2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 23-27, 2022, Kyoto, Japan

Safe Drone Flight with Time-Varying Backup Controllers

Andrew Singletary, Aiden Swann, Ivan Dario Jimenez Rodriguez, and Aaron D. Ames

***Abstract*— The weight, space, and power limitations of small aerial vehicles often prevent the application of modern control techniques without significant model simplifications. Moreover, high-speed agile behavior, such as that exhibited in drone racing, make these simplified models too unreliable for safety-critical control. In this work, we introduce the concept of time-varying backup controllers (TBCs): user-specified maneuvers combined with backup controllers that generate reference trajectories which guarantee the safety of nonlinear systems. TBCs reduce conservatism when compared to traditional backup controllers and can be directly applied to multi-agent coordination to guarantee safety. Theoretically, we provide conditions under which TBCs strictly reduce conservatism, describe how to switch between several TBC's and show how to embed TBCs in a multi-agent setting. Experimentally, we verify that TBCs safely increase operational freedom when filtering a pilot's actions and demonstrate robustness and computational efficiency when applied to decentralized safety filtering of two quadrotors.**

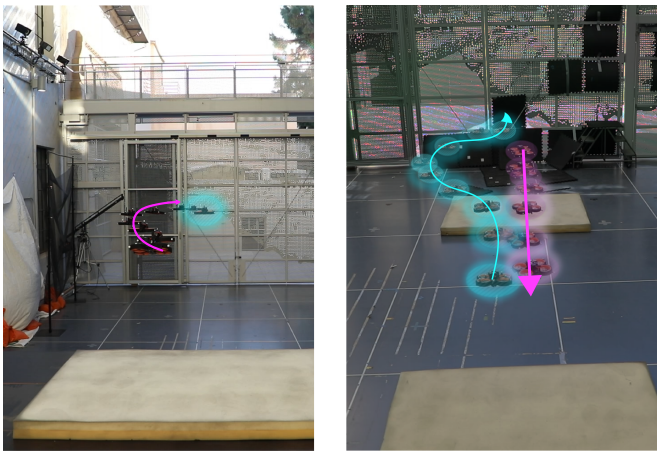


Fig. 1: Illustration of the experimental results demonstrating safe flight of multiple drones through the use of time-varying backup controllers.

I. INTRODUCTION

Drones have emerged as a staple robotic technology that have made the transition to the general public, used by hobbyists and videographers, and are finding applications in a variety of domains. Contributions from the robotics research community (see [1]–[4] to name a few) led to this transition to practice, and drones remain an active area of research due to their availability, small size and weight, nonlinear dynamics with important controllability properties, e.g., differential flatness [5], and increases in mobile computation. The result is the demonstration of impressive feats of agility, but these generally require either significant offline computation [6], or use computationally expensive predictive controllers [7], [8]. Lacking from these successes in acrobatic flying are guarantees of safety.

Safety concerns and regulations have thus far prevented the wide-scale adoption of applications that involve close proximity to humans, e.g., drone delivery. Due to their significant kinetic and potential energy, even a small drone can pose a life-threatening risk when operated incorrectly. This leads to the concept of regulating an operator's desired input to ensure safety which has been explored in several contexts. In [9], the authors leverage GPUs to generate thousands of potential trajectories via Model Predictive Control (MPC), and utilize the closest to the desired input. In [10], the authors present a promising MPC-based approach to safety filtering for vehicle racing, but the method has not shown to be applicable to higher-dimensional systems with less computational power.

Control barrier functions [11] have been widely utilized in recent years due to their computational efficiency and their ability to handle general nonlinear dynamical systems of arbitrary dimension. They have been utilized in a wide variety of quadrotor applications [12]–[14], but none of these results were computed online on flight hardware. Multi-agent drone systems have also been considered in the context of control barrier functions, as shown in [15], albeit utilizing simplified model dynamics and centralized computing. In [16], the authors present a decentralized approach to safety filtering of multiple agents utilizing the shared knowledge of backup control policies that keep the individual agents safe, but this was only tested in simulation, and could not be computed on flight hardware.

This paper presents a new approach for achieving flight with formal guarantees of safety that can be realized in practice onboard highly dynamic drones. To this end, we build upon previous work [17], wherein the authors introduce a method for utilizing backup controllers to guarantee safety in a way that did not require solving an optimization problem or computing expensive gradients online. In this work, we extend the method to include multi-agent aerial systems, as well as present a new framework for drastically increasing the operational freedom of the user through the use of maneuvers. The notion of maneuvers is formally encoded by the concept of *time-varying backup controllers (TBCs)* which formally guarantee safety. We establish that theoretically time-varying backup controllers produce larger safe sets than backup controllers without this time-varying component, thereby giving the operator more control authority without

Andrew Singletary, Aiden Swann, and Aaron D. Ames are with Department of Mechanical and Civil Engineering, Ivan Dario Jimenez Rodriguez with Deparment of Computational Math and Science, California Institute of Technology, Pasadena CA 91125, U.S.A. Email addresses: {asinglet, aswann, ivan.jimenez, ames}@caltech.edu. This work is supported by AeroVironment, NSF CPS award #1932091, and BP.

2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 978-1-6654-7927-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/IROS47612.2022.9981741

sacrificing safety. These theoretic results are demonstrated in simulation and on hardware in the context of collision avoidance, both for a single agent and for two agents. The computational benefits of the approach are illustrated through the online implementation, and the theory is validated through the resulting safe behaviors.

The layout of the paper is as follows. Section II overviews the underlying mathematical principles we will use to provide safety guarantees. Section III showcases Time-varying backup controllers, the algorithms and methods used to practically implement them and the conditions under which they provide safety guarantees with no added conservatism. We also show how to apply TBCs to multi-agent systems in a decentralized manner. Section IV outlines how to generate and implement these backup maneuvers, and demonstrates several of the concepts on a quadrotor in simulation. Finally, Section V demonstrates these ideas with a hardware implementation of two quadrotors with only small microprocessors for computation.

