

# R3R: Decentralized Multi-Agent Collision Avoidance with Infinite-Horizon Safety

Thomas Marshall Vielmetti<sup>1</sup>, Devansh R. Agrawal<sup>2</sup>, and Dimitra Panagou<sup>3</sup>

**Abstract**—Existing decentralized methods for multi-agent motion planning lack formal, infinite-horizon safety guarantees, especially for communication-constrained systems. We present **R3R**, to our knowledge the first decentralized and asynchronous framework for multi-agent motion planning under distance-based communication constraints with infinite-horizon safety guarantees for systems of nonlinear agents. **R3R**’s novelty lies in combining our *gatekeeper* safety framework with a geometric constraint called *R-Boundedness*, which together establish a formal link between an agent’s communication radius and its ability to plan safely. We constrain trajectories to within a fixed planning radius that is a function of the agent’s communication radius, which enables trajectories to be shown provably safe for all time, using only local information. Our algorithm is fully asynchronous, and ensures the forward invariance of these guarantees even in time-varying networks where agents asynchronously join, leave, and replan. We validate our approach in simulations of up to 128 Dubins vehicles, demonstrating 100% safety in dense, obstacle rich scenarios. Our results demonstrate that **R3R**’s performance scales with agent density rather than problem size, providing a practical solution for scalable and provably safe multi-agent systems<sup>1</sup>.

## I. INTRODUCTION

From last mile delivery [1] to warehouse robotics [2] to autonomous drone swarms [3], deploying multi-agent systems safely in the real world is bottlenecked by a lack of formal safety guarantees. This challenge, known as multi-agent motion planning (MAMP), forces a trade-off between guarantees, coordination, and scalability. Centralized planners can provide strong safety guarantees but fail to scale and are prone to single-node failure. Conversely, decentralized methods scale well to large teams but have thus far been unable to provide formal, infinite-horizon safety guarantees under distance-based communication constraints for non-trivial (i.e. beyond single/double integrator) systems.

Multi-agent path finding (MAPF), a discrete state precursor to MAMP, is typically formulated as a graph search problem. Classical MAPF methods, such as Conflict-Based Search and its variants [4], [5], [6], [7], provide strong optimality or bounded suboptimality guarantees, but these

approaches are centralized and limited to discrete domains, thus may not produce dynamically feasible solutions. Some MAMP methods have extended these concepts to continuous space, and trajectory optimization and sampling-based planners have been proposed, but these centralized approaches struggle with scalability and real-time performance [8].

Planning-based methods comprise the majority of the recent state-of-the-art. These methods guarantee safety by generating safe trajectories over some finite horizon. This also allows for agents to take preemptive actions, reducing deadlocks and allowing for more direct paths compared to reactive methods. Despite being distributed in computation, these methods are often centralized in information, or use synchronization to enable coordination. In MADER [9], [10], agents construct trajectories individually, but must share and receive full trajectories with all other agents in the network. Other methods require synchronous replanning/state updates [11], [12], [13], [14] and full state sharing among all agents in the network [15], [16], [17]. While these methods have been shown to outperform reactionary methods, they still provide only finite-horizon guarantees.

Critically, no existing decentralized, asynchronous approach guarantees infinite-horizon safety under distance-based communication constraints. This gap motivates our proposed framework, **R3R**. To the best of our knowledge, **R3R** represents the first decentralized and asynchronous framework to provide infinite-horizon safety guarantees for multi-agent motion planning under distance-based communication constraints, and generalize to broad classes of systems under certain assumptions. This is achieved by integrating a novel geometric constraint, which we term *R-Boundedness* with *gatekeeper* [18], our safety framework developed for single agents, to provide recursive feasibility guarantees in multi-agent systems. Our contributions are as follows:

- 1) **R3R**, a decentralized and asynchronous framework for multi-agent motion planning under communication constraints that provides infinite-horizon safety guarantees.
- 2) The key enabler of **R3R** is *R-Boundedness*, a novel geometric constraint on trajectory generation, which we show links an agent’s ability to communicate with its ability to plan safely.
- 3) We demonstrate the efficacy of **R3R** in simulations of up to 128 agents, showcasing safe, and scalable multi-agent coordination.

The paper is organized as follows: Section II introduces preliminaries and the problem statement, and section III

The authors would like to acknowledge the support of the National Science Foundation (NSF) under grant no. 2137195.

<sup>1</sup>Thomas Marshall Vielmetti is with the Department of Electrical and Computer Engineering, University of Michigan, Ann Arbor, MI 48109, USA [mvielmet@umich.edu](mailto:mvielmet@umich.edu)

<sup>2</sup>Devansh R. Agrawal is the Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA [devansh@umich.edu](mailto:devansh@umich.edu)

<sup>3</sup>Dimitra Panagou is with the Department of Robotics and Department of Aerospace Engineering, University of Michigan, Ann Arbor, MI 48109, USA [dpanagou@umich.edu](mailto:dpanagou@umich.edu)

<sup>1</sup>Code available at [github.com/MarshallVielmetti/r3r\\_decentralized\\_gatekeeper](https://github.com/MarshallVielmetti/r3r_decentralized_gatekeeper)

presents the proposed approach. The methodology is evaluated in simulation in section IV, while conclusions and thoughts for future work are given in section V.

## II. PRELIMINARIES AND PROBLEM STATEMENT

*Notation:*  $\mathbb{R}, \mathbb{R}_{\geq 0}, \mathbb{R}_{> 0}$  are the sets of reals, non-negative reals, and positive reals. Let  $\mathcal{B}(x, R) = \{y : \|x - y\| \leq R\}$  be the closed ball of radius  $R \geq 0$  centered at  $x \in \mathbb{R}^n$ .  $\mathcal{A}(t) = \{1 \dots N\}$  denotes the set of all agents at time  $t$ , where  $N$  is the total number of agents in the network at that time. Let  $\mathcal{A}_{-i}(t) = \mathcal{A}(t) \setminus \{i\}$  for  $i \in \mathcal{A}(t)$ . Let  $\delta \in \mathbb{R}_{> 0}$  denote the inter-agent collision radius.

### A. Preliminaries

Consider a dynamical system

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

where  $x \in \mathcal{X} \subseteq \mathbb{R}^n$  is the state,  $u \in \mathcal{U} \subseteq \mathbb{R}^m$  is the control input. The function  $f : \mathbb{R}_{\geq 0} \times \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}^n$  is piecewise continuous and locally Lipschitz in  $x$  and  $u$ .

**Definition 1** (Agent). *An agent represents a robot or other actor that interacts with the system of other agents. An agent  $i \in \mathcal{A}$  has the following properties:*

- 1) The agent's state  $x_i$  evolves through a system satisfying (1), and includes a position component  $p \in \mathbb{R}^d$ .
- 2)  $\delta \in \mathbb{R}_{> 0}$  Denotes the inter-agent avoidance distance. No other agent may come within a distance  $< \delta$  of agent  $i$ .
- 3)  $R^{comm} \in \mathbb{R}_{> 0}$  represents the communication radius.
- 4) A planning radius  $R^{plan}$  which constrains an agent's future trajectory.

