a controller commanding the drone to stop and hover in place; we will give more details on the backup set and controller selection later.

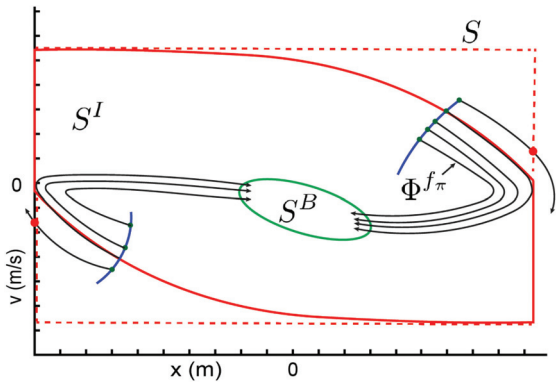


Fig. 2. The flow Φ_π and the backup set S_B are used to construct the invariant set S_I, following (4).

With the small backup set defined, a point $x_0 \in S$ is in the control invariant set S_I if the system is able to reach the backup set S_B over some time T without leaving the safe set S under the predefined backup controller [21]. This makes the description of our control invariant set:

$$S_I = \left\{ x_0 \in \mathbb{R}^n \;\middle|\; \underset{\forall \tau \in [0,T]}{\Phi_\pi(x_0, \tau) \in S} \wedge \Phi_\pi(x_0, T) \in S_B \right\} \quad (4)$$

where $\Phi_\pi(x_0, t)$ is the flow of the system from initial condition x_0 after time t, governed by $\dot{x} = f(x) + g(x)\pi(x) = f_\pi(x)$, where $\pi(x)$ is the backup control policy.

The concept of obtaining an invariant set S_I from S using backup controllers is illustrated in Fig. 2.

As shown in [22], the set generated by a proper backup controller can approach the size of the maximal control invariant set. However, for drone racing, where the extraordinary high-speed is coupled with limited onboard computational ability, these types of CBFs are not a realistic option. This is due to the computationally expensive gradient computation at each time-step to determine $\frac{\partial h}{\partial x}$.

The following section introduces our proposed method, which is similar to backup-set CBFs, but does not require any gradient computations or optimization solvers.

III. Theory

A. Smooth Safety Filtering Along the Backup Controller

In the previous section, we established that a backup controller and the corresponding backup set can be used to define a control invariant set as in (4). Let us describe the safe set S and the backup set S_B as the zero super-level sets of smooth functions $h(x)$ and $h_B(x)$. We rewrite S_I defined in (4) as:

$$S_I = \left\{ x \in \mathbb{R}^n \;\middle|\; \underset{\forall \tau \in [0,T]}{\left(h\left(\Phi_\pi(x, \tau)\right) \geq 0\right)} \wedge \left(h_B\left(\Phi_\pi(x, T)\right) \geq 0\right) \right\}. \quad (5)$$

Furthermore, following [22], we know that

$$S_I = \left\{ x \in \mathbb{R}^n \;\middle|\; \min_{\tau \in [0,T]} \left\{ h\left(\Phi_\pi(x, \tau)\right), h_B\left(\Phi_\pi(x, T)\right) \right\} \geq 0 \right\}. \quad (6)$$

For ease of notation, we define the following implicit CBF.

Definition 3: Given a control system (1) with flow map $\Phi_\pi(x, T)$ associated with a backup controller $u = \pi(x)$, the *implicit control barrier function (ImCBF)* h_I is defined as

$$h_I(x) := \min_{\tau \in [0,T]} \left\{ h\left(\Phi_\pi(x, \tau)\right), h_B\left(\Phi_\pi(x, T)\right) \right\}. \quad (7)$$

Associated with the ImCBF is the safe set given as in (6): $S_I = \{x \in \mathbb{R}^n \mid h_I(x) \geq 0\}$.

The ImCBF h_I is implicitly defined as it relies on the flow map of the system, which is also implicit. h_I can be evaluated by forward simulating the system under the backup controller.

While invariance could be enforced using the QP (3) with the ImCBF h_I as its constraint, called backup CBF QP, the implicitly defined CBF requires additional computation power to solve. Typically, the backup CBF QP is not achievable in real-time without a desktop-grade CPU. A simplification must be made in order to guarantee safety with a low-weight, low-power microcontroller suitable for a small racing drone, such as the Teensy 4.

Instead, we choose an approach utilizing the backup controller directly. Rather than switching to the backup controller when the state is about to leave S_I, our approach smoothly switches between the desired controller and the backup controller as the boundary of S_I is approached. We define the function that regulates this smooth switching the *regulation function*, and denote it $k(x, h_I(x))$.