II. Preliminaries

A. Safe flight and set invariance

We begin by reviewing safety as set invariance. Consider a general nonlinear control-affine dynamic system:

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

with state $x \in \mathbb{R}^n$, input $u \in U$ for the admissible input set $U \subseteq \mathbb{R}^m$.

For a policy $u = \pi(x)$ and initial condition $x(t_0) = x_0 \in \mathbb{R}^n$, the solution to this closed-loop system is given by the flow map

$$x(t) = \Phi_\pi(x_0, t) = x_0 + \int_{t_0}^{t} (f(x) + g(x)\pi(x))\, dt, \; t \geq t_0. \tag{2}$$

Consider S to be the set of all states in which the system is safe. In this work, we consider the set of all states where the drones are not in collision. We introduce the notion of set invariance with the goal of ensuring that our drone stays in this set for all time without being overly restrictive on the actions the drone takes inside this S.

Definition 1 (Set Invariance)**.** A set S is called *invariant* if the system's state stays in S for all time, i.e. $\forall t \geq t_0,\; x(t) \in S$.

We restrict our attention to safe sets defined by the 0-level set of a continuously differentiable function as follows:

Definition 2 (Safe Set)**.** Let $S \subset \mathbb{R}^n$ be the set defined by a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$:

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

B. Constructing Safe Sets with Backup Controllers

Suppose we have a subset of the state-space we wish to render forward invariant $S = \{x \in \mathbb{R}^n | h(x) \geq 0\}$. Furthermore we assume there exists a set $S_B = \{x \in \mathbb{R}^n | h_B(x) \geq 0\}$ that is *control invariant* as follows:

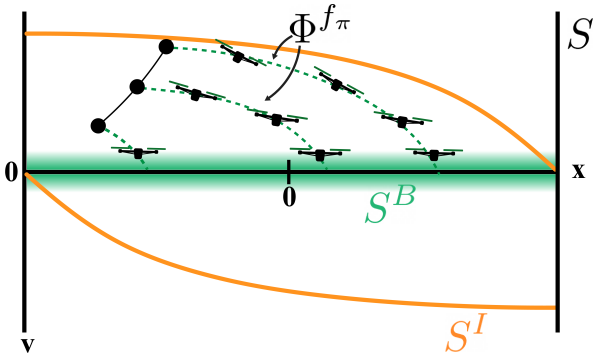


Fig. 2: Construction of invariant set S^I by the flow of the backup controller Φ^{f_π} to the backup set S^B

Definition 3 (Control Invariant Set)**.** Safe set S is *control invariant* if there exists a control policy $\pi : \mathbb{R}^n \to U$ that renders S invariant.

We assume that this $S_B \subset S$ and it usually corresponds to a trivially controllable subset of the system's state-space. We will call the particular control invariant set S_B the *backup set*. In the case of a drone, this could correspond a small region around a steady hover. The set S_B can be significantly smaller than S, and the size of S_B has almost no effect on the performance of this method.

As shown in [18], given a control policy π we can define a corresponding implicit control-invariant set:

$$S_I(\pi) = \left\{ x_0 \in \mathbb{R}^n \,\middle|\, \Phi_\pi(x_0,t) \in S \wedge \Phi_\pi(x_0,T) \in S_B \underset{\forall t \in [0,T]}{} \right\} = \tag{3}$$

$$\left\{ x_0 \in \mathbb{R}^n \,\middle|\, \min_{t \in [0,T]} \{h(\Phi_\pi(x_0,t)), h_B(\Phi_\pi(x_0,T))\} \geq 0 \right\} \tag{4}$$

In other words, this implicit safe set corresponds to the set of initial conditions such that flows of the system under the policy π remain in S for $t \in [0,T]$ and reach S_B by time T. We call policies with a non-empty S_I *backup controllers*.

This particular construction allows for the creation of an *implicit control barrier function (ImCBF)*:

Definition 4 (Implicit control barrier function (ImCBF))**.** The flow map of a system Φ_π with associated backup controller π defines an *implicit control barrier function(ImCBF* h_I defined as:

$$h_I(x) := \min_{t \in [0,T]} \{h(\Phi_\pi(x_0,t)), h_B(\Phi_\pi(x_0,T))\} \tag{5}$$

So that h_I defines the set $S_I(\pi) = \{x \in \mathbb{R}^n | h_I(x) \geq 0\}$.

While these ImCBF's can be used in the context of a CBF-based QP controller, as demonstrated first in [19], this implementation is not practical for small aerial vehicles with limited computational power due to the expensive gradient computations involved.

C. Control Filtering with Regulation Functions

In this setting, we assume there exists a desired control input u_{des} that we wish to render safe. This means we

want a filtering function $u_{\text{act}}(x,u_{\text{des}}) \in U$ that renders a safe set S forward invariant while still remaining as close as possible to u_{des}. This can be achieved using ImCBF's under the assumption that we have a backup controller u_B that renders the set $S_I \subseteq S$ control invariant. The function $u_{\text{act}} : \mathbb{R}^n \times U \to U$ must satisfy the following conditions:

(i) The backup controller is applied when x is at the boundary of the safe set:

$$h_I(x) = 0 \to u_{\text{act}}(x,u_{\text{des}}) = \pi(x) \tag{6}$$

(ii) u_{act} is locally Lipschitz continuous in its arguments.