This relationship between  $R^{comm}$  and  $R^{plan}$ , which we show must be  $R^{comm} = 3R^{plan} + \delta$ , (R3R) is the key insight of this work, and will be explored in detail.

**Assumption 1.** *All agents have a common  $\delta$  and  $R^{comm}$ , and thus a common  $R^{plan}$ .*

**Definition 2** (Trajectory). *A **trajectory**<sup>2</sup> is the tuple  $\mathcal{T} = (T, x, u)$ , where  $T \subset \mathbb{R}_{\geq 0}$  is a time-interval,  $x : T \rightarrow \mathcal{X}$  is the state-trajectory, and  $u : T \rightarrow \mathcal{U}$  is a control-trajectory, satisfying*

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

**Definition 3.** *Let  $\mathcal{G}(t) = (\mathcal{A}(t), E(t))$  be an undirected, time-varying graph, with nodes representing agents, and edges representing communication links between them. The edge set  $E(t)$ , defined for some communication radius  $R^{comm} > 0$ , is given by:*

$$E(t) := \{(i, j) \in \mathcal{A}(t) : \|p_i(t) - p_j(t)\| \leq R^{comm}\} \quad (3)$$

where  $p_i(t) \in \mathbb{R}^d$  is the position of agent  $i \in \mathcal{A}(t)$  at time  $t$ .

**Assumption 2.** *We assume the communication is instantaneous and range-limited, but not bandwidth limited. That is,*

<sup>2</sup>The time-interval  $T$  can be a finite horizon, e.g.,  $T = [t_0, t_1]$  or an infinite horizon, e.g.,  $T = [t_0, \infty)$ .

### Single Agent Constructs Committed Trajectory

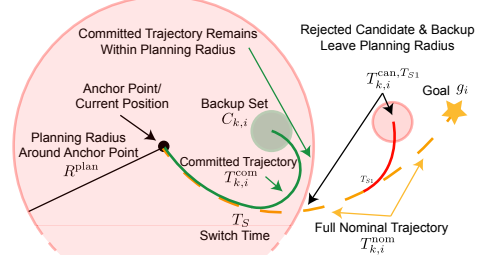


Fig. 1. **Constructing a Safe, Committed Trajectory.** An agent at the anchor point plans a nominal trajectory (orange) toward its goal. To ensure safety, it must find a candidate trajectory that remains entirely within its planning radius  $R^{plan}$  (red circle). An unsafe candidate that exits this radius (red) is rejected. A safe candidate that stays within the radius (green) is verified and becomes the committed trajectory for the agent to follow.

if two agents  $i, j$  are neighbors at some time  $t$ , they can communicate an arbitrary amount of information instantaneously.

We define the set of neighbors of agent  $i$  at time  $t \in \mathbb{R}_{\geq 0}$  as  $\mathcal{N}_i(t)$ , where

$$\mathcal{N}_i(t) = \{j \in \mathcal{A}(t) : (i, j) \in E(t)\}. \quad (4)$$

**Definition 4** (dist). *Let  $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^d$ ,  $x \in \mathbb{R}^d$ , define*

$$\text{dist}(\mathcal{S}_1, x) = \inf_{y \in \mathcal{S}_1} \|x - y\|_2, \quad (5)$$

$$\text{dist}(\mathcal{S}_1, \mathcal{S}_2) = \inf_{x \in \mathcal{S}_1, y \in \mathcal{S}_2} \|x - y\|_2. \quad (6)$$

### B. gatekeeper Preliminaries

R3R builds upon our prior work for single-agent safety gatekeeper [18], which ensures that agents always track a *committed trajectory* that is known to be safe for all future time. This is done by properly appending the goal-oriented *nominal trajectory* with a safe infinite-horizon *backup trajectory*. This becomes a *candidate trajectory*, which must be validated as safe prior to being committed. The steps are discussed later, and the principle is shown in fig. 1. Our key extension is to impose an additional geometric constraint on candidate trajectories, as illustrated in fig. 1, which enables decentralized verification.

Let  $\mathcal{S} \subset \mathcal{X}$  be the safe set, to be defined in Problem 1.

**Definition 5** (Backup Set). *A set  $\mathcal{C} \subset \mathcal{X}$  is a **backup set** if  $\mathcal{C} \subset \mathcal{S}$ , and there exists a controller  $\pi^B : \mathbb{R} \times \mathcal{X} \rightarrow \mathcal{U}$  such that  $\forall t_i \in \mathbb{R}_{\geq 0}$ , the closed loop system  $\dot{x} = f(t, x, \pi^B(t, x))$  satisfies*

$$x(t_i) \in \mathcal{C} \implies x(t) \in \mathcal{C}, \forall t \geq t_i. \quad (7)$$

**Definition 6** (Nominal Trajectory). *Given state  $x(t_k) \in \mathcal{X}$  at some time  $t_k \in \mathbb{R}_{\geq 0}$ , a **nominal trajectory**, typically generated by a planner, is denoted*

$$(T_k^{\text{nom}}, x_k^{\text{nom}}, u_k^{\text{nom}}) \quad (8)$$

and defined over the horizon  $T_k^{\text{nom}} = [t_k, t_k + T_H]$ , where  $T_H \in \mathbb{R}_{> 0}$  is the planning horizon.

A nominal trajectory is not guaranteed to satisfy constraints, nor guarantee safety beyond  $T_H$ . In the context of com-

munication constrained planning, this is because nominal trajectories are constructed without full knowledge of the system state, and thus may collide with agents outside the planning agents communication range.

**Definition 7** (Backup Trajectory). *Let  $T_B \in \mathbb{R}_{>0}$ . For any  $t_s \in \mathbb{R}_{>0}$ , and  $x_s \in \mathcal{X}$ , a trajectory  $(T^{\text{bak}}, x^{\text{bak}}, u^{\text{bak}})$ , defined on  $T^{\text{bak}} = [t_s, \infty)$  and terminating in a backup set  $\mathcal{C}$  is a **backup trajectory** if*

$$x^{\text{bak}}(t_s + T_B) \in \mathcal{C}, \quad (9)$$

and for all  $t \geq t_s + T_B$ ,

$$u^{\text{bak}}(t) = \pi^B(t, x^{\text{bak}}(t)). \quad (10)$$

A backup trajectory takes a system from a state  $x \in \mathcal{X}$  at time  $t_s$  such that at time  $t_s + T_B$ , the system lies in a backup set. The system then executes the backup controller for all future time, thus remaining in the backup set. If the finite horizon trajectory defined over  $[t_s, t_s + T_B]$  safely takes the agent into the backup set, then by the properties of the backup set we guarantee the agent will be safe over  $[t_s, \infty)$ , transforming a finite horizon guarantee into an infinite horizon one.

