Multi-Agent `gatekeeper`: Safe Flight Planning and Formation Control for Urban Air Mobility

Marshall Vielmetti*
University of Michigan, Ann Arbor, MI, 48109, USA

Devansh R. Agrawal†
University of Michigan, Ann Arbor, MI, 48109, USA

Dimitra Panagou‡
University of Michigan, Ann Arbor, MI, 48109, USA

We present Multi-Agent `gatekeeper`, a framework that provides provable safety guarantees for leader-follower formation control in cluttered 3D environments. Existing methods face a trad-off: online planners and controllers lack formal safety guarantees, while offline planners lack adaptability to changes in the number of agents or desired formation. To address this gap, we propose a hybrid architecture where a single leader tracks a pre-computed, safe trajectory, which serves as a shared trajectory backup set for all follower agents. Followers execute a nominal formation-keeping tracking controller, and are guaranteed to remain safe by always possessing a known-safe backup maneuver along the leader's path. We formally prove this method ensures collision avoidance with both static obstacles and other agents. The primary contributions are: (1) the multi-agent `gatekeeper` algorithm, which extends our single-agent gatekeeper framework to multi-agent systems; (2) the trajectory backup set for provably safe inter-agent coordination for leader-follower formation control; and (3) the first application of the gatekeeper framework in a 3D environment. We demonstrate our approach in a simulated 3D urban environment, where it achieved a 100% collision-avoidance success rate across 100 randomized trials, significantly outperforming baseline CBF and NMPC methods. Finally, we demonstrate the physical feasibility of the resulting trajectories on a team of quadcopters.

I. Nomenclature

$\mathcal{X}, \mathcal{X}_{\text{obs}}, \mathcal{S}$	=	State space, Obstacle space, Safe set
$\mathcal{U}$	=	Set of admissible control inputs.
$\mathcal{C}$	=	Backup set
$\psi, \gamma, \omega \in \mathbb{R}$	=	Heading angle (rad), Climb angle (rad), Turning Rate (rad/s)
$\delta \in \mathbb{R}_{\geq 0}$	=	Minimum inter-agent separation distance
$\epsilon \in \mathbb{R}_{\geq 0}$	=	Separation buffer to account for curvature
$d_i^* \in \mathbb{R}^d$	=	Desired offset of follower agent i from leader
$r_i \in \mathbb{R}^d$	=	Position component of state of agent i.
$N \in \mathbb{N}$	=	Number of agents
$\pi^{\text{nom}}, \pi^{\text{bak}}$	=	Nominal controller, Backup controller
$p^{\text{nom}}_{k,i}, u^{\text{nom}}_{k_i,i}$	=	k-th Nominal Trajectory of agent i.
$p^{\text{can}}_{k,i}, u^{\text{nom}}_{k_i,i}$	=	k-th Candidate Trajectory of agent i.
$p^{\text{com}}_{k,i}, u^{\text{nom}}_{k_i,i}$	=	k-th Committed Trajectory of agent i.
p_L, u_L	=	Precomputed leader's trajectory
$T_H, T_B \in \mathbb{R}_{\geq 0}$	=	Planning horizon, backup time horizon

*Masters Student, Department of Electrical & Computer Engineering, 1301 Beal Ave, Ann Arbor, MI 48109
†Postdoc, Department of Robotics, 1320 Beal Ave, Ann Arbor, MI 48109.
‡Associate Professor, Department of Robotics and Department of Aerospace Engineering, 2505 Hayward St, Ann Arbor, MI 48109).

II. Introduction

Multi-agent formation control has a wide variety of applications, including surveillance [1], exploration [2], and search and rescue [3]. However, providing rigorous safety guarantees for complex multi-agent systems operating in real-world environments remains a significant challenge in deploying robotics in sensitive applications.

Methods for multi-agent formation control can be broadly classified into three categories: (1) leader-follower, (2) virtual structures, and (3) behavior-based methods [4, 5]. Leader-follower methods designate some agents as "leaders" and the rest as "followers". Leader agents execute some desired trajectory, while followers attempt to maintain some fixed offset from the leader [6, 7]. These methods are appealing due to their simplicity, but often lack rigorous safety guarantees, as followers may collide with obstacles or each other while attempting to maintain formation [8, 9]. Virtual-structure based methods treat the entire formation as a single entity, and attempt to maintain a rigid formation by controlling the motion of this virtual structure [10–12]. These methods can provide strong formation tracking guarantees, but often struggle to adapt to changes in the number of agents or desired formation, and are difficult to scale to large numbers of agents [13]. Behavior-based methods define a set of behaviors for each agent, such as obstacle avoidance, formation keeping, and goal seeking, and combine these behaviors to generate control inputs [14]. These methods are often robust to changes in the environment and number of agents, but can be difficult to tune and may not provide formal safety guarantees [4]. In some cases, it is desirable to pre-plan safe, dynamically feasible trajectories offline, such that they may be executed by agents without having to worry about achieving real time performance. Methods like multi-agent RRT* [15] and other planning techniques [5, 16] can be used to generate safe trajectories offline, but these methods fail to scale to large numbers of agents, or high-dimensional state spaces. Furthermore, if the desired formation or number of agents change, these offline solutions become unusable.

Thus, there is a need for methods that can provide rigorous safety guarantees, while being adaptable to changes in the number of agents or desired formation, and computationally efficient enough to run online in real time. This gap motivates our approach, multi-agent `gatekeeper`. We propose a hybrid online-offline leader-follower approach, where a single path is computed offline for a designated leader-agent, and followers attempt to maintain a formation around the leader while deviating to ensure safety. Our approach provides rigorous theoretical safety guarantees, while being adaptable to changes in the number of agents or desired formation, and is computationally efficient enough to run online in real time. This paper extends our lab's previous work in online safety verification and control for single-agents, `gatekeeper` [17], which we extend to multi-agent systems for the first time, in the context of a distributed, leader-follower formation control problem. Our contributions are as follows:

1) We present multi-agent `gatekeeper`, an adaptation of our single-agent safety framework `gatekeeper` to multi-agent systems to achieve formal safety guarantees.
2) We present the first application of `gatekeeper` in a 3D environment, demonstrating our approach on a 3D Dubins aircraft model in a dense, urban-like environment.
3) We apply our method to a leader-follower formation control problem in both simulation and on hardware, demonstrating the ability to maintain formation while ensuring safety, and showing our method achieves safety where baselines fail.

This paper is organized as follows: Section III introduces the preliminaries and problem statement, section IV details our proposed solution, section V describes our simulated experiments and baseline comparisons, section VI describes our hardware experiments, and finally conclusions and future directions are discussed in section VII.

III. Preliminaries & Problem Statement

We will first introduce some preliminaries and notation, then formally state the problem we are trying to solve.

A. Preliminaries

Consider a nonlinear system,

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

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$ is the state and $u \in \mathcal{U} \subseteq \mathbb{R}^m$ is the control input. $f : \mathcal{X} \times \mathcal{U} \to \mathbb{R}^n$ is piecewise continuous and locally Lipschitz in x and u. Let $r \in \mathbb{R}^d$, $d \in \{2, 3\}$ denote the position component of the state x.