The proposed method for safety filtering using the regulation function is formalized in the following proposition.

Proposition 1 ([18]): Consider the open-loop system (1) under a continuous control law given by $k(x, h_I(x))$. If

$$k(x, 0) = \pi(x) \quad \forall x \in S_I, \quad (8)$$

for backup control law $\pi(x)$, then the closed-loop system under $k(x, h_I(x))$ is forward invariant, i.e. safe.

Proof: By definition, S_I is invariant under the control law $\pi(x)$. Therefore, $\forall x_0 \in S_I, \forall t \geq t_0, \Phi_\pi(x_0, t) \in S_I$. By continuity of the flow operator, we know that Φ_k is continuous, and h_I is continuous since it is the composition of continuous functions h and h_B and the flow Φ_k. Therefore, the state must pass through $h_I(x) = 0$ before exiting S_I. Without loss of generality, label any such point x_{h_0}. Since $f_k = f_\pi$ at such a point, the system will remain in S_I for all time, as $x_{h_0} \in S_I \Rightarrow \Phi_\pi(x_0, t) \in S_I, \forall t \geq t_{h_0}$. ■

Additionally, to improve performance, we require that for $h_I(x) \gg 0, k(x, h_I(x)) = u_{\text{des}}(x)$. This way, the filter does not modify pilot's actions unless there is an eminent risk of leaving the safe set.

Other than the reduced computational complexity, there are several advantages to this filtering approach. Unlike control barrier functions, there is no notion of relative degree for the input, giving complete freedom in the choice of $h(x)$. Moreover, input bounds can be handled trivially as, by design, we choose $\pi(x) \in U$, and k can be constructed so that the input bound is always satisfied. Another benefit of this method is showcased in the following proposition.

Theorem 1: If $k(\cdot, \cdot)$ is locally Lipschitz in its arguments, and the dynamics under the backup controller f_π are continuous and

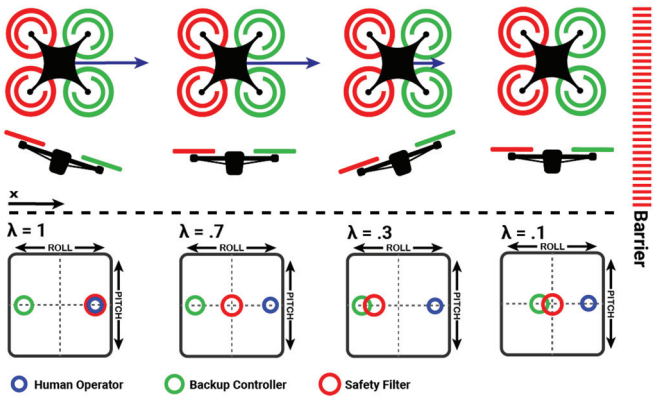


Fig. 3. As the drone approaches the barrier, λ decreases, resulting in the backup controller being utilized more.

bounded on S, then the resulting filter $k(x, h_I(x))$ is locally Lipchitz continuous.

Proof: By the definition of a CBF, $h(x)$ and $h_B(x)$ are differentiable. Moreover, the flow of the system $\Phi_\pi(x,t)$ is differentiable for continuous dynamics and backup controller by the second fundamental theorem of calculus, and therefore locally Lipschitz (since it is also bounded). Since Lipschitz continuity is preserved under the $\min$ operator, we have that $h_I(x)$ is locally Lipschitz continuous. Lastly, since Lipschitz continuity is preserved under compositions, we have that $k(x, h_I(x))$ is locally Lipschitz continuous. ■

While switching controllers can result in discontinuous inputs, the locally Lipschitz property demonstrated in Theorem 1 guarantees smoother control inputs with this method.

B. Choice of Regulation Function

The two requirements for our filtering function is that (i) the backup controller $\pi(x)$ is applied when $h_I(x) = 0$, (ii) it is Lipschitz continuous in its arguments. When $h_I(x) > 0$, we want the filter to mimic $u_{\text{des}}(x)$ as much as possible. To achieve this, we first choose the mixing function

$$k\left(x, h_I(x)\right) = \lambda\left(x, h_I(x)\right) u_{\text{des}}(x) + \left(1 - \lambda(x, h_I(x))\right)\pi(x), \tag{9}$$

for $\lambda(x, h_I(x)) : \mathbb{R}^n \times \mathbb{R} \to [0, 1]$. Fig. 3 illustrates how this mixing works. When $\lambda(x, h_I(x)) = 1$, the human operator is in complete control. As the drone approaches the wall, the value of $\lambda(x, h_I(x))$ decreases, and it begins to slow down. Finally, near the boundary, a steady-state is reached between the operator's control action and the backup control action, and the drone stops. It is important that the backup controller attempts to move away from the boundary, so that the system does not get "stuck" near the boundary, i.e. $\lambda > 0$ always.

