2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Decentralized Deadlock-free Trajectory Planning for Quadrotor Swarm in Obstacle-rich Environments

Jungwon Park, Inkyu Jang, and H. Jin Kim[1]

***Abstract*— This paper presents a decentralized multi-agent trajectory planning (MATP) algorithm that guarantees to generate a safe, deadlock-free trajectory in an obstacle-rich environment under a limited communication range. The proposed algorithm utilizes a grid-based multi-agent path planning (MAPP) algorithm for deadlock resolution, and we introduce the subgoal optimization method to make the agent converge to the waypoint generated from the MAPP without deadlock. In addition, the proposed algorithm ensures the feasibility of the optimization problem and collision avoidance by adopting a linear safe corridor (LSC). We verify that the proposed algorithm does not cause a deadlock in both random forests and dense mazes regardless of communication range, and it outperforms our previous work in flight time and distance. We validate the proposed algorithm through a hardware demonstration with ten quadrotors.**

I. INTRODUCTION

Multi-agent trajectory planning (MATP) is essential to utilize a large group of unmanned vehicles in various applications such as search and rescue, surveillance, and transportation. Among many MATP algorithms, decentralized approaches have received much attention due to their high scalability and low computation load, which enables online planning. However, most decentralized algorithms do not consider obstacles [1–3] or have a risk of causing a deadlock in an obstacle-rich environment [4–6].

This paper presents a decentralized multi-agent trajectory planning (MATP) algorithm that guarantees to generate a safe, deadlock-free trajectory in a cluttered environment. The proposed method solves a deadlock through the following three steps. First, we compute the waypoint of each agent using a decentralized grid-based multi-agent path planning (MAPP) algorithm. Then, we optimize a subgoal of each agent considering the collision constraints and communication range so that the agent can reach the waypoint without deadlock. Finally, we conduct trajectory optimization to make the agent converge to the waypoint. As a result, the proposed algorithm allows the agent to reach the goal by following the waypoints from the grid-based MAPP. We utilize a linear safe corridor (LSC) [7] to guarantee the feasibility of the optimization problem and collision avoidance. Moreover,

This research was supported by Unmanned Vehicles Core Technology Research and Development Program through the National Research Foundation of Korea (NRF) and Unmanned Vehicle Advanced Research Center(UVARC) funded by the Ministry of Science and ICT, the Republic of Korea (NRF-2020M3C1C1A01086411). This research was supported by a research grant from Hyundai Motor Company / Kia and Hyundai NGV.

[1]The authors are with the Department of Mechanical and Aerospace Engineering, Seoul National University (SNU), and Automation and Systems Research Institute (ASRI), Seoul 08826, South Korea {qwerty35, leplusbon, hjinkim}@snu.ac.kr

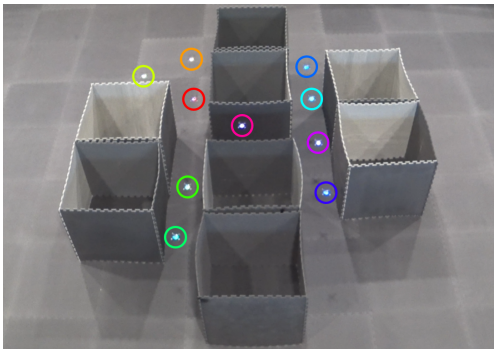


Fig. 1: Experiment with 10 quadrotors in a dense maze.

the proposed algorithm can be employed for robots with a limited communication range as long as they can configure an ad-hoc network. To the best of our knowledge, this is the first decentralized MATP algorithm that guarantees the feasibility of the optimization problem, collision avoidance, and deadlock-free in a dense maze-like environment. We conducted a hardware demonstration to verify the operability of the proposed algorithm, as shown in Fig. 1. We release the source code in `https://github.com/qwerty35/lsc_dr_planner`.

We summarize the main contributions as follows:

- Decentralized multi-agent trajectory planning algorithm that guarantees to prevent deadlock in a dense maze-like environment.
- Constraint generation method that ensures the feasibility of the optimization problem and collision avoidance under the limited communication range.
- Subgoal optimization method that allows the agent to converge to the waypoint without causing a deadlock.

II. RELATED WORK

MATP algorithms can be divided into two approaches: centralized and decentralized methods. The authors of [8, 9] present centralized planning algorithms that utilize a grid-based multi-agent path planning (MAPP) algorithm such as ECBS [10] to plan an initial trajectory and optimize it. This approach guarantees deadlock-free in a maze-like environment, but it is not scalable to the number of agents. On the other hand, decentralized methods [2, 4, 5, 11] show higher scalability than centralized ones, but they often suffer deadlock in a narrow corridor.

For deadlock resolution, many decentralized algorithms adopt the right-hand rule [1, 6, 12–14], which moves the goal point to the right side after the deadlock is detected. This approach works well in an obstacle-free environment,

but there is a risk of another deadlock even after changing the goal point. Another deadlock resolution method is to replan each robot's trajectory sequentially. In [15], a local coordinator asks neighboring agents to plan different trajectories until the deadlock is resolved. The authors of [16] introduce a token-based cooperative strategy, that determines which robots to yield the path by bidding. However, under these methods, there are cases where deadlock cannot be resolved by replanning an alternative trajectory of individual agents. The authors of [17] introduce a centralized high-level coordinator for deadlock resolution. This method is suitable for deadlock resolution in a cluttered environment, but all agents must be connected to the centralized coordinator during the entire mission.

