Author's Copy. This paper has been accepted for publication to IEEE L-CSS 2025.

Online Safety under Multiple Constraints and Input Bounds using `gatekeeper`: Theory and Applications

Devansh R. Agrawal, *Student Member, IEEE*, and Dimitra Panagou, *Senior Member, IEEE*

***Abstract*—This letter presents an approach to guarantee online safety of a cyber-physical system under multiple state and input constraints. Our proposed framework, called `gatekeeper`, recursively guarantees the existence of an infinite-horizon trajectory that satisfies all constraints and system dynamics. Such trajectory is constructed using a backup controller, which we define formally in this paper. `gatekeeper` relies on a small number of verifiable assumptions, and is computationally efficient since it requires optimization over a single scalar variable. We make two primary contributions in this letter. (A) First, we develop the theory of `gatekeeper`: we derive a sub-optimality bound relative to a full nonlinear trajectory optimization problem, and show how this can be used in runtime to validate performance. This also informs the design of the backup controllers and sets. (B) Second, we demonstrate in detail an application of `gatekeeper` for multi-agent formation flight, where each Dubins agent must avoid multiple obstacles and weapons engagement zones, both of which are nonlinear, nonconvex constraints. [Code][†]**

***Index Terms*—Constrained control; Optimization algorithms; Aerospace**

I. Introduction

INCREASING use of robotic systems in real-world applications necessitates advanced controllers that ensure safety, robustness, and effectiveness in human-machine teaming [1].

This letter formalizes and builds upon our recent work on online safety verification and control [2], which introduces `gatekeeper` as a novel algorithmic component between the planner and the controller of the autonomous system. To briefly illustrate the principle behind `gatekeeper`, consider a Unmanned Aerial Vehicle (UAV) navigating an unknown environment. The UAV follows a nominal trajectory, generated by its planner and tracked by its controller. At each iteration, `gatekeeper` performs two key steps: (i) it evaluates the currently known safe set (derived from onboard sensing), and a backup set, which represents a region the UAV can retreat to if the nominal trajectory is predicted to exit the safe set in the future; (ii) it constructs a candidate trajectory by stitching together the nominal trajectory (up to a future time horizon) and a backup trajectory that leads safely into the backup set. The candidate is accepted if it remains within the known safe set. If so, it becomes the new committed trajectory to be tracked by the controller. Otherwise, the UAV continues to follow the previously committed trajectory. Because a new trajectory is only committed when guaranteed to be safe, the UAV is always lies within the safe set. Importantly, `gatekeeper` only forward propagates candidate trajectories, making it computationally efficient.

Literature Review: Various approaches guarantee safety of autonomous systems. Model Predictive Control (MPC) offers a natural framework, but solving nonlinear/nonconvex problems online can be computationally expensive or can fail without warning [3]–[5]. Control Barrier Functions (CBFs) enforce safety constraints through Quadratic Programs (QPs) [6]–[8], though finding a valid barrier function remains challenging in the face of multiple constraints and input bounds. Reachability-based methods offer strong safety guarantees [9], [10], but are intractable in high-dimensions.

Increasingly backup-based methods have become popular [11]–[17], where a precomputed fallback is used as the system approaches unsafe conditions. Our framework extends this idea while addressing key limitations. Compared to [11] it explicitly considers nonlinear systems and nonconvex constraints. Compared to [12], [13], [17] instead of mixing the nominal and backup control inputs, we check when it is necessary to switch to the backup, allowing the system to follow the nominal closely. By guaranteeing safety using trajectories rather than controllers, we can enable performant backup maneuvers. [14] reviews runtime assurance methods, each variant appropriate for a different application. A common challenge is in choosing when/how to intervene, which we address by analyzing the optimality of backups.

Contributions: One open question is, how optimal is the generated trajectory of `gatekeeper`, and how is this affected by the choice of the backup controller and set? This letter formalizes `gatekeeper` by deriving suboptimality bounds, and defining the formal construction of the backup controller. This also provides a framework to analyze the (sub)optimality of other safety architectures, since most methods (e.g. [11], [13]) do not consider/minimize the penalty on mission performance.

Second, we demonstrate the framework in a challenging multi-agent formation flight problem. Each Dubins agent must avoid multiple Engagement Zones (EZs), both of which are nonlinear, nonconvex constraints that depend on the robot's state. This combination of tight input bounds, multiple con-

The authors would like to acknowledge the support of the National Science Foundation (NSF) under grant no. 2137195. Both D Agrawal (devansh@umich.edu) and D Panagou (dpanagou@umich.edu) are with the Robotics Department, University of Michigan, Ann Arbor, USA.

[†] https://github.com/dev10110/GatekeeperFormationFlight.jl

straints, and nonconvexity means that most modern approaches fail to guarantee safety. Furthermore, we demonstrate that our solution is computationally efficient and close-to-optimal solutions can be computed in 1% of the time required to solve this problem using IPOPT.

II. THEORY

Notation: $\mathbb{N} = \{0, 1, 2, ...\}$ is the set of natural numbers. $\mathbb{R}, \mathbb{R}_{\geq 0}$ denote reals, and non-negative reals. $\mathbb{S}^n_+$ is the set of symmetric positive-definite matrices in $\mathbb{R}^{n\times n}$. The notation $\{1 : N\}$ defines the set $\{1, ..., N\} \subset \mathbb{N}$. For $v \in \mathbb{R}^n$, $\|v\| = \sqrt{v^T v}$, $\|v\|_P = \sqrt{v^T P v}$. The set of piecewise continuous functions $w : \mathcal{T} \to \mathcal{D}$ are denoted by $\mathcal{L}(\mathcal{T}, \mathcal{D})$, and $\mathcal{T} = \mathbb{R}$ when omitted. $\rightrightarrows$ denotes a set-valued map, e.g. $\mathcal{S} : \mathcal{T} \rightrightarrows \mathcal{X}$ means that for any $t \in \mathcal{T}$, $\mathcal{S}(t) \subset \mathcal{X}$ is a set.