**Definition 8** (Candidate Trajectory). *At time  $t_k \in \mathbb{R}_{\geq 0}$ , let agent  $i \in \mathcal{A}(t)$  be at state  $x_i(t_k) \in \mathcal{X}$ . Let the agent's nominal trajectory be  $\mathcal{T}_{k,i}^{\text{nom}} = ([t_k, t_k + T_H], x_{k,i}^{\text{nom}}, u_{k,i}^{\text{nom}})$ . A **candidate trajectory** with switch time  $t_s \in [t_k, t_k + T_H]$ , denoted  $\mathcal{T}_{k,i}^{\text{can}, t_s} = ([t_k, \infty), x_{k,i}^{\text{can}}, u_{k,i}^{\text{can}})$  is defined by:*

$$\mathcal{T}_{k,i}^{\text{can}} = \begin{cases} (x_{k,i}^{\text{nom}}(\tau), u_{k,i}^{\text{nom}}(\tau)) & \text{if } \tau \in [t_k, t_s] \\ (x_{k,i}^{\text{bak}}(\tau), u_{k,i}^{\text{bak}}(\tau)) & \text{if } \tau \geq t_s \end{cases} \quad (11)$$

where  $(x_{k,i}^{\text{bak}}, u_{k,i}^{\text{bak}})$  is a backup trajectory from  $t_s, x_k^{\text{nom}}(t_s)$ , as in definition 7.

### C. Problem Statement

**Problem 1.** *Subject to assumptions 1 and 2, let  $\mathcal{A}(t) = \{1, 2, \dots, N\}$  denote a time-varying set of agents satisfying definition 1. These agents communicate over a connectivity graph  $\mathcal{G}(t)$  defined in definition 3. Let  $\mathcal{X}_{\text{obs}} \subset \mathcal{X}$  be the subset of the domain occupied by static obstacles, and  $\mathcal{S} \subset \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$  be the safe set. Assume this set is known to all agents. Let the goal state of each agent be denoted as  $g_i \in \mathcal{S}$ . Agents attempt to navigate to their goal state, such that*

$$\lim_{t \rightarrow \infty} x_i(t) = g_i, \quad \forall i \in \mathcal{A}, \quad (12)$$

while satisfying the static and inter-agent collision avoidance constraints

$$x_i(t) \in \mathcal{S}, \quad \forall t \in \mathbb{R}_{\geq 0}, \quad (13)$$

$$\|p_i(t) - p_j(t)\| \geq \delta, \quad \forall j \in \mathcal{A}_{-i}(t), \forall t \in \mathbb{R}_{\geq 0}. \quad (14)$$

To solve Problem 1, we propose R3R, a decentralized framework for multi-agent motion planning under communication constraints. R3R combines the gatekeeper framework with the concept of  $R$ -bounded trajectories, which we introduce in the next section. We then present a geometric

argument that allows an agent to certify its trajectory as safe for all future time using only local information, and leverage gatekeeper to ensure forward invariance.

### III. PROPOSED APPROACH AND SOLUTION

We first present  $R$ -Boundedness, the theoretical component which allows us to ensure inter-agent collision avoidance, then propose a decentralized protocol based on this constraint that leverages gatekeeper to ensure infinite-horizon safety.

#### A. $R$ -Boundedness

Our core challenge is to prove network-wide safety using only local information. To solve this, we must first make an agent's future motion predictable and spatially contained. We achieve this by introducing a geometric constraint called  $R$ -Boundedness, which ensures an agent's entire future trajectory remains within a known, finite volume. This is the critical property that allows agents to formally reason about which other agents they can safely ignore.

**Definition 9** ( $R$ -Bounded). *Some trajectory  $\mathcal{T} = ([t_0, \infty), x(t), u(t))$  is  **$R$ -Bounded** if*

$$\|p(t) - p(t_0)\|_2 \leq R, \quad \forall t \geq t_0. \quad (15)$$

That is, a trajectory is  $R$ -bounded if it remains within a ball of radius  $R$  around its starting position, its *anchor point*, for all future time.

**Lemma 1** (Collision of Bounded Trajectories). *Consider two  $R$ -Bounded trajectories  $([t_1, \infty), x_1, u_1)$ ,  $([t_2, \infty), x_2, u_2)$ , and an inter-agent collision radius of  $\delta < R$ . Without loss of generality, let  $t_1 \leq t_2$ . If:*

$$\|p_1(t_1) - p_2(t_2)\| \geq 2R + \delta \quad (16)$$

Then:

$$\|p_1(t) - p_2(t)\| \geq \delta, \quad \forall t \geq t_2. \quad (17)$$

*Proof.* Since trajectories are  $R$ -bounded,  $p_1(t) \in B_1 = \mathcal{B}(p_1(t_1), R)$ ,  $p_2(t) \in B_2 = \mathcal{B}(p_2(t_2), R)$ ,  $\forall t \geq t_2$ . Then,

$$\begin{aligned} \min_{t \geq t_2} \|p_1(t) - p_2(t)\| &\geq \text{dist}(B_1, B_2) \\ &= \|p_1(t_1) - p_2(t_2)\| - 2R \geq \delta. \end{aligned}$$

Hence, the trajectories are collision-free.  $\square$

Lemma 1 provides a powerful safety condition based on the trajectories' anchor points. However, for an agent replanning at time  $t_2$ , the anchor point of another agent's trajectory (created at an earlier time  $t_1$ ) is not immediately relevant. To be useful, we must re-state this condition in terms of the agents' current positions at the moment of planning.

**Lemma 2** (R3R). *Consider two  $R$ -Bounded trajectories  $([t_1, \infty), x_1, u_1)$ ,  $([t_2, \infty), x_2, u_2)$ , with starting times  $t_1 < t_2$  for some radius  $R$ . If at some  $t \geq t_2$  the agents collide*