We call any filter $\rho(x,h_I(x),u_{\text{des}}$ that satisfies these conditions a *regulation function*, as it regulates the desired inputs u_{des} such that safety is guaranteed. An example regulation function that satisfies both of these conditions is

$$\begin{aligned}&\rho(x,h_I(x),u_{\text{des}}) = \\ &\quad \lambda(x,h_I(x))u_{\text{des}}(x) + (1-\lambda(x,h_I(x)))\,\pi(x)\end{aligned} \tag{7}$$

such that $\lambda : \mathbb{R}^n \times \mathbb{R} \times U \to [0,1]$ is given by

$$\lambda(x,h_I(x)) = 1 - \exp\left(-\beta \max\{0,h_I(x)\}\right). \tag{8}$$

Here, β is a tuning parameter that controls how smoothly to mix the backup controller with the desired controller as you approach the boundary. This regulation filter is demonstrated in [20] where it is used to keep a drone inside a geofence at speeds upwards of 100 km/h. While proven to work well in practice for single agents in simple, static environments, this strategy is vulnerable to overly conservative behavior along the boundary of S, and can lead to deadlock in the multi-agent settings. With this work, we propose an extension to backup controllers that overcomes deadlocks in the multi-agent setting while strictly increasing the size of S_I.

III. Theory

This section introduces the theoretical framework for the usage of backup controllers experimentally in a multi-agent decentralized setting.

A. Time-varying backup controller

The intuition behind a time-varying backup controller stems from the desire to execute a maneuver before engaging the backup controller. This makes it easier for the backup controller to return to the backup set S_B while staying in S. In this section, we will show that even a bad maneuver will not reduce the size of S_I when compared to the original backup controller when properly formulated.

Definition 5 (Time-Varying Backup Controller)**.** Let $u_M : \mathbb{R}^n \to U$, $u_B : \mathbb{R}^n \to U$ and $u_{M\to B}\mathbb{R}^n \times [T_M, T_M+\delta] \to U$ for some $\delta > 0$. We call policies of the following form *time-varying backup controllers*:

$$\pi(x,\tau) = \begin{cases} u_M(x) & \tau \le T_M \\ u_{M\to B}(x,\tau) & T_M < \tau \le T_M + \delta \\ u_B(x) & \tau > T_M + \delta \end{cases}. \tag{9}$$

In eq. (9), π is executing two controllers (u_M and u_B) in sequence and continuously transitioning between them with $u_{M\to B}$. The first controller, u_M describes a maneuver to be performed before the backup controller is engaged. The controller u_B is the backup controller that brings you to S_B. Finally, $u_{M\to B}$ is a time-varying controller that (locally Lipschitz) continuously transitions the input from $u_M(x)$ to $u_B(x)$ over time δ. The potential benefit from introducing a maneuver is illustrated in Figure 3.

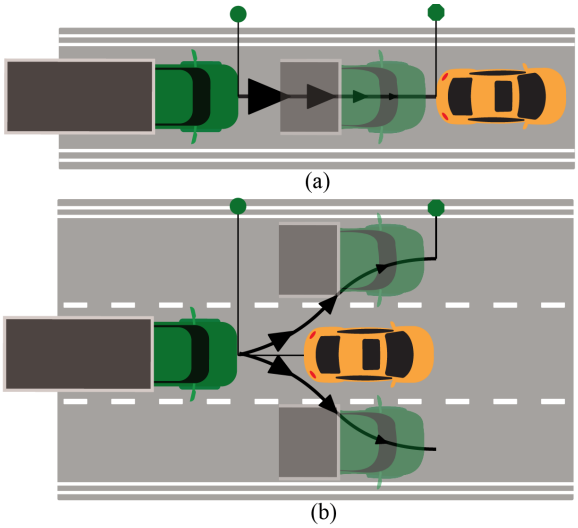


Fig. 3: Illustration of the benefits of time-varying backup controllers. (a) A high-inertia semi truck driving along the highway must keep a large distance behind lighter cars in order to stop before reaching them. (b) By adding a maneuver to switch lanes before stopping, the truck can follow much more closely, under the condition that no one is in the lane.

For a policy $\pi(x,t)$ and initial condition $x(t_0) = x_0 \in \mathbb{R}^n$, the solution to this closed-loop system is given by the flow:

$$x(t) = \Phi_\pi(x_0,t) = x_0 + \int_{t_0}^{t} (f(x) + g(x)\pi(x,\tau))\,d\tau. \tag{10}$$

In this case, for any initial time $t_0 \ge T_M$, the maneuver will not be performed. To remedy this, we introduce an alternative notion for the flow of the system with a time-varying backup controller:

$$\begin{aligned}x(t) &= \Phi_\pi(x_0,t,\tau_0) \\ &= x_0 + \int_{t_0}^{t} \left(f(x) + g(x)^\top \pi(x,\tau-\tau_0)\right) d\tau.\end{aligned} \tag{11}$$

This flow will implicitly maintain forward-invariance of the following set:

$$\begin{aligned}&S_I(\tau_0) = \\ &\left\{ x \in \mathbb{R}^n \,\middle|\, \min_{t\in[0,T]} \left\{h\left(\Phi_\pi(x,t,\tau_0)\right), h_B\left(\Phi_\pi(x,T,\tau_0)\right)\right\} \ge 0 \right\}\end{aligned} \tag{12}$$

which is equivalent to the formulation in eq. (4) where the policy π corresponds to π with the system time offset by τ_0. The choice of time-offset parameter τ_0 ultimately determines the shape S_I.

Example 1. Let us consider two examples that demonstrate how the parameter τ_0 transforms the final invariant set. First,

suppose $\tau_0 = t_0 - T_M - \delta$. In this case,

$$\begin{aligned}\Phi_\pi(x_0, t, t_0 - T_M - \delta) &= x_0 + \int_{t_0}^{t} \Big(f(x) + g(x)^\top u_B(x) \Big)\, d\tau \\ &= \Phi_{u_B}(x_0, t), \end{aligned} \tag{13}$$

which implies that $S_I(t_0 - T_M - \delta)$ recovers the safe-set of the nominal backup controller u_B.

Second, suppose $\tau_0 = t_0 + T - T_M$, which corresponds to the case of only the maneuver being performed over time T: $\Phi_\pi(x_0, T, t + T - T_M) = \Phi_{u_M}(x_0, T)$. This would likely result in $S_I = \emptyset$, as the backup maneuver $u_M(x)$ would not bring the system back into the backup set S_B.