To allow for maximum freedom of the operator, the function should exactly match $u_{\text{des}}(x)$ when $h_I(x) \gg 0$. One way to achieve this is by exploiting the exponential function:

$$\lambda(x, h_I(x)) = 1 - \exp\left(-\beta h_I^+(x)\right), \tag{10}$$

where constant β is used to tune how quickly the function $\lambda(x, h_I(x))$ decays, and $h_I^+(x) = \max(h_I(x), 0)$ ensures that $\lambda(x, h_I(x)) \in [0, 1]$. With this filtering controller, we get the desired behavior of $\lambda(x, h_I) \approx 1$ when $h_I(x) \gg 0$, while providing a smooth decay to 0 when as $h_I(x) \to 0$. The constant β is used to tune how quickly the function λ decays.

C. Comparison to Backup CBF Condition

To demonstrate the effectiveness of this method, we compare its performance to the backup set CBF approach [21]. The two relevant metrics for this comparison are the computational times and the conservatism, as both methods provide guarantees of safety.

While no perfect comparison can be made, due to the freedom in the selection of both $\lambda(x)$ and $k(x)$, these functions are chosen independently to result in smooth transitions when approaching the boundary of the set, while minimizing conservatism.

Example 1: Consider an inverted pendulum with state x dynamics $\dot{x}$

$$x = \begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0 \\ 1\end{bmatrix} u, \quad \dot{x} = \begin{bmatrix}\dot{\theta} \\ \sin(\theta)\end{bmatrix} + \begin{bmatrix}0 \\ 1\end{bmatrix} u, \tag{11}$$

and backup control law

$$\pi(x) = -Fx, \tag{12}$$

which attempts to stabilize the system to the backup set

$$h_B(x) = \min\left\{\left(\frac{\pi}{12}\right)^2 - x_1^2, \delta^2 - x_2^2\right\}, \tag{13}$$

for a small velocity value δ chosen to be 0.1 rd/s, while staying in the set

$$h(x) = \min\left\{1 - x_1^2, 2 - x_2^2\right\}. \tag{14}$$

For the exact formulation of the backup-set CBF, see [22]. The filtering function used here is the same as that used on the drone (9), to be detailed in the following section. The inverted pendulum was commanded a constant angular acceleration of 2 rd/s^2, and the resulting positions, velocities, and filtered inputs are displayed in Fig. 4.

Two benefits of the smooth filter can be seen in this comparison. While filtering performance is similar, the regulation function is an order of magnitude faster to evaluate. Moreover, when the gains are increased to allow a rapid approach of the boundary of the safe set, the CBF begins to oscillate near the boundary, whereas the smooth filter does not suffer from such behavior. These oscillations occur due to numerical instability of the optimization problem as the system pushes against boundary of the safe set.

It is important to note that the optimization-based CBF still has a distinct advantage in some situations. Utilizing gradient information allows quick motion along the boundary of the set, whereas with this switching approach, the value of $\lambda(x, h_I(x))$ will be low, limiting performance near the boundary. In future work, we will explore how this ability can be utilized in a derivative-free approach. While it is not particularly important for geofencing, it would be important in the context of collision avoidance while drone-racing.

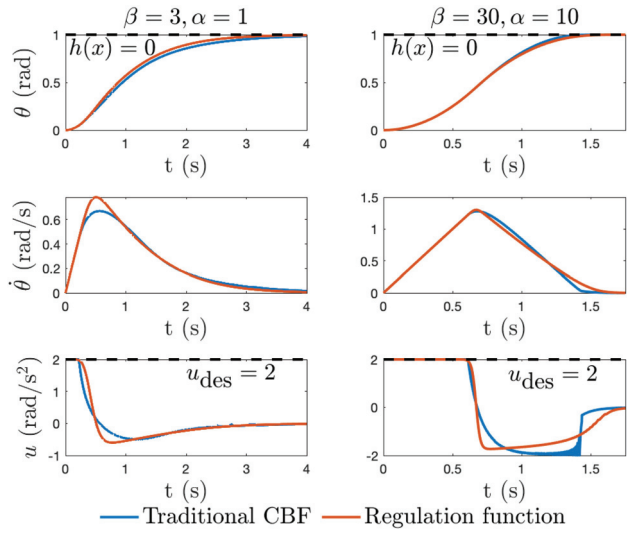


Fig. 4. The filtering performance of the traditional CBF compared to the proposed regulation function, for two parameters β in (10) and the scalar α for the CBF (3).