$$\|p_2(t) - p_1(t)\| < \delta \quad (18)$$

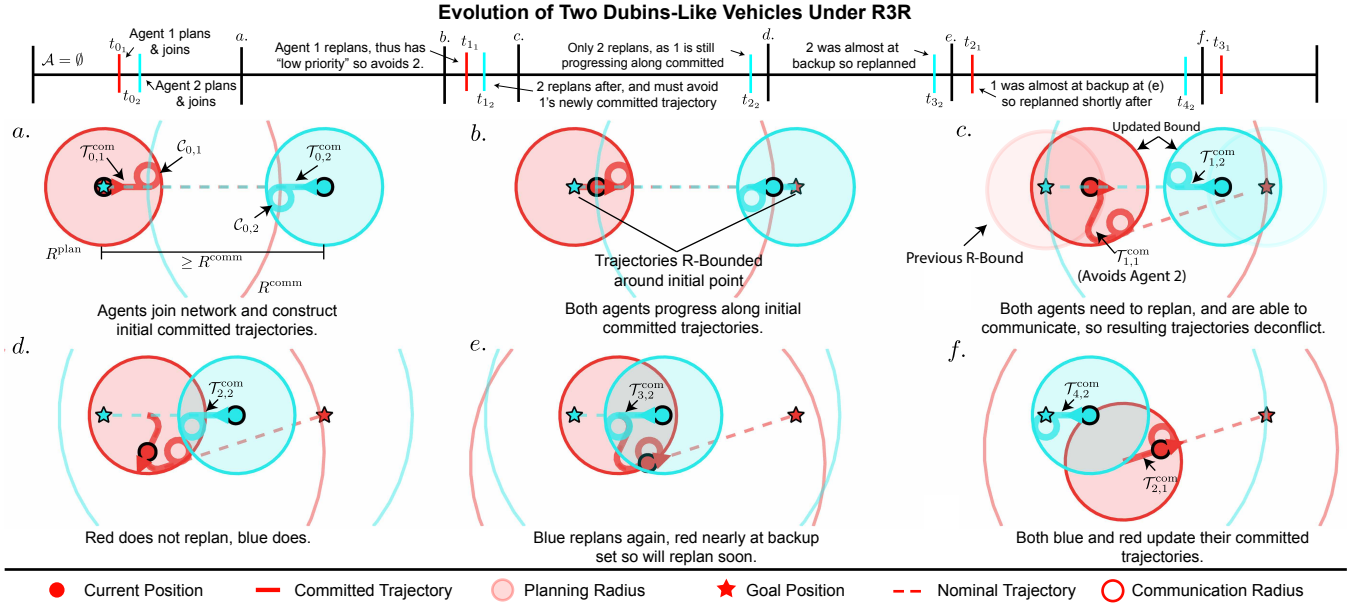


Fig. 2. **Demonstration of R3R over time for two Dubins agents.** The timeline indicates when each agent (Red: Agent 1, Blue: Agent 2) plans or replans. *Committed trajectories* (solid lines) are certified to be safe for all time, and remain within a planning radius  $R^{\text{plan}}$  (filled circles), around where they are constructed. Agents first plan goal-oriented nominal trajectories (dashed lines). A portion of this nominal is combined with a backup trajectory, to form a candidate. If this candidate is deemed safe, it becomes a committed trajectory. At times  $t_{0,1} \neq t_{0,2}$  agents 1 and 2 wish to join the network. Each agent independently constructs a valid committed trajectory, which is then simultaneously committed and broadcast as the agent enters the network. (a.) The initial committed trajectories of both agents remain within the planning radius. The agents could not communicate at the time of planning, so we see their nominal trajectories, which may be unsafe, intersect. (b.) Both agents have continued to track their initial committed trajectories, and are now within communication range. Both agents are near their backup sets, and thus should soon replan to continue making progress. (c.) Agent 1's logic dictated performing a replan first, so at time  $t_{1,1}$ , it constructs a new nominal trajectory to its goal, which attempts to avoid agent 2. It then constructs a new committed trajectory, which it certifies as safe with respect to the committed trajectory of agent 2. At some time  $t_{1,2} > t_{1,1}$ , agent 2 constructs a new committed trajectory which must avoid the recently committed trajectory of agent 1. This reflects the asynchronous nature of the algorithm, where agents replan independently and assume a low priority with respect to all neighbors. (d-f.) The process continues throughout the encounter. We see that agent 2 replans again at time  $t_{2,2}$ , but as agent 1 has not yet reached its backup set, its internal logic does not necessitate a replan.

then they must have been within  $3R + \delta$  at time  $t_2$ , i.e.,

$$\|p_2(t_2) - p_1(t_2)\| < 3R + \delta. \quad (19)$$

*Proof.* The contrapositive of lemma 1 gives

$$\|p_2(t) - p_1(t)\| < \delta \Rightarrow \|p_2(t_2) - p_1(t_1)\| < 2R + \delta. \quad (20)$$

By the definition of  $R$ -Boundedness, we have

$$\|p_1(t_2) - p_1(t_1)\| \leq R, \quad (21)$$

so by the triangle inequality,

$$\begin{aligned} \|p_2(t_2) - p_1(t_2)\| &\leq \|p_2(t_2) - p_1(t_1)\| + \|p_1(t_2) - p_1(t_1)\| \\ &< 2R + \delta + R = 3R + \delta \quad \square \end{aligned}$$

Lemma 2 provides a condition under which two  $R$ -Bounded trajectories can collide given the state of the system at the time agent 2 is constructing its committed trajectory. Letting agents communicate within a ball of  $3R + \delta$  allows us to rule out collisions with agents outside the replanning agent's communication range. We will thus require that the planning radius of all agents satisfy

$$R^{\text{comm}} = 3R^{\text{plan}} + \delta, \quad (22)$$

which we term the R3R constraint.

## B. Decentralized Safety Protocol

We now integrate the concept of  $R$ -Bounded trajectories into a decentralized safety protocol. We use `gatekeeper` as the building block for safety, and leverage  $R$ -Boundedness to enable local safety verification of candidate trajectories.

Each agent  $i$  executes `gatekeeper` independently, at discrete iterations indexed by  $k_i \in \mathbb{N}$ , which occur at times  $t_{k_i} \in \mathbb{R}_{\geq 0}$ , where  $t_{k_i} < t_{(k+1)_i}$ . Iteration  $0_i$  coincides with agent  $i$  entering  $\mathcal{A}$ . The agent is required to have a valid candidate at this time. At later iterations  $k_i > 0$ , agent  $i$  performs the following:

- 1) Receives the current committed trajectories  $\mathcal{T}_{k_j,j}^{\text{com}}$  of all its neighbors  $j \in \mathcal{N}_i(t_{k_i})$ .
- 2) Plans a nominal trajectory  $\mathcal{T}_{k_i,i}^{\text{nom}}$  to its goal, attempting to avoid the committed trajectories of all its neighbors.
- 3) Construct candidate trajectories as in algorithm 1 and defined in definition 8, until either one is found that satisfies the validity conditions of definition 10, or none is found.
- 4) As in algorithm 2, if a valid candidate is found, then it is committed, otherwise the previous committed trajectory is used.

This algorithm is fully decentralized, requiring only communication between neighboring agents, and asynchronous.