Several works guarantee deadlock-free in obstacle-free or sparse environments. The authors of [3] introduce a warning band to prevent the agents from clumping together. In [18], an artificial potential field (APF) is extended to solve the deadlock. The authors of [19] conduct deadlock analysis and resolution for 2 to 3 agents. However, there is a limitation that these methods cannot solve deadlock in a cluttered environment such as a maze. In [20, 21], the grid-based MAPP is utilized to solve deadlock, similar to the proposed method. The authors of [20] adopt a mode-switching strategy, which converts the planner mode to follow the waypoint from MAPP when the deadlock is detected. The authors of [21] utilize the discrete path from MAPP as an initial trajectory. However, these methods do not provide a theoretical guarantee for deadlock resolution. Compared to the previous work [7], the proposed algorithm does not require a fully connected network for collision avoidance, and it guarantees deadlock-free for dense maze-like environments.

III. PROBLEM STATEMENT

We suppose that N agents with radius r are deployed in a 2-dimensional space with static obstacles. Our goal is to plan a safe and deadlock-free trajectory for the agents under a limited communication range. The start and goal points of the agent i are $\mathbf{s}^i$ and $\mathbf{g}^i_{des}$, respectively. We denote a set that includes all agents as $\mathcal{I}$ and a set consisting of agent i and the agents that can communicate with the agent i as a *connected group* $\mathcal{N}^i$.

A. Assumption

1) Obstacle: The position of the static obstacles is given as prior knowledge.

2) Grid: All agents share the same grid space $G = (V, E)$, where the grid size d is larger than $2\sqrt{2}r$. If the agent is on the grid, then the agent does not collide with static obstacles.

3) Mission: The start and desired goal points of all agents are located at the vertex of the grid space, and inter-agent collision do not occur at the start point. All agents start the mission at the same time.

4) Communication: The agents i and j can share the information without a communication loss or delay if the agents satisfy the following:

$$\|\mathbf{p}^i(t) - \mathbf{p}^j(t)\|_\infty \leq r_c \tag{1}$$

where $\mathbf{p}^i(t)$ is the position of the agent i, $\|\cdot\|_\infty$ is the L-infinity norm, and $r_c > 2d$ is the communication range. All agents can configure an ad-hoc network to relay information between the agents within the communication range.

B. Agent Description

1) Trajectory representation: We represent the agent's trajectory to the M-segment piecewise Bernstein polynomial [22], thanks to the differential flatness of quadrotor dynamics [23]. More precisely, the m^{th} segment of the trajectory of the agent i is formulated as follows:

$$\mathbf{p}^i_k(t) = \sum_{l=0}^{n} \mathbf{c}^i_{k,m,l} b_{l,n}\left(\frac{t - T_{k+m-1}}{\Delta t}\right), \forall t \in [T_{k+m-1}, T_{k+m}] \tag{2}$$

where k is the current replanning step, $\mathbf{p}^i_k(t)$ is the trajectory of the agent i, $\mathbf{c}^i_{k,m,l} \in \mathbb{R}^2$ is the control point, $n > 4$ is the degree of the polynomial, $b_{l,n}(t)$ is Bernstein basis polynomial, T_0 is the mission start time, $T_k = T_0 + k\Delta t$, and Δt is the duration of the trajectory segment.

2) Collision avoidance: We define that the agent i is safe from a collision if the following conditions hold:

$$\|\mathbf{p}^i_k(t) - \mathbf{p}^j_k(t)\| \geq 2r, \forall t, j \in \mathcal{I}\backslash\{i\} \tag{3}$$

$$(\mathbf{p}^i_k(t) \oplus \mathcal{C}^{i,o}) \cap \mathcal{O} = \varnothing, \forall t \tag{4}$$

$$\mathcal{C}^{i,o} = \{\mathbf{x} \in \mathbb{R}^2 \mid \|\mathbf{x}\| < r\} \tag{5}$$

where $\oplus$ is the Minkowski sum, $\mathcal{C}^{i,o}$ is the obstacle collision model, $\mathcal{O}$ is the space occupied by the obstacles, and $\|\cdot\|$ is the Euclidean norm.

3) Dynamical limit: We model the dynamical limit of the agent as follows:

$$\|\mathbf{v}^i(t)\|_\infty \leq v_{max}, \forall t \tag{6}$$

$$\|\mathbf{a}^i(t)\|_\infty \leq a_{max}, \forall t \tag{7}$$

where $\mathbf{v}^i(t)$ and $\mathbf{a}^i(t)$ are the velocity and acceleration of the agent i, respectively, and v_{max} and a_{max} are the agent's maximum velocity and acceleration, respectively.

IV. METHOD

As described in Alg. 1, the proposed algorithm consists of the communication phase (lines 3-4) and trajectory generation phase (lines 5-13). During the communication phase, each agent configures an ad-hoc network between the agents within the communication range. After network configuration, we conduct a grid-based multi-agent path planning (MAPP) algorithm to determine the waypoint of the agent (line 3, Sec. IV-A). Then, the agent shares the previously planned trajectory and subgoal with the connected group (line 4). In the trajectory generation phase, we generate initial trajectories using the previously planned trajectories (line 6,

Sec. IV-B). The initial trajectories are utilized to construct feasible collision constraints (lines 7-9, Sec. IV-C). Next, we search for the subgoal that does not cause deadlock (line 10, Sec. IV-D). Finally, we conduct trajectory optimization and execute it (lines 11-12, Sec. IV-E). We repeat the above process until all agents reach the desired goal.