A. Preliminaries

Consider a (possibly time-varying) dynamical system

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

where $x \in \mathcal{X} \subset \mathbb{R}^n$ is the state, and $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input. The dynamics $f : \mathbb{R}_{\geq 0} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ are piecewise continuous in t and locally Lipschitz in x and u. Given a feedback policy $u = \pi(t, x)$ with π piecewise continuous in t and locally Lipschitz in x, the closed-loop system admits a unique solution over some interval.

Definition 1 (Trajectory)**.** *Let $\mathcal{T} = [t_i, t_f) \subset \mathbb{R}$. A trajectory is a pair of functions $(p : \mathcal{T} \to \mathcal{X}, u : \mathcal{T} \to \mathcal{U})$ satisfying*

$$\dot{p}(t) = f(t, p(t), u(t)) \quad \forall t \in (t_i, t_f). \tag{2}$$

The set of all trajectories from $(t, x) \in \mathbb{R} \times \mathcal{X}$ is

$$\Phi(t, x) = \{(p, u) : p(t) = x \text{ and } (p, u) \text{ is a trajectory}\}. \tag{3}$$

Let $\mathcal{S} : \mathbb{R} \rightrightarrows \mathcal{X}$ denote the (possibly time-varying) set of states satisfying constraints, e.g., a polytope $\{x : Ax \leq b\}$ or superlevel set $\{x : h(x) \geq 0\}$. The system satisfies the constraints if $x(t) \in \mathcal{S}(t) \quad \forall t \geq t_0$. In principle, one can compute the set of initial states that admit a safe trajectory:

$$\mathcal{F}(t) = \Big\{x \in \mathcal{X} : \exists (p, u) \in \Phi(t, x) \text{ satisfying } p(\tau) \in \mathcal{S}(\tau)\ \forall \tau \geq t\Big\}. \tag{4}$$

Computing $\mathcal{F}(t)$ is generally intractable, as it involves solving a reachability problem. Instead, we assume a known backup set $\mathcal{C} : \mathbb{R} \rightrightarrows \mathcal{X}$:

Definition 2 (Backup set)**.** *$\mathcal{C} : \mathbb{R} \rightrightarrows \mathcal{X}$ is a backup set if*

$$\mathcal{C}(t) \subset \mathcal{S}(t) \quad \forall t \in \mathbb{R}, \tag{5}$$

and the controller $\pi^B : \mathbb{R} \times \mathcal{X} \to \mathcal{U}$ is such that for all $t_i \in \mathbb{R}$ the closed-loop system $\dot{x} = f(t, x, \pi^B(t, x))$ satisfies

$$x(t_i) \in \mathcal{C}(t_i) \implies x(t) \in \mathcal{C}(t)\ \forall t \geq t_i. \tag{6}$$

Lemma 1. *If $\mathcal{C}$ is a backup set, then*

$$\mathcal{C}(t) \subset \mathcal{F}(t) \subset \mathcal{S}(t) \quad \forall t \in \mathbb{R}. \tag{7}$$

Proof. If $x \in \mathcal{F}(t)$, then by definition, $x = p(t) \in \mathcal{S}(t)$. If $x \in \mathcal{C}(t)$, the backup controller ensures $x(\tau) \in \mathcal{C}(\tau) \subset \mathcal{S}(\tau)$ for all $\tau \geq t$, hence $x \in \mathcal{F}(t)$. □

We specify the mission objectives in terms of a desired/nominal trajectory for the system to follow. Formally,

Definition 3 (Nominal Trajectory)**.** *Given state $x_k \in \mathcal{X}$ at time $t_k \in \mathbb{R}$, a planner generates a nominal trajectory,*

$$(p_k^{\text{nom}}, u_k^{\text{nom}}) \in \Phi(t_k, x_k)$$

defined over $\mathcal{T} = [t_k, t_k + T_H]$.

We cannot directly execute the nominal trajectory since it may violate safety constraints and may not end in $\mathcal{F}(t_k + T_H)$, risking future constraint violation.

B. Problem Statement

To address this, we seek a modified trajectory that is both safe and tracks the nominal plan. We can pose this as:

$$\begin{aligned}
\underset{\substack{p \in \mathcal{L}(\mathcal{X}),\\ u \in \mathcal{L}(\mathcal{U})}}{\text{minimize}} \int_{t_k}^{t_k+T_H} & L\left(t, p(t), u(t), p_k^{\text{nom}}(t), u_k^{\text{nom}}(t)\right) dt && \text{(8a)}\\
\text{s.t. } \dot{p} &= f(t, p(t), u(t)), \quad \forall t \in \mathcal{T}, && \text{(8b)}\\
p(t) &\in \mathcal{S}(t), \quad \forall t \in \mathcal{T}, && \text{(8c)}\\
p(t_k) &= x_k, && \text{(8d)}\\
p(t_k + T_H + T_B) &\in \mathcal{C}(t_k + T_H + T_B), && \text{(8e)}
\end{aligned}$$

with $\mathcal{T} = [t_k, t_k + T_H + T_B]$.

