Prescribed-Time Control Barrier Functions for Semiautonomous Navigation

Carmen Jimenez Cortes* and Samuel Coogan †
Georgia Institute of Technology, Atlanta, GA, 30332

This paper proposes a novel use of Control Barrier Functions (CBFs) to enforce Prescribed-Time Safety in a semiautonomous navigation scenario in which an aerial vehicle navigates through a sequence of waypoints. In particular, we use Prescribed-Time Control Barrier Functions (PT-CBFs) to ensure a minimum traversal time between when the vehicle approaches the vicinity of a waypoint and passes through the waypoint itself. Motivating applications are those in which onboard personnel are required to, e.g., provide visual confirmation of the waypoint availability. PT-CBFs are guaranteed to achieve the prescribed minimum waypoint traversal time, and, as demonstrated via simulation, they also allow for faster mission completion than a simple strategy that activates a traditional CBF for a specified duration.

I. Nomenclature

T_{safe}	=	prescribed safe time
T_{marg}	=	time margin around T_{safe}
t_0	=	initial time
$\nu_m(t)$	=	m power of the function $\nu(t)$
$\mu_m(t)$	=	*blow-up* function: inverse of the m power of the function ν
ψ_{rj}	=	jth waypoint rth order high-order CBF
$d^j(y)$	=	jth waypoint proximity function to point y
D^j_{crit}	=	critical proximity
D^j_{thresh}	=	threshold proximity

II. Introduction

An automated future is a shared milestone across many engineering fields such as driving [1, 2] manufacturing [3, 4], and aviation [5]. Before an autonomous world can become a reality, a more immediate scenario will consist of a collaboration between humans and partially autonomous systems to achieve common objectives. We denote these systems, in which one or more partially autonomous agents interact with a human collaborator or supervisor, as semiautonomous systems. Ideally, under nominal conditions, the system should be able to operate independently without any human input. However, no task or mission is ever completely nominal, and it is in such unpredictable situations when the human has to take action and rectify or approve commands generated by the system's autonomous capabilities. In such scenarios, and especially for dynamical systems operating in complex environments, safety must be guaranteed while the system awaits action from the human. For example, consider a semiautonomous navigation scenario in which a partially autonomous aerial vehicle, with onboard personnel, tracks a set of ordered waypoints at a given nominal speed, as depicted in Figure 1. At each waypoint, after crossing an outer threshold proximity, a minimum traversal time is required before the vehicle is allowed to cross an interior critical proximity nearer to the waypoint. This time gives the onboard personnel the chance to verify the waypoint availability, its visibility conditions, or update the route if required by the mission. This particular scenario constitutes the main focus of the paper, and will be further discussed in the case study. Control Barrier Functions (CBFs) [6–8] are a popular tool for safety control applications, as they guarantee forward invariance of a safe set. However, for the previous example, the system does not need to remain inside the initial safe set indefinitely. The paper [9] introduces the concept of Prescribed-Time Safety (PTSf), a new approach to CBFs for systems that only need to remain inside the safe set for a prescribed period of time. Inspired by this PTSf

*Ph.D. Student, School of Electrical and Computer Engineering, AIAA Student Member
†Associate Professor, School of Electrical and Computer Engineering and School of Civil Engineering

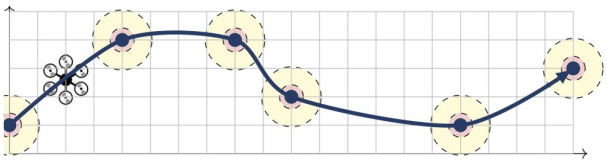


Fig. 1 Semiautonomous navigation problem set-up. Threshold proximities of waypoints could overlap but not their critical proximities, as it would mean that two waypoints have the same location.

technique, we now provide the mathematical underpinnings to guarantee the success of a semiautonomous navigation mission using CBFs that are only active for a prescribed time. We name these CBFs as Prescribed-Time Control Barrier Functions (PT-CBFs). Hence, the focus of this paper is on the theoretical foundations for a prescribed-time safety approach to semiautonomous control. These PT-CBFs were tested against two alternative control strategies: one that activates and deactivates a High-Order CBF (HO-CBF), and one that uses the nominal controller without any safety filters. The trajectory generated by the HO-CBF is shown to result in a slower time to complete the mission than the PT-CBFs, and the trajectory generated from the nominal controller is shown to be unsafe. Therefore, one can conclude that PT-CBFs are the best suited strategy to address this semiautonomous control problem. We also present the results of using PT-CBFs in a Microsoft Flight Simulator testing environment that we have developed and that will enable testing with humans in the control loop in future research.

The rest of this paper is structured as follows: Section III includes an overview of Control Barrier Functions and High-Order Control Barrier Functions in Subsection III.A, as well as the mathematical derivation of the PT-CBFs in Subsection III.B. The semiautonomous system navigation problem is defined in Section IV. Section V includes a sample case study with its simulated results, and an example of PT-CBFs running in our Microsoft Flight Simulator testing platform. Finally, Section VI summarizes the main findings of this research.