Algorithm 1: Trajectory planning for the agent i

Input: Start point $\mathbf{s}^i$, desired goal point $\mathbf{g}^i_{des}$, and obstacle space $\mathcal{O}$
Output: Trajectory of the agent i $\mathbf{p}^i_k(t)$

1 $k \leftarrow 0$;
2 **while** *not all agents at desired goal* **do**
  // Communication phase
3 $\mathbf{w}^{j\in\mathcal{N}^i}_k \leftarrow$ decentralizedMAPP($\mathbf{g}^i_{k-1}, \mathbf{w}^i_{k-1}, \mathbf{g}^i_{des}$);
4 $\mathbf{p}^{j\in\mathcal{N}^i}_{k-1}(t), \mathbf{g}^{j\in\mathcal{N}^i}_{k-1} \leftarrow$ communicate($\mathbf{p}^i_{k-1}(t), \mathbf{g}^i_{k-1}$);
  // Trajectory generation phase
5 **for** $\forall j \in \mathcal{N}^i$ **do**
6 $\hat{\mathbf{p}}^j_k(t) \leftarrow$ planInitialTraj($\mathbf{p}^j_{k-1}(t)$);
7 $\mathcal{L}^{i,j}_{k,m,l} \leftarrow$ buildLSC($\hat{\mathbf{p}}^i_k(t), \hat{\mathbf{p}}^j_k(t)$);
8 **end**
9 $\mathcal{S}^i_{k,m} \leftarrow$ buildSFC($\hat{\mathbf{p}}^i_k(t), \mathcal{O}$);
10 $\mathbf{g}^i_k \leftarrow$ subgoalOpt($\mathbf{g}^i_{k-1}, \mathbf{w}^i_k, \mathcal{S}^i_{k,m}, \mathcal{L}^{i,j}_{k,m,l}$);
11 $\mathbf{p}^i_k(t) \leftarrow$ trajOpt($\mathcal{S}^i_{k,m}, \mathcal{L}^{i,j}_{k,m,l}, \mathbf{g}^i_k$);
12 executeTrajectory($\mathbf{p}^i_k(t)$);
13 $k \leftarrow k+1$;
14 **end**

A. Decentralized Multi-agent Path Planning

We introduce a decentralized MAPP to plan the waypoint, which provides guidance on deadlock resolution. For every replanning step, each agent configures the ad-hoc network between agents within the communication range, and one agent among the connected group is selected as a local coordinator. The local coordinator collects the subgoals, waypoints, and desired goals of the agents in the connected group. Then, the coordinator plans collision-free discrete paths using the MAPP algorithm on the grid space G, where the start points of MAPP are the previous waypoints $\mathbf{w}^{i\in\mathcal{N}^i}_{k-1}$, and the goal points are the desired goals. If it is the first step of the planning, we set the start point as $\mathbf{s}^i$ instead. In this work, we adopt Priority Inheritance with Backtracking (PIBT) [24] for MAPP algortihm because it is a scalable algorithm that guarantees *reachability*, which ensures that all agents can reach the desired goal within a finite time. Next, the coordinator updates the agent's waypoint $\mathbf{w}^i_k$ to the second waypoint of the discrete path (the point one step after the start point) if the following two conditions are satisfied. First, the subgoal and waypoint at the previous step must be equal (8). Second, the distance between the updated waypoint and the endpoints of the previous trajectory's segments must be shorter than $r_c/2$ (9):

$$\mathbf{g}^i_{k-1} = \mathbf{w}^i_{k-1} \tag{8}$$

$$\|\mathbf{w}^i_k - \mathbf{p}^i_{k-1}(T_{k+m-2})\|_\infty < \frac{r_c}{2}, \forall m = 1, \cdots, M \tag{9}$$

where $\mathbf{g}^i_k$ and $\mathbf{w}^i_k$ are the subgoal and waypoint at the replanning step k, respectively. Otherwise, we reuse the previous waypoint as the current waypoint. Lastly, we check whether the waypoints are duplicated in the connected group. If there are the same ones, we restore one of them to the previous waypoint. We repeat this process until there is no duplicated waypoint. Lemma 1 shows that the proposed waypoint update rule prevents duplicated waypoints.

Lemma 1. *For any pair of the agents $i \in \mathcal{I}$ and $j \in \mathcal{I}\backslash\{i\}$, $\boldsymbol{w}^i_k \neq \boldsymbol{w}^j_k$ holds for every replanning step $k > 0$.*

Proof. Due to the page limit, the proof can be found in [25]. ■

B. Initial Trajectory Planning

As in our previous work [7], we utilize an initial trajectory to construct feasible collision constraints. We plan the initial trajectory using the previously planned trajectories, or the current position if it is the first step of the planning:

$$\hat{\mathbf{p}}^i_k(t) = \begin{cases} \mathbf{s}^i & k = 0, t \in [T_0, T_M] \\ \mathbf{p}^i_{k-1}(t) & k > 0, t \in [T_k, T_{k+M-1}] \\ \mathbf{p}^i_{k-1}(T_{k+M-1}) & k > 0, t \in [T_{k+M-1}, T_{k+M}] \end{cases} \tag{10}$$

where $\hat{\mathbf{p}}^i_k(t)$ is the initial trajectory at the replanning step k. The control point of the initial trajectory is represented as follows:

$$\hat{\mathbf{c}}^i_{k,m,l} = \begin{cases} \mathbf{s}^i & k = 0 \\ \mathbf{c}^i_{k-1,m+1,l} & k > 0, m < M \\ \mathbf{c}^i_{k-1,M,n} & k > 0, m = M \end{cases} \tag{11}$$

where $\hat{\mathbf{c}}^i_{k,m,l}$ is the control point of the initial trajectory.