This is a finite-horizon optimal control problem. The terminal constraint (8e) ensures the trajectory ends in a backup set, which is stricter than requiring $p(t_k + T_H) \in \mathcal{F}(t_k + T_H)$. We choose the former since $\mathcal{F}$ is unknown. The objective (8a) is to minimize the cost of deviating from the nominal trajectory. Note, the cost only integrates over $[t_k, t_k + T_H]$ a subset of $\mathcal{T}$. We make an assumption on L:

Assumption 1. *$L : \mathbb{R} \times \mathcal{X} \times \mathcal{U} \times \mathcal{X} \times \mathcal{U} \to \mathbb{R}_{\geq 0}$ is positive definite about (x_2, u_2):*

$$\begin{aligned}
&L(t, x_1, u_1, x_2, u_2) \geq 0,\\
&L(t, x_1, u_1, x_2, u_2) = 0 \iff (x_1 = x_2 \text{ and } u_1 = u_2).
\end{aligned}$$

for all $t \in \mathbb{R}, x_1, x_2 \in \mathcal{X}, u_1, u_2 \in \mathcal{U}$.

Examples satisfying this include:

$$\begin{aligned}
L_1(\cdot) &= \|x_1 - x_2\|_Q^2 + \|u_1 - u_2\|_R^2, && \text{(9a)}\\
L_2(\cdot) &= e^{-\gamma(t-t_k)} L_1(\cdot), && \text{(9b)}\\
L_3(\cdot) &= \begin{cases} 0 & \text{if } x_1 = x_2 \text{ and } u_1 = u_2,\\ 1 & \text{else.}\end{cases} && \text{(9c)}
\end{aligned}$$

where $Q \in \mathbb{S}^n_+$, $R \in \mathbb{S}^m_+$, $\gamma > 0$

The problem addressed in this paper is as follows:

Problem 1. *Design a method to solve* (8) *under Assumption 1 and assuming a backup set is known for the system. If solutions are suboptimal, quantify the suboptimality.*

In summary, the goal is to compute a *committed trajectory* $(p^{\text{com}}, u^{\text{com}})$ that guarantees safety over $[t_k, \infty)$ while minimizing deviation from the nominal trajectory. As (8) is typically nonconvex, our focus is on efficient computation of feasible solutions and real-time suboptimality bounds.

C. Constructing candidate trajectories

Here we describe our solution, a recursive method for constructing committed trajectories. The main result, Theorem 1, proves feasibility and quantifies suboptimality. While the core principles of `gatekeeper` as presented in [2] remain unchanged, key differences are highlighted in Remark 2.

`gatekeeper` triggers at discrete times $t_k \in \mathbb{R}$ for $k \in \mathbb{N}$. Each candidate trajectory is a concatenation of two segments: the nominal trajectory (as defined in Definition 3) and a backup trajectory, defined as follows:

Definition 4 (Backup Trajectory)**.** *Let $T_B \geq 0$. For any $t_s \in \mathbb{R}$ and $x_s \in \mathcal{X}$, a trajectory $(p^{\mathrm{bak}}, u^{\mathrm{bak}}) \in \Phi(t_s, x_s)$ defined on $[t_s, \infty)$ is a backup trajectory from (t_s, x_s) if*

$$p^{\mathrm{bak}}(t_s + T_B) \in \mathcal{C}(t_s + T_B), \tag{10}$$

and for all $t \geq t_s + T_B$, u^{bak} satisfies

$$u^{\mathrm{bak}}(t) = \pi^B(t, p^{\mathrm{bak}}(t)). \tag{11}$$

In words, a trajectory $(p^{\mathrm{bak}}, u^{\mathrm{bak}})$ is a backup trajectory from the specified (t_s, x_s), if (A) trajectory is dynamically feasible starting from (t_s, x_s) (since $(p^{\mathrm{bak}}, u^{\mathrm{bak}}) \in \Phi(t, x)$), (B) the trajectory reaches $\mathcal{C}$ within T_B seconds, and (C) after reaching $\mathcal{C}$, the control input corresponds to the backup controller. Recall from the definition of π^B this ensures the trajectory remains within $\mathcal{C}$. Thus for any backup trajectory, we have $p^{\mathrm{bak}}(\tau) \in \mathcal{C}(\tau)$ for all $\tau \geq t_s + T_B$.

A candidate trajectory is one that switches between executing the nominal trajectory and a backup trajectory. This idea of stitching together a section of the nominal trajectory with a backup trajectory is a core principle of `gatekeeper`.

Definition 5 (Candidate trajectory)**.** *Consider a system with state $x_k \in \mathcal{X}$ at time $t_k \in \mathbb{R}$. Let the nominal trajectory be $(p_k^{\mathrm{nom}}, u_k^{\mathrm{nom}}) \in \Phi(t_k, x_k)$ defined over $[t_k, t_k + T_H]$. A candidate trajectory with switch time $t_s \in [t_k, t_k + T_H]$ is $(p_k^{\mathrm{can}}, u_k^{\mathrm{can}}) \in \Phi(t_k, x_k)$ defined by*

$$(p_k^{\mathrm{can}}(\tau), u_k^{\mathrm{can}}(\tau)) = \begin{cases} (p_k^{\mathrm{nom}}(\tau), u_k^{\mathrm{nom}}(\tau)) & \text{if } \tau \in [t_k, t_s), \\ (p_k^{\mathrm{bak}}(\tau), u_k^{\mathrm{bak}}(\tau)) & \text{if } \tau \geq t_s, \end{cases} \tag{12}$$

where $(p_k^{\mathrm{bak}}, u_k^{\mathrm{bak}})$ is a backup trajectory from $(t_s, p_k^{\mathrm{nom}}(t_s))$.