Clearly, the choice of τ_0 is critical to the performance of the time-varying backup controller and, as demonstrated below, the right choice of τ_0 can yield a control invariant set that is larger than the one guaranteed by the nominal controller.

B. Time-Offset for Time-Varying Backup Control Filters

Rather than pick a single τ_0 for the entire evolution of our dynamical system, we will allow it to change as a function of the system time and the current state:

$$\tau_0^*(x,t) \coloneqq \min_{\tau_0 \in [-t, T-t]} \tau_0 \tag{14}$$

$$\text{s.t.} \quad \Phi_\pi(x, t+\tau, \tau_0) \in S \quad \forall \tau \in [0, T] \tag{15}$$

$$\Phi_\pi(x, t+T, \tau_0) \in S_B \tag{16}$$

Recall that in eq. (4), we give the policy a maximum time horizon of T to bring the system into the backup set. We operate under the assumption that $T > T_M + \delta$ so that the horizon can, in principle, cover both the maneuver and leave enough time for the backup set controller to take the system back to S_B. Equation (15) ensures that the chosen τ_0^* will be safe for states in a horizon of length T. The constraint in eq. (16) ensures that the final state of the horizon belongs to the backup set S_B. Notice that these two constraints, by definition, imply that $\pi(x, \tau, \tau_0)$ is a valid backup controller with time-offset τ_0^* since the constraints match the definition in eq. (3). The linear objective ensures we include as much of our maneuver behavior as possible, otherwise we could simply set $\tau_0 = t_0 - T_M - \delta$ and only execute the backup controller.

We now show that a time-offset chosen by this optimization problem results in a strict improvement over the nominal case of a single backup controller.

Theorem 1. *The safe set for a nominal backup controller $u_B(x)$ is strictly smaller than the safe set provided by a time-varying backup controller $\pi(x, \tau - \tau_0)$ with time-offset τ_0^* as defined in eq.* (14) *so that the following condition holds true:*

$$S_I(u_B) \subseteq S_I(\tau_0^*) \tag{17}$$

Proof. Suppose for the sake of contradiction that there exists an $x \in S_I(u_B)$ so that $x \notin S_I(\tau_0^*)$. Now $x \in S_I(u_B)$ implies that $\Phi_{u_B}(x,t) \in S \ \forall t \in [t_0, t_0 + T]$ and $\Phi_{u_B}(x, t_0 + T) \in S_B$ by definition. Recall that $\tau_0 = t_0 - T_M - \delta$ implies that

$$\Phi_\pi(x, t, t_0 - T_M - \delta) = \Phi_{u_B}(x, t)$$

Therefore $\tau_0 = t_0 - T_m - \delta$ is feasible for the optimization problem τ_0^*. This is a contradiction since $x \notin S(\tau_0^*)$ implies t_0^* has no feasible solution but $\tau_0 = t_0 - T_m - \delta$ is a feasible solution. □

Notice that because τ_0^* is an optimization problem over time rather than state or control input, it is amenable to the time discretization required to numerically integrate ODE's and can be efficiently approximated in practice. We introduce algorithm 1 to approximate τ_0^* online within the mixing framework described in eq. (7).

Algorithm 1 Online Approximation of τ_0^*

Inputs:
x, t ▷ Current State and System Time
Δ ▷ Discrete-Time Increment
$\tau_0^*(x(t-\Delta), t-\Delta)$ ▷ Previous τ_0^* Solution
π ▷ Time-Varying Backup Controller
procedure RESETTIMEOFFSET
  $\mathscr{P} \leftarrow \{\Phi_\pi(x, t+\tau, -t) \,|\, 0 \le \tau \le T\}$
  if $(\mathscr{P} \subseteq S) \wedge (\Phi_\pi(x, t+T, -t) \in S_B)$ **then**
    return $\tau_0^*(x(t), t) \leftarrow -t$
  else
    return $\tau_0^*(x(t), t) \leftarrow \tau_0^*(x(t-\Delta), t-\Delta) + \Delta$
  end if
end procedure

The end result of Algorithm 1 is that the time-offset τ_0 gets set to the the negation of system time whenever the system is able to perform the backup maneuver u_M for the entirety of T_M and remain safe. Otherwise τ_0^* is allowed to increase with the system time.

Remark 1. The maneuver $u_M(x)$ is chosen not to be time-varying is so that τ_0 may be reset freely before $t - \tau_0 < T_M$ without discontinuities. While there is likely to be a discontinuity in the input when reset after $t - \tau_0 > T_M$, it can only occur every T_M seconds, and the resulting backup maneuver corresponding to the new τ_0 is Lipschitz continuous. Therefore, the reset-induced discontinuities do not affect the safety guarantees, nor do they lead to high-frequency oscillations in the input.

C. Multiple backup control policies

Now that the theory is established for the time-varying backup maneuvers, we can introduce the utilization of multiple maneuvers. For a maneuver i, the time-varying backup controller takes the following form:

$$\pi^i(x, \tau) = \begin{cases} u_{M^i}(x) & \tau \le T_M \\ u_{M^i \to B}(x, \tau) & T_M < \tau \le T_M + \delta \\ u_B(x) & \tau > T_M + \delta \end{cases}. \tag{18}$$

For ease of notation, we assume that T_M is constant among the different maneuvers, but in practice, there is no need for this to be the case.

Much like the introduction of a single maneuver $u_M(x)$ resulted in a safe set S^I at least as large as, and generally

larger, than the original set, the inclusion of multiple maneuvers can further increase the size of S^I. For example, in the case of Figure 3, having multiple maneuver options could correspond to switching to different lanes, increasing the likelihood that one of them is empty.

As shown in Figure 3, naive mixing of backup maneuvers may not yield a safe control input and could cause discontinuities in the resulting control action taken. In the case of the truck, going left or going right are mutually exclusive. As a secondary consideration we also want to avoid the expensive computation of rolling out trajectories for all possible maneuvers every time we wish to generate a u_{act}.