**Remark 1.** *Replanning iterations can be triggered by an*

---

**Algorithm 1: AttemptReplan**

---

**Input:**  $x_i, g_i, \mathcal{N}^{com}, \mathcal{S}$   
**Result:** *success* (bool),  $\mathcal{T}^{can}, T_S$   
 $\mathcal{T}^{nom} \leftarrow \text{PlanNominal}(x_i, g_i, \mathcal{N}^{com}, \mathcal{S})$   
**for**  $T_S \in [T_H, 0]$  **do**  
     $\mathcal{T}^{can}, T_S \leftarrow \text{ConstructCandidate}(\mathcal{T}^{nom}, T_S)$   
    **if**  $\text{IsValid}(\mathcal{T}^{can}, T_S, \mathcal{N}^{com}, \mathcal{S})$  **then**  
        **return** *True*,  $\mathcal{T}^{can}, T_S$   
**return** *False*,  $\emptyset$

---

---

**Algorithm 2: UpdateState**

---

// Trigger: Agent  $i$  wants to  
    join/replan.  
**success**,  $\mathcal{T}^{can} \leftarrow \text{AttemptReplan}()$   
**if** **success** **then**  
     $\mathcal{T}_{k_i+1,i}^{com} \leftarrow \mathcal{T}^{can}$    // Commit to the new  
    valid trajectory  
    **if** agent  $i \notin \mathcal{A}(t)$  **then**  
         $\mathcal{A}(t) \leftarrow \mathcal{A}(t) \cup \{i\}$    // Join the  
        network  
    **else if** agent  $i \in \mathcal{A}(t)$  **then**  
         $\mathcal{T}_{k_i+1,i}^{com} \leftarrow \mathcal{T}_{k_i,i}^{com}$    // Gatekeeper: Keep  
        the old trajectory

---

internal clock (i.e. every  $T_{replan}$  seconds), or by some other internal logic (i.e. when the agent is close to the switch time of its current committed trajectory). In general, the replanning times  $t_{k_i}$  are not synchronized between agents, and can be completely independent of each other, subject to Assumption 3, defined later. No assumptions need to be placed on the frequency of replanning to ensure safety.

The safety of this protocol hinges on the *IsValid* check within algorithm 1. Validity represents the conditions a candidate trajectory must satisfy in order to be considered safe. We formally define those criteria as follows.

**Definition 10** (Valid). A candidate trajectory  $\mathcal{T}_{k_i,i}^{can}, T_S$  as defined in definition 8 is **valid** if all the following hold:

1) The trajectory remains in the safe set during  $[t_{k_i}, t_{k_B}]$

$$x_{k_i,i}^{can}(t) \in \mathcal{S}, \forall t \in [t_{k_i}, t_{k_B}]. \quad (23)$$

2) The trajectory reaches the backup set  $\mathcal{C}_{k_i,i}$  at  $t_{k_i,B}$

$$x_{k_i,i}(t_{k_i,B}) \in \mathcal{C}_{k_i,i}, \quad (24)$$

where  $t_{k_i,B} = t_{k_i} + T_S + T_B$  is the time the trajectory enters the backup set.

3) The trajectory remains within the planning bound  $R^{plan}$  for all time:

$$\|p_{k_i,i}^{can}(t) - p_i(t_{k_i})\| \leq R^{plan}, \forall t \in [t_{k_i}, t_{k_i,B}], \quad (25)$$

$$\text{dist}(\mathcal{C}_{k_i,i}, p_i(t_{k_i})) \leq R^{plan}. \quad (26)$$

4) The candidate trajectory is collision free with respect to

the committed trajectories of all neighboring agents,

$$\|p_{k_i,i}^{can}(t) - p_{k_j,j}^{com}(t)\| \geq \delta, \forall t \in [t_{k_i}, \infty), \forall j \in \mathcal{N}_i(t_{k_i}) \quad (27)$$

We can now establish the central pillar of our safety argument: if every agent executes a trajectory that meets these local validity conditions, then the entire multi-agent system is provably safe for all future time. This is because the  $R$ -Boundedness condition geometrically precludes collisions with agents outside of the local communication radius.

**Theorem 1** (R3R - Safety of Valid Trajectories). *Consider the system described by Problem 1 at some arbitrary time  $t_k$ , with the set of active agents being  $\mathcal{A}(t_k)$ . Assume all agents satisfy the R3R condition (22). If at time  $t_k$ , every agent  $i \in \mathcal{A}(t_k)$  is executing a trajectory certified as valid per definition 10 at its respective time of creation  $t_{k_i} < t_k$ , then the system is guaranteed to be safe for all future time. That is,  $\forall t \geq t_k$ :*

$$\|p_i(t) - p_j(t)\| \geq \delta, \quad \forall i, j \in \mathcal{A}, i \neq j, \quad (28)$$

$$x_i(t) \in \mathcal{S}, \quad \forall i \in \mathcal{A}. \quad (29)$$

*Proof. [Proof Strategy]* The proof is conducted in two parts. First, we prove inter-agent collision avoidance by contradiction, showing that a collision would violate the trajectory's validity conditions at the time of creation. Second, we directly prove that agents remain in the safe set by construction.

*[Collision Avoidance]* Seeking contradiction, assume that at some  $t \geq t_k$ , distinct agents  $i, j \in \mathcal{A}$  collide, i.e.,  $\exists t \geq t_k$  such that  $\|p_i(t) - p_j(t)\| < \delta$ . Without loss of generality, let  $t_{k_i} < t_{k_j}$ .<sup>3</sup> Both trajectories are valid by definition 10, and thus  $R$ -Bounded by  $R^{plan}$ . Applying lemma 2, the condition  $\|p_i(t) - p_j(t)\| < \delta$  implies that at time  $t_{k_j}$ , the agents' positions must have satisfied  $\|p_i(t_{k_j}) - p_j(t_{k_j})\| < 3R^{plan} + \delta$ , which per the R3R condition (22) is precisely  $R^{comm}$ , thus  $i \in \mathcal{N}_j(t_{k_j})$ . This is a contradiction, since by definition 10, agent  $j$ 's trajectory  $p_j(t)$  being valid at  $t_{k_j}$  requires that it is collision free with respect to the trajectories of all agents in  $\mathcal{N}_j(t_{k_j})$ , and hence collision free with respect to the trajectory  $p_i(t)$  of agent  $i \in \mathcal{N}_j(t_{k_j})$ ,  $\forall t \geq t_{k_j}$ . This contradicts our assumption that agents  $i$  and  $j$  collide at some time  $t \geq t_{k_j}$ . Therefore our assumption of a collision must be false, i.e., the agents do not collide.

*[Safe Set]* We now show the agent remains in the safe set. By construction, a valid trajectory satisfies  $x_i(t) \in \mathcal{S}, \forall t \in [t_{k_i}, t_{k_i,B}]$ , and  $x_i(t) \in \mathcal{C}_i, \forall t \geq t_{k_i,B}$ , which by definition 5,  $\mathcal{C}_i \subset \mathcal{S}$  thus  $x_i(t) \in \mathcal{S}, \forall t \geq t_{k_i}$ .  $\square$

Theorem 1 provides a condition for infinite-horizon safety at a single instant, given agents all track valid trajectories. We must now show all agents in the network always track valid trajectories, thus showing the theorem is satisfied after every iteration of *gatekeeper*. This is the principle of forward invariance, which we achieve by formalizing how an agent "commits" to a trajectory. *gatekeeper*'s key function is to ensure that an agent only replaces its current safe plan with

<sup>3</sup>In the case of  $t_{k_i} = t_{k_j}$ , collision is precluded by lemma 1 and Assumption 3, to be defined.

a new, provably safe plan, or else continues with the old one. This is formalized with the notion of the **committed trajectory**, which is the trajectory tracked by the agent, and which is constructed to satisfy the validity conditions of definition 10 as follows.