A candidate is *valid* if it is safe over a finite horizon:

Definition 6 (Valid)**.** *A candidate trajectory $(p_k^{\mathrm{can}}, u_k^{\mathrm{can}}) \in \Phi(t_k, x_k)$ with switch time $t_s \in \mathbb{R}$ is valid if*

$$p_k^{\mathrm{can}}(t) \in \mathcal{S}(t) \quad \forall t \in [t_k, t_s + T_B], \tag{13}$$

where $T_B \geq 0$ is the horizon of the backup trajectory.

This immediately leads to the following lemma:

Lemma 2. *Consider a system with state $x_k \in \mathcal{X}$ at time $t_k \in \mathbb{R}$. If $(p_k^{\mathrm{can}}, u_k^{\mathrm{can}}) \in \Phi(t_k, x_k)$ is a valid candidate trajectory, then*

$$p_k^{\mathrm{can}}(t) \in \mathcal{S}(t) \quad \forall t \geq t_k. \tag{14}$$

Proof. First we prove that $p_k^{\mathrm{can}}(t) \in \mathcal{S}(t)$ for all $t \geq t_k$. Since the candidate is valid, we have $p_k^{\mathrm{can}}(t) \in \mathcal{S}(t)\ \forall t \in [t_k, t_b]$ where $t_b = t_s + T_B$. Since the candidate trajectory follows the backup trajectory for all $t \in [t_s, t_b]$, it reaches the backup set at t_b, i.e., $p_k^{\mathrm{can}}(t_b) \in \mathcal{C}(t_b)$. For all $t \geq t_b$ the control input matches the backup controller, and because π^B renders $\mathcal{C}$ forward invariant, it follows that $p_k^{\mathrm{can}}(t) \in \mathcal{C}(t) \forall t \geq t_b$. Finally since for any backup set $\mathcal{C}(t) \subset \mathcal{S}(t)\ \forall t \in \mathbb{R}$, we have $p_k^{\mathrm{can}}(t) \in \mathcal{C}(t) \subset \mathcal{S}(t) \quad \forall t \geq t_b$. Therefore, we have $p_k^{\mathrm{can}}(t) \in \mathcal{S}(t)$ for all $t \geq t_k$. □

This proves that any valid candidate trajectory is a safe trajectory for all future time, but only requires one to check safety over a finite horizon $[t_k, t_s + T_B]$. This finite-horizon check enables practical implementation of the framework.

D. Optimality of candidate trajectories

Beyond safety, we would also like our trajectories to be optimal. Here we address: (A) how should one construct the backup trajectory, and (B) how should one select the best candidate trajectory, i.e., select the best switching time t_s?

Recall that $L(\cdot)$ is the running cost, as defined in (8a). The objective functional (8a) can be split into two intervals:

$$\begin{aligned} J(p, u) &= \int_{t_k}^{t_k + T_H} L(\cdot) dt && \text{(15a)} \\ &= \underbrace{\int_{t_k}^{t_s} L(\cdot) dt}_{J_1(p,u,t_s)} + \underbrace{\int_{t_s}^{t_k + T_H} L(\cdot) dt}_{J_2(p,u,t_s)} && \text{(15b)} \end{aligned}$$

where $(\cdot) = (t, p(t), u(t), p_k^{\mathrm{nom}}(t), u_k^{\mathrm{nom}}(t))$, and $t_s \in [t_k, t_k + T_H]$ is a switch time. Then we have the following:

$$\begin{aligned} &J(p^{\mathrm{can}}, u^{\mathrm{can}}) \\ &\quad = J_1(p^{\mathrm{can}}, u^{\mathrm{can}}, t_s) + J_2(p^{\mathrm{can}}, u^{\mathrm{can}}, t_s) && \text{(16a)} \\ &\quad = J_1(p^{\mathrm{nom}}, u^{\mathrm{nom}}, t_s) + J_2(p^{\mathrm{bak}}, u^{\mathrm{bak}}, t_s) && \text{(16b)} \\ &\quad = 0 + J_2(p^{\mathrm{bak}}, u^{\mathrm{bak}}, t_s) && \text{(16c)} \end{aligned}$$

where the J_1 term is zero, since for all $t \in [t_k, t_s]$, the candidate trajectory matches the nominal trajectory. Thus, by Assumption 1, the integrand is $L(\cdot) = 0$, and therefore $J_1(\cdot) = 0$. We can thus propose a solution to Problem 1:

Theorem 1. *Suppose at time $t_k \in \mathbb{R}$ the system state is $x_k \in \mathcal{X}$. Let $(p_k^{\mathrm{nom}}, u_k^{\mathrm{nom}}) \in \Phi(t_k, x_k)$ be the nominal trajectory. Suppose $(p_k^{\mathrm{can}}, u_k^{\mathrm{can}}) \in \Phi(t_k, x_k)$ is a candidate trajectory with switch time $t_s \in [t_k, t_k + T_H]$.*

If $(p_k^{\mathrm{can}}, u_k^{\mathrm{can}})$ is a valid candidate trajectory, then

(A) *the candidate trajectory is a feasible solution to* (8),

(B) *the suboptimality of the candidate trajectory with respect to* (8) *is upper-bounded by* [1]

$$\bar{B} = \int_{t_s}^{t_k + T_H} L(\cdot) dt \tag{17}$$

[1] The integral in (17) is over $[t_s, t_k + T_H]$, not $[t_k, t_k + T_H]$ as in (8a).

where $(\cdot) = (t, p_k^{\text{can}}(t), u_k^{\text{can}}(t), p^{\text{nom}}(t), u^{\text{nom}}(t))$.