One possible condition for switching is once it is no longer possible to perform the current maneuver and has engaged only the backup controller portion, i.e. $\tau_0 \geq t_0 - T_M - \delta$. In this case, the backup controller is already engaged, switching to a different policy $\pi^i(x,t,\tau_0)$ will not result in a discontinuity. This process can be continued for all possible backup maneuvers, until one of the maneuvers allows a reset of τ_0. If no choice of backup maneuver allows a reset, the backup controller continues executing maintaining the safety of the system. This is formalized in Algorithm 2.

Algorithm 2 Maneuver Switching Algorithm

Inputs:
x, t ▷ Current State and System Time
j ▷ Current Maneuver
Δ ▷ Discrete-Time Increment
$\tau_0^*(x(t-\Delta),t-\Delta)$ ▷ Previous τ_0^* Solution
$\{\pi^i\}_{i=1}^J$ ▷ Time-Varying Backup Controllers
procedure SWITCHMANEUVER
  if $\tau_0^*(x(t-\Delta),t-\Delta) \geq t - T_M - \Delta$ **then**
    $j \leftarrow j+1$
    if $j > J$ **then**
      $j = 1$
    end if
    $\mathscr{P}^j \leftarrow \{\Phi_{\pi^j}(x,t+\tau,-t) \,|\, 0 \leq \tau \leq T\}$
    if $\left(\mathscr{P}^j \subseteq S\right) \wedge (\Phi_{\pi^j}(x,t+T,-t) \in S_B)$ **then**
      return $\tau_0^*(x(t),t) \leftarrow -t$
    else
      return $\tau_0^*(x(t),t) \leftarrow \tau_0^*(x(t-\Delta),t-\Delta)+\Delta$
    end if
  end if
end procedure

Remark 2. Similar to Algorithm 1, this algorithm results in a discontinuity no more frequent than every T_M seconds. Moreover, only one maneuver is evaluated at each time-step, resulting in minimal computational burden.

D. Decentralized, multi-agent safety

For backup CBFs, the decentralized multi-agent case was handled in [16]. In this work, we follow a similar approach with our regulation functions along backup maneuvers.

Suppose we have a set of agents $a \in \mathscr{A}$, each with its own independent state $x^a \in \mathbb{R}^n$ and nominal safe set defined by continuously differentiable $h^a : \mathbb{R}^n \to \mathbb{R}$. We are also concerned with avoiding collisions between agents using pair-wise barrier $h^{ab} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$. Combining these two conditions we define a safe set for each agent:

$$S^a = \left\{ x_0^a \in \mathbb{R}^n \,\middle|\, \left(\bigwedge_a h^a(x^a) \geq 0 \right) \wedge \left(\bigwedge_{a \neq b} h^{(a,b)}(x^a,x^b) \geq 0 \right) \right\} \tag{19}$$

This leads to the following result on safety in the decentralized case.

Theorem 2 (Decentralized Safety)**.** *Consider a set of agents $\mathscr{A}$, with each agent $a \in \mathscr{A}$ described by* (1)*, i.e., $\dot{x}^a = f^a(x^a) + g^a(x^a)\pi^a(x)$ with each agent following a TBC π^a using independent time-offsets τ_0^{*a}. Then*

- *For all $a \in \mathscr{A}$ if the initial state $x_0^a \in S^a$ then S^a in* (19) *is safe (forward invariant) so long as each agent has state information, x^b for $b \in \mathscr{A} \setminus \{a\}$.*
- *The global safe set*

$$S = \left\{ x_0 \in \mathbb{R}^{n|\mathscr{A}|} \,\middle|\, \left(\bigwedge_a h^a(x^a) \geq 0 \right) \wedge \left(\bigwedge_{a \neq b} h^{ab}(x^a,x^b) \geq 0 \right) \right\} \tag{20}$$

is forward invariant where $x_0 \in \mathbb{R}^{n|\mathscr{A}|}$ is the concatenation of all agent states: the global state.

Proof. Recall that by theorem 1, a TBC using τ_0^{*a} as offset results in a backup controller that renders S^a safe (forward invariant) for each agent $a \in \mathscr{A}$. Clearly, if for all $a \in \mathscr{A}$ S^a is forward invariant then $\cap_{a \in \mathscr{A}}$ is forward invariant. This allows us to perform the following derivation:

$$\cup_{a \in A} S^a = \left\{ x \in \mathbb{R}^{n|\mathscr{A}|} \,\middle|\, \bigwedge_{a \in A} x^a \in S^a \right\} \tag{21}$$

$$= \left\{ x \in \mathbb{R}^{n|\mathscr{A}|} \,\middle|\, \bigwedge_{a \in \mathscr{A}} h^a(x^a) \geq 0 \wedge \forall_{b \in \mathscr{A} \setminus \{a\}} h^{ab}(x^a,x^b) \geq 0 \right\} \tag{22}$$

$$= \left\{ x_0 \in \mathbb{R}^{n|\mathscr{A}|} \,\middle|\, \left(\bigwedge_a h^{(a)}(x^{(a)}) \geq 0 \right) \wedge \left(\bigwedge_{a \neq b} h^{(ab)}(x^{(a)},x^{(b)}) \geq 0 \right) \right\} \tag{23}$$

$$= S \tag{24}$$

Where eq. (22) follows from the application of the definition of S^a and eq. (23) follows from the application of associativity and commutativity the logic and operator ($\wedge$). □

If multiple agents have multiple maneuvers, the current maneuver must be known by all agents. As a consequence of $u_B^i(x)$ being constant in time throughout all maneuvers for an agent i, the drones are able to independently update their maneuvers at any time $\tau_0 \geq t - T_M - \delta$. However, all drones must have knowledge of the new maneuver being utilized, in order to know whether or not they can reset τ_0.