C. Collision Constraints Construction

In our previous work [7], we utilized a safe flight corridor (SFC) and linear safe corridor (LSC) for collision avoidance. However, these constraints may cause deadlock if the agent is blocked by the constraints before reaching the waypoint. For this reason, we modify the collision constraints so that the agent can proceed to the waypoint.

1) Obstacle avoidance: For obstacle avoidance, we construct the SFC as follows:

$$\mathcal{S}^i_{k,m} = \begin{cases} \mathcal{S}(\{\mathbf{s}^i, \mathbf{w}^i_k\}) & k = 0 \\ \mathcal{S}^i_{k-1,m+1} & k > 0, m < M \\ \mathcal{S}(\{\hat{\mathbf{c}}^i_{k,M,n}, \mathbf{g}^i_{k-1}, \mathbf{w}^i_k\})) & k > 0, m = M, \\ \mathcal{S}(\{\hat{\mathbf{c}}^i_{k,M,n}, \mathbf{g}^i_{k-1}\}) & \text{else} \end{cases} \tag{12}$$

$$(\text{Conv}(\{\hat{\mathbf{c}}^i_{k,M,n}, \mathbf{g}^i_{k-1}, \mathbf{w}^i_k\}) \oplus \mathcal{C}^{i,o}) \cap \mathcal{O} = \varnothing \tag{13}$$

where $\mathcal{S}^i_{k,m}$ is the SFC for m^{th} trajectory segment, $\mathcal{S}(\mathcal{P})$ is a convex set that includes the point set $\mathcal{P}$ and satisfies $(\mathcal{S}(\mathcal{P}) \oplus \mathcal{C}^{i,o}) \cap \mathcal{O} = \varnothing$, and $\text{Conv}(\cdot)$ is the convex hull operator that returns a convex hull of the input set. We generate the SFC using the axis-search method [9].

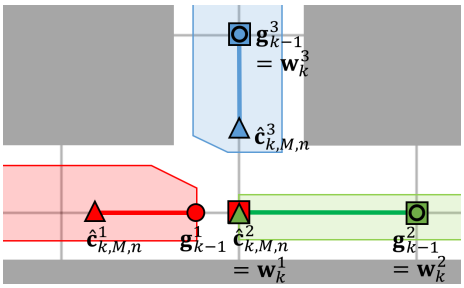


Fig. 2: Collision constraints for the last trajectory segment. The squares are the waypoints, the triangles are the final points of the initial trajectories, and the circles are the previously planned subgoals. The gray box is the static obstacle, and the color-shaded region is the feasible region that satisfies the collision constraints. We generate the collision constraint for the last segment to include the line segment between the final point and the subgoal, which is depicted as the thick line.

2) Inter-agent collision avoidance: If it is the first step of the planning or $m < M$, we construct the LSC using the same approach in [7]:

$$\mathcal{L}^{i,j}_{k,m,l} = \{\mathbf{x} \in \mathbb{R}^2 \mid (\mathbf{x} - \hat{\mathbf{c}}^j_{k,m,l}) \cdot \mathbf{n}^{i,j}_m - d^{i,j}_{m,l} \geq 0\} \quad (14a)$$

$$d^{i,j}_{m,l} = r + \frac{1}{2}(\hat{\mathbf{c}}^i_{k,m,l} - \hat{\mathbf{c}}^j_{k,m,l}) \cdot \mathbf{n}^{i,j}_m \quad (14b)$$

where $\mathcal{L}^{i,j}_{k,m,l}$ is the LSC between the agent i and j, $\mathbf{n}^{i,j}_m$ is the normal vector that satisfies $\mathbf{n}^{i,j}_m = -\mathbf{n}^{j,i}_m$, $d^{i,j}_{m,l}$ is the safety margin. The detailed LSC construction can be found in [7]. If it is not the first step of the planning and $m = M$, then we generate the LSC as follows:

$$\mathcal{L}^{i,j}_{k,M,l} = \{\mathbf{x} \in \mathbb{R}^2 \mid (\mathbf{x} - \mathbf{p}^j_{cls,i}) \cdot \mathbf{n}^{i,j}_M - d^{i,j}_{M,l} \geq 0\} \quad (15a)$$

$$\mathbf{n}^{i,j}_M = \frac{\mathbf{p}^i_{cls,j} - \mathbf{p}^j_{cls,i}}{\|\mathbf{p}^i_{cls,j} - \mathbf{p}^j_{cls,i}\|} \quad (15b)$$

$$d^{i,j}_{M,l} = r + \frac{1}{2}\|\mathbf{p}^i_{cls,j} - \mathbf{p}^j_{cls,i}\| \quad (15c)$$

where $\mathbf{p}^i_{cls,j} \in \langle \hat{\mathbf{c}}^i_{k,M,n}, \mathbf{g}^i_{k-1} \rangle$ and $\mathbf{p}^j_{cls,i} \in \langle \hat{\mathbf{c}}^j_{k,M,n}, \mathbf{g}^j_{k-1} \rangle$ are the closest points between $\langle \hat{\mathbf{c}}^i_{k,M,n}, \mathbf{g}^i_{k-1} \rangle$ and $\langle \hat{\mathbf{c}}^j_{k,M,n}, \mathbf{g}^j_{k-1} \rangle$, respectively, and $\langle \mathbf{a}, \mathbf{b} \rangle$ is the line segment between two points $\mathbf{a}$ and $\mathbf{b}$.