Proof. Claim A: Since $(p_k^{\text{can}}, u_k^{\text{can}}) \in \Phi(t_k, x_k)$ it must satisfy (8b) and (8d). Since it is valid, it must satisfy (8c), and at time $t_b = t_s + T_B \leq t_k + T_H + T_B$ (since $t_s \in [t_k, t_k + T_H]$) the trajectory reaches $p_k^{\text{can}}(t_b) \in \mathcal{C}(t_b)$. Since for $t \geq t_b$ the candidate trajectory remains within $\mathcal{C}$, it satisfies (8e).

Claim B: For convenience, let $t_H = t_k + T_H$ and $t_{HB} = t_k + T_H + T_B$. Suppose $(p_k^{\text{opt}}, u_k^{\text{opt}}) \in \Phi(t_k, x_k)$ is the optimal solution of (8), which exists since $(p_k^{\text{can}}, u_k^{\text{can}})$ is feasible.

First, notice that it is possible for $J(p_k^{\text{opt}}, u_k^{\text{opt}}) \geq 0$. Suppose the nominal trajectory is safe (i.e., $p_k^{\text{nom}}(t) \in \mathcal{S}(t)\ \forall t \in [t_k, t_H]$) and terminates in the backup set (i.e., $p_k^{\text{nom}}(t_H) \in \mathcal{C}(t_H)$). Then, the candidate trajectory (p', u') with switch time $t_s = t_H$ is valid. Notice that $J(p', u') = 0$ since for all $t \in [t_k, t_H]$ the candidate trajectory is equal to the nominal trajectory, and thus $L(\cdot) = 0$. Therefore, it is possible for $J(p, u) = 0$ in (8), and thus $J(p_k^{\text{opt}}, u_k^{\text{opt}}) \geq 0$.

Second, notice that the cost of the candidate trajectory is

$$J(p_k^{\text{can}}, u_k^{\text{can}}) = \underbrace{J_1(p_k^{\text{can}}, u_k^{\text{can}}, t_s)}_{=0} + \underbrace{J_2(p_k^{\text{can}}, u_k^{\text{can}}, t_s)}_{=\bar{B}}$$

where the the first term is zero since over the interval $t \in [t_k, t_s]$ the candidate trajectory equals the nominal trajectory. Therefore, $0 \leq J(p_k^{\text{opt}}, u_k^{\text{opt}}) \leq J(p_k^{\text{can}}, u_k^{\text{can}}) = \bar{B}$, and thus

$$J(p_k^{\text{can}}, u_k^{\text{can}}) - J(p_k^{\text{opt}}, u_k^{\text{opt}}) \leq \bar{B},$$

i.e., $\bar{B}$ is the maximum suboptimality. □

Corollary 1. *The optimal candidate trajectory has*

$$t_s \in \underset{t'_s \in [t_k, t_k + T_H]}{\operatorname{argmin}} \ J_2(p^{\text{bak}}, u^{\text{bak}}, t'_s). \tag{18}$$

where for any $t'_s \in \mathbb{R}$, $(p^{\text{bak}}, u^{\text{bak}})$ *is a backup trajectory from* $(t'_s, p_k^{\text{nom}}(t'_s))$.

E. Optimal backup trajectories

In principle, as part of solving (18), one could also optimize over the set of backup trajectories. For any given t'_s, suppose the backup trajectory solves

$$\begin{aligned} \underset{(p,u) \in \Phi(t'_s, p_k^{\text{nom}}(t'_s))}{\text{minimize}} \quad & J_2(p, u, t'_s) && \text{(19a)} \\ \text{s.t.} \quad & p(t) \in \mathcal{S}(t), \qquad \forall t \in \mathcal{T}, && \text{(19b)} \\ & p(t'_s + T_B) \in \mathcal{C}(t'_s + T_B), && \text{(19c)} \end{aligned}$$

where $\mathcal{T} = [t'_s, t_k + T_B]$. In this case, we can conclude:

Lemma 3. *Let* $(p_k^{\text{nom}}, u_k^{\text{nom}}) \in \Phi(t_k, x_k)$ *be the nominal trajectory. A candidate trajectory* $(p^{\text{can}}, u^{\text{can}}) \in \Phi(t_k, x_k)$ *with switch time* t_s *is an optimal solution of* (8) *if*

(A) *the candidate trajectory is valid,*
(B) *the backup trajectory is the solution of* (19), *and*
(C) *the switch time* t_s *is chosen according to* (18).

Proof. Notice that a candidate trajectory satisfying conditions (A, B, C) is the solution of the problem

$$\begin{aligned} \underset{\substack{p \in \mathcal{L}(\mathcal{X}), u \in \mathcal{L}(\mathcal{U}) \\ t_s \in [t_k, t_k + T_H]}}{\text{minimize}} \quad & J_1(p, u, t_s) + J_2(p, u, t_s) && \text{(20a)} \\ \text{s.t.} \quad & \text{(8b), (8c), (8d), (8e)} && \text{(20b)} \end{aligned}$$

which is equivalent to problem (8), except with the additional variable t_s. This additional variable does not affect the feasibility or optimality of the problem, therefore these optimization problems (and solutions) are equivalent. □

Remark 1. *The key insight from Lemma 3 is that if an optimal backup trajectory is known, solving* (18) *yields the optimal solution to* (8), *but without requiring one to solve a trajectory optimization problem.* `gatekeeper` *is particularly useful when feasible (but not necessarily optimal) backup trajectories can be efficiently generated. In such cases,* (18) *— a scalar line search over a bounded interval — yields a suboptimal solution to* (8), *with a suboptimality bound given by* (17).