Figure 4 demonstrates the idea behind requiring knowledge of the other agents' maneuvers to guarantee safety of the individual.

IV. MODELING AND IMPLEMENTATION

A. Drone modeling and control

By utilizing the high-performance flight controllers on modern racing drones, capable of tracking angular rates at

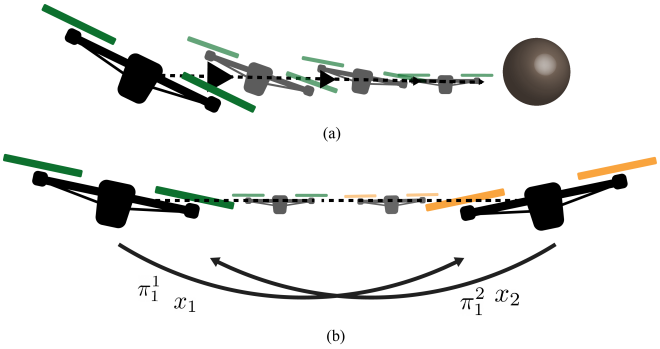


Fig. 4: Illustration contrasting single agent and multi-agent implementation. (a) shows the single agent case, while (b) shows the multi-agent case where state and backup policy information is required to guarantee safety.

frequencies upwards of 8 kHz, we are able to abstract away the motor and propeller dynamics, and treat the control system as a rigid body in $\mathbb{R}^3$ that attempts to track angular rates. Note that we do not assume perfect angular rate tracking, meaning that this model is still able to capture the dynamics introduced by the motors, propellers, and delays, albeit in a simpler form.

As in the preceding work [17], the abstracted system of the drone and lower-level flight controller 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

$$x = [p_w, q, v_w, \omega_b]^T, \tag{25}$$

where $p_w = [p_x, p_y, p_z]^T$ is the drone position in the world frame, $v_w = [v_x, v_y, v_z]^T$ is the velocity vector in the world frame, q is the the orientation with respect to the world frame represented by a quaternion, and $\omega_b = [\omega_x, \omega_y, \omega_z]^T$ is the angular velocity vector in the body frame.

The angular velocity vector attempts to capture the underlying dynamics via the function

$$\dot{\omega} = C(x)(\omega_{\text{des}} - \omega), \tag{26}$$

where $C(x) : \mathbb{R}^n \to \mathbb{R}_+$ governs the response time of the tracking controller.

The throttle-to-thrust curve was fit via a second-order polynomial, using accelerometer and motion capture data. An integral term in the altitude controller is used to eliminate any error in this modeling.

B. Safe sets, backup controller, and regulation function

For this work, the nominal safe set of each agent i is chosen as a box of free space centered at $(x_c, y_c, z_c) \in \mathbb{R}^3$ with side lengths r_x, r_y, r_z,

$$h^i(x) = \min\left\{r_x^2 - (p_x^i - x_c)^2, r_y^2 - (p_y^i - y_c)^2, r_z^2 - (p_z^i - z_c)^2\right\}. \tag{27}$$

The safety constraint between agents i and j is given as

$$h^{ij}(x) = (p_x^i - p_x^j)^2 + (p_y^i - p_y^j)^2 + (p_z^i - p_z^j)^2 - 4r^2, \tag{28}$$

which ensures that no collision occurs between the drones, which are modeled as spheres of radius r.

As shown in Section III, the new time-varying backup controllers consist of a maneuver $u_M(x)$ to be performed for time T_M, a backup controller $u_B(x)$ to be used after time $T_M + \delta$, and a smoothing controller $u_{M\to B}(x,t)$ that switches inputs from the two over time δ.

The backup controller $u_B(x)$ is the same velocity controller on SE(3), inspired by [21], that was used in [17]. The backup controller attempts to stop the drone, and repel it from the boundary of the safe set if it gets too close.

The backup set S_B is defined by the function

$$h_B = -\sqrt{v_x^2 + v_y^2 + v_z^2} + 0.1, \tag{29}$$

which guarantees that the drone is able to slow itself to a speed of 0.1 m/s, which is forward invariant under the backup controller.

C. Backup Maneuvers

In this section, we will outline two simple maneuvers, and demonstrate how they improve performance on hardware. We call these maneuvers the "carry on" and "evade" maneuvers.

Carry on maneuver. The carry on maneuver is exactly how it sounds: the maneuver attempts to propagate the current desired input forward. This input is held constant throughout T_M, and is updated every time τ_0 is allowed to reset. With the carry on maneuver, the overall policy is defined as

$$\rho^i(x,\tau) = \begin{cases} u_{\text{des}} & \tau \le T_M \\ u_{M^i\to B}(x,\tau) & T_M < \tau \le T_M + \delta \\ u_B(x) & \tau > T_M + \delta \end{cases}. \tag{30}$$

This policy shines when a skilled pilot wants to move along the edge of the safe set, when one might have $\lambda << 1$. This is illustrated in Figure 5a.

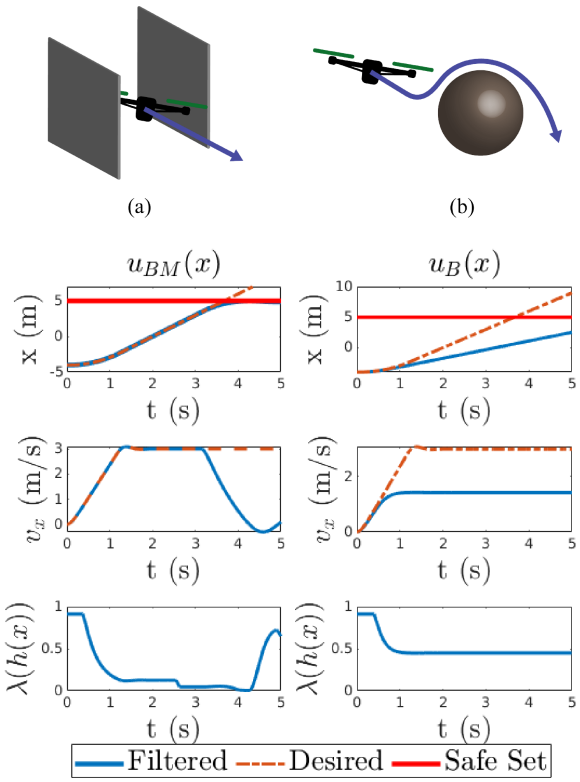
Evade maneuver. The evade maneuver is a maneuver that attempts to reposition the drone before coming to a stop. This is similar to a lane change, as in Figure 3. For this specific maneuver, we attempt to bring the drone upwards, but similar results could be achieved with evading in different directions. The policy is written as

