`gatekeeper`: Safety Verification and Control for Nonlinear Systems in Unknown and Dynamic Environments

Devansh Agrawal, Ruichang Chen and Dimitra Panagou

***Abstract*— This paper presents the `gatekeeper` algorithm, a real-time and computationally-lightweight method to ensure that nonlinear systems can operate safely within unknown and dynamic environments despite limited perception. `gatekeeper` integrates with existing path planners and feedback controllers by introducing an additional verification step that ensures that proposed trajectories can be executed safely, despite nonlinear dynamics subject to bounded disturbances, input constraints and partial knowledge of the environment. Our key contribution is that (A) we propose an algorithm to recursively construct committed trajectories, and (B) we prove that tracking the committed trajectory ensures the system is safe for all time into the future. The method is demonstrated on a complicated firefighting mission in a dynamic environment, and compares against the state-of-the-art techniques for similar problems.**

I. INTRODUCTION

Designing autonomous systems that can accomplish mission specifications with strict guarantees of safety is still a bottleneck to deploying such systems in the real world. Safety is often posed as requiring the system's trajectories to lie within a predefined set of allowable states, called the safe set. If the safe set is not known a priori, but is instead built based on output (sensor) measurements on-the-fly, then ensuring safety is even more challenging. We consider the problem where a robot with limited sensing capabilities (hence limited information about the environment) has to move while remaining safe with respect to a dynamic environment using only available sensory information.

Navigating within a non-convex safe set is often tackled by path planning techniques [1]–[4]. Since simplified (often linear) dynamics are used to generate trajectories that lie within the safe set, these may not be trackable by a nonlinear system. Furthermore, since trajectories are planned over finite horizons, in dynamic environments these methods can fail to find solutions, leading to safety violations. Recently, Control Barrier Functions (CBFs) [5] have gained interest since they offer a computationally-efficient method to maintain forward invariance of a safe set. However, a suitable CBF needs to be found, either analytically or using computationally expensive offline methods [5], [6]. Constructive methods are applicable to certain classes of dynamics and safe sets, but do not handle time-varying or multiple safety conditions well [7]–[10].

*The authors would like to acknowledge the support of the National Science Foundation (NSF) under grant no. 1942907.

Devansh Agrawal is with the Department of Aerospace Engineering, University of Michigan; Ruichang Chen is with the Department of Electrical and Computer Engineering; Dimitra Panagou is with the Department of Robotics and the Department of Aerospace Engineering, University of Michigan, Ann Arbor, USA. {devansh, chenrc dpanagou}@umich.edu

Approaches that couple the path planning and control problems comprise Model Predictive Control (MPC) techniques [11], [12], where dynamically feasible trajectories in the nonconvex safe set are designed. The nonconvexity of the problem implies that guaranteeing convergence, stability and recursive feasibility is challenging. In [13], such guarantees are obtained by exploiting the differential flatness of the system, although the resulting mixed-integer problem makes the method expensive in cluttered/complicated environments.

In this paper, we propose a technique to bridge path planners (that can solve the nonconvex trajectory generation problem) and low-level control techniques (that have robust stability guarantees) in a way that ensures safety, *without modifying either*. The idea is that given a nominal trajectory generated by the path planner (potentially unsafe and/or not dynamically feasible), using a backup controller we construct a committed trajectory that is safe, feasible, and defined for all future time. The low-level controller always tracks the committed trajectory, and therefore will always remain safe. This paper's key contribution is the algorithm to construct such committed trajectories, and a proof that the proposed approach ensures safety.

The method takes inspiration from [14] and [15], both of which also employ the idea of a backup planner/controller. In [14], a backup trajectory is constructed using a linear model to ensure the trajectory lies within the known safe set at any given time. However, ensuring that a trajectory (that may not be feasible for the nonlinear system dynamics) can be tracked is challenging. In [15], safety is guaranteed by blending the nominal control input with a backup control input. The mixing fraction is determined by numerically forward propagating the backup controller. However, since the nominal and backup control inputs are mixed, the nominal trajectory is never followed exactly, even when it is safe to do so. In this paper, by combining elements from both methods in a novel manner, we address the respective limitations of each. Furthermore, we explicitly account for robustness against disturbances since it turns out to be non-trivial.

Notation: Let $\mathbb{N} = \{0, 1, 2, ...\}$, and $\mathbb{R}, \mathbb{R}_{>0}, \mathbb{R}_{\geq 0}$ denote the set of reals, positive reals, and non-negative reals. Lowercase t is used for specific time points, while uppercase T is for intervals. $\|\cdot\|$ refers to the vector 2-norm. Norm balls are denoted $\mathbb{B}(x_0, r) = \{x : \|x - x_0\| \leq r\}$. $A \ominus B$ is the Pontryagin set difference. A function $\alpha : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is class $\mathcal{K}$ if it is continuous, strictly increasing and $\alpha(0) = 0$. $\beta : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a class $\mathcal{KL}$ function if it is continuous, for each $t \geq 0$, $\beta(\cdot, t)$ is class $\mathcal{K}$, and for each $r > 0$, $\beta(r, \cdot)$ is strictly decreasing and $\lim_{t\to\infty} \beta(r, t) = 0$.

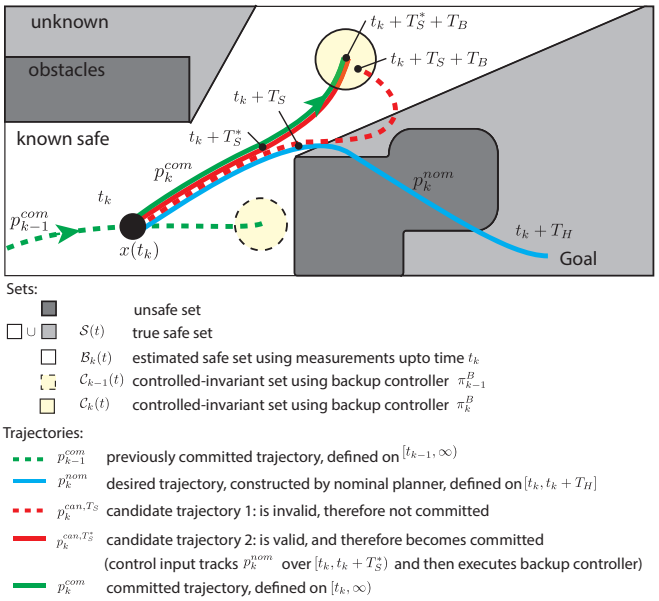

Fig. 1. Notation used in this paper. The nominal planner can plan trajectories into unknown spaces, but `gatekeeper` ensures the committed trajectory lies within the estimated safe sets, for all future time.

II. MOTIVATING EXAMPLE

We present an example to illustrate the key concepts in this paper, and challenges when dealing with dynamic environments and limited perception. A common wildfire fire-fighting mission is the "firewatch" mission, where a helicopter is deployed to trace the fire-front, the outer perimeter of the wildfire. The recorded GPS trace is used to create a map of the wildfire, used to efficiently deploy resources. Today, the firewatch mission is often carried out by human pilots, but in this example, we design an autonomous controller for a UAV to trace the fire-front without entering or being surrounded by the fire. Fig. 1 depicts the notation.

The fire is constantly evolving, and expanding outwards. Thus the safe set, the set of states located outside the fire, is a time-varying set denoted $\mathcal{S}(t)$. Since the rate of spread of fire is different at each location, (it depends on various environmental factors like slope, vegetation and wind [16], [17]), the evolution of the safe set $\mathcal{S}(t)$ is unknown.

That said, it is often possible to bound the evolution of $\mathcal{S}(t)$. In this example, we assume the maximum fire spread rate is known. To operate in this dynamic environment, the UAV makes measurements, for example thermal images that detect the fire-front. However, due to a limited field-of-view, only a part of the safe set can be measured.