F. The `gatekeeper` architecture

`gatekeeper` is an intermediary module between the planner and the low-level controller. It uses the planner's nominal trajectory to construct a committed trajectory that is defined for all future time, guaranteed to be safe, and minimally deviates from the nominal. At each planning step k at time t_k and state x_k, given a nominal trajectory $(p_k^{\text{nom}}, u_k^{\text{nom}})$ `gatekeeper` constructs candidate trajectories for each switch time $t_s \in [t_k, t_k + T_H]$ (see Definition 5). Valid candidates are checked, and if any exist, the one minimizing the cost in (18) is selected; otherwise, the committed trajectory remains unchanged. This guarantees safety for all future time if a valid committed exists initially.

While the optimal backup could be found by solving (19), this may be computationally expensive. Instead, efficient suboptimal backups can still yield good performance, with online-computable suboptimality bounds given in Theorem 1, enabling runtime monitoring of mission progress.

Remark 2. *The* `gatekeeper` *framework was first introduced in [2], but this work includes several key extensions:*

- *Whereas [2] selected the switch time* t_s *to maximize the validity duration of the nominal trajectory (i.e., minimizing the backup duration), we generalize this by allowing arbitrary cost functions in* (18).
- *Disturbances are handled in [2] but are omitted here for clarity; the analysis can be extended to include them.*
- *We introduce a formal optimality framework, including sufficient conditions for optimality (Lemma 3) and suboptimality bounds (Theorem 1), which were not considered in [2] or prior works like [11], [13].*

III. APPLICATION

We demonstrate the proposed architecture to a multi-agent formation flight problem, where agents must safely navigate through a domain with multiple EZs.

Problem Setup: Consider a team of N_A agents:

$$\dot{x}_i = \begin{bmatrix} v_i \cos x_{i,3}, & v_i \sin x_{i,3}, & \omega_i \end{bmatrix}^T \tag{21}$$

where $x_i \in \mathcal{X} \subset \mathbb{R}^3$ and $u_i = [v_i, \omega_i] \in \mathcal{U} \subset \mathbb{R}^2$. The state defines 2D position and heading; inputs are bounded: $v_i \in [0.8, 1.0]$, $\omega_i \in [-10.0, 10.0]$. Units are normalized (LU/TU $= 1$). Agents must avoid N_Z EZs:

$$x_i \in \mathcal{S} = \{x : h_j(x) \geq 0, \ \forall j \in \{1, \dots, N_Z\}\}, \tag{22}$$

where h_j is defined in [18]. We omit its expression for brevity, and as it is not critical to the algorithm. The leader's trajectory (p^L, u^L) is precomputed using a Dubins-based RRT* method [19], generating a safe path in ∼13 seconds. Followers track offset curves from the leader's path, using a forward-propagated controller to ensure feasibility. Inter-agent collision constraints are not considered.

This problem is challenging since it includes (A) multiple safety constraints, (B) tight input bounds, and (C) complicated h_j expressions that are difficult to handle analytically.

Applying `gatekeeper`: We define the backup set as the leader's trajectory:

$$\mathcal{C} = \{x \in \mathcal{X} : \exists \tau \in [t_0, t_f] \text{ s.t. } p^L(\tau) = x\}. \tag{23}$$

This set is forward invariant under the backup controller:

$$\pi^B(t, x) = u^L(t - t' + \tau), \quad \forall t \geq t', \tag{24}$$

where at time t' the robot joins the leaders path: $x(t') = p^L(\tau)$. Thus, π^B keeps the agent on $\mathcal{C}$ for all $t \geq t'$. To compute backup trajectories, we use Dubins shortest paths (using [20], [21]) to a discrete set of points in $\mathcal{C}$, rejecting unsafe paths and selecting the shortest safe one. Candidate trajectories are constructed by minimizing the cost in (18), using the norm $\|x - x_{\text{nom}}\|_Q$ with $Q = \text{diag}(1, 1, 0)$.

Figure 1 depicts committed trajectories. They follow the nominal until nearing an EZ, then switch to tracking the leader's path. Since committed trajectories are updated, agents can rejoin the nominal trajectory after passing an EZ.

Results: Figure 2 summarizes the results. We compare `gatekeeper` against (A) Control Barrier Function Quadratic Programs (CBF-QPs) and (B) nonlinear trajectory optimization. The CBF-QP minimizes deviation from nominal input while satisfying $L_f h_j(x_i) + L_g h_j(x_i) u \geq -\alpha(h_j(x_i))$, for all j subject to input bounds. This may become infeasible due to multiple constraints or invalidity of h_j as a CBF. When infeasible, we re-solve a slacked version of the QP, minimizing the norm of the slack. The nonlinear trajectory optimization uses IPOPT [22], which replans every 0.2 TU for a 0.5 TU horizon, initialized using the nominal trajectory.

Comparing Figures 2a, 2b, 2c, we can see closed-loop trajectories of the agents as they navigate through the environment. Only the `gatekeeper` method shows no safety violations. For the CBF-QP method, the h_j functions may not be CBFs, especially under input constraints and the presence of multiple constraints, and therefore can lead to violations. The trajectory optimization method also has collisions, since there is no guarantee that IPOPT will find a feasible solution satisfying all the constraints.

Figures 2d, 2e, 2f compare the deviation of the trajectories from the desired trajectories. Here, we see that the CBF-QP method has large deviations, since the controller reduces the v to try to meet safety constraints. Furthermore, notice that the leader also deviates from the reference trajectory, even though the leaders path is safe. This is because even though the trajectory is safe, the trajectory approaches the boundary too quickly for the CBF condition to be satisfied. Comparing Figures 2e, 2f, notice that the deviations in `gatekeeper` and the trajectory optimization methods are comparable, indicating that `gatekeeper` has minimal sub-optimality.