$$\rho^i(x,\tau) = \begin{cases} u_z(x) & \tau \le T_M \\ u_{M^i\to B}(x,\tau) & T_M < \tau \le T_M + \delta \\ u_B(x) & \tau > T_M + \delta \end{cases}, \tag{31}$$

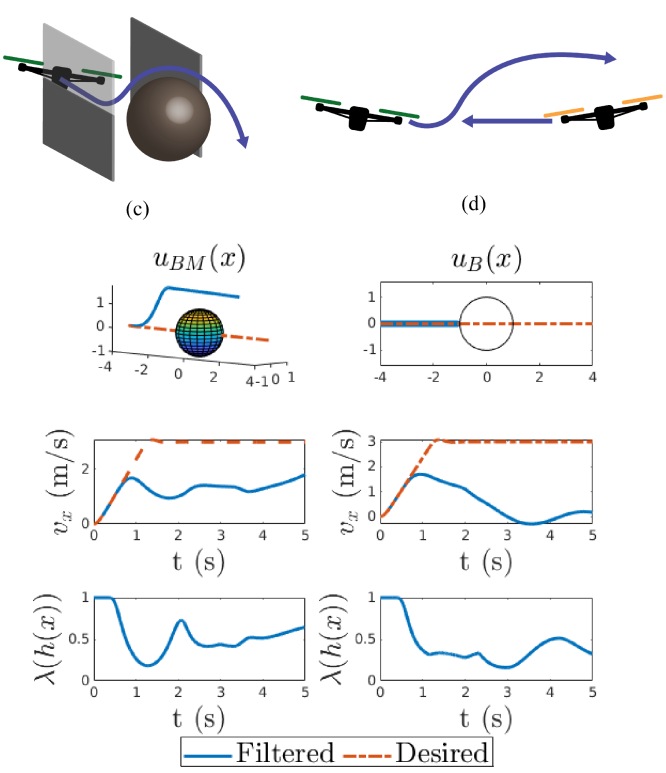
with $u_z(x)$ being the velocity controller attempting to track to the desired height. This policy shines when the pilot does not react to avoid an obstacle fast enough, and relies on the safety filter to do it for them. This is illustrated in Figure 5b.

D. Switching backup policies

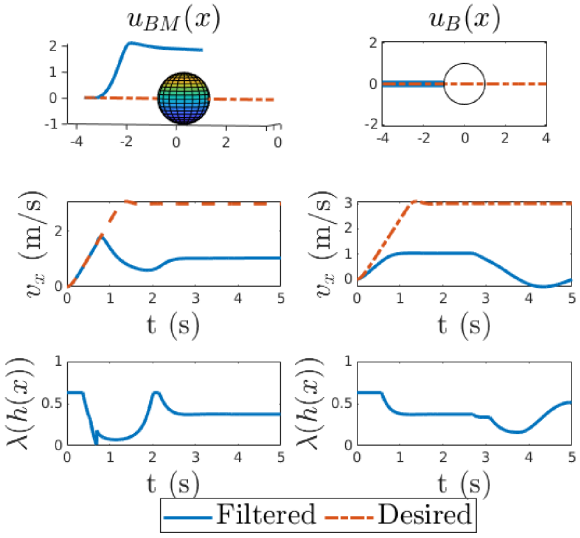
By implementing multiple backup policies that the agents can switch between, you can can further increase performance by expanding the size of S_I. By combining the carry on and evade maneuvers, and allowing free switching between them in the event that $t \ge T_M$ for the backup maneuver, we achieve the result obtained in Figure 5c.



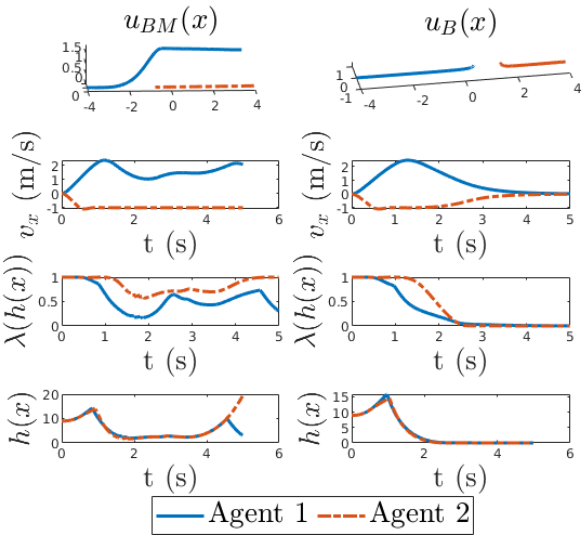
(a) Carry on maneuver compared to only $u_B(x)$ in tightly constrained space.



(b) Evade maneuver compared to only $u_B(x)$ when encountering a spherical obstacle.



(c) Carry on maneuver switching to evade maneuver when encountering a spherical obstacle in tightly constrained space



(d) Multi-agent sim

Fig. 5: Simulation results capturing the performance benefits of allowing backup maneuvers. While the value of λ is often lower, due to being closer or faster near the barrier, there is significantly better alignment with the desired velocities.