Definition 1 (Agent)**.** *An **agent** $i \in V$ represents a robotic system, which satisfies the following:*

1) The agent's state x_i evolves through a system satisfying (1)*.*
2) Collision radius $\delta \in \mathbb{R}_{>0}$. Agents must maintain a minimum separation of δ.
3) Agents can communicate over some undirected graph $\mathcal{G}$ as in Def. 4.

*Agents can be classified as **leaders** or **followers**. Leaders execute some desired trajectory, while followers attempt to maintain some fixed offset from the leader.*

Agents attempt to navigate through a known environment, which contains some set of static obstacles $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$. The safe set is defined as $\mathcal{S} = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$.

Definition 2 (Trajectory)**.** *A **trajectory** is the tuple (T, p, u), where $T \subseteq \mathbb{R}_{\geq 0}$ is a time interval, $p : T \to \mathcal{X}$ is the state trajectory, and $u : T \to \mathcal{U}$ is a control trajectory, which together satisfy the system dynamics,*

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

The position component of the trajectory is denoted $r(t)$, where $r(t)$ is the position component of $p(t)$.

Definition 3 (Arc Length)**.** *Given a trajectory (T, p, u) the **arc length** between times $t_1, t_2 \in T$ is defined as,*

$$d(t_1, t_2) = \int_{t_1}^{t_2} \|\dot{r}(\tau)\| \, d\tau \tag{3}$$

Definition 4 (Undirected Graph)**.** *Let $\mathcal{G} = (V, E)$ be an undirected graph, where V is the set of vertices and $E \subseteq V \times V$ is the set of edges. Each vertex $i \in V$ represents an agent, and an edge $(i, j) \in E$ indicates that agents i and j can communicate with each other.*

Assumption 1 (Communications)**.** *We assume $\mathcal{G}$ is fully connected, and has no communication delay or bandwidth limitations.*

Remark 1. *Assumption 1 is adopted to simplify the safety proof. Future work will investigate relaxing this assumption to allow for time-varying graphs, communication delays, and bandwidth constraints.*

B. `gatekeeper` Preliminaries

We now introduce definitions and assumptions from `gatekeeper` [17, 18], which forms the basis of our approach.

Definition 5 (Backup Set)**.** *A **backup set**, denoted $C \subseteq \mathcal{S}$ is some subset of the safe set, defined with some feedback controller $\pi^{\text{bak}} : \mathcal{X} \to \mathcal{U}$ (the **backup controller**), such that the backup controller renders the backup set control-invariant. That is, the closed-loop system $\dot{x} = f(x, \pi^{\text{bak}}(x))$ satisfies*

$$x(t_k) \in C \implies x(t) \in C, \forall t \geq t_k. \tag{4}$$

Definition 6 (Nominal Trajectory)**.** *From some state $x(t_k) \in \mathcal{X}$ at time t_k, a **nominal trajectory** is the tuple $([t_k, t_k + T_H], p_k^{\text{nom}}, u_k^{\text{nom}})$, where $T_H \in \mathbb{R}_{\geq 0}$ is the planning horizon.*

Nominal trajectories are often generated by online planners, and may violate safety constraints, or terminate in some unrecoverable state.

Definition 7 (Backup Trajectory)**.** *Let $T_B \in \mathbb{R}_{\geq 0}$ be finite. For any $t_s \in \mathbb{R}_{\geq 0}$, and $x_s \in \mathcal{X}$, a trajectory $([t_s, \infty), p_k^{\text{bak}}, u_k^{\text{bak}})$ is a **backup trajectory** if:*

$$p_k^{\text{bak}}(t_s + T_B) \in C, \tag{5}$$

and for all $t \geq t_s + T_B$

$$u^{\text{bak}}(t) = \pi^{\text{bak}}(p^{\text{bak}}(t)), \tag{6}$$

where $C \subset S$ is a backup set as in Def. 5, and π^B is a backup controller that renders C control-invariant.

Definition 8 (Candidate Trajectory)**.** *Given a state $x(t_k) \in \mathcal{X}$ at some time $t_k \in \mathbb{R}_{\geq 0}$, a **candidate trajectory** consists of a nominal trajectory, switch time, and backup trajectory. A candidate with switch time t_s is denoted $(p^{\text{can},t_s}, u^{\text{can},t_s})$ and defined over $[t_k, \infty)$. The candidate is defined as a piecewise trajectory,*

$$p^{\text{can},t_s}(t), u^{\text{can},t_s}(t) = \begin{cases} p^{\text{nom}}(t), u^{\text{nom}}(t), & t \in [t_k, t_s) \\ p^{\text{bak}}(t), u^{\text{bak}}(t), & t \in [t_s, \infty). \end{cases} \tag{7}$$

A candidate controller is thus defined by an agent executing its nominal trajectory until it reaches the switch time, then a backup controller for all future time.

C. Problem Statement

We now introduce a formation control problem, where agents attempt to maintain some fixed offset from a designated leader, while avoiding obstacles and each other.

Definition 9 (Formation Control)**.** *Leader-follower **formation control** is defined by designating one agent as a leader, and the rest as followers. Each follower agent i then attempts to maintain some fixed offset d_i^* from the leader, while the leader executes its desired trajectory.*

Definition 10 (Formation Error)**.** *Given a formation control problem with $i = 1 \dots N-1$ follower agents attempting to maintain some fixed displacement $d_i^* \in \mathbb{R}^d$ relative to a designated leader agent, we can write the **formation error** at time t_k as*

$$E(t_k) = \sum_{i=1}^{N-1} \left\| (r_L(t_k) - r_i(t_k)) - d_i^* \right\|^2, \tag{8}$$

where $r_L(t_k)$ is the position of the leader agent at time t_k, $r_i(t_k)$ is the position of the follower agent, and d_i^* is the desired displacement of the follower agent.

We will seek to minimize formation error of follower agents, but prioritize deviations from the nominal formation keeping controller to maintain safety.

Problem 1 (Formation Control Problem)**.** *Consider a set V of N homogeneous agents defined as in Def. 1, consisting of one designated leader and $N-1$ followers. Agents navigate a known environment, and must remain within the safe set $\mathcal{S}$ as defined in section III.A, and avoid inter-agent collisions. Agents communicate over an undirected graph $\mathcal{G} = (V, E)$ satisfying Assumption 1.*

The leader agent executes a pre-computed, safe, dynamically feasible trajectory $([t_0, t_f], p_L, u_L)$ while each follower i seeks to track a fixed offset d_i^* relative to the leader. We aim to minimize the total formation error $E(t)$ over the mission horizon $[t_0, t_f]$, while ensuring safety constraints are satisfied. Formally, the global formation control problem is stated as,

$$\min \quad \int_{t_0}^{t_f} E(t)\, dt \tag{9a}$$

$$s.t. \quad x_i(t) \in \mathcal{S}, \quad i = 1, \dots, N,\ t \in [t_0, t_f] \tag{9b}$$

$$\left\| r_i(t) - r_j(t) \right\| \geq \delta, \quad \forall i \neq j,\ t \in [t_0, t_f] \tag{9c}$$

$$\dot{x}_i(t) = f(x_i(t), u_i(t)), \quad i = 1, \dots, N,\ t \in [t_0, t_f] \tag{9d}$$

IV. Proposed Solution

We propose multi-agent `gatekeeper`, an extension of our single-agent safety framework `gatekeeper` [17] to multi-agent systems. While the original framework provides formal safety guarantees in for single agents, even in the presence of dynamic obstacles, this extension addresses the challenge of avoiding other agents.