Fig. 2 shows the collision constraints for the last trajectory segment. We can observe that the feasible region of the agent always contains $\langle \mathbf{g}^i_{k-1}, \hat{\mathbf{c}}^i_{k,M,n} \rangle$. Thus, each agent can secure the free space to proceed to the subgoal $\mathbf{g}^i_{k-1}$, which will converge to the waypoint $\mathbf{w}^i_k$.

D. Subgoal Optimization

Suppose that the waypoint from MAPP does not satisfy the collision constraints. If we directly set this waypoint as the target point, this may lead to deadlock since the agent cannot reach the waypoint by the constraints. Therefore, we designate the point that is closest to the waypoint and satisfies the collision constraints as the subgoal. More precisely, we determine the subgoal by solving the following linear programming (LP) problem:

$$\begin{aligned} \underset{\mathbf{g}^i_k}{\text{minimize}} \quad & \|\mathbf{g}^i_k - \mathbf{w}^i_k\| \\ \text{subject to} \quad & \mathbf{g}^i_k \in \langle \mathbf{s}^i, \mathbf{w}^i_k \rangle \quad \text{if } k = 0 \\ & \mathbf{g}^i_k \in \langle \mathbf{g}^i_{k-1}, \mathbf{w}^i_k \rangle \quad \text{if } k > 0 \\ & \mathbf{g}^i_k \in \mathcal{S}^i_{k,M} \\ & \mathbf{g}^i_k \in \mathcal{L}^{i,j}_{k,M,n} \quad \forall j \in \mathcal{N}^i \end{aligned} \quad (16)$$

where $\mathbf{g}^i_k$ is the subgoal at the replanning step k. We will prove that the subgoal in (16) does not cause deadlock in section V. Lemma 2 shows the properties of the subgoal.

Lemma 2. *For the agents $i \in \mathcal{I}$, $j \in \mathcal{I} \backslash \{i\}$, (i) there exists a grid edge $e \in E$ such that $\langle \mathbf{g}^i_k, \mathbf{w}^i_k \rangle \subset e$, (ii) $\mathbf{g}^i_k \neq \mathbf{g}^j_k$, (iii) if there exists an edge $e \in E$ such that $\mathbf{g}^i_k \in e$ and $\mathbf{g}^j_k \in e$, then $\mathbf{g}^i_k$ or $\mathbf{g}^j_k$ is on the vertex of the grid G.*

Proof. Due to the page limit, the proof can be found in [25]. ■

E. Trajectory Optimization

1) Objective function: We formulate the objective function to minimize both the distance to the current subgoal and the jerk of the trajectory as follows:

$$J^i_{err} = w_{err}\|\mathbf{p}^i_k(T_M) - \mathbf{g}^i_k\|^2 \quad (17)$$

$$J^i_{der} = w_{der}\int_{T_0}^{T_M} \left\| \frac{d^3}{dt^3}\mathbf{p}^i_k(t) \right\|^2 dt \quad (18)$$

where $w_{err}, w_{der} > 0$ are the weight coefficients.

2) Communication range: If we do not consider the communication range when generating the trajectory, the agent may collide with an agent outside the range. Also, if the distance between the agent and its waypoint is longer than half the communication range, an agent outside the range can assign the same waypoint. Hence we add the following constraints to prevent the collision and duplicated waypoints between agents outside the range:

$$\|\mathbf{c}^i_{k,m+h,l} - \mathbf{c}^i_{k,m,0}\|_\infty \leq \frac{r_c}{2} - r, \forall h \geq 0, m, l, \quad (19)$$

$$\|\mathbf{c}^i_{k,m,n} - \mathbf{w}^i_k\|_\infty \leq \frac{r_c}{2}, \forall m \quad (20)$$

3) Other constraints: The trajectory must satisfy the initial condition to match the agent's current state, and we impose continuity constraints to make the trajectory continuous up to the acceleration. We add the final stop condition for the feasibility of the optimization problem (i.e., $\mathbf{c}^i_{k,M,n} = \mathbf{c}^i_{k,M,n-1} = \mathbf{c}^i_{k,M,n-2}$). The dynamical limit (6), (7) can be represented to affine inequality using the convex hull property of the Bernstein polynomial. We can reformulate the above constraints as the following affine constraints:

$$A_{eq}\mathbf{c}^i_k = \mathbf{b}_{eq} \quad (21)$$

$$A_{dyn}\mathbf{c}^i_k \preceq \mathbf{b}_{dyn} \quad (22)$$

where $\mathbf{c}^i_k$ is the vector that concatenates the control points of $\mathbf{p}^i_k(t)$.

4) Optimization problem: We conduct the trajectory optimization by solving the following quadratic programming (QP) problem:

$$\begin{aligned} \underset{\mathbf{c}_k^i}{\text{minimize}} \quad & J_{err}^i + J_{der}^i \\ \text{subject to} \quad & \mathbf{c}_{k,m,l}^i \in \mathcal{S}_{k,m}^i \qquad \forall m, l \\ & \mathbf{c}_{k,m,l}^i \in \mathcal{L}_{k,m,l}^{i,j} \qquad \forall j \in \mathcal{N}^i, m, l \\ & (19), (20), (21), (22) \end{aligned} \tag{23}$$

V. Theoretical Guarantee

We present the theoretical guarantee of the proposed algorithm. We omit many details in the proof due to the page limit, and the omitted part can be found in [25].