The challenge, therefore, is to design a controller for the nonlinear system that uses the on-the-fly measurements to meet mission objectives, while ensuring the system state $x(t)$ remains within *the unknown* safe set at all times, i.e.,

$$x(t) \in \mathcal{S}(t), \ \forall t \geq t_0. \tag{1}$$

This paper proposes a technique to ensure this. When new information from the perception system arrives, we construct a *candidate trajectory* and check whether (1) holds for the candidate. If it does, the candidate trajectory becomes a *committed trajectory*. The low-level controller always tracks the last committed trajectory, ensuring safety.

Since $\mathcal{S}$ is unknown, verifying (1) directly is not possible. We ask a related question: given the information available at some time t_k, does a candidate trajectory $p_k^{can}(t)$ satisfy

$$p_k^{can}(t) \in \mathcal{B}_k(t), \ \forall t \geq t_k, \tag{2}$$

where $\mathcal{B}_k(t)$ is the *estimated* safe set (for each $t \geq t_k$), constructed using only the sensory information available up to time t_k. If we assume the perception system is not incorrect, $\mathcal{B}_k(t) \subset \mathcal{S}(t) \ \forall t \geq t_k$, then any candidate trajectory that satisfies (2) will also satisfy $p_k^{can}(t) \in \mathcal{S}(t)$. However, since the check in (2) needs to be performed over an infinite horizon $t \geq t_k$, it is not numerically feasible. *A key contribution of this paper is to show how we can perform this check by verifying only a finite horizon into the future.*

Referring back to the firewatch mission, if the UAV is able to fly faster than the maximum spread rate of the fire, a safe course of action could be to simply fly perpendicular to the firefront, i.e., radially from the fire at a higher speed than the maximum fire spread rate. This maneuver is an example of a *backup controller*, since it encodes the idea that if the system state reaches a backup set $\mathcal{C}_k(t)$ at some time $t_{kB} \geq t_k$, then the backup controller π_k^B ensures that $x(t) \in \mathcal{C}_k(t)$ for all $t \geq t_{kB}$. In the firewatch mission, π_k^B is controller that makes the UAV fly perpendicular to the firefront, and $\mathcal{C}_k(t)$ is the set of states that are "sufficiently far from fire, with a sufficiently high speed perpendicular to the fire."[1] Since the fire is constantly expanding, the $\mathcal{C}_k(t)$ set is also changing in time: the set of safe states needs to continue moving outwards radially. Furthermore, at each k, the backup controller and set can be a different, so we index these by k as well.

Using the notion of backup controllers, (2) reduces to:

$$p_k^{can}(t) \in \mathcal{S}(t) \ \forall t \geq t_k \tag{3}$$

$$\iff \begin{cases} p_k^{can}(t) \in \mathcal{S}(t) & \text{if } t \in [t_k, t_{kB}) \\ p_k^{can}(t) \in \mathcal{S}(t) & \text{if } t \in [t_{kB}, \infty) \end{cases} \tag{4}$$

$$\impliedby \begin{cases} p_k^{can}(t) \in \mathcal{B}_k(t) & \text{if } t \in [t_k, t_{kB}) \\ p_k^{can}(t_{kB}) \in \mathcal{C}_k(t_{kB}) \end{cases} \tag{5}$$

for any $t_{kB} \geq t_k$, provided (I) $\mathcal{B}_k(t) \subset \mathcal{S}(t)$, (II) $\mathcal{C}_k(t) \subset \mathcal{S}(t) \ \forall t \geq t_{kB}$, and (III) for $t \geq t_{kB}$ the control input to the candidate trajectory is π_k^B. These conditions can be verified easily: (I) is the correctness of the perception system, (II) is the defining property of a backup controller, and (III) will be true based on how we construct the candidate trajectory.

Notice that in (5), we only need to verify the candidate trajectory over a finite interval $[t_k, t_{kB}]$, but this is sufficient to proving that the candidate is safe for all $t \geq t_k$.

In the following sections, we formalize the `gatekeeper` as a method to construct safe trajectories that balance between satisfying mission objectives and ensuring safety.

[1] A worked example with exact expressions for $\mathcal{S}(t), \mathcal{B}_k(t), \mathcal{C}_k(t)$ is in the appendix of an extended version of this paper, uploaded here: [18].

III. PROBLEM FORMULATION

Consider a nominal and a perturbed nonlinear system:

$$\dot{x} = f(x, u), \tag{6}$$
$$\dot{x} = f(x, u) + d(t), \tag{7}$$

respectively, where $x \in \mathcal{X} \subset \mathbb{R}^n$ is the state, $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, and $f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ defines the (nominal) system dynamics. The additive disturbances $d : [t_0, \infty) \to \mathbb{R}^n$ are bounded, $\sup_{t \geq t_0} \|d(t)\| = \bar{d} < \infty$.

Given a control policy $\pi : [t_0, \infty) \times \mathcal{X} \to \mathcal{U}$, an initial condition $x(t_0) = x_0 \in \mathcal{X}$, and a bounded disturbance signal $d(t) \leq \bar{d}$, $t \geq t_0$, the initial-value problem describing the closed-loop system dynamics are:

$$\dot{x} = f(x, \pi(t, x)), \qquad x(t_0) = x_0, \tag{8}$$
$$\dot{x} = f(x, \pi(t, x)) + d(t), \qquad x(t_0) = x_0. \tag{9}$$

We assume that for each bounded disturbance signal $d(t)$, the solution $x(t)$ exists and is unique for all time $t \in [t_0, \infty)$.

Our method is based on concepts in forward invariance.

Definition 1. *For system (6), a controller $\pi : [t_0, \infty) \times \mathcal{X} \to \mathcal{U}$* renders a set $\mathcal{C}(t) \subset \mathcal{X}$ controlled-invariant *on t_0 if, for the closed-loop system (8) and any $\tau \geq t_0$,*

$$x(\tau) \in \mathcal{C}(\tau) \implies x(t) \in \mathcal{C}(t), \ \forall t \geq \tau. \tag{10}$$

Definition 2. *For system (7), a controller $\pi : [t_0, \infty) \times \mathcal{X} \to \mathcal{U}$* renders a set $\mathcal{C}(t) \subset \mathcal{X}$ robustly controlled-invariant *on t_0 if, for the closed-loop system (8) and any bounded disturbance $d(t)$ with $\sup_{t \geq t_0} \|d(t)\| \leq \bar{d}$, for any $\tau \geq t_0$,*

$$x(\tau) \in \mathcal{C}(\tau) \implies x(t) \in \mathcal{C}(t), \ \forall t \geq \tau. \tag{11}$$

The objective of this paper is to design a controller that ensures that system trajectories remain within $\mathcal{S}(t)$. We assume the following are available:

- a perception system that can estimate the safe set
- a nominal planner that uses simplified dynamics to generate desired trajectories to satisfy mission requirements (for example reaching a goal state, or exploring a region)
- a trajectory tracking controller
- a backup controller that can stabilize the system to a control invariant set.

Each is described in further detail below.

1) Perception System: Let $\mathcal{S}(t) \subset \mathcal{X}$ be the time-varying set of safe states, which in general is unknown. We assume that the perception system can construct *estimates* of the safe set that are updated as new information is acquired. The information is available at discrete times t_k, $k \in \mathbb{N}$. Let the estimated safe set at the k-th iteration (i.e., at time t_k) be $\mathcal{B}_k(t)$. We assume that:

Assumption 1. The estimated safe set $\mathcal{B}_k(t)$ satisfies

$$\mathcal{B}_k(t) \subset \mathcal{B}_{k+1}(t) \subset \mathcal{S}(t) \quad \forall t \geq t_k, \ \forall k \in \mathbb{N}. \tag{12}$$

This is essentially a correctness assumption: if the k-th measurement classifies a future state $x(t)$ (for $t \geq t_k$) as safe, we assume the next measurement will not reclassify $x(t)$ as unsafe. This assumption (while stated more generally) is common in the literature on path planning in dynamic/unknown environments [19], [20]. Depending on the application, various methods can be used to computationally represent such sets, including SDFs [21] or SFCs [22].