IV. Modeling and Implementation

A. Modeling the Drone and Onboard Flight Controller

Flight controllers on modern racing drones are able to track desired angular rates extremely well. With the availability of low-cost, high-speed electronic speed controllers (ESCs) and rate gyros, state-of-the-art controllers can track angular rates at control frequencies of 8 kHz. We utilize this by wrapping our controller around the closed-loop system of the drone with the onboard flight controller. Therefore, rather than the control inputs being the torques of the four motors, we command throttle and angular rates. This choice of architecture greatly simplifies the task of modeling the drone dynamics, and allows us to better filter the system in a way that minimizes the impact on the pilot. Moreover, the same filter can be applied to different drones with different dynamics, including those with six or eight rotors.

By modeling the response of the system to desired angular rate commands, the proposed method does not rely on perfect angular rate tracking. Through this, we also account for any delay in the filter stemmed from communication between the sensors and the onboard flight controller.

The drone and flight controller system is modelled as rigid body motion in the Special Euclidean Group in 3 dimensions, SE(3). The state-space model $x \in \mathbb{R}^{13}$ is chosen to be $x = [p_w, q, v_w, \omega_b]^T$ where $p_w = [x, y, z]^T$ is the position in the world frame, $v_w = [v_x, v_y, v_z]^T$ are the world-frame velocities, q is the quaternion representation of the orientation with respect to the world frame, and $\omega_b = [\omega_x, \omega_y, \omega_z]^T$ are the body-frame angular velocities.

To model the system's response to an angular rate command, we set the derivative of the angular rates to

$$\dot{\omega} = C(x)(\omega_{\text{des}} - \omega), \tag{15}$$

where $C(x) : \mathbb{R}^n \to \mathbb{R}_+$ is a (potentially state-dependent) function that determines how quickly the desired rates are tracked. For a well-tuned racing drone with minimal filter delay, its value should be on the order of 50, and can be treated as state-independent.

Lastly, to map the throttle command to thrust, we fit a second-order polynomial with data from the accelerometer and GPS. While this mapping will be dependent on the voltage, the inclusion of an integral term in the altitude controller is generally sufficient to eliminate any drift.

B. Safe Sets and Backup Controllers

The primary goal of this work is to constrain the position of the drone inside a large polytope in the 3D space, inside of which the pilot has almost complete control, but is unable to leave. To this end, we define the safe set

$$h(x) = \min\{r_x^2 - (x - x_c)^2, r_y^2 - (y - y_c)^2, r_z^2 - (z - z_c)^2\}, \tag{16}$$

which is positive inside of a box with side lengths (r_x, r_y, r_z) centered at (x_c, y_c, z_c), and negative outside.

The backup controller $\pi(x)$ is a velocity controller on SE(3), inspired by [23]. The backup controller attempts to bring the drone to zero velocity in the x, y, z, but has one other goal which is very important: to bring the drone away from the boundary if it is too close. This is critical, as if the drone were to simply stop at the boundary, the pilot would be stuck at the edge of the safe set due to $\lambda(x)$ approaching 0. To achieve this, we set the desired velocity to

$$v_x = \begin{cases} 0 & r_x^2 - (x - x_c)^2 \geq \delta, \\ -(\delta - r_x^2 + (x - x_c)^2) & \text{otherwise.} \end{cases} \tag{17}$$

Under this backup controller, the drone will move a distance δ from the boundary before stopping. The desired velocities are identical for y and z directions.