Finally, we compare the computation load. The total computation time required to generate the closed-loop trajectory all three agents is as follows: CBF-QP: 9.49 s, trajectory optimization: 302.65 s, `gatekeeper`: 3.61 s. Ergo, CBF-QP and `gatekeeper` required only 3.1% and 1.2% of the computational time of nonlinear trajectory optimization, but only `gatekeeper` resulted in safe trajectories. Our proposed approach is significantly computationally cheaper and has strong guarantees of constraint satisfaction, despite multiple state and input constraints.

IV. Conclusions

We have presented `gatekeeper`, a flexible safety framework that guarantees safe execution in real-time planning systems by committing to a safety-verified trajectory with a known backup strategy. In particular, we quantify the suboptimality of the `gatekeeper` approach, a quantity that we can compute in real-time.

We demonstrated `gatekeeper` on a challenging multi-agent formation flight task with tight safety margins and complex constraints. Compared to CBF-based control and trajectory optimization, `gatekeeper` was the only method to maintain safety throughout, while also being significantly faster and only minimally suboptimal.

These results suggest `gatekeeper` is a promising direction for safety-critical robotics, especially when robustness and computational efficiency are paramount. Future work includes extending `gatekeeper` to handle inter-agent collision constraints in a distributed communication network.

References

[1] E. E. Alves, D. Bhatt, B. Hall, K. Driscoll, A. Murugesan, and J. Rushby, "Considerations in assuring safety of increasingly autonomous systems," Tech. Rep., 2018.

[2] D. R. Agrawal, R. Chen, and D. Panagou, "gatekeeper: Online safety verification and control for nonlinear systems in dynamic environments," *IEEE TRO*, 2024.

[3] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[4] B. T. Lopez, J.-J. E. Slotine, and J. P. How, "Dynamic tube mpc for nonlinear systems," in *IEEE ACC*. IEEE, 2019, pp. 1655–1662.

[5] J. Yin, O. So, E. Y. Yu, C. Fan, and P. Tsiotras, "Safe beyond the horizon: Efficient sampling-based mpc with neural control barrier functions," *arXiv preprint arXiv:2502.15006*, 2025.

[6] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE TAC*, vol. 62, no. 8, pp. 3861–3876, 2016.

[7] K. Garg, J. Usevitch, J. Breeden, M. Black, D. Agrawal, H. Parwana, and D. Panagou, "Advances in the theory of control barrier functions: Addressing practical challenges in safe control synthesis for autonomous and robotic systems," *Annual Reviews in Control*, vol. 57, p. 100945, 2024.

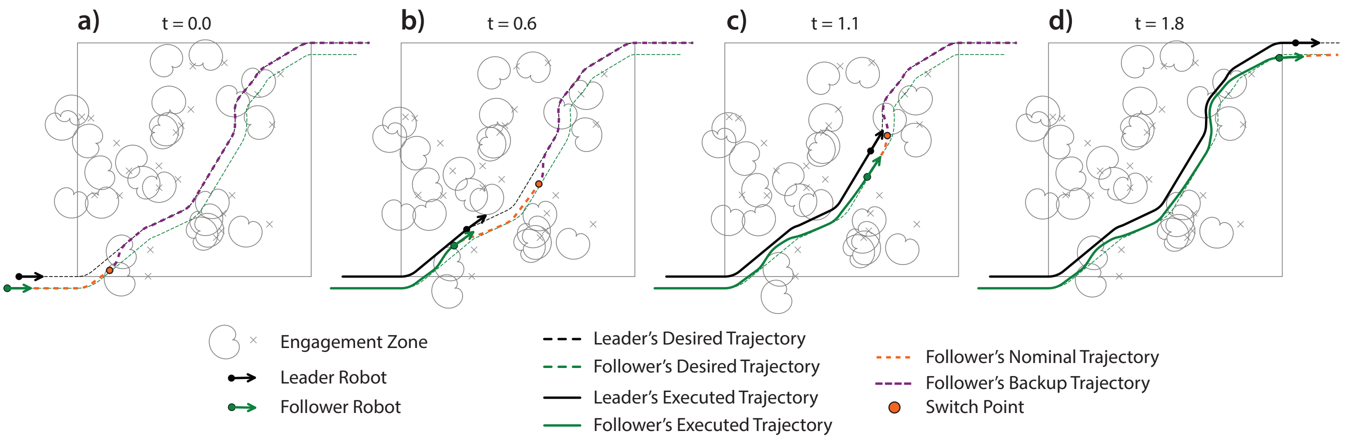
a)
t = 0.0
b)
t = 0.6
c)
t = 1.1
d)
t = 1.8
Engagement Zone
Leader Robot
Follower Robot
Leader's Desired Trajectory
Follower's Desired Trajectory
Leader's Executed Trajectory
Follower's Executed Trajectory
Follower's Nominal Trajectory
Follower's Backup Trajectory
Switch Point

Fig. 1. Formation flight using `gatekeeper`. The leader follows an RRT* path (black dashed); followers track offset curves. The committed trajectory (green) includes nominal (orange dashed) and backup (purple dashed) trajectories.