Our key insight is that inter-agent communication allows agents to treat other agents as predictable, time-varying obstacles. By sharing committed trajectories, agents can certify candidates as safe in a distributed manner.

We first present the general multi-agent `gatekeeper` algorithm in section IV.A. We then apply this algorithm to the leader-follower formation control Problem 1 in section IV.B, leveraging the leader's path as a shared backup maneuver to provide formal proofs of safety in section IV.C.

A. Multi-Agent gatekeeper

Each follower agent i executes multi-agent `gatekeeper` independently (algorithm 1) at discrete iterations $k_i \in \mathbb{N}$, which occur at times t_{k_i}, such that $t_{k_i} < t_{k_i+1}$. Our method is *computationally distributed and asynchronous* (agents plan locally) but *informationally centralized* (agents require the committed trajectories of all other agents).

At every iteration k_i, agent i performs the following:

1) **Receive** current committed trajectories of all other agents, $\mathcal{P}^{\text{com}}$.
2) **Construct** a goal-oriented nominal trajectory $([t_{k_i,i}, t_{k_i,i} + T_H], p_{k_i,i}^{\text{nom}}, u_{k_i,i}^{\text{nom}})$ over the horizon T_H.
3) **Attempt** to construct a candidate trajectory $([t_{k_i,i}, \infty), p_{k_i,i}^{\text{can}}, u_{k_i,i}^{\text{can}})$ as in Def. 8 using algorithm 2.
4) **Commit** to the candidate trajectory if valid (Def. 11), otherwise, continue the previous committed trajectory.

Algorithm 1: Multi-Agent `gatekeeper` (Agent i at time t_{k_i})

$\mathcal{P}_{\text{com}} \leftarrow p^{\text{com}}_{k_j,j},\ \forall j \neq i$ `// Receive other committed trajectories.`

$([t_{k_i,i}, t_{k_i,i} + T_H], p^{\text{nom}}_i, u^{\text{nom}}_i) \leftarrow (x^{\text{nom}}_i(t), \pi^{\text{nom}}_i(x^{\text{nom}}_i(t)))$ `// Propagate Nominal.`

$([t_{k_i,i}, \infty), p^{\text{can},t_s}_i, u^{\text{can},t_s}_i) \leftarrow \textsc{ConstructCandidate}(p^{\text{nom}}_i, u^{\text{nom}}_i, \mathcal{P}_{\text{com}})$ `// Try to construct candidate.`

if $([t_{k_i,i}, \infty), p^{\text{can},t_s}_i, u^{\text{can},t_s}_i) \neq \textit{null}$ **then**

  $([t_{k_i,i}, \infty), p^{\text{com}}_{k_i,i}, u^{\text{com}}_{k_i,i}) \leftarrow ([t_{k_i,i}, \infty), p^{\text{can},t_s}_i, u^{\text{can},t_s}_i)$ `// Commit to valid candidate.`

else

  $([t_{k_i,i}, \infty), p^{\text{com}}_{k_i,i}, u^{\text{com}}_{k_i,i}) \leftarrow ([t_{k_i,i}, \infty), p^{\text{com}}_{k_i-1,i}, u^{\text{com}}_{k_i-1,i})$ `// Failure: continue previous.`

Assumption 2 (Sequential Consistency)**.** *We assume the planning iterations of agents are non-overlapping. That is, for agents i, j, the planning iterations k_i, k_j occur at distinct times $t_{k_i} \neq t_{k_j}$ for all $i \neq j$. Furthermore, we assume the computation time Δt_i is negligible with respect to the system dynamics, such that the state of the system remains effectively constant during planning iterations.*

Remark 2. *This assumption guarantees sequential consistency, preventing race conditions where multiple agents simultaneously attempt to update their committed trajectories based on outdated information. In practice, this is enforced by a Time-Division Multiple Access (TDMA) communication scheme, where each agent is assigned a unique time slot to perform planning iterations if necessary. This is also a common assumption in the literature, e.g. [19].*

We will now define the notions of validity and a committed trajectory, then the multi-agent `gatekeeper` algorithm.

Definition 11 (Valid)**.** *The k-th candidate trajectory of agent i, denoted $([t_{k_i,i}, \infty), p^{\text{can}}_{k_i,i}, u^{\text{can}}_{k_i,i})$, is considered **valid** if, 1) it remains in the safe set,*

$$p^{\text{can}}_{k_i,i}(t) \in \mathcal{S}, \forall t \geq t_{k_i}, \tag{10}$$

2) reaches a backup set $\mathcal{C}_{k_i,i}$ at some finite time $t_{k_i} + t_s + T_B$, and remains there for all future time,

$$p^{\text{can}}_{k_i,i}(t) \in \mathcal{C}_{k_i,i}, \forall t \geq t_s + T_B, \tag{11}$$

3) and is collision free with respect to the committed trajectories of all other agents,

$$\left\| p^{\text{can}}_{k_i,i}(t) - p^{\text{com}}_{k_j,j}(t) \right\| \geq \delta, \forall j \neq i, \forall t \geq t_{k_i}. \tag{12}$$

Definition 12 (Committed Trajectory)**.** *At the kth iteration of agent i, let the agent construct some goal-oriented nominal trajectory $[t_{k_i,i}, t_{k_i,i} + T_H], (p^{\text{nom}}_{k_i,i}, u^{\text{nom}}_{k_i,i})$. Define the set of all valid switch times,*

$$\mathcal{I}_{k_i,i} = \{t_S \in [0, T_H] \mid ([t_{k_i,i}, \infty), p^{\text{can},t_s}_{k_i,i}, u^{\text{can},t_s}_{k_i,i}) \textit{ is valid}\}, \tag{13}$$

where $([t_{k_i,i}, \infty), p^{\text{can},t_s}_{k_i,i}, u^{\text{can},t_s}_{k_i,i})$ is the candidate trajectory with switch time t_s as in Def. 8, and validity is determined by Def. 11. If $\mathcal{I}_{k_i,i} \neq \varnothing$, let $t_s = \max(\mathcal{I}_{k_i,i})$. The agent then commits the candidate trajectory with switch time t_s,

$$([t_{k_i,i}, \infty), p^{\text{com}}_{k_i,i}, u^{\text{com}}_{k_i,i}) = ([t_{k_i,i}, \infty), p^{\text{can},t_s}_{k_i,i}, u^{\text{can},t_s}_{k_i,i}), \tag{14}$$

and thus must execute this trajectory until a new trajectory is committed. If $\mathcal{I}_{k_i,i} = \varnothing$, the agent continues executing its previously committed trajectory, $([t_{k_i,i}, \infty), p^{\text{com}}_{k_i,i}, u^{\text{com}}_{k_i,i}) = ([t_{k_i-1,i}, \infty), p^{\text{com}}_{k_i-1,i}, u^{\text{com}}_{k_i-1,i})$.