Finally, the backup set S_B is defined by the function

$$h_B = -\sqrt{v_x^2 + v_y^2 + v_z^2} + \epsilon. \tag{18}$$

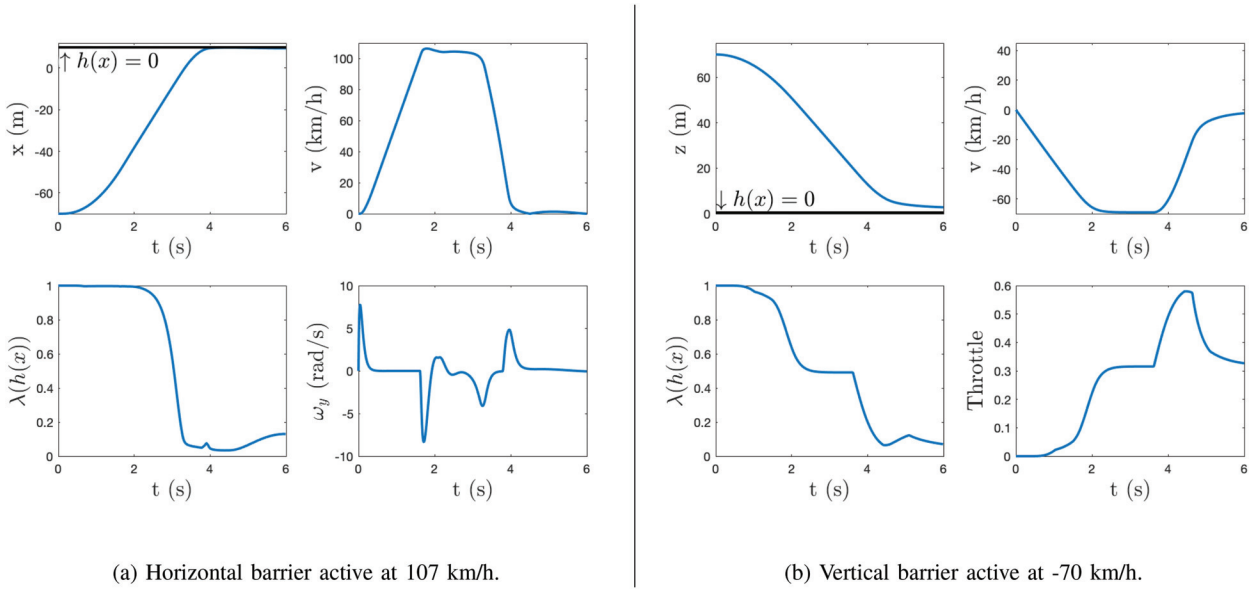
This backup set ensures that the drone is able to slow itself to a speed of ϵ, chosen to be 0.1 m/s, thus guaranteeing that the drone is able to stop before hitting the boundary.

C. Modification of Safety Filter for Very High Speeds

Modifications must be made to the function $\lambda(x, h_I(x))$ to work well at very high-speeds. This is because β must be made large to have smooth breaking at high speeds, which would make the filter overly conservative near the boundary at low speeds. This can be fixed simply by scaling the value of $h_I(x)$ by the inverse of the velocity towards the barrier. The safety filtering function used by the drone is

$$\lambda(x, h_I(x)) = 1 - \exp\left(-\frac{\beta h_I(x)^+}{v_\perp^+}\right), \tag{19}$$

where $v_\perp$ is the velocity in the direction of the barrier.



(a) Horizontal barrier active at 107 km/h.

(b) Vertical barrier active at -70 km/h.

Fig. 5. Simulation results of the two primary hardware test cases. On the left, the drone accelerates towards the barrier at $x_w = 10$ m. On the right, the drone free-falls from 70 m towards the barrier at $z_w = 0.5$ m.

D. Simulation

Before testing the barrier functions on hardware, we first devise the test cases in simulation. While the safety filters are the same in simulation and on hardware, the hardware is operated by a human pilot, while the sim has its own desired controllers, so some discrepancies will arise because of this.

Two primary test cases were run in simulation, a high-speed horizontal test, with the goal of successful filtering at 100 km/h, and a free fall from 70 m. The results of the horizontal simulation are shown in Fig. 5a. The drone accelerates to a maximum speed of 107 km/h before being forced to stop. The minimum distance to barrier was 0.12 m. The free fall simulation was also successful: the drone accelerated to a top speed of 70 km/h downwards, before reaching a hover at a distance of 1.2 m above the barrier.

V. Results

A. Hardware Setup

Our quadrotor is built on a Chimera 7" frame with four iFlight XING X2806.5 1300 KV brushless motors, a T-Motor F55 A Pro II 4-in1 ESC, a MAMBA BASIC F722 Flight Controller (FC), a Teensy 4.1 microcontroller, a Vectornav VN-200 IMU+GPS, a FrSky R-XSR receiver, a DJI FPV air unit, and a Cadex FPV camera. We use a FrSky QX7 radio to send desired angular rates commands to Teensy microcontroller through the receiver. The VN-200 fuses GPS and IMU data with a built in extended Kalman filter. This data is sent to the Teensy as navigation data at 400 Hz. Using this data, the microcontroller then modifies these angular rates commands with the regulation function and then forwards them on to the FC. The FC runs betaflight, an open source software, to track the commanded angular rates. The PID loop runs at the gyro update rate at 8 kHz. The FC sends digital commands to the ESC using DSHOT600 at the same 8 kHz. FPV video is digitally streamed with a end to end latency of 25 ms from the DJI FPV air unit to DJI FPV goggles, which are worn by the operator.