III. Review of Control Barrier Functions

A. Foundations of High-Order Control Barrier Functions

Control Barrier Functions [10] are a mathematical tool used to generate controllers that render a given safe set $C = \{x : h(x) \geq 0\}$ forward invariant for the control-affine system

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

where $x \in \mathbb{R}^n$ is the system state and $u \in \mathbb{R}^m$ is the control input. The set C is *forward invariant* if, for any initial condition within the set C, the system will remain inside C for all time $t \geq 0$ [11]. The function $h(x)$ used in the definition of C is a *Control Barrier Function (CBF)* for the system if it is continuously differentiable and if there exists a locally Lipschitz function $\alpha : \mathbb{R} \to \mathbb{R}$ satisfying $\alpha(0) = 0$, so that

$$\sup_{u \in \mathbb{R}^m} L_f h(x) + L_g h(x) u \geq -\alpha(h(x)) \tag{2}$$

where $L_f h(x) = \nabla h(x)^T f(x)$ and $L_g h(x) = \nabla h(x)^T g(x)$ are the *Lie derivatives* of $h(x)$ with respect to $f(x)$ and $g(x)$. Defining $U(x) = \{u \mid L_f h(x) + L_g h(x) u \geq -\alpha(h(x))\}$, choosing any Lipschitz continuous feedback control strategy $u(x)$ such that $u(x) \in U(x)$ for all x ensures that C is forward invariant. The resulting control law is called a *safe controller*. Given a smooth nominal feedback control strategy $\hat{u}(x)$, a common approach is to obtain a safe control strategy by choosing $u(x)$ to minimize $\|u(x) - \hat{u}(x)\|$ subject to $u(x) \in U(x)$. Equation (2) includes the directional derivative $L_g h(x)$, which could be equal to zero. In that case, the control input u no longer intervenes in the equation. A possible solution is to use High-Order Control Barrier Functions. To do so, a new function ψ is defined as $\psi(x) = L_f h(x) + \alpha(h(x))$. If there exists a locally Lipschitz function α_2 that satisfies $\sup_{u \in \mathbb{R}^m} L_f \psi(x) + L_g \psi(x) u \geq -\alpha_2(\psi(x))$ then ψ is a CBF, and we call it a HO-CBF to emphasize its construction from higher order derivatives of h. The first time derivative of the new HO-CBF candidate ψ is given by

$$\dot{\psi}(x) = L_f^2 h(x) + L_g L_f h(x) u + \dot{\alpha}(h(x)). \tag{3}$$

Continuing this process if needed, for a system of uniform relative degree r from u to $h(x)$, the recursion of HO-CBFs is built as follows [12]:

$$\begin{aligned}\psi_1(x) &= h(x)\\ \psi_i(x) &= L_f\psi_{i-1}(x) + \alpha_{i-1}(\psi_{i-1}(x)) \quad \forall i = 2, ..., r\end{aligned} \tag{4}$$

And the inequality constraint (2) is

$$\sup_{u\in\mathbb{R}^m} L_f\psi_r(x) + L_g\psi_r(x)u \geq -\alpha_r(\psi_r(x)). \tag{5}$$

Note that for a system with well-defined relative degree r, the process is repeated $r-1$ times to obtain a well-defined HO-CBF and $\dot{\psi}_r(x,u)$ is the only explicit function of the control input u.

B. Prescribed-Time Safety with Control Barrier Functions

The focus of this paper is on applications where the system generally does not need to remain inside C for an indefinite period of time. That is, there exists a time, denoted by T_{safe}, that expresses the time from which the system is allowed to exit the original safe set in order to continue with its mission. One immediate option is to deactivate or eliminate the CBF once T_{safe} expires. Instead, this paper proposes an alternative approach using *Prescribed-Time CBFs*, which are CBFs that possess the prescribed-time safety property first introduced in [9]. PT-CBFs use auxiliary *blow-up* functions in their definition that effectively make the right hand side of (2) increasingly negative so that the inequality is easier to satisfy as time progresses. This allows the safe controller to match a nominal controller $\hat{u}$ more closely, especially as time approaches T_{safe}, and enabling a smooth switch when deactivating the barrier.

We also introduce an updated version of blow-up functions defined in [9]. With this new formulation μ_m does not take effect until t is "close enough" to T_{safe}. Only when t enters a predefined margin T_{marg} around T_{safe}, μ_m increases its value. Otherwise, without the use of T_{marg}, the constraint in (2) will be increasingly relaxed as time progresses, leading to greater disparity between the CBF and the PT-CBF, allowing more aggressive control inputs, and thus compromising safety. These new blow-up functions are defined as

$$\nu(t) = \begin{cases} 1 & 0 \leq t - t_0 \leq T_{\text{safe}} - T_{\text{marg}} \\ \frac{T_{\text{safe}}}{T_{\text{marg}}} - \frac{t-t_0}{T_{\text{marg}}} & T_{\text{safe}} - T_{\text{marg}} < t - t_0 \leq T_{\text{safe}} \end{cases} \tag{6}$$