B. Formation Control With Multi-Agent `gatekeeper`

We now discuss the specific application of multi-agent `gatekeeper` to the leader-follower formation control Problem 1. Our key insight is to leverage the pre-computed leader trajectory as a *shared, guaranteed-safe backup maneuver* for all followers. Since the trajectory is known to be safe and dynamically feasible, any follower that merges onto this path and *maintains appropriate separation from other agents* is guaranteed to remain safe for all future time.

Definition 13 (Trajectory Backup Set)**.** *Let* $([t_0, t_f], p_L, u_L)$ *be a safe, dynamically feasible leader trajectory. For any state* x *coinciding with the leader's path at parameter* $t_L \in [t_0, t_f]$ *such that* $x = p_L(t_L)$*, and any current time* t_{now}*, we define the* ***trajectory backup controller*** *as,*

$$\pi^{\text{bak}}(x, t) = u_L(t_L + (t - t_{now})), \tag{15}$$

Executing this controller ensures the agent tracks the leader's path p_L *starting from* t_L*, effectively rendering the leader's trajectory a backup set.*

However, due to the risk of inter-agent collisions, we must still verify that the trajectory backup set is safe with respect to other agents before it can be committed.

This requires us to pose the following requirements on the leader's trajectory.

Definition 14 (Valid Trajectory Backup Set)**.** *A trajectory* $([t_0, t_f], p_L(t), u_L(t))$ *is a* ***valid trajectory backup set*** *if,* ***1)*** *It is safe with respect to static obstacles,*

$$p_L(t) \in \mathcal{S}, \forall t \in [t_0, t_f], \tag{16}$$

2) *It is dynamically feasible, and* ***3)*** *there exists a curvature margin* $\epsilon \in \mathbb{R}_{\geq 0}$ *such that for all path parameters* $t_1, t_2 \in [t_0, t_f]$ *and future offsets* $\tau \geq 0$*,*

$$d(t_1, t_2) \geq \delta + \epsilon \implies \|r_L(t_1 + \tau) - r_L(t_2 + \tau)\| \geq \delta \tag{17}$$

where $d(\cdot, \cdot)$ *is the arc length along the leader's trajectory as in Def. 3.*

The condition eq. (17) accounts for the curvature of the leader's trajectory. It ensures that an arc length separation of $\delta + \epsilon$ guarantees at least a safe δ of Euclidian distance for all future time. In practice, a conservative bound on ϵ can be computed by sampling the leader path and computing pairwise Euclidean distances over bounded lookahead windows.

Lemma 1 (Validity of a Trajectory Backup Set)**.** *Let the leader trajectory* $([t_0, t_f], p_L(t), u_L(t))$ *be a valid trajectory backup set by Def. 14. Suppose agents* i *and* j *begin to execute* π^{bak} *from the leader path at times* t_i *and* t_j *respectively, at path parameters* t_{Li} *and* t_{Lj} *such that,*

$$p_i(t_i) = p_L(t_{Li}), \quad p_j(t_j) = p_L(t_{Lj}). \tag{18}$$

If the arc-length separation between the agents at time $t_c = \max(t_i, t_j)$ *satisfies,*

$$d(t_{Li} + (t_c - t_i), t_{Lj} + (t_c - t_j)) \geq \delta + \epsilon \implies \|r_i(t) - r_j(t)\| \geq \delta, \ \forall t \geq t_c, \tag{19}$$

i.e. the agents remain safe with respect to each other for all future time.

Proof. WLOG, assume $t_i \leq t_j$, thus $t_c = t_j$. At time t_c, agent j has just joined the path at parameter t_{Lj}, and agent i has been traveling on the path for $(t_j - t_i)$ seconds, reaching parameter $t_{Li} + (t_j - t_i)$. By assumption, the arc length distance between these two parameters satisfies the condition in Def. 14. Thus, for all $\tau \geq 0$,

$$\left\|r_L((t_{Li} + (t_j - t_i)) + \tau) - r_L(t_{Lj} + \tau)\right\| \geq \delta. \tag{20}$$

Substituting $\tau = t - t_j$ yields $\left\|r_i(t) - r_j(t)\right\| \geq \delta, \forall t \geq t_j$. □

Therefore, by ensuring that the trajectory backup set satisfies Def. 14, we can guarantee that if agents joining the leader's trajectory are spaced sufficiently far apart (satisfy the conditions of lemma 1), they will remain safe for all future time.

C. Backup Trajectory Generation and Coordination

While lemma 1 provides the theoretical condition for safety, it does not specify how agents distributively coordinate to claim "slots" on the leader's path, nor how they generate the transition trajectories to reach them.

Algorithm 2: Construct Candidate

1 $\mathcal{I}_{valid} \leftarrow \varnothing$
2 **for** $t_s \in [t_k, t_k + T_H]$ **do**
3 $\quad x_s \leftarrow p_i^{nom}(t_s)$
$\quad$ `// Plan rejoin leader maneuver`
4 $\quad ([t_{k_i,i}+t_s, t_{k_i,i}+t_s+T_B], p_i^{join}, u_i^{join}), t_{Li} \leftarrow$ PlanJoinToBackup(x_s, p_L)
5 $\quad$ **if** $t_{Li} = null$ **then**
6 $\quad\quad$ **continue**
7 $\quad (t_{k_i,i}, \infty), p_i^{can}, u_i^{can}) \leftarrow (p_i^{nom}|_{[t_k,t_s]}, p_i^{join}, p_L|_{[t_{Li},\infty)})$
8 $\quad$ **if** ValidateCandidate$(p_i^{can}, u_i^{can}, \mathcal{P}_{com}, t_{Li})$ **then**
9 $\quad\quad \mathcal{I}_{valid} \leftarrow \mathcal{I}_{valid} \cup \{t_s\}$
10 **if** $\mathcal{I}_{valid} \neq \varnothing$ **then**
11 $\quad t_s^* \leftarrow \max(\mathcal{I}_{valid})$
12 $\quad$ **return** candidate with switch time t_s^*
13 **else**
14 $\quad$ **return** null

Algorithm 3: Validate Candidate

$\quad$ `// Check static obstacle collisions`
1 **for** $t \in [t_k, t_{Li}]$ **do**
2 $\quad$ **if** $p_i^{can}(t) \notin \mathcal{S}$ **then**
3 $\quad\quad$ **return false**
$\quad$ `// Validate against all other agents' committed trajectories`
4 **forall** $p_j^{com} \in \mathcal{P}_{com}$ **do**
$\quad\quad$ `// Check inter-agent collisions`
5 $\quad\quad t_{max} \leftarrow \max(t_{Li}, t_{Lj})$ **for** $t \in [t_k, t_{max})$ **do**
6 $\quad\quad\quad$ **if** $\|r_i^{can}(t) - r_j^{com}(t)\| < \delta$ **then**
7 $\quad\quad\quad\quad$ **return false**
$\quad\quad$ `// Ensure backup slot separation (Lemma 1)`
8 $\quad\quad$ **if** $\|r_i^{can}(t_{max}) - r_j^{com}(t_{max})\| < \delta + \epsilon$ **then**
9 $\quad\quad\quad$ **return false**
10 **return true**