2) Nominal Planner: We assume that a nominal planner enforces the mission requirements by specifying the desired state of the robot for a short horizon T_H into the future.

Definition 3. *A* trajectory *p with horizon T_H is a piecewise continuous function $p : \mathcal{T} \to \mathcal{X}$ defined on $\mathcal{T} = [t_k, t_k + T_H] \subset \mathbb{R}$. A trajectory p is* dynamically feasible *for the system (6) if there exists a control policy $u : \mathcal{T} \to \mathcal{U}$ s. t.*

$$p(t) = p(t_k) + \int_{t_k}^{t} f(p(\tau), u(\tau)) d\tau, \quad \forall t \in \mathcal{T}. \tag{13}$$

Denote the nominal trajectory available at the k-th iteration by p_k^{nom}, defined on $[t_k, t_k + T_H]$. We do not require p_k^{nom} to be dynamically feasible.

3) Tracking Controller: We assume a state feedback controller $\pi^T : \mathcal{X} \times \mathcal{X} \to \mathcal{U}$ that computes a control input $u = \pi^T(x, p(t))$ to track a given trajectory $p(t)$; we refer to this policy as the tracking controller [23]–[25]. We assume that the tracking controller is disturbance-to-state stable:

Definition 4. *For any trajectory $p(t)$ defined on $\mathcal{T} = [t_k, t_k + T_H]$ that is dynamically feasible for the nominal system (6), the closed-loop dynamics of the perturbed system (7) under the tracking controller π^T given by:*

$$\dot{x} = f(x, \pi^T(x, p(t)) + d(t) \tag{14}$$

is disturbance-to-state stable*, i.e., for any disturbance $d(t)$,*

$$\|x(t_k) - p(t_k)\| \leq \delta \implies \|x(t) - p(t)\| \leq \beta(\delta, t - t_k) + \gamma(\bar{d}), \ \forall t \in \mathcal{T}, \tag{15}$$

where $\beta : \mathbb{R}_{\geq 0} \times \mathcal{T} \to \mathbb{R}_{\geq 0}$ is class $\mathcal{KL}$, $\gamma : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is class $\mathcal{K}$, and $\bar{d} = \sup_{t \geq t_0} \|d(t)\|$.

4) Backup Controller: In the case when a safe set $\mathcal{S}$ can not be rendered controlled invariant for given system dynamics, the objective can be reduced to finding a set $\mathcal{C} \subset \mathcal{S}$, and a controller $\pi : \mathcal{C} \to \mathcal{U}$ that renders the set $\mathcal{C}$ controlled invariant. For example, by linearizing (6) around a stabilizable equilibrium point x_e, an LQR controller can render a (sufficiently) small set of states around x_e forward invariant [26, Thm. 4.13, 4.18]. This observation leads to the notion of backup safety [15], [27].

Definition 5. *A controller $\pi_k^B : \mathcal{T} \times \mathcal{X} \to \mathcal{U}$ is a* backup controller *to a set $\mathcal{C}_k(t) \subset \mathcal{X}$ defined for $t \in \mathcal{T} = [t_k, \infty)$ if, for the closed-loop system*

$$\dot{x} = f(x, \pi_k^B(t, x)), \tag{16}$$

(A) there exists a neighborhood $\mathcal{N}_k(t) \subset \mathcal{X}$ of $\mathcal{C}_k(t)$, s.t. $\mathcal{C}_k(t)$ is reachable in fixed time T_B:

$$x(\tau) \in \mathcal{N}_k(\tau) \implies x(\tau + T_B) \in \mathcal{C}(\tau + T_B), \tag{17}$$

and (B) π_k^B renders $\mathcal{C}_k(t)$ controlled-invariant:

$$x(\tau) \in \mathcal{C}(\tau) \implies x(t) \in \mathcal{C}(t)\ \forall t \geq \tau. \tag{18}$$

We make the following assumption:

Assumption 2. At the k-th iteration, a set $\mathcal{C}_k(t)$ and a backup controller $\pi_B : [t_k, \infty) \times \mathcal{X} \to \mathcal{U}$ to $\mathcal{C}_k(t)$ can be found s.t.

$$\mathcal{C}_k(t) \subset \mathcal{S}(t),\ \forall t \geq t_k. \tag{19}$$

Note that while we assume $\mathcal{C}_k(t) \subset \mathcal{S}(t)$, we *do not* assume the trajectory to reach $\mathcal{C}_k(t)$ is necessarily safe, or that the set $\mathcal{C}_k(t)$ is always reachable. This is in contrast to backward reachability based methods [9], [10], [28], [29].

In summary, the problem statement is

Problem 1. Consider a dynamical system (7) with a perception system satisfying assumption 1, a nominal planner that generates desired trajectories, a disturbance-to-state stable tracking controller, and a backup controller satisfying assumption 2. Design an algorithm to track desired trajectories while ensuring safety, i.e., $x(t) \in \mathcal{S}(t)$ for all $t \geq t_0$.

IV. PROPOSED SOLUTION

`gatekeeper` is an additional module that lies *between* the traditional planning and control modules. It takes trajectories generated by the nominal planner, and instead of passing them directly to the low-level tracking controller, it computes a safe *committed* trajectory that is passed to the low-level tracking controller instead. In this section, we will demonstrate how to construct these committed trajectories. To aid the reader, the analysis is first presented for the nominal case, and later extended to the perturbed case. The various trajectories and times are depicted in Fig. 1.

A. Nominal Case

Suppose at the k-th iteration, $k \in \mathbb{N} \setminus \{0\}$, the previously committed trajectory is p_{k-1}^{com}. `gatekeeper` constructs a candidate trajectory p_k^{can,T_S} by forward propagating a controller that tracks p_k^{nom} over an interval $[t_k, t_k + T_S)$, and executes the backup controller for $t \geq t_k + T_S$. T_S is a switching time that `gatekeeper` will optimize, as described later. Formally,

Definition 6. *Suppose at $t = t_k$,*

- *the state is $x(t_k) = x_k$*
- *the nominal trajectory is p_k^{nom} defined on $[t_k, t_k + T_H]$*
- *π_k^B is a backup controller to the set $\mathcal{C}_k(t)$*

Given a $T_S \in [0, T_H]$, the candidate trajectory p_k^{can,T_S} *is the solution to the initial value problem*

$$\dot{p} = f(p, u(t)),\ p(t_k) = x_k, \tag{20}$$

$$u(t) = \begin{cases} \pi^T(p(t), p_k^{nom}(t)) & t \in [t_k, t_k + T_S) \\ \pi_k^B(t, p(t)) & t \geq t_k + T_S. \end{cases} \tag{21}$$

The candidate trajectory p_k^{can,T_S} is defined on $[t_k, \infty)$.

We say a candidate trajectory is *valid* if the following hold:

Definition 7. *A candidate trajectory p_k^{can,T_s} defined by (20) is* valid *if the trajectory is safe wrt the estimated safe set:*

$$p_k^{can,T_S}(t) \in \mathcal{B}_k(t),\ \forall t \in [t_k, t_{k,SB}], \tag{22}$$

and the trajectory reaches $\mathcal{C}_k(t)$ at the end of the horizon:

$$p_k^{can,T_S}(t_{k,SB}) \in \mathcal{C}_k(t_{k,SB}), \tag{23}$$

where $t_{k,SB} = t_k + T_S + T_B$.