E. Multi-agent backup policies

Finally, the decentralized, multi-agent approach was tested for two agents moving towards each other. Agent 1 is utilizing with the evade maneuver, while Agent 2 attempts the carry-on maneuver. The desired velocity v_x of Agent 1 is 3 m/s, and for Agent 2, it is -1 m/s.

Figure 5d demonstrates the advantage over simply utilizing the backup controller $u_B(x)$ in preventing the drones from reaching a deadlock.

V. Hardware Results

A. Hardware setup

The quadrotor used in our experiments is a consumer "Cinewhoop" drone shown in Figure 6. We add several

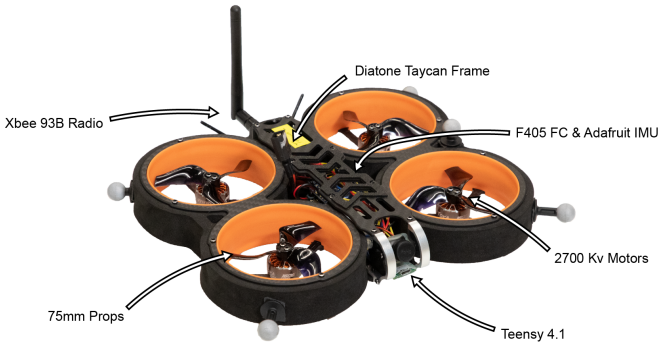


Fig. 6: The Cinewhoop quadrotor used in the experiments

components to this off the shelf drone for the computation of the barrier functions: a Teensy 4.1 microcontroller, a Xbee 93b radio and an Adafruit BNO055 IMU. Optitrack position and attitude data is streamed to the drone at 20 Hz via a Xbee93B radio. The Teensy 4.1 reads this position data along with IMU data at 100 Hz. The on-board barrier computation issues angular rates commands at 100 Hz to a flight controller running Betaflight, an open source flight control software which is tightly integrated with the flight controller. This allows angular rate tracking at the internal IMU update frequency of 8 kHz.

B. Hardware results

To validate the tractability and robustness of our algorithms in flight, we recreate the simulation results showcased in Figures 5b and 5d with two identical "Cinewhoop" drones. The results are demonstrated in Figure 7.

The only major difference between the simulation and the hardware was the choice of the evade maneuver to move to the side rather than up. This was done due to limit the effects of the downward airflow on the other agent. As demonstrated in the plots, the TBC's ran quite well despite the very noisy velocity data, as well as the computational constraints of a microcontroller. However, the CBF and all other onboard computations were easily ran at the update rate of the Betaflight's desired inputs, 100 Hz.

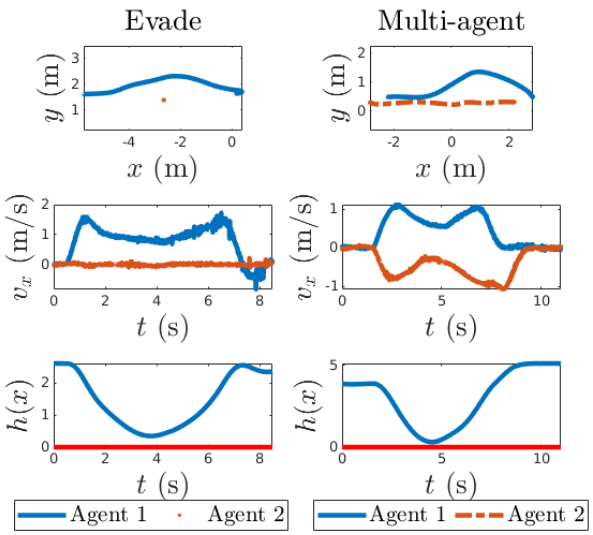


Fig. 7: Plots of the hardware results showcased in Figure 1. For more hardware results, please refer to the attached video.

REFERENCES

[1] 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.

[2] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 45–52.

[3] K. Sreenath, T. Lee, and V. Kumar, "Geometric control and differential flatness of a quadrotor uav with a cable-suspended load," in *52nd IEEE Conference on Decision and Control*. IEEE, 2013, pp. 2269–2274.

[4] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[5] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 620–626, 2017.

[6] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," *Proceedings of Robotics: Science and Systems XVI*, 2020.

[7] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[8] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *arXiv preprint arXiv:2109.01365*, 2021.

[9] A. Broad, T. Murphey, and B. Argall, "Highly parallelized data-driven mpc for minimal intervention shared control," *arXiv preprint arXiv:1906.02318*, 2019.

[10] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, "A predictive safety filter for learning-based racing control," *arXiv preprint arXiv:2102.11907*, 2021.

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

[12] G. Wu and K. Sreenath, "Safety-critical control of a 3d quadrotor with range-limited sensing," in *Dynamic Systems and Control Conference*, vol. 50695. American Society of Mechanical Engineers, 2016, p. V001T05A006.

[13] M. Khan, M. Zafar, and A. Chatterjee, "Barrier functions in cascaded controller: Safe quadrotor control," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 1737–1742.

[14] B. Xu and K. Sreenath, "Safe teleoperation of dynamic uavs through control barrier functions," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7848–7855.

[15] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2460–2465.

[16] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, 2020.

[17] A. Singletary, A. Swann, Y. Chen, and A. Ames, "Onboard safety guarantees for racing drones: High-speed geofencing with control barrier functions," *IEEE Robotics and Automation Letters*, 2022.

[18] Y. Chen, M. Jankovic, M. Santillo, and A. D. Ames, "Backup control barrier functions: Formulation and comparative study," *arXiv preprint arXiv:2104.11332*, 2021.

[19] T. Gurriet, M. Mote, A. D. Ames, and E. Feron, "An online approach to active set invariance," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 3592–3599.

[20] A. Singletary, T. Gurriet, P. Nilsson, and A. D. Ames, "Safety-critical rapid aerial exploration of unknown environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 270–10 276.

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