1. Algorithm 2: Construct Candidate

The goal of algorithm 2 is to identify a valid switch time t_s that allows the agent to transition from its nominal trajectory to the leader's trajectory, while maintaining safety. The algorithm iterates through discrete times steps within the planning horizon (Line 2). At each potential switch time t_s, the agent attempts to compute a feasible "join" trajectory (p_i^{join}, u_i^{join}) that connects the agent's nominal $x_i^{\text{nom}}(t_s)$ to some point on the leader's trajectory p_L (Line 4). If a feasible trajectory is found, the agent identifies the merge parameter t_{Li} (point on the leader's path where the join occurrs). It then constructs a full candidate (Line 7) as Nominal -> Join -> Leader Path, and validates it agains static obstacles and other agents' committed trajectories using algorithm 3 (Line 8). Finally, if any valid candidates were found, the candidate with the maximum switch time is returned (Lines 11-15).

2. Algorithm 3: Validate Candidate

Algorithm 3 provides the computational procedure for validating a candidate trajectory (Def. 11). Three conditions must be verified,

1) *Static Safety*: The candidate must remain in the safe set (Lines 1-3).
2) *Transition Safety*: The candidate must avoid collisions with all other agents' committed trajectories during the transition period (Lines 4-7).
3) *Backup Slot Safety*: Once the agent merges with the leader's trajectory, it must satisfy the separation conditions of lemma 1 with respect to all other agents (Lines 8-9).

Note that line 8 uses Euclidean distance as a conservative approximation of the arc length condition. Since the arc length is always greater than or equal to the Euclidean distance, $(d(a, b) \geq \|r_L(a) - r_L(b)\|)$, satisfying the Euclidean condition is a computationally simpler sufficient condition for lemma 1

3. Implementation Details

In order to guarantee safety, we require that every agent is able to construct a valid committed trajectory at the initial time t_0. Each agent should construct their initial committed trajectory sequentially, with respect to the committed trajectories of all previous agents. If an agent is unable to construct a valid committed trajectory at time t_0, the algorithm is infeasible.

Note the use of euclidean distance in algorithm 3 to verify sufficient separation along the leader's trajectory at the join time t_{Li}. This is a conservative approximation of the arc length condition in Def. 14, but is simpler to compute and

suffices in practice. The arc length separation condition implies Euclidian separation if the maximum curvature of the path is bounded.

The function PlanJoinToBackup(x_s, p_L) computes a dynamically feasible trajectory from state x_s to some point on the leader's trajectory p_L, returning both the join trajectory and the time t_{Li} at which the agent will reach the leader's trajectory. If no such trajectory can be found, it returns null. In practice, we found that selecting multiple points along the leader's trajectory as potential targets for the join maneuver, and attempting to plan a trajectory to each point in sequence until one is found, works well.

We will now provide a formal proof of safety for the multi-agent formation control Problem 1.

Theorem 1. *Consider a set of N agents satisfying Assumption 1 and Assumption 2. If*

1) The leader trajectory $([t_0, t_f], p_L, u_L)$ satisfies Def. 14, with known $\epsilon \in \mathbb{R}_{\geq 0}$ satisfying lemma 1,

2) All agents possess valid committed trajectories at time t_0, and update their committed trajectories according to multi-agent `gatekeeper`*,*

Then, $\forall t \geq t_0$, all agents i remain in $\mathcal{S}$, and maintain inter-agent safety $\left\|r_i(t) - r_j(t)\right\| \geq \delta, \forall i \neq j$.

Proof. The proof is by induction.

Base Case: At time t_0, by assumption, each agent i has a valid committed trajectory $([t_0, \infty), p^{\text{com}}_{0,i}, u^{\text{com}}_{0,i})$, and safety holds by hypothesis.

Inductive Step: Assume agent i is safe at iteration k_i. At iteration $k_i + 1$, it generates a candidate.

1) *Case A:* The candidate is valid by Def. 11. By Def. 11, the candidate is safe with respect to static obstacles, reaches the backup set, and is collision free with respect to all other agents' committed trajectories.
2) *Case B:* The agent finds no valid candidate. It continues executing its previous committed trajectory, valid by the inductive hypothesis, and thus remains safe.

Since the algorithm guarantees agents never switch to an invalid candidate, and the initial committed trajectory is valid, safety is preserved at each iteration. □

Remark 3. *The safety guarantees in theorem 1 rely on assumptions which require timely, and asynchronous sharing of committed trajectories among agents, and the existence of initial valid committed trajectories constructed sequentially at t_0. In practice, communications are subject to delays and packet drops, and initial construction may fail for large numbers of agents. We therefore treat the present results as a proof of concept that the trajectory backup set idea yields strong safety guarantees under these conditions, and addressing delayed/partial communications and decentralized initial allocation is important future work section VII.*

V. Simulation Results

We simulate a team of agents modeled as 3D Dubins vehicles executing Problem 1 in a dense urban-like 3D environment. We demonstrate the efficacy of our approach through a series of randomized trials, comparing multi-agent `gatekeeper` to baseline methods. We consider 3D curvature- and pitch-angle constrained vehicles, which are useful models for fixed-wing aircraft, and reflect the dynamics of Dubins Airplanes [20, 21].

A. Setup

Consider a team of N_A agents, modeled as a 3D variant of the common unicycle model, with pitch, curvature and velocity constraints. This model is useful due to its compatibility with 3D Dubins paths.

$$s_i = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix}, \quad \dot{s}_i = \begin{bmatrix} v_{\max} \cos\psi \cos\gamma \\ v_{\max} \sin\psi \cos\gamma \\ v_{\max} \sin\gamma \\ \omega \end{bmatrix}, \quad u = \begin{bmatrix} \omega \\ \gamma \end{bmatrix}, \quad \begin{matrix} |\omega| \leq \omega_{\max} \\ \gamma_{\min} \leq \gamma \leq \gamma_{\max} \end{matrix} \tag{21}$$

Using this model, we can compute the leader's path using a Dubins-based *RRT** method [22], specifying the turn radius for the Dubins primitives accordingly. We used [23] to generate Dubins curves in 3D which respect these pitch and curvature constraints.