**Definition 11** (Committed Trajectory). *At the  $k$ th iteration, let agent  $i \in \mathcal{A}(t_{k_i})$  be at state  $x_i(t_{k_i}) \in \mathcal{X}$ . Let  $\mathcal{T}_{k_i,i}^{\text{can}}$  be the candidate trajectory constructed at time  $t_{k_i}$  by agent  $i$ , as in definition 8. Define the set of valid switch times*

$$\mathcal{I}_{k_i} = \{T_S \in [0, T_H] : \mathcal{T}_{k_i,i}^{\text{can}} \text{ is valid.}\}, \quad (30)$$

where validity is defined by definition 10. If  $\mathcal{I}_{k_i} \neq \emptyset$ , let  $T_S^* = \max \mathcal{I}_{k_i}$ , then  $\mathcal{T}_{k_i,i}^{\text{com}} = \mathcal{T}_{k_i,i}^{\text{can}, T_S^*}$ . Otherwise, let  $\mathcal{T}_{k_i,i}^{\text{com}} = \mathcal{T}_{k_i-1,i}^{\text{com}}$  (i.e. continue tracking the previously committed trajectory).

To handle the asynchronous nature of a decentralized system, where multiple agents may replan at different times, we must introduce an assumption to prevent conflicts where two neighbors invalidate each other's plans simultaneously.

**Assumption 3.** *No two agents which can communicate update their committed trajectories simultaneously.*

**Remark 2.** *This assumption is necessary to ensure replanning agents have accurate information their neighbors [19]. If two neighboring agents replan simultaneously, they could each construct and validate colliding candidate trajectories, as neither would have knowledge of the other. An approach to relaxing this assumption is explored in [10], uses a check-recheck scheme to ensure agents have sufficiently up-to-date information about their neighbors'. Alternatively, a randomized backoff could be implemented, such that if two agents attempt to replan simultaneously, both wait for a short, randomized period before trying again. We acknowledge this is a significant limitation and an area of future work.*

Finally, we combine these elements into a single theorem that proves the forward-invariant safety of the entire protocol. This theorem asserts that as long as agents join the network and update their trajectories according to the specified rules, every agent will possess a valid committed trajectory at all times, thereby guaranteeing perpetual system-wide safety.

**Theorem 2** (Forward-Invariant Safety via Update Protocols). *Consider the system described in Problem 1, initialized at time  $t_0$  with an empty set of agents  $\mathcal{A}(t_0) = \emptyset$ . If all agents satisfy the R3R condition (22), perform all actions (i.e. joining and replanning) by algorithm 2 subject to Assumption 3, then for all  $t \geq t_0$ , the system is guaranteed to remain safe: (1). Collision Avoidance:  $\|p_i(t) - p_j(t)\| \geq \delta$  for all distinct  $i, j \in \mathcal{A}(t)$ . (2). Safe Set:  $x_i(t) \in \mathcal{S}$ ,  $\forall i \in \mathcal{A}(t)$ .*

*Proof.* The proof is by induction.

[Initial Conditions] The initial condition  $\mathcal{A}(t_0) = \emptyset$  is trivially safe. Consider the first agent  $i$  to join the network at time  $t_{0,i}$ . By algorithm 2, agent  $i$  can only join the network if algorithm 1 finds a valid candidate trajectory  $\mathcal{T}_{0,i}^{\text{can}, T_S}$ . By definition 10, this trajectory must remain in the safe set  $\mathcal{S}$

for all time, and because there are no other agents, the inter-agent collision constraint is trivially satisfied.

[Inductive Hypothesis] Assume at some time  $t_k$ , all agents in the network  $\mathcal{A}(t_k)$  are tracking committed trajectories that were valid at their respective times of construction.

[Inductive Step] We must show the system remains safe after the next replanning event at time  $t_{k+1} > t_k$ . This can either be a new agent joining, or existing agent replanning.

*Case 1: A New Agent Joins* Agent  $j \notin \mathcal{A}(t_{k+1})$  must execute algorithm 2, and construct a valid committed trajectory  $\mathcal{T}_{0,j}^{\text{com}}$ . By theorem 1, because agent  $j$ 's trajectory and all existing agents trajectories are valid, the entire system is guaranteed to be safe.

*Case 2: Existing Agent Replans* Assume agent  $i \in \mathcal{A}(t_{k+1})$  executes algorithm 2. If the AttemptReplan algorithm fails, the agent continues tracking its previous committed trajectory. The set of active trajectories in the system is unchanged, so the system remains safe by the inductive hypothesis.

If AttemptReplan succeeds, the agent adopts a new Valid committed trajectory,  $\mathcal{T}_{k+1,i}^{\text{com}}$ , which as before is guaranteed to be safe by the validity conditions as shown in theorem 1.

Thus, since the system starts in a safe state, and the protocols ensure state transitions are required to be safe, the system must remain safe for all future time.  $\square$

## IV. SIMULATION RESULTS

As a case study, we consider 2D curvature-constrained agents operating under the Dubins [20] vehicle dynamics

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v \cos \theta \\ v \sin \theta \\ \omega \end{bmatrix}, \quad (31)$$

where  $v$  is a fixed velocity, and  $\omega$  is the angular acceleration input, which is bounded by  $|\omega| \leq \omega_{\max}$ . This system is representative of a fixed-winged aircraft operating at a constant velocity. We perform simulations of up to 128 agents, operating in different environments, with static obstacles represented by occupancy grids. All simulations were performed in `julia`, using `Agents.jl` [21], running on a 2022 MacBook Air (Apple M2, 16 GB). All simulation code, as well as scripts to recreate all figures in this paper, is made public.

The nominal planner is a Dynamic-RRT\*<sup>4</sup>, a dubins-primitive based RRT\* planner that plans dynamically feasible trajectories, while avoiding static obstacles and other agents. It is important to note that *R3R* and *gatekeeper* are planner-agnostic—safety is a result of *R3R* and *gatekeeper*, not the planner.

Committed trajectories are constructed by maximizing the time for which the agent's nominal can be tracked before switching to the backup controller. The backup controller for the Dubins dynamics was implemented as a time-parameterized loiter circle.