$$\mu_m(t) = \frac{1}{\nu^m(t)} \qquad \forall\, t - t_0 \geq 0 \tag{7}$$

where t is the current time, t_0 is the time when the PT-CBF is triggered, T_{safe} is the time since t_0 that the system needs to remain inside the safe region, and T_{marg} is the time before T_{safe} when μ_m will start to increase its value. PT-CBFs are constructed in an analogous way to HO-CBFs using the auxiliary blow-up functions. For a control-affine system (1) with uniform relative degree r from u to $h(x)$, define for all $i = 1, ..., r$

$$\begin{aligned}&\psi_1(x) = h(x)\\ &\dot{\psi}_1(x) = L_f h(x)\\ &\psi_i(x,t) = \dot{\psi}_{i-1}(x,t) + \alpha_{i-1}(\psi_{i-1}(x,t)) \quad \forall i \geq 2 \qquad (8)\\ &\dot{\psi}_i(x,t) = \ddot{\psi}_{i-1}(x,t) + c_{i-1}\left[2\mu_2(t)\frac{\mu_1(t)}{T_{\text{marg}}}\psi_{i-1}(x,t) + \mu_2(t)\dot{\psi}_{i-1}(x,t)\right] \quad \forall i \geq 2 \qquad (9)\\ &\alpha_i(\psi_i) = \mu_2(t)c_i\psi_i(x,t)\,, \quad c_i \in \mathbb{R}_{>0}. \qquad (10)\end{aligned}$$

Note that similarly to HO-CBFs, for a system with well-defined relative degree r, $\dot{\psi}_r(x,t,u)$ is the only explicit function of the control input u.

Remark 1 *The first time derivative of $\mu_2(t)$ can also be expressed as*

$$\dot{\mu}_2(t) = 2\frac{T_{marg}^2}{(T_{safe} - t + t_0)^3} = 2\mu_2(t)\frac{\mu_1(t)}{T_{marg}}. \tag{11}$$

The new problem statement to implement PT-CBFs is as follows: given a control-affine system (1) and a continuously differentiable function $y = h(x)$, find a safe controller u that modifies the nominal controller $\hat{u}$ in a minimally invasive manner during a fixed time T_{safe}:

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & ||u - \hat{u}||^2 \\ s.t. \quad & \dot{\psi}_r \geq -\alpha_r(\psi_r) \end{aligned} \tag{12}$$

According to [9, Thm. 1], using the safe controller u instead of the nominal one $\hat{u}$ guarantees that whenever the execution time t is lower than T_{safe}, the system will stay within the safe region $C = \{x : h(x) \geq 0\}$, while progressing towards its goal. This result is formalized in Proposition III.1.

Proposition III.1 *For a system $\dot{x} = f(x) + g(x)u$ that is initially safe ($h \geq 0$ at t_0), the control law that solves* (12) *guarantees that $h \geq 0$ for all time $t \in [t_0, t_0 + T_{safe})$ for sufficiently large constants $c_1, ..., c_r$ in* (10)*.*

IV. Semiautonomous System Navigation

Consider an autonomous aerial vehicle with dynamics

$$\dot{x} = f(x) + g(x)u \tag{13}$$

$$y = o(x) \tag{14}$$

for state $x \in \mathbb{R}^n$ and input $u \in \mathbb{R}^{m_i}$ along with output $y \in \mathbb{R}^{m_o}$. The system is assumed to have a well-defined relative degree $r \geq 1$ from input u to output $o(x)$ uniform in x. The mission objective is for the resulting output trajectory $y(t)$ to pass through a given sequence of waypoints $\{z^1, z^2, \ldots, z^N\}$ defined by $z^j \in \mathbb{R}^{m_o}$ for each j. Assume there exists a collection $\hat{u}^j(x)$ of nominal feedback controllers for each waypoint j that drives the system output to the waypoint. Thus, a nominal strategy is to apply in succession $\hat{u}^1(x)$ until waypoint z^1 is reached, then $\hat{u}^2(x)$ until waypoint z^2 is reached, etc. However, consider the following safety constraint that generally prevents direct application of this nominal strategy: assume that each waypoint also possesses a *proximity function* $d^j(y)$ characterizing a distance between the point y and the waypoint z^j. For example, a choice for d^j could be $d^j(y) = \|y - z^j\|_2^2$, although the proposed method accommodates more general proximity functions, as demonstrated in the case study. Further suppose there exists a threshold distance D^j_{thresh} and a critical distance $D^j_{\text{crit}} < D^j_{\text{thresh}}$ for each waypoint j. Then impose the following safety constraints:

S1) If waypoint j is the current target waypoint, and the vehicle output crosses the threshold proximity at time t_0, then the output cannot cross the critical proximity until time $t_0 + T_{\text{safe}}$. Defining $t^* = t - t_0$ such that t_0 is the time instant when $d_j(y(t_0)) = D^j_{\text{thresh}}$, then $d^j(y(t^*)) \geq D^j_{\text{crit}}$ for all $t^* \leq T_{\text{safe}}$.