Theorem 1. *(Collision avoidance) The trajectory from (23) does not cause inter-agent collision or collision between agent and obstacle.*

Proof. The agent does not collide with obstacles since $\mathbf{p}_k^i(t) \in \mathcal{S}_{k,m}^i$ by the convex hull property of Bernstein polynomial and $(\mathcal{S}_{k,m}^i \oplus \mathcal{C}^{i,o}) \cap \mathcal{O} = \varnothing$ by (12). For the agent $j \in \mathcal{N}^i$, we can use the similar method as the proof of Lemma 2 in [7] to prove that there is no collision between agents i and j. For the agent $j \notin \mathcal{N}^i$, then we can show that $\|\mathbf{p}_k^i(t) - \mathbf{p}_k^j(t)\| \geq \|\mathbf{p}_k^i(t) - \mathbf{p}_k^j(t)\|_\infty \geq 2r$ using (19). Thus, the trajectory from (23) does not cause collision. ■

Theorem 2. *(Feasibility of optimization problem) If the replanning period is same as the segment duration Δt, the solution of (23) always exists for every replanning step.*

Proof. Similar to the proof of Theorem 1 in [7], we can prove this by showing that the initial trajectory $\hat{\mathbf{p}}_k^i(t)$ always satisfies the constraints in (23) for every replanning step. ■

Theorem 3. *(Deadlock resolution) If $d > 2\sqrt{2}r$ and the MAPP in section IV-A does not cause deadlock, then Alg. 1 does not cause deadlock.*

Proof. We can show that agent i does not cause deadlock if:

$$\mathbf{c}_{k,M,n}^i \neq \mathbf{g}_k^i \tag{24}$$

$$\mathbf{g}_k^i \in \mathcal{S}_{k,M}^i, \quad \mathbf{g}_k^i \in \mathcal{L}_{k,M,n}^{i,\forall j \in \mathcal{N}^i}, \quad \|\mathbf{g}_k^i - \mathbf{w}_k^i\|_\infty \leq \frac{r_c}{2} \tag{25}$$

Here, $\mathbf{g}_k^i$ always satisfies (25) due to the constraints in (16) and the assumption that $r_c > 2d$. Therefore, a deadlock occurs if and only if there exists the agent i such that:

$$\mathbf{g}_k^i = \mathbf{g}_{k_0}^i \neq \mathbf{w}_{k_0}^i, k \geq k_0 \tag{26}$$

where k_0 is the replanning step when the deadlock happens. Let $\mathcal{D}$ be a set of agents that satisfy (26). Then, by the KKT conditions [26] of (16), the agent i in $\mathcal{D}$ must have a *blocking agent* j that prevents the agent i from reaching its waypoint as follows:

$$\|\mathbf{g}_k^i - \mathbf{g}_k^j\| = 2r, \forall k > k_0 \tag{27}$$

$$(\mathbf{g}_k^j - \mathbf{g}_k^i)^T (\mathbf{w}_k^i - \mathbf{g}_k^i) > 0, \forall k > k_0 \tag{28}$$

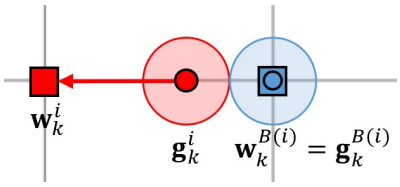


Fig. 3: Illustration for the proof of Theorem 3.

We denote the blocking agent of the agent i as $B(i)$. Suppose that $B(i) \notin \mathcal{D}$ for some $i \in \mathcal{D}$. Then, the agents i and $B(i)$ must be on the same grid edge after the deadlock happens, as shown in Fig. 3. It is because $B(i)$ converges to its waypoint and the distance between two agents is $2r$ by (27). However, the waypoints of two agents must be different by Lemma 1, so $B(i)$ cannot satisfy (28). Thus, $B(i) \in \mathcal{D}$ for $\forall i \in \mathcal{D}$. Let us choose an agent i in $\mathcal{D}$ that satisfies the following:

$$\Delta(i) \geq \Delta(j), \forall j \in \mathcal{D} \tag{29}$$

where $\Delta(i) = \|\mathbf{w}_k^i - \mathbf{g}_k^i\|$. Since $i \in \mathcal{D}$, the agent i has its blocking agent $B(i) \in \mathcal{D}$, and $\Delta(B(i))$ is computed as:

$$\Delta(B(i)) = \begin{cases} d - 2r + \Delta(i), & \text{if } \mathbf{n}_{dir} > 0 \\ d - \sqrt{4r^2 - \Delta(i)^2}, & \text{else} \end{cases} \tag{30}$$

$$\mathbf{n}_{dir} = (\mathbf{w}_k^{B(i)} - \mathbf{g}_k^{B(i)})^T (\mathbf{w}_k^i - \mathbf{g}_k^i) \tag{31}$$

Here, we can observe that $\Delta(B(i)) > \Delta(i)$ because $d > 2\sqrt{2}r$. However, it is inconsistent with the assumption (29). Therefore, there is at least one agent that does not have a blocking agent. Thus, Alg. 1 does not cause deadlock.