Nominal follower paths are offset by a fixed displacement vector d_i^*, transformed to the body frame of the leader aircraft at each point along the path. The resulting paths may not be dynamically feasible nor are they guaranteed to be

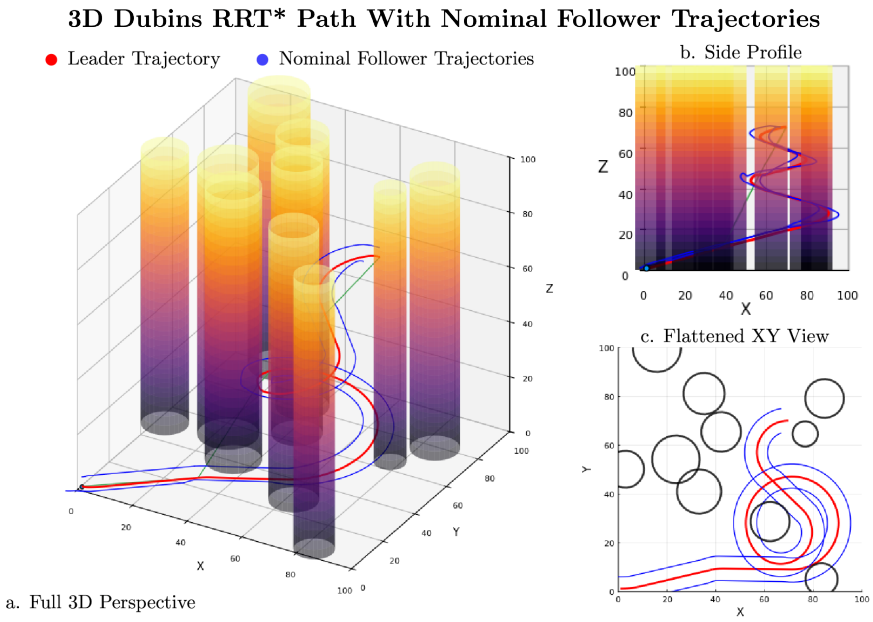

Fig. 1 A leader trajectory (red) generated using Dubins RRT* in a 3D environment with cylindrical obstacles. (a) Full 3D perspective. (b) Side profile, demonstrating the path's adherence to pitch constraints. (c) Top-down XY projection. While the leader's path is safe and dynamically feasible, the nominal follower trajectories (blue) , created from a fixed offset , are shown intersecting with obstacles and violating dynamic constraints.

safe. Figure 1 shows a leader trajectory generated using this method, and the nominal follower trajectories generated by offsetting the leader trajectory. We see that while the leader trajectory avoids obstacles and respects dynamics constraints, nominal follower trajectories intersect with obstacles and violate minimum curvature constraints impose by the dynamics model. This method is scalable to many agents, as once the leader trajectory is generated offline, additional followers can be added online.

B. Simulation Results

We perform simulations of one leader agent and two follower agents operating in randomly-generated environments. All simulations were performed in `julia`, running on a 2022 MacBook Air (Apple M2, 16GB).

The simulation environment is $100 \times 100 \times 100$ meters and contains 25 randomly placed cylindrical obstacles with varying radii. The leader trajectory is generated from $(0, 0, 0)$ to $(100, 100, 70)$, with the start and end orientation fixed as level and towards positive x. Two follower agents are offset by $(-3.0, 5.0, 0.0)$ and $(-3.0, -5.0, 0.0)$ relative to the body frame of the leader. A minimum turning radius of 10 and pitch-limits of $(-15°, 20°)$ are used. An inter-agent collision distance $\delta = 1$ is used. Each environment is generated by placing each cylinder at a random location, and sampling the radius uniformly between 2 and 5 units. The leader trajectory is then generated using the Dubins *RRT** method described above.

Figure 2 shows the evolution of a sample simulation over time. We see that at various points in time, agents rejoin the leader's path to avoid collision with obstacles, while still maintaining a safe distance. Figure 3 provides a more detailed view of the same scenario, showing the full 3D trajectory, a 2D projection of the scenario, as well as the minimum interagent distance plotted over the course of the trial. We see that agents deviate as necessary to avoid obstacles while remaining safe with respect to other agents. Agents must also deviate to make tight turns that are dynamically infeasible, but overall remain close to the nominal trajectory

C. Baseline Comparison

To validate our approach, we compare multi-agent `gatekeeper` with baseline methods (CBF-QP and NMPC) by simulating each algorithm in 100 randomly-generated scenarios. We allow the baseline methods to replan at a frequency of 60 Hz, as well as vary velocity between $[0.8, 1.0]$ to improve their ability to avoid collisions. The other simulation

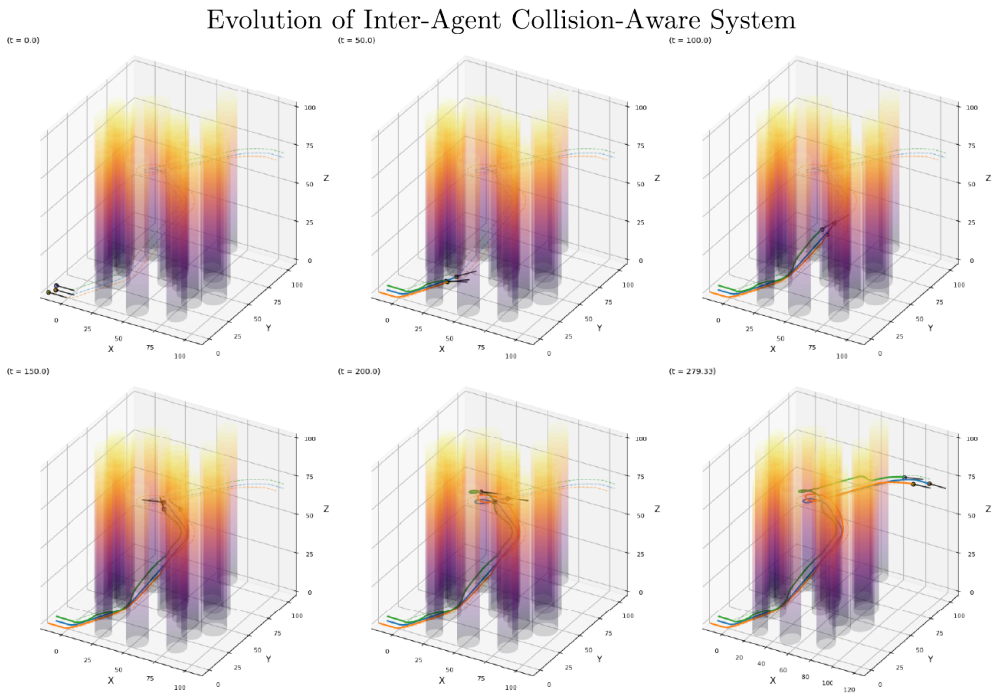

Fig. 2 Evolution of a sample simulation over time, shown at six distinct time steps. The agents navigate a dense 3D environment with cylindrical obstacles. Agents can be seen deviating from their nominal formation and rejoining the leader's path to safely avoid collisions with obstacles.