S2) Moreover, the distance to all other nontarget waypoints should not be less than the waypoint's critical proximity i.e., $d^{j'}(y(t)) \geq D^{j'}_{\text{crit}}$ for all $j' \neq j$ for all t.

The purpose of imposing constraint S1 is to provide sufficient time for a human supervisor to override the tracking controller in the event that a waypoint is compromised, and the assumption is that the human is only able to determine the status of the waypoint when within the threshold proximity. The proposed solution is to use HO-CBFs to ensure constraint S2 and PT-CBFs to ensure constraint S1 as shown in Algorithm 1. To implement these safety filters in a minimally invasive manner, the following optimization problem needs to be solved at each time t:

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & ||u - \hat{u}||^2 \\ s.t. \quad & \dot{\psi}_{rj} \geq -\alpha_{rj}(\psi_{rj}) \quad \forall j = 1, ..., N \end{aligned} \tag{15}$$

where each ψ_{rj} is either an HO-CBF or a PT-CBF, depending on the target waypoint at time t. This approach is formalized in Algorithm 1 and in Theorem IV.1. Algorithm 1 explains how to update the different safety filters and nominal controller $\hat{u}^j$, depending on which waypoint z^j is the target and if the system has crossed its D^j_{thresh}. Theorem IV.1 establishes that safety is guaranteed. Note that for clarity purposes, Algorithm 1 omits the code to initialize and update the system's state but addresses where in the code it would take place with text.

Theorem IV.1 *If the quadratic program* (15) *is feasible for all time t when the constraint set is defined as in Algorithm 1, then safety is always guaranteed.*

Proof 1 *Safety guarantees follow from [10, Thm. 2] when using CBFs, from [12, Thm. 1] when using HO-CBFs, and from [9, Thm. 1] when using PT-CBFs.* ■

Note that when the waypoints are sufficiently apart, no more than one constraint will be active at a time and therefore the quadratic program (15) will always be feasible.

V. Case Study

We now demonstrate the performance benefits of PT-CBFs in a simulated semiautonomous navigation scenario. The aerial vehicle is modeled as a second-order unicycle and its mission is to track a set of N ordered waypoints. The simulation results show how PT-CBFs satisfy all safety constraints while having an improved performance, measured as total mission duration, compared to an alternative strategy that only uses HO-CBFs.

Algorithm 1 Semiautonomous Navigation

System Initialization
$j = 1$ ▷ Initialize waypoint list index
$j' = 1$ ▷ Initialize target list index. First target is z^1
$\hat{u} = \hat{U}(j')$ ▷ First nominal controller is $\hat{u}^1$
$cstr = \{\}$ ▷ Constraint set starts empty
$t = t_0 = 0$
while true **do**
  $t = t + \Delta t$
  $t^* = t - t_0$
  for $j = 1:1:N$ **do**
    if $j \neq j'$ **then** ▷ j is not target waypoint
      $cstr = cstr \cup \{\psi_{rj} = \dot{\psi}_{r-1j} + c_{rj}\psi_{r-1j}\}$ ▷ Use HO-CBF
    else ▷ j is target waypoint
      if $d^j(y(t^*)) > D^j_{\text{thresh}}$ **then** ▷ System outside threshold. Use HO-CBF
        $cstr = cstr \cup \{\psi_{rj} = \dot{\psi}_{r-1j} + c_{rj}\psi_{r-1j}\}$
      else ▷ System inside threshold
        if $t_0 == 0$ **then** ▷ First time crossing D^j_{thresh}
          $t_0 = t$ ▷ Save D^j_{thresh} crossing time
          $t^* = t - t_0$ ▷ Update t^*
        end if
        if $t^* < T_{\text{safe}}$ **then** ▷ Use PT-CBF
          $cstr = cstr \cup \{\psi_{rj} = \dot{\psi}_{r-1j} + \mu_2 c_{rj}\psi_{r-1j}\}$
        else ▷ $d^j(y(t^*)) \leq D^j_{\text{crit}}$ allowed
          $cstr = cstr \cup \{\psi_{rj} = 0\}$ ▷ Remove constraint
        end if
      end if
    end if
  end for
  $u = min(||u - \hat{u}||^2 \quad s.t. \quad cstr)$ ▷ Solve QP
  Update System
  if $d^{j'}(y(t^*)) == 0$ **then** ▷ Target waypoint reached
    $\psi_{rj'} = \dot{\psi}_{r-1j'} + c_{rj'}\psi_{r-1j'}$ ▷ HO-CBF active again
    $j' += 1$ ▷ Update j'
    $\hat{u} = \hat{U}(j')$ ▷ Update $\hat{u}$
    $t_0 = 0$ ▷ Reset D^j_{thresh} crossing time
  end if
end while