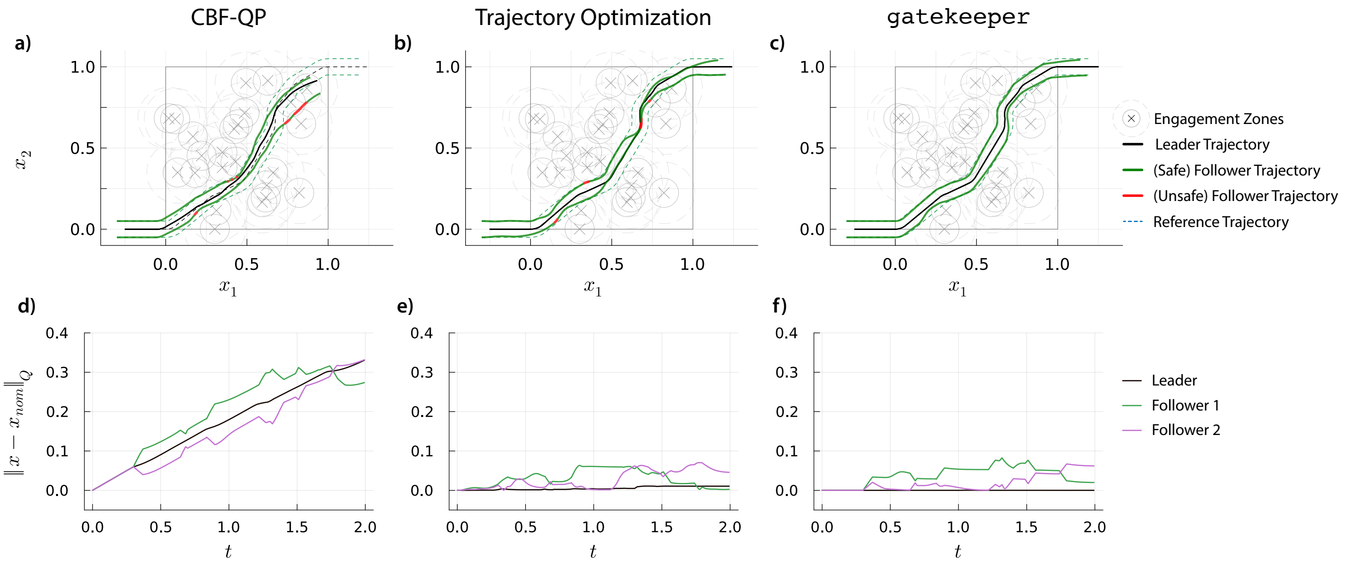

Fig. 2. Simulation Results. (a, b, c) depict the trajectories of the leader and the follower agents through the domain with 24 EZs using the different planning methods. The leader's trajectory is drawn in black, while the follower's trajectories are drawn in green (when not in collision with a EZ), and in red (when in collision). While CBF-QP and IPOPT have safety violations, `gatekeeper` does not. (d, e, f) show the deviation of the agents from the desired trajectory, with $\boldsymbol{Q} = \mathbf{diag}(\mathbf{1}, \mathbf{1}, \mathbf{0})$. While the deviation in IPOPT and `gatekeeper` are similar, the deviation is higher for the CBF-QP.

[8] M. H. Cohen, T. G. Molnar, and A. D. Ames, "Safety-critical control for autonomous systems: Control barrier functions via reduced-order models," *Annual Reviews in Control*, vol. 57, p. 100947, 2024.

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

[10] M. Ganai, S. Gao, and S. Herbert, "Hamilton-jacobi reachability in reinforcement learning: A survey," *IEEE Open Journal of Control Systems*, 2024.

[11] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *IEEE/RSJ IROS*. IEEE, 2019, pp. 1934–1940.

[12] 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 (CDC)*. IEEE, 2021, pp. 6835–6841.

[13] A. Singletary, A. Swann, I. D. J. Rodriguez, and A. D. Ames, "Safe drone flight with time-varying backup controllers," in *IEEE/RSJ IROS*. IEEE, 2022, pp. 4577–4584.

[14] K. L. Hobbs, M. L. Mote, M. C. Abate, S. D. Coogan, and E. M. Feron, "Runtime assurance for safety-critical systems: An introduction to safety filtering approaches for complex control systems," *IEEE Control Systems Magazine*, vol. 43, no. 2, pp. 28–65, 2023.

[15] L. Jung, A. Estornell, and M. Everett, "Contingency constrained planning with mppi within mppi," *arXiv preprint arXiv:2412.09777*, 2024.

[16] N. C. Janwani, E. Daş, T. Touma, S. X. Wei, T. G. Molnar, and J. W. Burdick, "A learning-based framework for safe human-robot collaboration with multiple backup control barrier functions," in *IEEE ICRA*. IEEE, 2024, pp. 11 676–11 682.

[17] D. Ko and W. K. Chung, "A backup control barrier function approach for safety-critical control of mechanical systems under multiple constraints," *IEEE/ASME Transactions on Mechatronics*, 2024.

[18] T. Chapman, I. E. Weintraub, A. Von Moll, and E. Garcia, "Engagement zones for a turn constrained pursuer," *arXiv preprint arXiv:2502.00364*, 2025.

[19] A. Wolek, I. E. Weintraub, A. Von Moll, D. Casbeer, and S. G. Manyam, "Sampling-based risk-aware path planning around dynamic engagement zones," *IFAC-PapersOnLine*, vol. 58, no. 28, pp. 594–599, 2024.

[20] A. M. Shkel and V. Lumelsky, "Classification of the dubins set," *Robotics and Autonomous Systems*, vol. 34, no. 4, pp. 179–202, 2001.

[21] K. Sundar, "Dubins.jl," 2018. [Online]. Available: https://github.com/kaarthiksundar/Dubins.jl

[22] J. L. Pulsipher, W. Zhang, T. J. Hongisto, and V. M. Zavala, "A unifying modeling abstraction for infinite-dimensional optimization," *Computers & Chemical Engineering*, vol. 156, 2022.