Notice that checking whether a candidate is valid only the solution p_k^{can,T_S} over the finite interval $[t_k, t_k + T_S + T_B]$. This means that the candidate can be constructed by numerical forward integration over a finite horizon.

Next, we define how to construct a committed trajectory.

Definition 8. *At the k-th iteration, define*

$$\mathcal{I}_k = \Big\{ T_S \in [0, T_H] : p_k^{can,T_S} \text{ is valid} \Big\}, \tag{24}$$

where p_k^{can,T_S} is as defined in (20), and Def. 7 is used to check validity.

If $\mathcal{I}_k \neq \emptyset$, let $T_S^* = \max \mathcal{I}_k$. The committed trajectory *is*

$$p_k^{com}(t) = p_k^{can,T_S^*}(t), \quad t \in [t_k, \infty). \tag{25}$$

If $\mathcal{I}_k = \emptyset$, the committed trajectory *is*

$$p_k^{com}(t) = p_{k-1}^{com}(t), \quad t \in [t_k, \infty). \tag{26}$$

Def. 8 defines how the k-th committed trajectory is constructed using the nominal trajectory p_k^{nom} and the backup controller π_k^B.

Finally, we prove the proposed strategy guarantees safety.

Theorem 1. *Suppose p_0^{can,T_S} is a dynamically feasible candidate on $[t_0, \infty)$ that is valid by Def. 7 for some $T_S \geq 0$.*

If, for every $k \in \mathbb{N}$, p_k^{com} is determined using Def. 8, then

$$p_k^{com}(t) \in \mathcal{S}(t), \quad \forall t \in [t_k, \infty). \tag{27}$$

Furthermore, if $x(t_0) = p_0^{com}(t_0)$, and control input to the nominal system (6) is $u(t) = \pi_k^T(x(t), p_k^{com}(t)$, for $t \in [t_k, t_{k+1})$, then the closed-loop dynamics (8) satisfy $x(t) \in \mathcal{S}(t)$ for all $t \geq t_0$.

Proof. The first claim, i.e., $p_k^{com}(t) \in \mathcal{S}(t)$ for $t \geq t_k$ is proved by induction. *Base Case:* $k = 0$. Since p_0^{can} is a valid trajectory, it is committed, i.e., $p_0^{com} = p_0^{can,T_S}$. Then,

$$p_0^{com}(t) \in \begin{cases} \mathcal{B}_0(t) & \text{for } t \in [t_0, t_{0,SB}) \\ \mathcal{C}_0(t) & \text{for } t = t_{0,SB} \end{cases}$$

$$\implies p_0^{com}(t) \in \begin{cases} \mathcal{S}(t) & \text{for } t \in [t_0, t_{0,SB}) \\ \mathcal{S}(t) & \text{for } t \geq t_{0,SB} \end{cases}$$

$$\iff p_0^{com}(t) \in \mathcal{S}(t) \text{ for } t \geq t_0$$

where $t_{0,SB} = t_0 + T_S + T_B$.

Induction Step: Suppose the claim is true for some $k \in \mathbb{N}$. We will show the claim is also true for $k+1$. There are two possible definitions for p_k^{com}:

Case 1: When $\mathcal{I}_{k+1} \neq \emptyset$, p_{k+1}^{can,T_S^*} is a valid candidate, i.e.,

$$\begin{aligned} p_{k+1}^{com}(t) &= p_{k+1}^{can,T_S^*}(t) \ \forall t \geq t_0 \\ &\in \begin{cases} \mathcal{B}_{k+1}(t) & \text{for } t \in [t_{k+1}, t_{k+1,SB}) \\ \mathcal{C}_{k+1}(t) & \text{for } t \geq t_{k+1,SB} \end{cases} \\ &\in \mathcal{S}(t) \text{ for } t \geq t_{k+1} \end{aligned}$$

Case 2: If $\mathcal{I}_{k+1} = \emptyset$, the committed is unchanged,

$$p_{k+1}^{com}(t) = p_k^{com}(t) \in \mathcal{S}(t), \ \forall t \geq t_{k+1}.$$

This completes the first claim. Next, we prove that $x(t) \in \mathcal{S}(t)$ for all $t \geq t_0$. We do so by proving that $\forall k \in \mathbb{N}$, $x(t) = p_k^{com}(t)$ for all $t \in [t_k, t_{k+1})$. Again, we use induction.

Base Case: Since we are considering the nominal system dynamics (6), if $x(t_0) = p_0^{com}(t_0)$, and the tracking controller is disturbance to state stable (15),

$$\begin{aligned} \|x(t) - p_0^{com}(t)\| &\leq \beta(0, t - t_0) + \gamma(0) = 0 \\ \therefore x(t) &= p_0^{com}(t) \ \forall t \in [t_0, t_1) \end{aligned}$$

Induction Step: Suppose for some $k \in \mathbb{N}$, $x(t) = p_k^{com}(t)$ for $t \in [t_k, t_{k+1})$. There are two cases for p_{k+1}^{com}: *Case 1:* a new candidate is committed, $\therefore p_{k+1}^{can,T_S}(t_{k+1}) = x(t_{k+1})$. Since the tracking controller is disturbance-to-state stable, this implies $x(t) = p_{k+1}^{com}(t)$ for $t \in [t_{k+1}, t_{k+2})$. *Case 2:* A new candidate is not committed, $\therefore p_{k+1}^{com}(t) = p_k^{com}(t)$ for $t \in [t_{k+1}, t_{k+2})$. Since $x(t_{k+1}) = p_k^{com}(t_{k+1})$, the tracking controller ensures $x(t) = p_{k+1}^{com}(t)$ for $t \in [t_{k+1}, t_{k+2})$.

Therefore, $x(t) = p_k^{com}(t) \in \mathcal{S}(t) \ \forall t \in [t_k, t_{k+1})$, for each $k \in \mathbb{N}$. Thus, $x(t) \in \mathcal{S}(t)$ for all $t \geq t_0$. □

Remark 1. Notice that this construction method allows safe nominal trajectories to be followed closely: suppose at t_k a candidate trajectory p_k^{can,T_S} is committed, i.e., tracking the nominal over $[t_k, t_k + T_S)$ is safe. If the next iteration starts within this interval ($t_{k+1} \in [t_k, t_k + T_S)$), and a new valid candidate p_{k+1}^{can,T_S} is found, the system will track it over the larger interval $[t_k, t_{k+1} + T_S)$ and the backup controller is not used. As such, when `gatekeeper` is run frequently (i.e. $t_{k+1} - t_k$ is small), the committed trajectory is closer to the nominal trajectory. In practice, often, planners compute paths based on a global map, updated less frequently than the local map. Running `gatekeeper` on every local map update allows the system to be more reactive to new information.

B. Perturbed Case

We now address the case where the disturbances are non-zero. The algorithm is identical to that presented above, except that the validation step will be redefined. First, we highlight the problem that disturbances introduce. Recall Assumption 4, which defines the maximum tracking error.

A specific scenario is visualized in Fig. 2. If instead of (22), we checked that the tube containing the system trajectories lies within the safe set (green tube in Fig. 2a), then indeed, the system can remain safe. However, at the next iteration, for any new candidate, the new tube (red tube in Fig. 2a) will intersect with the unsafe set. Therefore, no new candidate trajectory can be committed, i.e., an undesired deadlock is reached: $x(t) \in \mathcal{C}_k(t)$ for all $t \geq t_{k,SB}$.