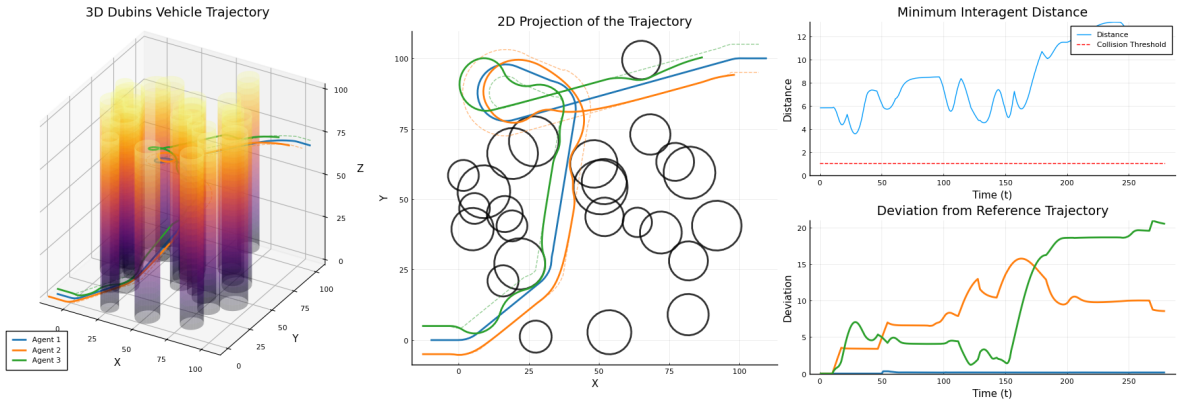

Fig. 3 Trajectory and performance metrics for the simulation scenario shown in fig. 2. (Left) Full 3D agent paths; (Center) 2D projection with static obstacles; (Top-Right) minimum inter-agent distance over time, which remains above the collision threshold; and (Bottom-Right) agent deviation from their nominal reference trajectories.

parameters remain unchanged. To ensure feasibility of the optimization-based solvers, in the event that the optimization problem becomes infeasible, we relax the inter-agent collision constraints by moving them to the cost function with a high penalty weight. We use a formation tracking error metric defined as the average Euclidean distance between each follower and its nominal position relative to the leader, averaged over the duration of the trial. Success is defined as no agents incurring a collision over the duration of the trial. The results of the trials are given in table 2. We see that multi-agent `gatekeeper` is able to successfully complete all 100 trials without collision, while maintaining low formation tracking error. The NMPC baseline achieves a moderate success rate, and with less formation error than multi-agent `gatekeeper`. This highlights the tradeoff between safety and performance, as our approach prioritizes safety over formation tracking performance, and thus must deviate more frequently from the nominal trajectory. The

CBF-QP method struggled significantly with the highly nonlinear and dynamic collision avoidance constraints, resulting in very poor success rates, even with a high replanning frequency and planning horizon, as well as velocity modulation, as shown in table 2. Our method was the **only** method to achieve a 100% success rate, demonstrating our superior safety guarantees.

Algorithm	Success Rate	Mean Formation Error
multi-agent `gatekeeper`	**100%**	6.32
CBF-QP	22%	7.48
NMPC	68%	**4.32**

Table 2 Restuls of the baseline comparison over 100 randomized trials. Each scenario was created by randomly generating a 3D environement with obstacles, and generating a leader trajectory using Dubins RRT*. The same leader trajectory was used across all algorithms for each scenario. Success Rate is the percentage of trials completed with no collisions. Mean Formation Error is the average deviation from the nominal reference trajectory.

VI. Hardware Demonstration

To validate the physical feasibility of the complex, 3D trajectories generated by the multi-agent `gatekeeper` algorithm, we executed a representative scenario on a team of Crazyflie 2.0 quadcopters.

A. Experiment Design

The experiment's goal was to confirm that the safe trajectories, which require agents to dynamically break and reform their formation, satisfies safety constraints. Due to the challenges of onboard, high-frequency replanning and information sharing onboard the quadcopters, the trajectories were pre-computed offline using the full multi-agent `gatekeeper` algorithm in a simulated environment. These trajectories were then used as a reference for the onboard low-level controller, which used state feedback from a Vicon motion capture system to track the path.

The quadcopters were controlled to emulate the 3D Dubins vehicle model used in simulation. This was achieved by commanding the low-level controller to track the pre-computed path's position, velocity, and yaw, thereby respecting the pitch and curvature constraints of the simulated Dubins airplanes.

B. Results

We ran a scenario where three agents, starting in a triangular formation, were required to navigate through a narrow gate. As shown in fig. 4, the only safe solution requires the follower agents to identify the future collision risk, break their formation and follow the leader's path (their 'trajectory backup set') in a single-file line, and then re-establish the formation after clearing the obstacle.

This experiment was performed five times. In all trials, the quadcopters successfully executed the maneuver, passed through the gate without collision, and maintained the required safety separation. This result validates that the trajectories produced by our algorithm are not just theoretically sound but are also dynamically feasible and trackable by physical robotic systems in a 3D environment.

VII. Conclusion

This paper introduced multi-agent `gatekeeper`, a novel, distributed safety framework for leader-follower formation control in complex 3D environments. Our approach extends the single-agent `gatekeeper` safety verification algorithm to a multi-agent system, and marks its first application in a 3D environment.

The core of our method is a hybrid online-offline approach, where a leader follows a pre-computed safe path, and follower agents compute trajectories online. We introduced the "trajectory backup set," which leverages the leader's safe path as a guaranteed backup maneuver for all followers, allowing followers to flexibly maintain formation, but maintain a safe fallback option. We then provided a rigorous proof that this approach ensures forward invariant safety under the assumption of ideal, delay-free communication and sequential decision making.

Empirically, we demonstrated the superior reliability of multi-agent `gatekeeper` compared to baseline methods, maintaining 100% safety complex 3D environments where CBF and NMPC methods frequently failed. Furthermore,

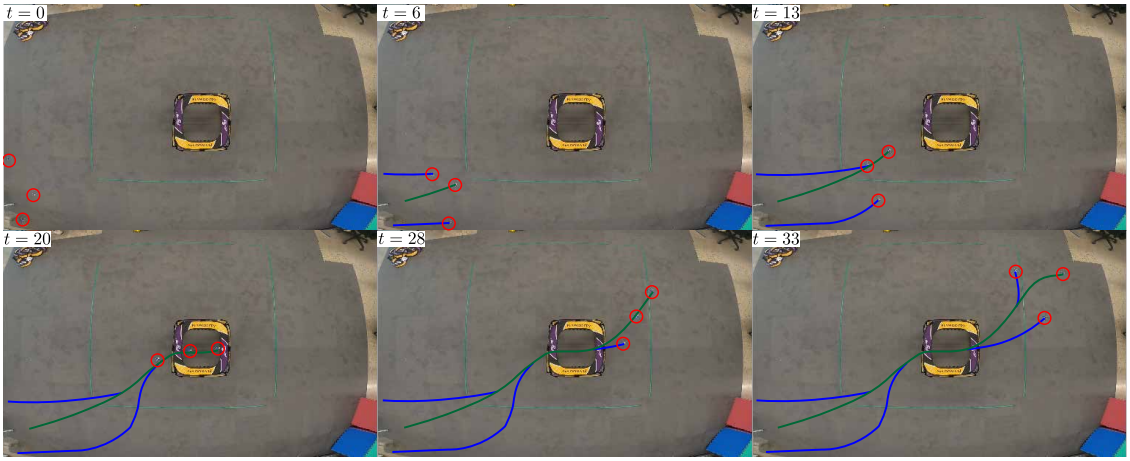

Fig. 4 Hardware demonstration of multi-agent `gatekeeper` with three Crazyflie 2.0 quadcopters navigating through a narrow gate.

we validated the approach on hardware with a team of quadcopters, successfully demonstrating the ability to break formation to pass through a narrow obstacle and reform afterward while still ensuring safety.