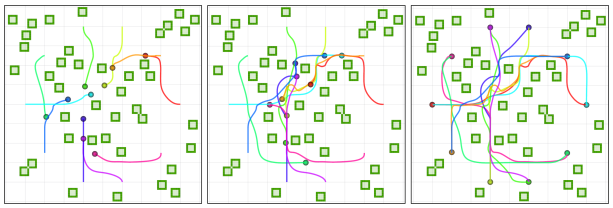
■

VI. Evaluation

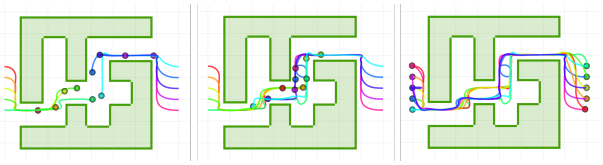
We compared the following algorithms to verify the performance of the proposed algorithm:

- LSC-PB (LSC with priority-based goal planning, [7])
- LSC-DR (LSC with deadlock resolution, **proposed**)

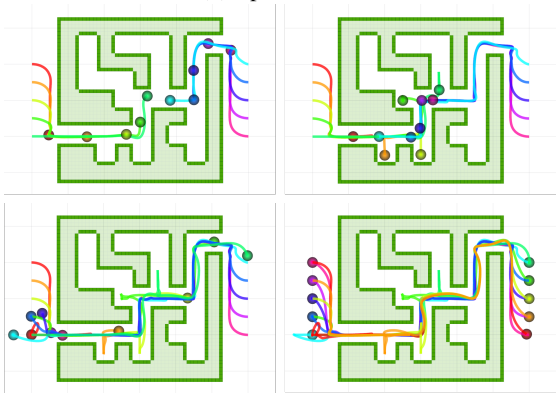
We modeled the agent with radius $r = 0.15$ m, maximum velocity $v_{max} = 1.0$ m/s, maximum acceleration $a_{max} = 2.0$ m/s^2 based on the experimental result with Crazyflie 2.1. To represent the trajectory, we set the degree of polynomials $n = 5$, the number of segments $M = 10$, and the segment time $\Delta t = 0.2$ s. Therefore, the total planning horizon is 2 s. We assigned the replanning period to be $\Delta t = 0.2$ s to satisfy the assumption in Thm. 2, so the trajectories are updated with the rate of 5 Hz at the same time. For decentralized MAPP, we implemented PIBT based on the source code of [27], and we set the grid size $d = 0.5$ m to fulfill the assumption that $d > 2\sqrt{2}r$. We used the Octomap library [28] to represent the obstacles, and we utilized randomized Prim's algorithm [29] to generate mazes. We set $w_{err} = 1$, $w_{der} = 0.01$ as the parameters of the objective function, and we used the CPLEX solver [30] for subgoal and trajectory optimization. The simulation was executed on a laptop with Intel Core i7-9750H @ 2.60GHz CPU and 32G RAM.



(a) Random forest



(b) Sparse maze



(c) Dense maze

Fig. 4: Trajectory generation result of the proposed method (r_c = 3 m). The circle and line are the agents and their trajectories respectively, and the green regions are obstacles.

A. Simulation

We conducted the comparison in the three environments:

(i) Random forest. We deploy 40 static obstacles in a random position and ten agents in a circle with 4 m radius. The goal point of the agent is at the antipodal point of the start point, as shown in Fig. 4a.

(ii) Sparse maze. It consists of 6 × 6 cells, and each cell size is 1.0 m × 1.0 m, thus three agents can pass the corridor simultaneously. The maze has two entrances, and there are five agents at each entrance. We assigned each agent's goal point to the entrance on the other side of the maze, as depicted in Fig. 4b.

(iii) Dense maze. It consists of 9 × 9 cells, and each cell size is 0.5 m × 0.5 m, thus only one agent can pass the corridor. We assigned the mission similar to the sparse maze, as illustrated in Fig. 4c.

We judge that the mission failed when a collision occurred or when the agent failed to reach the goal within 60 s. For each map, we ran 30 trials changing the obstacle's position.

Table I and Fig. 4 describe the simulation results in obstacle environments. LSC-PB shows the perfect success rate in sparse environments and does not cause a collision in all cases. However, LSC-PB fails to reach the goal in the dense maze because it cannot solve a deadlock when there is no space to yield to a higher-priority agent. On the other hand, our algorithm achieves the perfect success rate for all types of environments regardless of the communication range. It validates that the proposed algorithm can solve a deadlock even in a dense maze-like environment without a centralized coordinator.

TABLE I: Comparison with previous work [7]. The bold number indicates the best result (sr: success rate (%), T_f: flight time (s), L: flight distance per agent (m), T_c: computation time (ms)).

Env.	Method	sr	T_f	L	T_c
Random forest	LSC-PB [7] (r_c = ∞)	100	25.7	11.8	8.15
	LSC-DR (r_c = 2 m)	100	28.8	11.7	**7.86**
	LSC-DR (r_c = 3 m)	100	20.7	11.3	8.01
	LSC-DR (r_c = 4 m)	100	19.9	11.3	8.23
	LSC-DR (r_c = ∞)	100	**19.1**	**11.1**	8.16
Sparse maze	LSC-PB [7] (r_c = ∞)	100	33.7	13.9	9.05
	LSC-DR (r_c = 2 m)	100	34.4	13.5	**8.51**
	LSC-DR (r_c = 3 m)	100	27.1	13.1	8.62
	LSC-DR (r_c = 4 m)	100	**23.7**	**12.6**	8.66
	LSC-DR (r_c = ∞)	100	23.9	12.7	8.67
Dense maze	LSC-PB [7] (r_c = ∞)	0	-	-	-
	LSC-DR (r_c = 2 m)	100	61.4	**16.5**	7.30
	LSC-DR (r_c = 3 m)	100	51.0	16.6	7.44
	LSC-DR (r_c = 4 m)	100	50.9	17.1	7.31
	LSC-DR (r_c = ∞)	100	**48.3**	16.7	**7.24**

The proposed method shows a 27.6% shorter flight time and a 7.4% shorter flight distance compared to LSC-PB when r_c = ∞. It is because LSC-PB performs a deadlock resolution only when the distance between the agents is close enough. Conversely, the proposed algorithm utilizes the final trajectory point, not the current position, for deadlock resolution. Therefore, the agent does not need to wait until other agents clump together.