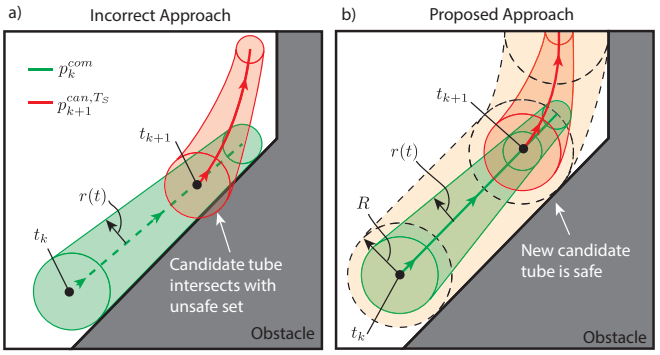

Fig. 2. Diagram depicting the challenge due to disturbances. (a) Green line shows the committed trajectory at iteration k, and the shaded region is the tube that contains the system trajectory. If the validation step only checks that the green tube lies within the safe set, a new candidate trajectory (red) cannot be committed, since the candidate tube (red shaded region) intersects with the unsafe set. (b) shows the proposed approach, where a tube of larger radius R is used to validate the trajectory. This ensures that at the next iteration, there is sufficient margin for a new trajectory to be committed.

To avoid this behavior, we must use a larger radius when performing the check:

Definition 9. *A candidate trajectory p_k^{can,T_s} defined by (20) is* robustly valid *with robustness level $r \geq 0$, if*

- *it is robustly safe over a finite interval:*

$$p_k^{can,T_S}(t) \in \mathcal{B}_k(t) \ominus \mathbb{B}(0, R) \ \forall t \in [t_k, t_{k,SB}], \quad (28)$$

- *at the end of the interval, it reaches the interior of $\mathcal{C}_k(t)$:*

$$p_k^{can,T_S}(t_{k,SB}) \in \mathcal{C}_k(t_{k,SB}) \ominus \mathbb{B}(0, m), \quad (29)$$

- *and the set $\mathcal{C}_k(t)$ is R away from the safe set boundary:*

$$\mathcal{C}_k(t) \subset \mathcal{S}(t) \ominus \mathbb{B}(0, R) \ \forall t \geq t_k, \quad (30)$$

where $m = \beta(r, T_S + T_B) + \gamma(\bar{d})$ and $R = \beta(r, 0) + \gamma(\bar{d})$.

Notice that $\gamma(\bar{d}) \leq m \leq R$ for each r, but in the limit as $r \to 0$, $\gamma(\bar{d}) = m = R$.

Theorem 2. *Suppose p_0^{can,T_S} is a dynamically feasible candidate trajectory on $[t_0, \infty)$ that is* robustly *valid by Def. 9 for some $r, T_S > 0$. Suppose $\left\| x(t_0) - p_0^{can,T_S}(t_0) \right\| \leq r$.*

If, for every $k \in \mathbb{N}$, p_k^{com} is determined using Def. 8 (except that validity is checked using Def. 9), and the control input to the perturbed system (7) is

$$u(t) = \pi_k^T(x(t), p_k^{com}(t)) \quad \forall t \in [t_k, t_{k+1}] \quad (31)$$

then the closed-loop (9) satisfies $x(t) \in \mathcal{S}(t), \ \forall t \geq t_0$.

Proof. This proof is almost identical to that of Thm. 1. We highlight the main differences. We need to prove two things: (A) for every iteration k, $\left\| x(t_k) - p_k^{can,T_S} \right\| < r$ and (B) if (A) is true, then tracking the k-th robustly valid candidate trajectory for $t \geq t_k$ ensures $x(t) \in \mathcal{S}(t)$.

Part (A): The base case, $k = 0$, is assumed in the theorem statement. For $k \geq 1$, notice that when p_k^{can,T_S} is constructed, according to (20), the candidate is a forward propagation from the initial condition $p_k^{can,T_S}(t_k) = x_k = x(t_x)$. Therefore, $\left\| x(t_k) - p_k^{can,T_S}(t_k) \right\| = 0 \leq r$.

Part (B): If p_k^{can,T_S} is robustly valid,

$$p_k^{can,T_S}(t) \in \begin{cases} \mathcal{B}_k(t) \ominus \mathbb{B}(0,R) & t \in [t_k, t_{k,SB}) \\ \mathcal{C}_k(t) \ominus \mathbb{B}(0,m) & t \in [t_{0,SB}, \infty) \end{cases}$$

Therefore, if $x(t)$ were to track p_k^{can,T_S} for all $t \geq t_k$,

$$x(t) \in \begin{cases} \mathcal{B}_k(t) & t \in [t_k, t_{k,SB}) \\ \mathcal{C}_k(t) & t \in [t_{k,SB}, \infty) \end{cases} \implies x(t) \in \mathcal{S}(t) \ \forall t \geq t_k$$

since by Assumption 4, $\left\| x(t_k) - p_k^{can,T_S}(t_k) \right\| \leq r$ implies

$$\left\| x(t) - p_k^{can,T_S}(t) \right\| \leq \beta(r, t - t_k) + \gamma(\bar{d}) \leq R, \quad \forall t \geq t_k.$$

To complete the proof, notice that since the committed trajectory over the interval $[t_k, t_{k+1})$ corresponds to a robustly valid candidate trajectory, tracking p_k^{com} implies $x(t) \in \mathbb{B}(p_k^{com}(t), R) \subset \mathcal{S}(t)$, for all $t \in [t_k, t_{k+1})$. Since this is true for all $k \in \mathbb{N}$, $x(t) \in \mathcal{S}(t)$ for all $t \geq t_0$. □

Remark 2. The theorem can be interpreted as guidelines/constraints on the nominal planner. For instance, requiring trajectories to lie in $\mathcal{B}(t_k) \ominus \mathbb{B}(0,R)$ corresponds to the common practice of inflating the unsafe sets by a radius R. However, what should the inflation radius be? The theorem shows that any $R \geq \gamma(\bar{d})$ is sufficient. Further increasing R (by increasing r) makes solutions more conservative, but robust to mismatch in initial conditions $\|x(t_k) - p_k^{com}(t_k)\| \leq r$. This can be used to account for errors due to state estimation or computation time.

Remark 3. The construction of committed trajectories is summarized in pseudo-code in Alg. 1. Determining $\max \mathcal{I}$ is not computationally expensive, since it is an optimization over a scalar variable in a bounded interval. We used a simple grid search with N points. Therefore, upto N initial value problems need to be solved. Using modern diffeq libraries, e.g. [30], this can be done very efficiently. In our simulations, with $N = 10$, the median computation time was only 3.4 ms.

Algorithm 1: `gatekeeper`

Parameters: $N > 0 \in \mathbb{N}$
`// Do a grid search backwards over the interval` $[0, T_H]$`:`
for *i in range(0, N):* **do**
  Using $\mathcal{B}_k(t)$, identify $\mathcal{C}_k(t)$ satisfying assum. 2.
  $T_S = (1 - i/N)T_H$
  Solve the initial value problem (20) to determine $p_k^{can,T_S}(t)$ over the interval $[t_k, t_k + T_S + T_B]$
  if p_k^{can,T_S} *is robustly valid by Def. 9* **then**
    $p_k^{com} = p_k^{can,T_S}$
    return
`// no candidate is valid,` $\mathcal{I} = \emptyset$
$p_k^{com} = p_{k-1}^{com}$
return

V. Extended Case Study

Code and Animations: [18].