A. Autonomous System Model: Second-Order Unicycle

Rotary-wing aircraft, as well as any other driftless differential drive systems, are often modeled as unicycles [8, 13, 14]. The most common unicycle model uses the system's position and its heading angle as state variables. Its available control inputs are the translational velocity or speed, and the angular velocity. In a more realistic navigation scenario, the inputs are the translational and angular accelerations of the system, as any force applied relates to the system's accelerations through its mass and moment of inertia. This motivates the use of a higher-order model, known as the second-order unicycle [15]. The second-order unicycle is a five-dimensional, nonholonomic, non-linear system with dynamics

$$(\dot{x}_1, \dot{x}_2) = (s\cos\theta, s\sin\theta) \qquad \dot{s} = u_a \qquad \dot{\theta} = \omega \qquad \dot{\omega} = u_\alpha \tag{16}$$

where $u = (u_a, u_\alpha)$ is the two dimensional control input vector, and the state vector $x = (x_1, x_2, s, \theta, \omega)$ consists of the following variables:

- (x_1, x_2): Unicycle center of mass position
- s: Unicycle translational velocity
- θ: Unicycle heading angle. Counter-clockwise rotations are considered positive
- ω: Unicycle angular velocity. Its sign criteria is the same as for the angle θ.

A common approach to set point stabilization or position tracking uses feedback linearization, but the second-order unicycle model is not feedback linearizable, as the requisite matrix of Lie derivatives is rank deficient. The problem of having a non-feedback linearizable system can be solved by controlling a displaced point instead of the center of mass of the unicycle [16]. This displaced point has coordinates $(y_1, y_2) = (x_1 + p\cos\theta, x_2 + p\sin\theta)$, and its distance to the system's center of mass p does not need to be large. By using the displaced point (y_1, y_2) instead of the center of mass of the system, the requisite matrix of Lie derivatives has now full rank. Applying the state feedback

$$\begin{aligned} \sigma_1 &= u_a\cos\theta - u_\alpha p\sin\theta - \omega s\sin\theta - \omega^2 p\cos\theta \\ \sigma_2 &= u_a\sin\theta + u_\alpha p\cos\theta + \omega s\cos\theta - \omega^2 p\sin\theta \end{aligned} \tag{17}$$

the second order unicycle displaced point dynamics are

$$\begin{aligned} &(\dot{y}_1, \dot{y}_2) = (v_{y_1}, v_{y_2}) \\ &(\dot{v}_{y_1}, \dot{v}_{y_2}) = (\sigma_1, \sigma_2) \\ &\dot{s} = -\omega^2 p + \sigma_1\cos\theta + \sigma_2\sin\theta \\ &\dot{\theta} = \omega \\ &\dot{\omega} = \frac{\sigma_2\cos\theta + \sigma_1\sin\theta + \omega s}{p} \end{aligned} \tag{18}$$

where $\sigma = (\sigma_1, \sigma_2)$ is now the new virtual nominal control input to be chosen.

B. Safety Filters

For this case study, the waypoints' proximity function $d^j(y)$ was characterized using Lamé curves [17]:

$$\left|\frac{y_1 - z_1^j}{r_{sj}}\right|^n + \left|\frac{y_2 - z_2^j}{r_{sj}}\right|^n = 1 \quad \forall j \in N \tag{19}$$

r_{sj} in (19) corresponds to the critical proximity to the jth waypoint D^j_{crit}, y_1 and y_2 are the coordinates of the displaced point, and z_1^j and z_2^j are the coordinates of the jth waypoint. The motivation behind using superellipses is to show how PT-CBFs can accommodate different shaped safe sets, which depend on the degree n. The resulting first-order CBFs and their derivative for all waypoints $j = 1, ..., N$ are

$$h_j = \psi_{1j} = (y_1 - z_1^j)^n + (y_2 - z_2^j)^n - r_{sj} \tag{20}$$

$$\dot{h}_j = \dot{\psi}_{1j} = n(y_1 - z_1^j)^{n-1} v_{y_1} + n(y_2 - z_2^j)^{n-1} v_{y_2} \tag{21}$$

The second order unicycle model from (18) requires a HO-CBF, as it has a relative degree of 2. The expressions for ψ and $\dot{\psi}$ used in (15) depend on the PT-CBFs being active or not, as shown in Algorithm 1. If the system has not entered the proximity threshold D^j_{thresh}, the safety filter applied uses HO-CBFs given by

$$\psi_{2j} = \dot{\psi}_{1j} + c_{1j}\psi_{1j} \tag{22}$$

$$\dot{\psi}_{2j} = n(n-1)(y_1 - z_1^j)^{n-2}v_{y_1}^2 + n(y_1 - z_1^j)^{n-1}\sigma_1 + n(n-1)(y_2 - z_2^j)^{n-2}v_{y_2}^2 + n(y_2 - z_2^j)^{n-1}\sigma_2 \tag{23}$$

Once the system enters D_{thresh}, the PT-CBFs are triggered:

$$\psi_{2j} = \dot{\psi}_{1j} + \mu_2 c_{1j}\psi_{1j} \tag{24}$$

$$\dot{\psi}_{2j} = n(n-1)(y_1 - z_1^j)^{n-2}v_{y_1}^2 + n(y_1 - z_1^j)^{n-1}\sigma_1 + n(n-1)(y_2 - z_2^j)^{n-2}v_{y_2}^2 + n(y_2 - z_2^j)^{n-1}\sigma_2 + c_{1j}(2\mu_2\frac{\mu_1}{T_{\text{marg}}}\psi_{1j} + \mu_2\dot{\psi}_{1j}) \tag{25}$$