Limitations of the current work include the offline computation of trajectories for the hardware demonstration, the assumption of ideal, delay-free communications, and the centralized nature of the information sharing. Future work will investigate scalability and decentralization of this approach, and its robustness to communication delays and bandwidth constraints.

References

[1] Xia, Z., Du, J., Wang, J., Jiang, C., Ren, Y., Li, G., and Han, Z., "Multi-Agent Reinforcement Learning Aided Intelligent UAV Swarm for Target Tracking," *IEEE Transactions on Vehicular Technology*, Vol. 71, No. 1, 2022, pp. 931–945. https://doi.org/10.1109/TVT.2021.3129504.

[2] Jo, Y., Lee, S., Yeom, J., and Han, S., "FoX: Formation-Aware Exploration in Multi-Agent Reinforcement Learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, No. 12, 2024, pp. 12985–12994. https://doi.org/10.1609/aaai.v38i12.29196.

[3] Afrazi, M., Seo, S., and Lee, K., "Density-Driven Formation Control of a Multi-Agent System with an Application to Search-and-Rescue Missions," *2025 American Control Conference (ACC)*, IEEE, Denver, CO, USA, 2025, pp. 3622–3627. https://doi.org/10.23919/ACC63710.2025.11107863.

[4] Lee, G., and Chwa, D., "Decentralized Behavior-Based Formation Control of Multiple Robots Considering Obstacle Avoidance," *Intelligent Service Robotics*, Vol. 11, No. 1, 2018, pp. 127–138. https://doi.org/10.1007/s11370-017-0240-y.

[5] Barfoot, T., and Clark, C., "Motion Planning for Formations of Mobile Robots," *Robotics and Autonomous Systems*, Vol. 46, No. 2, 2004, pp. 65–78. https://doi.org/10.1016/j.robot.2003.11.004.

[6] Roldão, V., Cunha, R., Cabecinhas, D., Silvestre, C., and Oliveira, P., "A Leader-Following Trajectory Generator with Application to Quadrotor Formation Flight," *Robotics and Autonomous Systems*, Vol. 62, No. 10, 2014, pp. 1597–1609. https://doi.org/10.1016/j.robot.2014.05.002.

[7] Xiao, H., and Chen, C. L. P., "Leader-Follower Consensus Multi-Robot Formation Control Using Neurodynamic-Optimization-Based Nonlinear Model Predictive Control," *IEEE Access*, Vol. 7, 2019, pp. 43581–43590. https://doi.org/10.1109/ACCESS.2019.2907960.

[8] Liu, Y., and Bucknall, R., "A Survey of Formation Control and Motion Planning of Multiple Unmanned Vehicles," *Robotica*, Vol. 36, No. 7, 2018, pp. 1019–1047. https://doi.org/10.1017/S0263574718000218.

[9] Wang, Y., Shan, M., Yue, Y., and Wang, D., "Vision-Based Flexible Leader–Follower Formation Tracking of Multiple Nonholonomic Mobile Robots in Unknown Obstacle Environments," *IEEE Transactions on Control Systems Technology*, Vol. 28, No. 3, 2020, pp. 1025–1033. https://doi.org/10.1109/TCST.2019.2892031.

[10] Kar-Han Tan, and Lewis, M., "Virtual Structures for High-Precision Cooperative Mobile Robotic Control," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS '96*, Vol. 1, IEEE, Osaka, Japan, 1996, pp. 132–139. https://doi.org/10.1109/IROS.1996.570643.

[11] Askari, A., Mortazavi, M., and Talebi, H. A., "UAV Formation Control via the Virtual Structure Approach," *Journal of Aerospace Engineering*, Vol. 28, No. 1, 2015, p. 04014047. https://doi.org/10.1061/(ASCE)AS.1943-5525.0000351.

[12] Zhou, D., Wang, Z., and Schwager, M., "Agile Coordination and Assistive Collision Avoidance for Quadrotor Swarms Using Virtual Structures," *IEEE Transactions on Robotics*, Vol. 34, No. 4, 2018, pp. 916–923. https://doi.org/10.1109/TRO.2018.2857477.

[13] Oh, K.-K., Park, M.-C., and Ahn, H.-S., "A Survey of Multi-Agent Formation Control," *Automatica*, Vol. 53, 2015, pp. 424–440. https://doi.org/10.1016/j.automatica.2014.10.022.

[14] Balch, T., and Arkin, R., "Behavior-Based Formation Control for Multirobot Teams," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 6, 1998, pp. 926–939. https://doi.org/10.1109/70.736776.

[15] Čáp, M., Novák, P., Vokřínek, J., and Pěchouček, M., "Multi-Agent RRT*: Sampling-based Cooperative Pathfinding (Extended Abstract)," , 2013. https://doi.org/10.48550/ARXIV.1302.2828.

[16] Zhou, M., Wang, Z., Wang, J., Dong, Z., and School of Electrical and Control Engineering, North China University of Technology No.5 Jinyuanzhuang Road, Shijingshan District, Beijing 100144, China, "A Hybrid Path Planning and Formation Control Strategy of Multi-Robots in a Dynamic Environment," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, Vol. 26, No. 3, 2022, pp. 342–354. https://doi.org/10.20965/jaciii.2022.p0342.

[17] Agrawal, D. R., Chen, R., and Panagou, D., "Gatekeeper: Online Safety Verification and Control for Nonlinear Systems in Dynamic Environments," *IEEE Transactions on Robotics*, Vol. 40, 2024, pp. 4358–4375. https://doi.org/10.1109/TRO.2024.3454415.

[18] Agrawal, D. R., and Panagou, D., "Online Safety Under Multiple Constraints and Input Bounds Using Gatekeeper: Theory and Applications," *IEEE Control Systems Letters*, Vol. 9, 2025, pp. 2309–2314. https://doi.org/10.1109/LCSYS.2025.3599718.

[19] Tordesillas, J., and How, J. P., "MADER: Trajectory Planner in Multiagent and Dynamic Environments," *IEEE Transactions on Robotics*, Vol. 38, No. 1, 2022, pp. 463–476. https://doi.org/10.1109/TRO.2021.3080235.

[20] Valavanis, K. P., and Vachtsevanos, G. J. (eds.), *Handbook of Unmanned Aerial Vehicles*, Springer Netherlands, Dordrecht, 2015. https://doi.org/10.1007/978-90-481-9707-1.

[21] Dubins, L. E., "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents," *American Journal of Mathematics*, Vol. 79, No. 3, 1957, p. 497. https://doi.org/10.2307/2372560.

[22] Lin, Y., and Saripalli, S., "Path Planning Using 3D Dubins Curve for Unmanned Aerial Vehicles," *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, Orlando, FL, USA, 2014, pp. 296–304. https://doi.org/10.1109/icuas.2014.6842268.

[23] Vana, P., Alves Neto, A., Faigl, J., and Macharet, D. G., "Minimal 3D Dubins Path with Bounded Curvature and Pitch Angle," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, Paris, France, 2020. https://doi.org/10.1109/icra40945.2020.9197084.