We simulate an autonomous helicopter performing the firewatch mission, around a fire with an initial perimeter of 16 km. The helicopter begins 0.45 km from the fire front, and is tasked to fly along the perimeter, without entering the fire, while maintaining a target airspeed of 15 m/s. The helicopter is modelled as:

$$\dot{x}_1 = x_3 \cos x_4 \qquad \dot{x}_2 = x_3 \sin x_4$$
$$\dot{x}_3 = u_1 \qquad \dot{x}_4 = (g/x_3) \tan u_2$$

where x_1, x_2 are the cartesian position coordinates of the helicopter wrt an inertial frame, x_3 is the speed of the vehicle along its heading, x_4 is the heading, and g is the acceleration due to gravity. The control inputs are u_1, the acceleration along the heading, and u_2, the roll angle. The inputs are bounded, with $|u_1| < 0.5g$ and $|u_2| < \pi/4$ rad. This system models a UAV that can control its forward airspeed and makes coordinated turns. Notice the model has a singularity at $x_3 = 0$, and the system is *not* control affine.

The fire is modeled using level-set methods [31]. In particular, the fire is described using the implicit function $\phi : \mathbb{R} \times \mathbb{R}^2 \to \mathbb{R}$, where $\phi(p,t)$ is the signed distance to the firefront from location p at time t. Hence, the safe set is

$$\mathcal{S}(t) = \{x : \phi(t, [x_1, x_2]^T) \geq 0$$

where $[x_1, x_2]$ is the position of the UAV. The evolution of the fire is based on the Rothermel 1972 model [16]. Each point p on the fire-front travels normal to the front at a speed $\sigma(p)$, satisfying: $\frac{\partial \phi}{\partial t}(t,p) + \sigma(p) \|\nabla \phi(t,p)\| = 0$, where $\sigma : \mathbb{R}^2 \to \mathbb{R}$ is the Rate of Spread (RoS). The RoS depends on various environmental factors including terrain topology, vegetation type, and wind speeds [16], [17] but can be bounded [32]. The simulated environment was assigned an RoS function that the controllers did not have access to. The only information the controllers were allowed to use was the thermal image (to detect the fire within a ± 1 km range) and assumption that the maximum rate of spread is 8 km/h.

We compare our approach against the nominal planner and two state of the art methods for similar problems, Fig. V. In particular, we compare (A) a nominal planner (black), (B) FASTER [14] (purple), (C) Backup Filters [15] (blue) and (D) `gatekeeper` (green). Since these methods were not originally developed for dynamic environments with limited perception, both methods (B, C) were modified to be applicable to this scenario. See [18] for details.

The simulation environment and each of the methods were implemented in `julia`, to allow for direct comparison. `Tsit5()` [30] with default tolerances was used to simulate

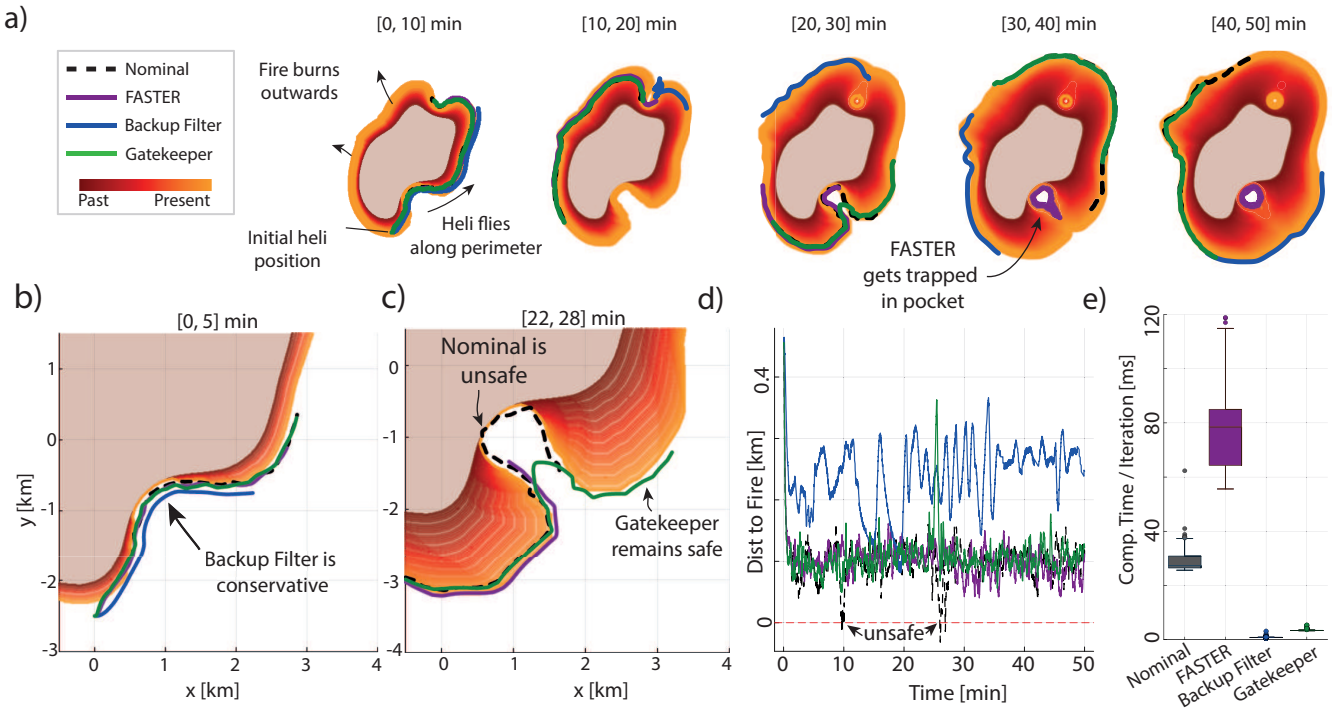

Fig. 3. Simulation results from Firewatch mission. (a) Snapshots of the fire and trajectories executed by each of three controller. The fire is spreading outwards, and the helicopters are following the perimeter. The black line traces the nominal controller, the blue line is based on the backup filter adapted from [15] and the green line shows the proposed controller. (b, c) show specific durations in greater detail. At $t = 0$, the `gatekeeper` controller behaves identically to the nominal controller, and makes small modifications when necessary to ensure safety. The backup filter is conservative, driving the helicopter away from the fire and slowing it down. (d) Plot of minimum distance to fire-front across time for each of the controllers. (e) The nominal controller becomes unsafe 3 times, while FASTER, the backup controller, and the `gatekeeper` controllers maintain safety. Animations are available at [18].

	Distance to Fire [km]			Velocity [m/s]		Comp. time [ms]		
	Minimum	Mean	Std.	Mean	Std.	Median	IQR	
Target	≥ 0	0.100	-	15.0	-	-	-	
Nominal Planner	-0.032	0.098	0.032	15.14	0.73	27.32	4.37	Unsafe
FASTER [14]	0.040	0.101	0.030	12.60	2.08	78.50	20.64	Safe, but gets trapped in pocket
Backup Filters [15]	0.081	0.240	0.054	10.11	3.52	0.87*	0.05	Safe, but conservative and slow
Gatekeeper (proposed)	0.049	0.108	0.034	14.91	1.35	3.39	0.11	Safe

TABLE I. Comparison of `gatekeeper` (ours) with the nominal planner, FASTER [14], and backup filters [15]. The distance to the firefront, velocity of the helicopter, and computation time per iteration are reported for each method. IQR = interquartile range. *Since the backup filter is run at each control iteration instead of every planning iteration, it runs 20 times as often as `gatekeeper`, i.e., is 5 times as computationally expensive as `gatekeeper`.

the different controllers. Each run simulates a flight time of 50 minutes. The controllers were implemented as zero-order hold, updated at 20 Hz. Measurements of the firefront are available at 0.1 Hz, triggering the planners to update, intentionally slow to highlight the challenges of slow perception/planning systems. The measurements are a bitmask image, defining the domain where $\phi \leq 0$, at a grid resolution of 10 meters. All experiments were performed on a 2019 Macbook Pro (Intel i9, 2.3 GHz, 16 GB).