<sup>4</sup>Implementation: <https://github.com/MarshallVielmetti/DynamicRRT.jl>

## Evolution of 64 Agents in City-Like Environment

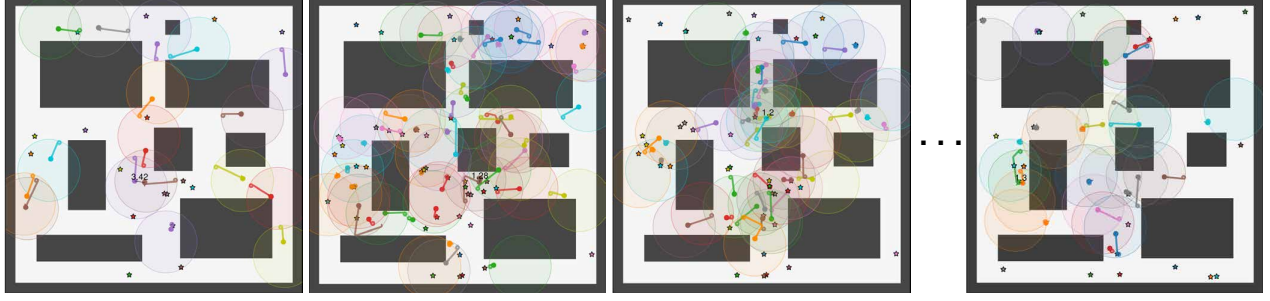


Fig. 3. 64 Dubins vehicles initialized with random start and goal positions in a city-like environment.

| #Agents - Environment | Safety % | Success % |
|-----------------------|----------|-----------|
| 8-swap                | 100%     | 100%      |
| 16-swap               | 100%     | 100%      |
| 8-city-like           | 100%     | 100%      |
| 16-city-like          | 100%     | 100%      |
| 32-city-like          | 100%     | 100%      |
| 64-city-like          | 100%     | 100%      |
| 128-city-like         | 100%     | 97%       |
| 8-willow-garage       | 100%     | 100%      |
| 16-willow-garage      | 100%     | 100%      |

TABLE I

SIMULATION RESULTS OF R3R & GATEKEEPER APPROACH.

DEMONSTRATES SAFETY AND DEADLOCK RESILIENCE.

Results from the simulations are shown in table I. Each scenario was run 5 times, with all simulation parameters held constant. For the city-like and willow garage scenarios, agent starting positions and goal positions were randomized at each trial. Success was defined as the percent of agents making it to their goal positions. Safety is defined as all environmental constraints being satisfied, and no agents coming within inter-agent collision avoidance distance. The simulation parameters for the city-like environment (approx.  $100 \times 100$ ) were:  $R^{\text{comm}} = 16.0$ ,  $\delta = 0.5$ ,  $R^{\text{plan}} = 5.16$ . For the 32 agent simulation, agents averaged 4.7 neighbors at each replanning iteration, with a maximum of 10. Replanning steps took an average of 1.4ms.

We see R3R achieves consistently high success and safety rates across all tested scenarios, demonstrating its effectiveness in complex multi-agent environments. See fig. 3 for an example of 64 agents operating in the city-like environment. The only failure case was in the 128-agent city-like scenario, where in one iteration a group of agents were unable to reach their goals within the simulation time limit. This was due to a combination of high density and limited free space, which led to deadlock situations that the current implementation could not resolve. Future work will focus on enhancing deadlock resolution strategies to further improve success rates in such challenging scenarios.

1) *Baseline Comparison:* We compare the performance of our solver to a baseline Nonlinear MPC in Julia using Ipopt[22] and JuMP[23]. For simplicity, we demonstrated this comparison in an empty environment. Agents are given the fully-planned trajectories of other agents within their communication radius. Inter-agent collision constraints are

encoded in the optimization problem as hard constraints. If the resulting problem is infeasible, we attempt to solve a relaxed version of the problem, with high penalties for constraint violation.

Due to the effects of the communication radius, the NMPC solver can find itself in unrecoverable states where no safe solution exists. It thus has a non-zero rate of safety violations, with performance degrading significantly as communication radius decreases and density increases.

| $n_{\text{agents}}$ | Safety %                |                         |                          |
|---------------------|-------------------------|-------------------------|--------------------------|
|                     | $R^{\text{comm}} = 1.0$ | $R^{\text{comm}} = 2.0$ | $R^{\text{comm}} = 10.0$ |
| 4                   | 87.5%                   | 100.0%                  | 100.0%                   |
| 6                   | 87.5%                   | 100.0%                  | 100.0%                   |
| 8                   | 75.0%                   | 87.5%                   | 100.0%                   |
| 10                  | 62.5%                   | 87.5%                   | 100.0%                   |
| 14                  | 50.0%                   | 75.0%                   | 100.0%                   |
| 16                  | 37.5%                   | 37.5%                   | 75.0%                    |

TABLE II

% OF TRIALS WITHOUT SAFETY VIOLATIONS FOR NMPC WITH DIFFERENT NUMBERS OF AGENTS AND COMMUNICATION RANGES.

Our empirical results highlight that the NMPC solver performed well when the communication radius was large enough to ensure a fully connected network, and the density was low enough to ensure that the problem was not too constrained. We see these results in Table II.

2) *Runtime Analysis:* Let  $N$  be total agents, environment area  $A$ , density  $\rho = N/A$ , and communication radius  $R^{\text{comm}}$ . Assuming agents are distributed uniformly, at some replanning iteration, the avg. number of neighbors within communication range is  $\lambda = \rho\pi(R^{\text{comm}})^2$ . All online costs depend on  $\lambda$  (local density), not  $N$ , provided  $\rho$  stays bounded as  $N$  grows. Thus, collision checking in nominal planning and candidate validation is  $O(\lambda)$ , not  $O(N)$  as with centralized or fully-connected approaches. For  $\lambda \ll N$ , this reflects significant savings.

We validate this analysis empirically. We consider a fixed number of agents  $N$  operating in a square with area  $d^2$ , where  $d$  is the length of the side of the square. To evaluate the effect of density on the algorithm, we fix the number of agents  $N$ , and vary the dimension  $d$  of the environment. To ensure the simulations are reflective of the varying density, we randomize start and goal positions for each agent, and conduct  $T = 20$  randomized trials for each  $d$ . Results are

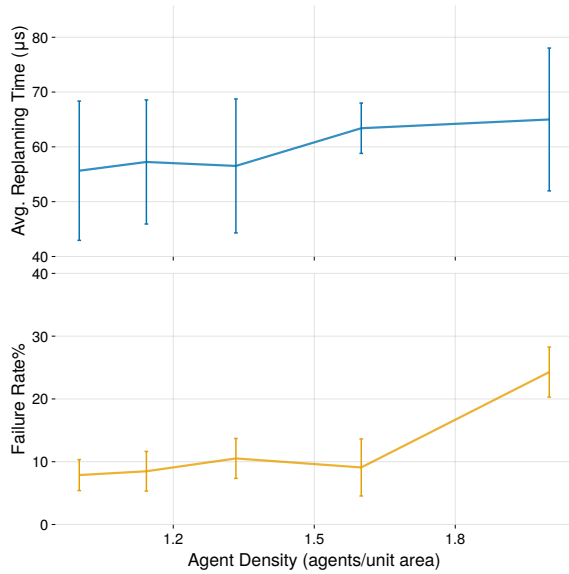


Fig. 4. Runtime performance of R3R as a function of agent density for  $N$  agents in a variable-sized environment over 20 randomized trials. (Top) The average time per replanning attempt increases with density due to a higher number of neighboring agents to consider in collision checks. (Bottom) The rate of replanning failures (no valid candidates found) also increases with density as it becomes more difficult to find a valid trajectory.

shown in fig. 4. As expected, we see that as the problem density increases the performance of the solver decreases.