C. Simulation Results

Simulation experiments were conducted using a set of three waypoints ($N = 3$). Two of these waypoints use (19) with $n = 2$, and the third uses $n = 4$. All of them have threshold and critical proximity radii equal to 10 and 1; respectively. The system is initialized with initial position $(x_{1_0}, x_{2_0}) = (30, -5)$, heading angle $\theta_0 = 0$, angular velocity $\omega = 0$, and speed $s_0 = -1.5$. The distance between the center of mass of the unicycle and the displaced point is $p = 0.5$. The nominal controller is a tracking derivative (D) controller with reference speed $\hat{s} = 1$ and control gains $k_{vy_1} = k_{vy_2} = 10$. The results consist of a comparison between three different output trajectories. The first one is obtained when the system uses PT-CBFs, the second corresponds to the system under the action of HO-CBFs that turn off when T_{safe} expires, and the last one is the output of the system when it is only driven by the nominal derivative controller. The constants c_{ij} of the class-k functions (10) have a value of 1.4 for all 3 waypoints. The intuition is that knowing in advance the time when the barrier has to be deactivated would lead to better results than simply turning the barrier off once this time expires. Additionally, it is also expected that the system without the action of any safety filters will violate the conditions S1 and S2 from Section IV.

The simulation starts with the autonomous aerial vehicle navigating towards the first waypoint, the purple circle in Figure 2. Due to its initial position, the system would fly over the target region of the third waypoint, the blue square in Figure 2, unless a safety filter is active around it. Both the PT-CBF and HO-CBF trajectories successfully avoid this area, whereas the output trajectory with only the D controller transverses across it. Once the target area around waypoint 1 is reached, the waypoint to track is updated and the system starts navigating towards the second waypoint and so on. Figures 3a, 3b, and 3c show the value of the PT-CBFs and HO-CBFs around waypoints 1, 2 and 3 respectively. In all three figures the PT-CBFs value is presented in a blue solid line, the HO-CBFs value corresponds to the solid orange line, the time instant when T_{safe} expires for the system using PT-CBFs is represented with a blue dashed vertical line, and the time instant when T_{safe} expires for the system using HO-CBFs corresponds to the orange dashed vertical line. Note that when h_j becomes negative it implies that the system has entered the critical proximity D^j_{crit}. All PT-CBFs decrease up to $h_j = -1$ because D^j_{crit} was chosen to be a radius $r_{sj} = 1$. Once the system reaches the waypoint, the corresponding PT-CBF and HO-CBF start increasing its value because the target waypoint has been updated, and now the autonomous aerial vehicle is flying towards the new goal. Figures 3a, 3b, and 3c prove how the system output using PT-CBFs enters the critical proximity faster than the system with HO-CBFs. As the simulation continues, the system using PT-CBFs reaches each target earlier than the one using only HO-CBFs. As a consequence, the time when each of these simulated outputs is allowed to enter the target region becomes further apart, which in addition to the faster transitions of the PT-CBFs once T_{safe} expires, leads to an overall better performance.

D. Implementation of PT- CBFs in Microsoft Flight Simulator

The final contribution of this work is to develop a Microsoft Flight Simulator (MSFS) based testing environment and implement PT-CBFs to generate safe navigation controllers. The testing platform has been built using Matlab-Simulink, where we have implemented the control strategy, the model dynamics, communications with MSFS and some real time monitoring of variables. The Simulink project is divided into four different subsystems as shown in Figure 4a. The System Model Subsystem, in Figure 4b, includes the second order unicycle dynamics and all the coordinate changes required to have an accurate visualization in MSFS. All simulations are run at a constant altitude. In Figure 4c the

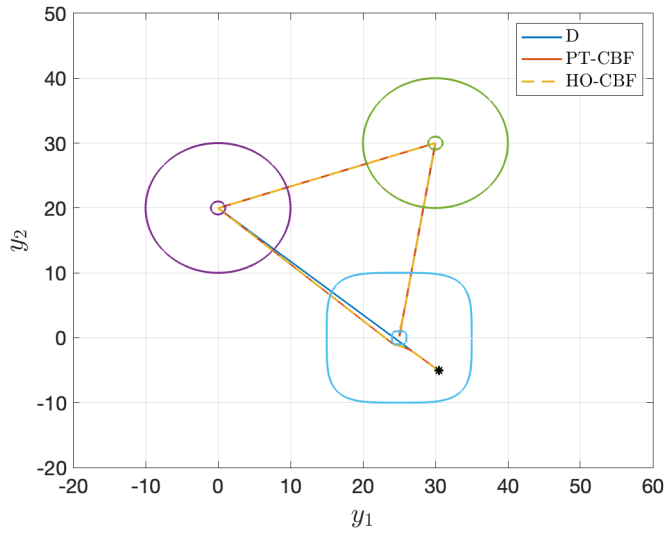
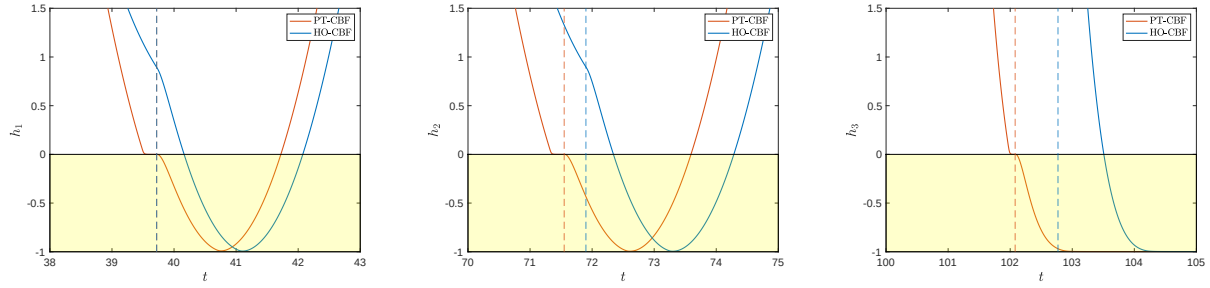