Fig. 6. The 7" racing drone used for experiments.

B. Filter Software and Execution

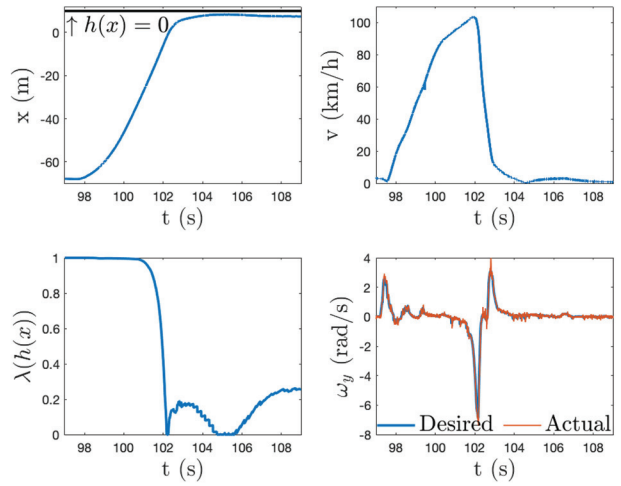
To simplify the transition from simulation to hardware, the barrier functions are first generated in MATLAB for simulation, and then codegen is used to create C++ code that will run on hardware. All code used to generate and run the barrier functions on a Teensy 4.1 can be found at https://github.com/DrewSingletary/racing_drone_geofencing. This codebase also includes the interface for our specific receiver and flight controller, but this could be modified to fit other drone and radio configurations.

The execution time of the filter on the Teensy is approximately 350 μs. The algorithm runs at the update rate of the navigation data, which is 400 Hz.

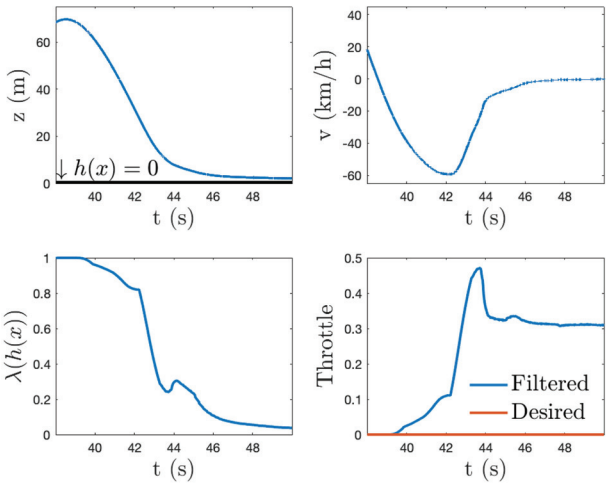
C. Outdoor Flight Tests

A large number of flight tests were done to verify the safety guarantees provided by our method. We emphasize two specific examples, but several more flight tests are displayed in Fig. 8.

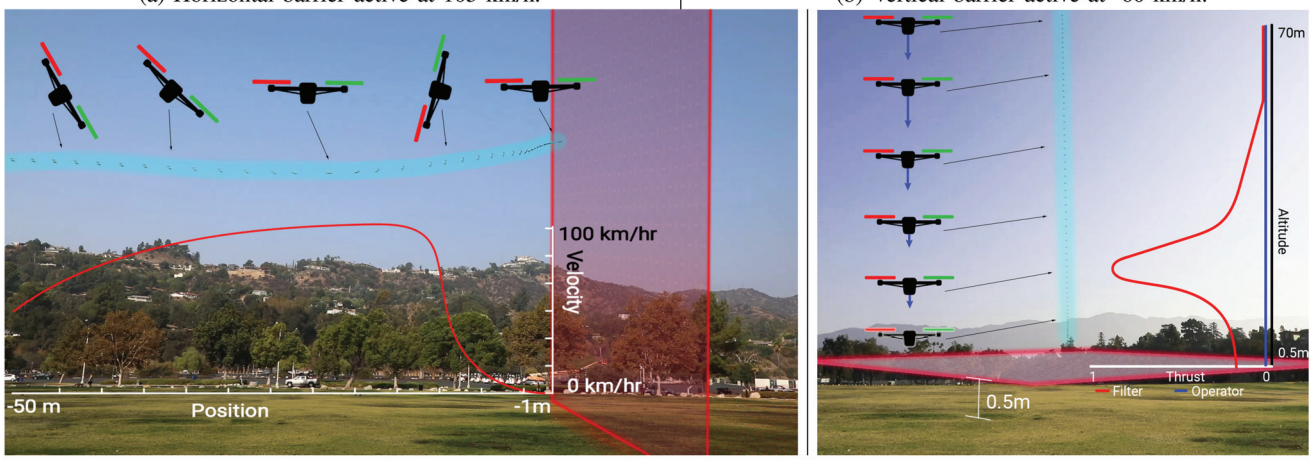
Test 1: Horizontal barrier at 104 km/h. For this test, the pilot commanded the drone to head north at high speeds. The active component of the barrier was 40 m north of the initial position. The drone was able to reach a top speed of 104 km/h,



(a) Horizontal barrier active at 105 km/h.