## V. CONCLUSION

This paper presented R3R, a novel framework that establishes a formal, geometric link between an agent’s communication range and the range within it can plan its trajectories to ensure safety. Specifically, we introduced  $R$ -Bounded trajectories, which remain within a radius  $R$  of their starting point. By ensuring  $R$  is one-third of the communication radius, we geometrically guarantee that local safety checks imply global safety. We then leverage *gatekeeper* to guarantee that all agents always track valid,  $R$ -Bounded trajectories, ensuring forward-invariant safety.

Our work demonstrates that for a broad class of systems, scalability and formal safety guarantees are not mutually exclusive goals but can be achieved simultaneously through principled geometric constraints. The efficacy of the algorithm is demonstrated via the asynchronous deconfliction of  $>100$  agents in obstacle environments.

Our future work will focus on addressing the method’s limitations, namely to enhance the framework’s robustness, resolve deadlocks and incorporate models of network uncertainty to provide probabilistic safety guarantees under realistic communication.

## REFERENCES

- [1] H. Eskandaripour and E. Boldsaikhan, “Last-Mile Drone Delivery: Past, Present, and Future,” *Drones*, vol. 7, no. 2, p. 77, Jan. 2023.
- [2] J. Wen, L. He, and F. Zhu, “Swarm Robotics Control and Communications: Imminent Challenges for Next Generation Smart Logistics,” *IEEE Communications Magazine*, vol. 56, no. 7, pp. 102–107, Jul. 2018.
- [3] S. Javed, A. Hassan, R. Ahmad, W. Ahmed, R. Ahmed, A. Saadat, and M. Guizani, “State-of-the-Art and Future Research Challenges in UAV Swarms,” *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19 023–19 045, Jun. 2024.

- [4] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, “Conflict-based search for optimal multi-agent pathfinding,” *Artificial Intelligence*, vol. 219, pp. 40–66, Feb. 2015.
- [5] E. Boyarski, A. Felner, R. Stern, G. Sharon, O. Betzalel, D. Tolpin, and E. Shimony, “ICBS: The Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding,” *Proc. of the International Symposium on Combinatorial Search*, vol. 6, no. 1, pp. 223–225, Sep. 2021.
- [6] G. Gange, D. Harabor, and P. J. Stuckey, “Lazy CBS: Implicit Conflict-Based Search Using Lazy Clause Generation,” *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, pp. 155–162, Jul. 2019.
- [7] J. Li, D. Harabor, P. J. Stuckey, H. Ma, and S. Koenig, “Symmetry-Breaking Constraints for Grid-Based Multi-Agent Path Finding,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 6087–6095, Jul. 2019.
- [8] M. Čáp, P. Novák, J. Vokřínek, and M. Pěchouček, “Multi-agent RRT\*: Sampling-based Cooperative Pathfinding,” 2013.
- [9] J. Tordesillas and J. P. How, “MADER: Trajectory planner in multi-agent and dynamic environments,” *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 463–476, 2021.
- [10] K. Kondo, R. Figueroa, J. Rached, J. Tordesillas, P. C. Lusk, and J. P. How, “Robust MADER: Decentralized Multiagent Trajectory Planner Robust to Communication Delay in Dynamic Environments,” 2023.
- [11] J. Park, D. Kim, G. C. Kim, D. Oh, and H. J. Kim, “Online Distributed Trajectory Planning for Quadrotor Swarm With Feasibility Guarantee Using Linear Safe Corridor,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4869–4876, Apr. 2022.
- [12] J. Park, Y. Lee, I. Jang, and H. J. Kim, “DLSC: Distributed Multi-Agent Trajectory Planning in Maze-Like Dynamic Environments Using Linear Safe Corridor,” *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3739–3758, Oct. 2023.
- [13] —, “Decentralized trajectory planning for quadrotor swarm in cluttered environments with goal convergence guarantee,” *The International Journal of Robotics Research*, vol. 44, no. 8, pp. 1336–1359, Jul. 2025.
- [14] Y. Lee and J. Park, “MC-Swarm: Minimal-Communication Multi-Agent Trajectory Planning and Deadlock Resolution for Quadrotor Swarm,” 2025.
- [15] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, “EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments,” 2020.
- [16] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, “Swarm of micro flying robots in the wild,” *Science Robotics*, vol. 7, no. 66, p. eabm5954, May 2022. [Online]. Available: <https://www.science.org/doi/10.1126/scirobotics.abm5954>
- [17] B. Şenbaşlar and G. S. Sukhatme, “DREAM: Decentralized Real-Time Asynchronous Probabilistic Trajectory Planning for Collision-Free Multirobot Navigation in Cluttered Environments,” *IEEE Transactions on Robotics*, vol. 41, pp. 573–592, 2025.
- [18] D. R. Agrawal, R. Chen, and D. Panagou, “Gatekeeper: Online Safety Verification and Control for Nonlinear Systems in Dynamic Environments,” *IEEE Transactions on Robotics*, vol. 40, pp. 4358–4375, 2024.
- [19] T. M. Vielmetti, D. R. Agrawal, and D. Panagou, “Graph-gatekeeper: Distributed safe flight planning and control for urban air mobility,” in *2026 AIAA SciTech Forum*. AIAA, 2026, in press.
- [20] L. E. Dubins, “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, p. 497, Jul. 1957.
- [21] G. Datseris, A. R. Vahdati, and T. C. DuBois, “Agents.jl: a performant and feature-full agent-based modeling software of minimal code complexity,” *Simulation*, vol. 100, no. 10, pp. 1019–1031, Oct. 2024.
- [22] A. Wächter and L. T. Biegler, “On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, no. (1), pp. 25–57, 2006.
- [23] M. Lubin, O. Dowson, J. D. Garcia, J. Huchette, B. Legat, and J. P. Vielma, “JuMP 1.0: recent improvements to a modeling language for mathematical optimization,” *Mathematical Programming Computation*, vol. 15, no. 3, pp. 581–589, Sep. 2023. [Online]. Available: <https://link.springer.com/10.1007/s12532-023-00239-3>