Fig. 2 Output trajectories compared: Nominal D (blue), PT-CBF (orange), and HO-CBF (dashed yellow). The autonomous aerial vehicle is tracking waypoints 1 (purple circle), 2 (green circle), and 3 (light blue square). The system's initial position is represented with an asterisk. The nominal trajectory (blue) violates safety as it fails to avoid the unsafe region.



(a) Waypoint 1 - h_1 **(b) Waypoint 2 - h_2** **(c) Waypoint 3 - h_3**

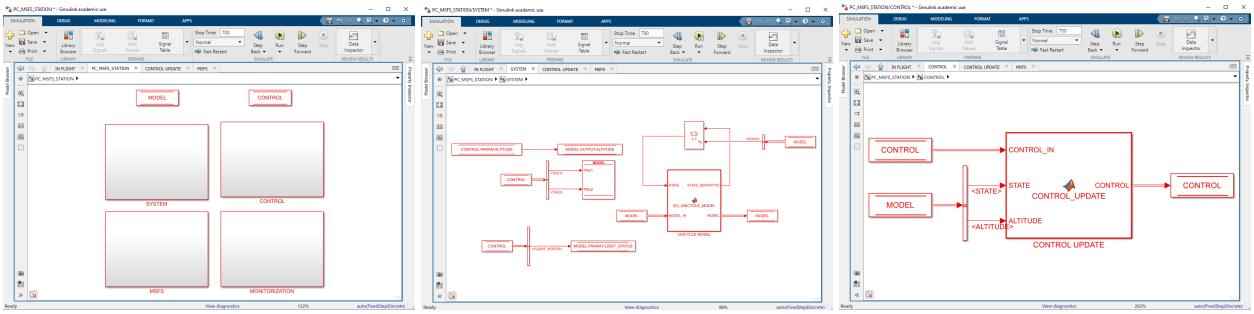
Fig. 3 Comparison between PT-CBFs (blue) and HO-CBFs (orange) for all waypoints in a semiautonomous navigation scenario. Both PT-CBFs and HO-CBFs are positive until their respective T_{safe} expires and they are removed.

Matlab function block includes the nominal controller, a Derivative waypoint tracking control, and the PT-CBFs used to avoid violating safety before the prescribed safe time T_{safe} expires.

The code used to run this experiment can be found in `https://github.com/gtfactslab/JimenezCortes_AIAA2024`. All three waypoints are modelled using $n = 2$ in Equation (19) with threshold and critical proximity radii equal to 1000 and 10 ft. The vehicle used for visualization is a Bell Model 407, Figure 5a, and the dynamics are modelled as a second order unicycle as they are run at a fixed altitude. Once the simulation starts, the helicopter tracks the waypoints with latitude and longitude $(33.7157°N, 84.2821°W)$, $(33.7472°N, 84.2821°W)$, and $(33.7759°N, 84.3107°W)$. Once the systems reaches the third waypoint, it hovers at the target location and the mission ends. The resulting trajectory can be seen in Figure 5b.

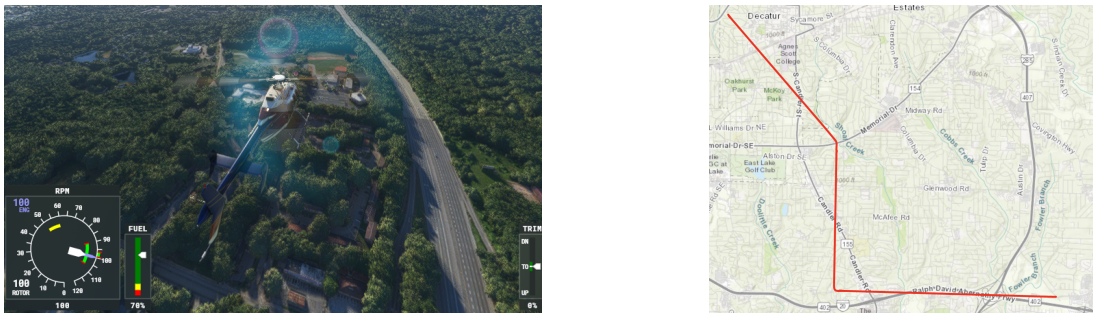
VI. Conclusions and Discussion

Coordinating humans' and autonomous systems' actions can be one of the most challenging aspects for the success of semiautonomous systems' missions. This collaboration between humans and autonomous systems becomes particularly challenging when the autonomy needs to perform an action that requires human intervention. While the system is