In the nominal planner, a linear MPC problem is solved to generate trajectories that fly along the local tangent 0.1 km away from firefront at 15 m/s. The planner uses a simplified dynamic model for the helicopter, a discrete-time double integrator. This problem is a convex quadratic program (QP), solved using `gurobi`. The median computation time is 27 ms, using $N = 40$ waypoints and a planning horizon of 120 seconds. The tracking controller is a nonlinear feedback controller that directly tracks trajectories of the double integrator, based on differential flatness [13], [25]. When the low level directly tracks nominal trajectories, the system becomes unsafe, going as far as 32 m into the fire.

In FASTER, the same double integrator model is assumed, and a similar MPC problem is solved. We impose additional safety constraints, that the committed trajectory must lie within a safe flight corridor [22] based on the signed distance field to the fire, corrected based on the maximum fire spread rate. While this approach does keep the helicopter outside the fire, it gets surrounded by the fire (Fig. Va). This is ultimately due to the fact that FASTER only plans trajectories over a finite planning horizon, and is therefore unable to guarantee recursive feasibility in a dynamic environment. Due to the additional constraints on the QP, FASTER is about 3 times slower than the nominal planner.

In the Backup Filters approach, the backup trajectory

is forward propagated on the nonlinear system over the same 120 second horizon. The backup controller makes the helicopter fly radially, i.e., is a simple feedback controller. Backup trajectories can be computed extremely efficiently, requiring less than 1 ms per iteration. While this approach keeps the system safe, it does so at the cost of performance: the mean distance to the fire is 0.24 km, more than twice the target value, and the average speed is 10 m/s, 33% less than the target. This is behaviour is because the desired flight direction is perpendicular to the backup flight direction, and therefore the trajectory is off-nominal.

In `gatekeeper`, the committed trajectories are constructed by maximizing the interval that the nominal trajectory is tracked, before implementing the backup controller. This allows the system to follow the nominal, and deviate only when required to ensure safety: in Fig. Vc, we see that `gatekeeper` chooses to not fly into the pocket, since it cannot ensure a safe path out of the pocket will exist in the future. `gatekeeper` is computationally lightweight, with a median run time of 3.4 ms, more than 20 times faster than FASTER. This is primarily because `gatekeeper` searches over a scalar variable in a bounded interval, instead of optimizing $\mathbb{R}^{4N+2N-2}$ variables as in the MPC problem.

VI. CONCLUSION

This paper proposes an algorithm ("`gatekeeper`") to safely control nonlinear robotic systems while information about dynamically-evolving safe states is received online. The algorithm constructs an infinite-horizon committed trajectory from a nominal trajectory using backup controllers. By extending a section of the nominal trajectory with the backup controller, `gatekeeper` is able to follow nominal trajectories closely, while guaranteeing a safe control input is known at all times. We applied the algorithm to an aerial firefighting mission, where we demonstrated `gatekeeper` is less conservative than similar methods, while remaining computationally lightweight. While we have demonstrated the approach on a firefighting scenario, the method is applicable to a wide range of scenarios where only limited safety information is known, for instance, overtaking or merging scenarios for autonomous vehicles. Future directions involve developing more general methods to identify backup controllers, and understanding how the method can be applied to adversarial multi-agent settings.

REFERENCES

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

[2] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *2011 IEEE ICRA*. IEEE, 2011, pp. 1478–1483.

[3] D. J. Webb and J. Van Den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE ICRA*, 2013, pp. 5054–5061.

[4] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *16th Int. Symposium on Robotics Research*. Springer, 2016, pp. 649–666.

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

[6] T. Gurriet, M. Mote, A. Singletary, E. Feron, and A. D. Ames, "A scalable controlled set invariance framework with practical safety guarantees," in *2019 IEEE CDC*. IEEE, 2019, pp. 2046–2053.

[7] W. S. Cortez and D. V. Dimarogonas, "Correct-by-design control barrier functions for euler-lagrange systems with input constraints," in *2020 American Control Conference*, 2020, pp. 950–955.

[8] J. Breeden and D. Panagou, "Guaranteed safe spacecraft docking with control barrier functions," *IEEE Control Systems Letters*, vol. 6, pp. 2000–2005, 2021.

[9] M. Abate and S. Coogan, "Enforcing safety at runtime for systems with disturbances," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 2038–2043.

[10] C. Llanes, M. Abate, and S. Coogan, "Safety from in-the-loop reachability for cyber-physical systems," in *Proceedings of the Workshop on Computation-Aware Algorithmic Design for Cyber-Physical Systems*, 2021, pp. 9–10.

[11] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2.

[12] U. Rosolia, A. Carvalho, and F. Borrelli, "Autonomous racing using learning model predictive control," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 5115–5120.

[13] D. R. Agrawal, H. Parwana, R. K. Cosner, U. Rosolia, A. D. Ames, and D. Panagou, "A constructive method for designing safe multirate controllers for differentially-flat systems," *IEEE Control Systems Letters*, vol. 6, pp. 2138–2143, 2021.

[14] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "Faster: Fast and safe trajectory planner for navigation in unknown environments," *IEEE TRO*, vol. 38, no. 2, pp. 922–938, 2021.

[15] A. Singletary, A. Swann, Y. Chen, and A. D. Ames, "Onboard safety guarantees for racing drones: High-speed geofencing with control barrier functions," *IEEE RAL*, vol. 7, no. 2, pp. 2897–2904, 2022.

[16] R. C. Rothermel, *A mathematical model for predicting fire spread in wildland fuels*. Intermountain Forest & Range Experiment Station, Forest Service, US ..., 1972, vol. 115.

[17] P. L. Andrews, "The rothermel surface fire spread model and associated developments: A comprehensive explanation," *Gen. Tech. Rep. RMRS-GTR-371. Fort Collins, CO: US Dept. of Agriculture, Forest Service, Rocky Mountain Research Station. 121 p.*, vol. 371, 2018.

[18] D. Agrawal and R. Chen, "Gatekeeper," https://github.com/dev10110/gatekeeper, 2022.

[19] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE TRO*, vol. 38, no. 1, pp. 463–476, 2021.

[20] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE TRO*, vol. 37, no. 6, pp. 1992–2009, 2021.

[21] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning," in *2017 IEEE/RSJ IROS*. IEEE, 2017, pp. 1366–1373.

[22] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE RAL*, vol. 2, no. 3, pp. 1688–1695, 2017.

[23] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi, "A stable tracking control method for an autonomous mobile robot," in *IEEE ICRA*. IEEE, 1990, pp. 384–389.

[24] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se (3)," in *49th IEEE CDC*. IEEE, 2010, pp. 5420–5425.

[25] P. Martin, R. M. Murray, and P. Rouchon, "Flat systems, equivalence and trajectory generation," 2003.

[26] H. K. Khalil, "Nonlinear systems, 3rd edition," *Prentice Hall*, 2002.

[27] Y. Chen, M. Jankovic, M. Santillo, and A. D. Ames, "Backup control barrier functions: Formulation and comparative study," in *2021 60th IEEE Conference on Decision and Control*, 2021, pp. 6835–6841.

[28] S. Tonkens and S. Herbert, "Refining control barrier functions through Hamilton-Jacobi reachability," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 13 355–13 362.

[29] T. Gurriet, P. Nilsson, A. Singletary, and A. D. Ames, "Realizable set invariance conditions for cyber-physical systems," in *2019 IEEE ACC*. IEEE, 2019, pp. 3642–3649.

[30] C. Rackauckas and Q. Nie, "`Differentialequations.jl`–a performant and feature-rich ecosystem for solving differential equations in `julia`," *J. Open Research Software*, vol. 5, no. 1, 2017.