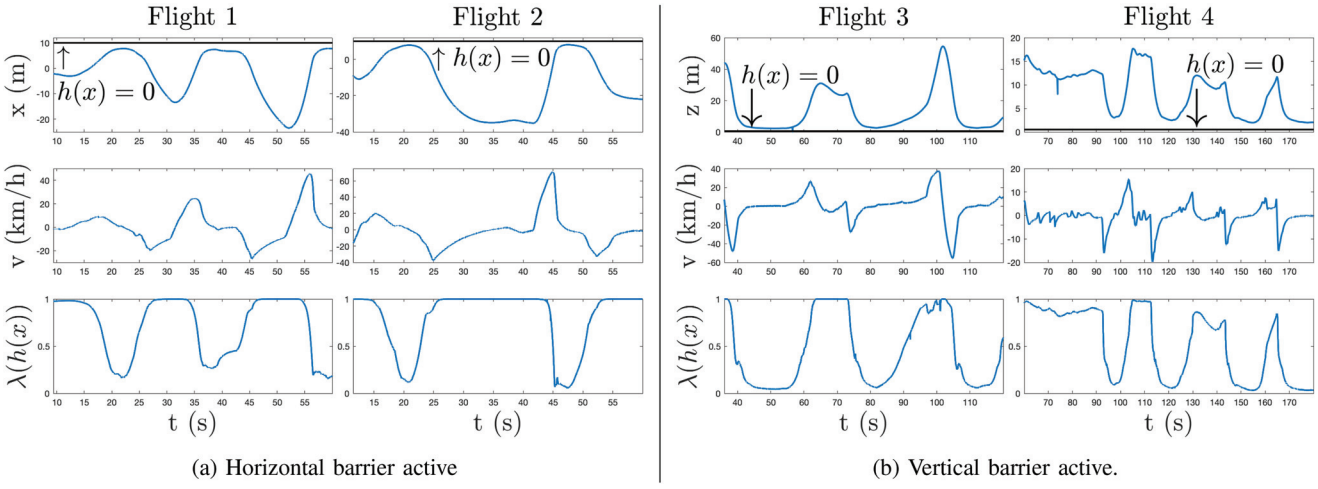


(b) Vertical barrier active at -60 km/h.



(c) Actual drone flight during the two showcased example is highlighted in blue.

Fig. 7. Two highlighted examples of geofencing with the high-speed racing drone.



(a) Horizontal barrier active

(b) Vertical barrier active.

Fig. 8. Four separate experimental runs where the drone is commanded to approach the barrier several times. λ never reaches zero, meaning the pilot always has some amount of control, and the drone never leaves the defined safe set.

before beginning to break at a distance of 15 m from the barrier. The pilot attempted to push the drone past the barrier, but was stopped at a minimum distance of 1.7 m from the barrier. Due to this, the pilot was then able to safely move away from the restricted airspace.

Fig. 7a showcases the results of the experiment. The results agree strongly with the simulation data in Fig. 5a, despite the human piloting commanding different desired inputs than the simulation. This is, in part, due to the very accurate angular rate tracking showcased at the bottom right of Fig. 7a. While there is a slight delay in this tracking, this is properly modeled in (15), thus it does not affect our ability to guarantee safety.

Fig. 7c shows the drone throughout this maneuver, highlighted in blue. Above this, the orientation of the drone at different snapshots are visualized. As shown, the drone reaches an angle of nearly 90° while breaking.

Test 2: Free fall from 70 m. At the beginning of this test, the pilot was flying at an altitude of 70 m and then sends no commands, mimicking a loss of radio connection. The barrier was chosen to be a distance of 0.5 m from the ground. After a short free-fall, reaching a vertical velocity of −60 km/h, the safety filter stabilizes the drone before coming to a stop at a distance of 1.8 m above the ground.

The data from this flight is visualized in Fig. 7b. Rather than plotting desired angular rates, we now showcase the desired throttle of the drone sent from the user compared to the throttle produced by the safety filter. Despite the pilot commanding no throttle for the entire duration of the descent, the drone is able to smoothly recover before crashing.