B. Hardware demonstration

Fig. 1 shows the hardware demonstration with ten Crazyflie 2.1 quadrotors in the dense maze. We set the communication range as 2 m, and we used the Crazyswarm [31] to broadcast the trajectory to the agents. We present the full demonstration in the supplemental video, and there was no collision or deadlock during the entire flight.

VII. CONCLUSIONS

We presented the online decentralized MATP algorithm that guarantees to generate a safe, deadlock-free in a cluttered environment. We utilized the decentralized MAPP for deadlock resolution, and generate the constraints so that the agent could reach the waypoint without deadlock. We proved that the proposed algorithm guarantees the feasibility of the optimization problem, collision avoidance, and deadlock-free for every replanning step. We verified that the proposed method does not cause collision or deadlock, regardless of the size of the free space or communication range. Moreover, the proposed algorithm has a 27.6% shorter flight time and a 7.4% shorter flight distance than our previous work. In future work, we plan to extend it to three-dimensional spaces or dynamic environments.

REFERENCES

[1] D. Zhou, Z. Wang, S. Bandyopadhyay, and M. Schwager, "Fast, online collision avoidance for dynamic vehicles using buffered voronoi cells," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1047–1054, 2017.

[2] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, "Online trajectory generation with distributed model predictive control for multi-robot motion planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 604–611, 2020.

[3] Y. Chen, M. Guo, and Z. Li, "Recursive feasibility and deadlock resolution in mpc-based multi-robot trajectory generation," *arXiv preprint arXiv:2202.06071*, 2022.

[4] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, "Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4101–4107.

[5] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, 2021.

[6] C. Toumieh and A. Lambert, "Decentralized multi-agent planning using model predictive control and time-aware safe corridors," *IEEE Robotics and Automation Letters*, 2022.

[7] 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, 2022.

[8] W. Hönig, J. A. Preiss, T. S. Kumar, G. S. Sukhatme, and N. Ayanian, "Trajectory planning for quadrotor swarms," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 856–869, 2018.

[9] J. Park, J. Kim, I. Jang, and H. J. Kim, "Efficient multi-agent trajectory planning with feasibility guarantee using relative bernstein polynomial," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 434–440.

[10] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Seventh Annual Symposium on Combinatorial Search*, 2014.

[11] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[12] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[13] H. Zhu and J. Alonso-Mora, "B-uavc: Buffered uncertainty-aware voronoi cells for probabilistic multi-robot collision avoidance," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*. IEEE, 2019, pp. 162–168.

[14] M. Abdullhak and A. Vardy, "Deadlock prediction and recovery for distributed collision avoidance with buffered voronoi cells," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 429–436.

[15] M. Jager and B. Nebel, "Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, vol. 3. IEEE, 2001, pp. 1213–1219.

[16] V. R. Desaraju and J. P. How, "Decentralized path planning for multi-agent teams with complex constraints," *Autonomous Robots*, vol. 32, no. 4, pp. 385–403, 2012.

[17] J. Alonso-Mora, J. A. DeCastro, V. Raman, D. Rus, and H. Kress-Gazit, "Reactive mission and motion planning with deadlock resolution avoiding dynamic obstacles," *Autonomous Robots*, vol. 42, no. 4, pp. 801–824, 2018.

[18] S. H. Semnani, A. H. de Ruiter, and H. H. Liu, "Force-based algorithm for motion planning of large agent," *IEEE Transactions on Cybernetics*, 2020.

[19] J. S. Grover, C. Liu, and K. Sycara, "Deadlock analysis and resolution for multi-robot systems," in *International Workshop on the Algorithmic Foundations of Robotics*. Springer, 2020, pp. 294–312.

[20] S. Dergachev and K. Yakovlev, "Distributed multi-agent navigation based on reciprocal collision avoidance and locally confined multi-agent path finding," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2021, pp. 1489–1494.

[21] J. Hou, X. Zhou, Z. Gan, and F. Gao, "Enhanced decentralized autonomous aerial robot teams with group planning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9240–9247, 2022.

[22] R. T. Farouki, "The bernstein polynomial basis: A centennial retrospective," *Computer Aided Geometric Design*, vol. 29, no. 6, pp. 379–419, 2012.

[23] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 477–483.

[24] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, vol. 310, p. 103752, 2022.

[25] J. Park, I. Jang, and H. J. Kim, "Decentralized deadlock-free trajectory planning for quadrotor swarm in obstacle-rich environments–extended version," *arXiv preprint arXiv:2209.09447*, 2022.

[26] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[27] K. Okumura, Y. Tamura, and X. Défago, "Iterative refinement for real-time multi-robot path planning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 9690–9697.

[28] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[29] M. Foltin, "Automated maze generation and human interaction," *Brno: Masaryk University Faculty Of Informatics*, 2011.

[30] I. CPLEX, "12.7. 0 user's manual," 2016.

[31] J. A. Preiss, W. Honig, G. S. Sukhatme, and N. Ayanian, "Crazyswarm: A large nano-quadcopter swarm," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3299–3304.