[31] A. Alessandri, P. Bagnerini, M. Gaggero, and L. Mantelli, "Parameter estimation of fire propagation models using level set methods," *Applied Mathematical Modelling*, vol. 92, pp. 731–747, 2021.

[32] M. G. Cruz and M. E. Alexander, "The 10% wind speed rule of thumb for estimating a wildfire's forward rate of spread in forests and shrublands," *Annals of Forest Science*, vol. 76, no. 2, pp. 1–11, 2019.

APPENDIX

A. Worked Example for the Firewatch Scenario

This example demonstrates how the sets $\mathcal{S}(t), \mathcal{B}_k(t), \mathcal{C}_k(t)$ are related.

Consider the firewatch mission. For simplicity, consider a double integrator system,

$$\dot{x} = Ax + Bu \tag{33}$$

where $x_{pos} = [x_1, x_2]^T$ is the position of the helicopter, and $x_{vel} = [x_3, x_4]^T$ is the velocity.

Say the fire starts at $t = t_0$, at the location $p = [0, 0]^T$. The fire expands radially, where the rate of spread is $\sigma : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$, i.e., $\sigma(p)$ is the rate of spread at a location $p \in \mathbb{R}^2$. To make the algebra easier, say the rate of spread only depends on $\|p\|$, i.e, $\sigma(p_1) = \sigma(p_2)$ for any $\|p_1\| = \|p_2\|$. This means that the fire always spreads out uniformly, and is always a circle.

Therefore, the safe set is time-varying, and can be described as

$$\mathcal{S}(t) = \left\{ x : \|x_{pos}\| \geq \int_0^t \sigma(r(\tau))d\tau \right\} \ \forall t \geq 0 \tag{34}$$

where $r(t)$ is the radius of the fire at time $t \geq t_0$.

Since we don't know σ, we don't actually know $\mathcal{S}(t)$. Instead, we assume a reasonable upper bound is known: $\sigma(r) \leq 2$ m/s for all $r \geq 0$.

Therefore, at the initial time, we can define an *estimated safe set*:

$$\mathcal{B}^0(t) = \{x : \|x_{pos}\| \geq 2(t - t_0)\} \ \forall t \geq t_0 \tag{35}$$

and therefore

$$\mathcal{B}^0(t) \subset \mathcal{S}(t) \ \forall t \geq 0 \tag{36}$$

Notice that $\mathcal{B}^0(t)$ is *not* a controlled invariant set for the double integrator.[2]

Suppose the system can directly measure the fire's radius. Let the k-th measurement be $r_k = r(t_k)$. This allows us to define the k-th *estimated* safe set:

$$\mathcal{B}_k(t) = \{x : \|x_{pos}\| \geq r_k + 2(t - t_k)\} \ \forall t \geq t_k \tag{37}$$

It is easy to verify that

$$\begin{aligned}
\mathcal{B}_k(t) &= \{x : \|x_{pos}\| \geq r_k + 2(t - t_k)\} \\
&= \left\{ x : \|x_{pos}\| \geq \int_{t_0}^{t_k} \sigma(r(\tau))d\tau + 2(t - t_k) \right\} \\
&\subset \left\{ x : \|x_{pos}\| \geq \int_{t_0}^{t_k} \sigma(r(\tau))d\tau + \int_{t_k}^{t} \sigma(r(\tau))d\tau \right\} \\
&= \left\{ x : \|x_{pos}\| \geq \int_{t_0}^{t} \sigma(r(\tau))d\tau \right\} \\
&= \mathcal{S}(t)
\end{aligned}$$

[2]Technically, a higher-order CBF could be used to design a QP controller that renders a subset of $\mathcal{B}^0(t)$ forward invariant, but this is only possible since $\mathcal{B}^0(t)$ is a sufficiently smooth function that we can analyze analytically.

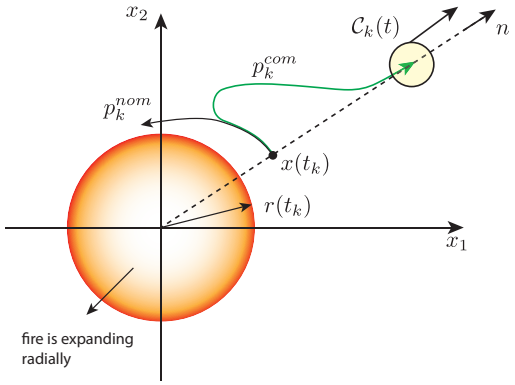

Fig. 4. Depiction of the scenario in the worked example.

i.e. $\mathcal{B}_k(t) \subset \mathcal{S}(t)$ for all $t \geq t_k$.

Similarly, we can verify that for any $k \geq 0$,

$$\begin{aligned}
\mathcal{B}_{k+1}(t) &= \{x : \|x_{pos}\| \geq r_{k+1} + 2(t - t_{k+1})\} \\
&= \left\{ x : \|x_{pos}\| \geq r_k + \int_{t_k}^{t_{k+1}} \sigma(r(\tau))d\tau + 2(t - t_{k+1}) \right\} \\
&\supset \{x : \|x_{pos}\| \geq r_k + 2(t_{k+1} - t_k) + 2(t - t_{k+1})\} \\
&= \{x : \|x_{pos}\| \geq r_k + 2(t - t_k)\} \\
&= \mathcal{B}_k(t)
\end{aligned}$$

i.e., $\mathcal{B}_k(t) \subset \mathcal{B}_{k+1}(t)$ for all $t \geq t_k$.

This shows that assumption 1 is satisfied for this perception system.

Next, we define the backup controllers.

For any $k \in \mathbb{N}$, suppose the state is $x_k = x(t_k)$. The backup controller should drive the system radially away from the fire. Define n_k as the unit vector pointed at $x(t_k)$:

$$n_k = \frac{x_{pos}(t_k)}{\|x_{pos}(t_k)\|} \tag{38}$$

Notice that if the position followed the reference

$$p_{ref}(t) = (1 + r_k + 2(t - t_k))n_k \tag{39}$$

then the reference is moving radially at a speed of 2 m/s, and therefore faster than the maximum spread rate of the fire. Thus $p_{ref}(t)$ is a safe trajectory for all $t \geq t_k$.

This leads to the following backup controller:

$$\pi_k^B(t, x) = -K \left(\begin{bmatrix} x_{pos} \\ x_{vel} \end{bmatrix} - \begin{bmatrix} p_{ref}(t) \\ 2n_k \end{bmatrix} \right) \tag{40}$$

where $K \in \mathbb{R}^{2\times 4}$ is a stabilizing LQR controller for the double integrator.

This controller will stabilizes the system to time varying set $\mathcal{C}_k(t)$, where

$$\mathcal{C}_k(t) = \left\{ x : \left\| x - \begin{bmatrix} p_{ref}(t) \\ 2n_k \end{bmatrix} \right\| \leq 1 \right\} \tag{41}$$

This set is controlled invariant using the backup controller π_k^B. Geometrically, $\mathcal{C}_k(t)$ is a unit norm ball that is moving radially at 2 m/s in the n_k direction. Therefore, it is apparent that $\mathcal{C}_k(t) \subset \mathcal{S}(t)$ for all $t \geq t_k$, since the set is moving

outwards radially at a speed higher than the maximum spread rate.

This example demonstrates how $\mathcal{S}(t)$, $\mathcal{B}_k(t)$, $\mathcal{C}_k(t)$ can be defined. The main validation step in `gatekeeper`, will confirm whether after following the nominal trajectory over $[t_k, t_k + T_S)$, the system is able to safely reach $\mathcal{C}_k(t)$ using the backup controller π_k^B.

While in this example the sets were described analytically, in the simulation case study, they were represented numerically using signed distance fields.