Again, when comparing this data to the simulation in Fig. 5(b), notice the extremely similar results. In fact, the only major discrepancy, which is the fact that the simulation reached a speed of 10 km/h faster downwards than the drone, can be easily explained by a lack of drag in the simulation model. This did not occur during the horizontal tests, as the velocity controller is able to correct for this drag in flight.

Testing for reliability and consistency. Fig. 8 highlights the reliability and consistency of this method in the application of geofencing. Four separate flights are plotted, two of which engage the horizontal barrier whereas two engage the vertical barrier. In each flight, the barrier is engaged two to four times, and every time, safety is maintained, and λ never reaches zero, meaning the pilot never lost complete control of the drone for any period of time.

VI. Conclusion

In this work, we showcased a novel safety filter intended to guarantee safe, high speed flight in the presence of a human operator. This method required no offline computations, and was implemented on a small microcontroller aboard a 7" racing drone. The filter was successful at keeping the drone inside of a desired region at speeds upwards of 100 km/h, and was easily able to recover from an 70 m free fall. Future work will consist of adding vision into the loop to dynamically construct the safe sets, as well as multi-robot collision avoidance at high-speeds for drone races.

References

[1] P. Foehn *et al.*, "Alphapilot: Autonomous drone racing," *Auton. Robots*, pp. 1–14, 2021.

[2] S. Jung, S. Hwang, H. Shin, and D. H. Shim, "Perception, guidance, and navigation for indoor autonomous drone racing using deep learning," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 2539–2544, Jul. 2018.

[3] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 6713–6719.

[4] J. Tordesillas, B. T. Lopez, J. Carter, J. Ware, and J. P. How, "Real-time planning with multi-fidelity models for agile flights in unknown environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 725–731.

[5] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 1934–1940.

[6] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: Learning agile flight in dynamic environments," in *Proc. Conf. Robot Learning. PMLR*, 2018, pp. 133–145.

[7] E. Kaufmann *et al.*, "Beauty and the beast: Optimal methods meet learning for drone racing," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 690–696.

[8] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2021.

[9] H. Moon *et al.*, "Challenges and implemented technologies used in autonomous drone racing," *Intell. Service Robot.*, vol. 12, no. 2, pp. 137–148, 2019.

[10] A. Broad, T. Murphey, and B. Argall, "Highly parallelized data-driven mpc for minimal intervention shared control," in *Proc. Robot.: Sci. Syst. (RSS)*, 2019.

[11] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, "A predictive safety filter for learning-based racing control," *IEEE Robot. Automat. Letters*, vol. 6, no. 4, pp. 7635–7642, 2021, doi: 10.1109/LRA.2021.3097073.

[12] S. Zhang, D. Wei, M. Q. Huynh, J. X. Quek, X. Ma, and L. Xie, "Model predictive control based dynamic geofence system for unmanned aerial vehicles," in *Proc. AIAA Inf. Syst.-AIAA Infotech, Aerosp.*, 2017, Art. no. 0675.

[13] E. Hermand, T. W. Nguyen, M. Hosseinzadeh, and E. Garone, "Constrained control of uavs in geofencing applications," in *Proc. IEEE 26th Mediterranean Conf. Control Automat.*, 2018, pp. 217–222.

[14] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 3861–3876, Aug. 2017.

[15] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf.*, 2019, pp. 3420–3431.

[16] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 2460–2465.

[17] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Contr. Syst. Lett.*, vol. 5, no. 1, pp. 127–132, Jan. 2021.

[18] A. Singletary, T. Gurriet, P. Nilsson, and A. D. Ames, "Safety-critical rapid aerial exploration of unknown environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 10270–10276.

[19] J.-P. Aubin, *Viability Theory*. Berlin, Germany: Springer Science, 2009.

[20] I. M. Mitchell, A. M. Bayen, and C. J. Tomlin, "A time-dependent hamilton-jacobi formulation of reachable sets for continuous dynamic games," *IEEE Trans. Autom. Control*, vol. 50, no. 7, pp. 947–957, Jul. 2005.

[21] T. Gurriet, M. Mote, A. D. Ames, and E. Feron, "An online approach to active set invariance," in *Proc. IEEE Conf. Decis. Control*, 2018, pp. 3592–3599.

[22] Y. Chen, M. Jankovic, M. Santillo, and A. D. Ames, "Backup control barrier functions: Formulation and comparative study," in *Proc. Conf. Decis. Control (CDC)*, 2021.

[23] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *Proc. 49th IEEE Conf. Decis. Control*, 2010, pp. 5420–5425.