(a) Main Simulator Screen **(b) System Model Subsystem** **(c) Control Subsystem**

Fig. 4 Microsoft Flight Simulator Testing Platform.



(a) MSFS visualization during the mission **(b) MSFS trajectory using PT-CBFs**

Fig. 5 Microsoft Flight Simulator example. The system tracks three ordered waypoints $(33.7157°N, 84.2821°W)$**,** $(33.7472°N, 84.2821°W)$**, and** $(33.7759°N, 84.3107°W)$**, but it is not allowed in their proximity before** T_{safe} **expires.**

waiting for the human response, which can take a maximum time of T_{safe}, the system must remain safe, and execute the next command immediately after this time expires. This work shows that Prescribed-Time Control Barrier Functions facilitate coordination of humans' and autonomous sytems' actions in an efficient and risk-free way, outperforming simple strategies with CBFs or without safety filters, in these semiautonomous scenarios. Prescribed-Time Control Barrier Functions were also tested in a Microsoft Flight Simulator based testing platform, which, in future developments of this research, will enable running real experiments with humans in the control loop.

Acknowledgment

This work was funded in part by the Office of Naval Research, Science of Autonomy grant N00014-21-1-2759 Human-AI Collaboration in Autonomous Aerial Vehicles.

References

[1] Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., et al., "Towards fully autonomous driving: Systems and algorithms," *2011 IEEE intelligent vehicles symposium (IV)*, IEEE, 2011, pp. 163–168.

[2] Yurtsever, E., Lambert, J., Carballo, A., and Takeda, K., "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, Vol. 8, 2020, pp. 58443–58469.

[3] Bourne, D. A., and Fox, M. S., "Autonomous manufacturing: automating the job-shop," *Computer*, Vol. 17, No. 09, 1984, pp. 76–86.

[4] Park, H.-S., and Tran, N.-H., "An autonomous manufacturing system based on swarm of cognitive agents," *Journal of Manufacturing Systems*, Vol. 31, No. 3, 2012, pp. 337–348.

[5] Frew, E., McGee, T., Kim, Z., Xiao, X., Jackson, S., Morimoto, M., Rathinam, S., Padial, J., and Sengupta, R., "Vision-based road-following using a small autonomous aircraft," *2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No. 04TH8720)*, Vol. 5, IEEE, 2004, pp. 3006–3015.

[6] Ames, A. D., Grizzle, J. W., and Tabuada, P., "Control barrier function based quadratic programs with application to adaptive cruise control," *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 6271–6278.

[7] Notomista, G., and Egerstedt, M., "Persistification of robotic tasks," *IEEE Transactions on Control Systems Technology*, Vol. 29, No. 2, 2020, pp. 756–767.

[8] Squires, E., Pierpaoli, P., and Egerstedt, M., "Constructive barrier certificates with applications to fixed-wing aircraft collision avoidance," *2018 IEEE Conference on Control Technology and Applications (CCTA)*, IEEE, 2018, pp. 1656–1661.

[9] Abel, I., Steeves, D., and Krstic, M., "Prescribed-Time Safety Design for a Chain of Integrators," *arXiv preprint arXiv:2201.09447*, 2022.

[10] Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., and Tabuada, P., "Control Barrier Functions: Theory and Applications," *2019 18th European Control Conference (ECC)*, 2019, pp. 3420–3431.

[11] Khalil, H. K., *Nonlinear systems; 3rd ed.*, Prentice-Hall, 2002.

[12] Nguyen, Q., and Sreenath, K., "Exponential control barrier functions for enforcing high relative-degree safety-critical constraints," IEEE, 2016, pp. 322–328.

[13] Lee, T.-C., Song, K.-T., Lee, C.-H., and Teng, C.-C., "Tracking control of unicycle-modeled mobile robots using a saturation feedback controller," *IEEE transactions on control systems technology*, Vol. 9, No. 2, 2001, pp. 305–318.

[14] Van den Broek, T. H., van de Wouw, N., and Nijmeijer, H., "Formation control of unicycle mobile robots: a virtual structure approach," *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, IEEE, 2009, pp. 8328–8333.

[15] LaValle, S. M., *Planning algorithms*, Cambridge university press, 2006.

[16] De Luca, A., Oriolo, G., and Vendittelli, M., "Control of wheeled mobile robots: An experimental overview," *Ramsete*, 2001, pp. 181–226.

[17] Lamé, G., *Lamé ovals. The mathematical Gazette*, N.T. Gridgeman, 1970.