Multi-Robot Trajectory Planning with Feasibility Guarantee and Deadlock Resolution: An Obstacle-Dense Environment

Yuda Chen, Chenghan Wang, Meng Guo, *Member, IEEE* and Zhongkui Li, *Senior Member, IEEE*

Abstract—This article presents a multi-robot trajectory planning method which not only guarantees optimization feasibility and but also resolves deadlocks in obstacle-dense environments. The method is proposed via formulating a recursive optimization problem, where a novel safe corridor is generated online to ensure obstacle avoidance in trajectory planning. A dynamic-priority mechanism is combined with the right-hand rule to handle potential deadlocks that are much harder to resolve due to static obstacles. Comparisons with other state-of-the-art results are conducted to validate the improved safety and success rate. Additional hardware experiments are carried out with up to eight nano-quadrotors in various cluttered scenarios.

Index Terms—Trajectory generation, motion planning, multi-robot system, collision avoidance, deadlock resolution.

Fig. 1. Eight nano-quadrotors fly through a cubic framework.

I. INTRODUCTION

COLLISION-free trajectory planning plays an essential role for a swarm of robots navigating in a shared environment [1]. The robots need to avoid collision with both other robots and obstacles while moving to their targets. Various techniques are proposed, among which optimization-based methods [2], [3] have gained significant attentions due to their flexibility of adding constraints, such as the convex constraints in [2], [4] and the non-convex constraints in [3]. Nevertheless, the constrained optimization may suffer from the infeasibility, leading to failed planning. This problem becomes specially severe in a crowed environment with dense obstacles. Moreover, without a central coordinator, it often happens that the robots block each other indefinitely and no further progresses can be made, which is also known as deadlocks [5].

To address the issue of infeasibility, the work in [6] converts hard constraints into soft ones by adding them to the objective function. But the safety properties associated with the constraints such as collision avoidance cannot be ensured anymore. Our previous work in [7] can provably ensure the feasibility of underlying optimization, which however cannot deal with the obstacle collision. To handle obstacles during
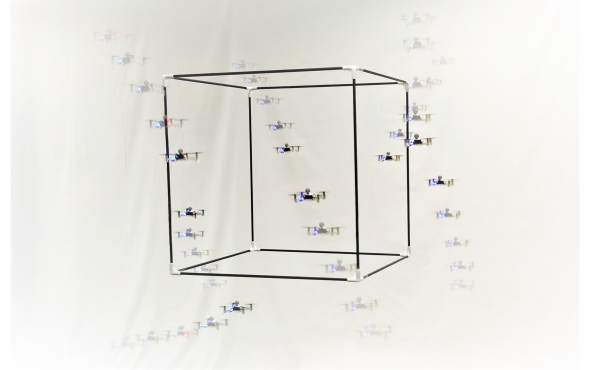
motion, one of the most widely-adopted method is safe corridor, e.g., [8] generates a high-quality corridor via semi-definite programming, and [9] combines the search-based path planning and geometry-based corridor construction. Unfortunately, the hard constraints introduced by the safe corridor may also lead to infeasible optimizations, as discussed in [10], [11]. In addition, the authors in [12], [13] propose a cubic corridor in a grid map. Although this method is computationally efficient to generate safe corridor and can guarantee feasibility, it is restricted by the cubic shape that may have a lower space efficiency. Furthermore, the method in [14] generates a safe corridor via supported vector machines, which has a higher utility rate of freespace, but is centralized and computed offline. Another way to construct a safe corridor is the voxel expansion proposed in [10], [15] on a grid map, which however is also computed offline.

As mentioned earlier, another prominent problem in distributed trajectory planning is deadlock when robots block each other indefinitely during motion, as also addressed in [13], [16]. Various heuristic approaches are proposed to resolve deadlocks, by introducing e.g., the right-hand rule [10], [17], priority planning [13], and detour points [18]. The common drawback of these approaches is that the safety of robots or the effectiveness of deadlock resolution is not ensured and usually can lead to livelock where deadlocks re-appear after resolution. Our previous work [7] proposes an adaptive right-hand rule for deadlock resolution in obstacle-free space. Unfortunately, it cannot be applied to obstacle-dense environments because static obstacles are non-cooperative and cannot be pushed away by the right-hand force.

To address these two issues formally in obstacle-dense environments, we first propose a novel method to construct safe corridors online which provide feasible space for trajectory generation. It adopts the path planning method ABIT* [19] to determine a reference path, based on which the separating hyperplanes are computed via quadratic programming between the imminent obstacles and the planned trajectory. Regarding deadlocks in obstacle-dense environments, a dynamic-priority mechanism is proposed along with the adaptive right-hand rule. Invalidation of the right-hand rule would induce a higher priority thus more egoistic to continue its motion.

The main contributions are summarized as follows.

- The proposed online generation of safe corridor as a sequence of polytopes has a much higher utilization of the workspace than the cubic safe corridor proposed in [12], [13]. It is shown to reduce computational cost and increase safety margin, compared with [8].
- The proposed dynamic-priority mechanism resolves potential deadlocks whenever the adaptive right-hand rule [7] is invalidated due to static obstacles.
- Comparisons with several state-of-the-art methods [3], [6], [13] are made in non-trivial cluttered scenarios. It is validated that the proposed method has a better performance in terms of guaranteeing collision avoidance as well as handling deadlocks.
- Extensive hardware experiments are conducted to validate the real-time performance, including eight nano-quadrotors navigating in obstacle-dense 3D scenarios, as shown in Fig. 1.

II. PROBLEM STATEMENT AND PRELIMINARIES

Consider a team of N robots that navigate to their respective destinations within a common 2D or 3D workspace that are cluttered with static obstacles. During the navigation, a robot cannot collide with either other robots or obstacles. Each robot can determine its own control input and exchange information with other robots via communication.

A. Robot Dynamics

Let $h > 0$ denote the sampling time. The dynamic model of the robot $i \in \mathcal{N} = \{1, \cdots, N\}$ is given by

$$x_k^i(t) = \mathbf{A} x_{k-1}^i(t) + \mathbf{B} u_{k-1}^i(t), \tag{1}$$

where $k \in \mathcal{K} \triangleq \{1, \cdots, K\}$ is the step within the planning horizon $K > 0$; $p_k^i(t)$, $v_k^i(t)$, $u_k^i(t)$ are the *planned* position, velocity, and control input at time $t + kh$, respectively; $x_k^i(t) = [p_k^i(t), v_k^i(t)]$ is the planned state at time $t + kh$ with $x_0^i(t) = x^i(t)$; the double-integrator dynamics are given by $\mathbf{A} = \begin{bmatrix} \mathbf{I}_d & h\mathbf{I}_d \\ \mathbf{0}_d & \mathbf{I}_d \end{bmatrix}$, $\mathbf{B} = \begin{bmatrix} \frac{h^2}{2}\mathbf{I}_d \\ h\mathbf{I}_d \end{bmatrix}$ with the dimension $d = 2, 3$. The planned trajectory of robot i at the time $t > 0$ is defined as $\mathcal{P}^i(t) = \{p_1^i(t), p_2^i(t), \cdots, p_K^i(t)\}$. Additionally, the velocity and input constraints are given as

$$\begin{aligned} \|\Theta_a u_{k-1}^i(t)\|_2 &\leq a_{\max}, \ k \in \mathcal{K}, \\ \|\Theta_v v_k^i(t)\|_2 &\leq v_{\max}, \ k \in \mathcal{K}, \end{aligned} \tag{2}$$

where Θ_v, Θ_a are given positive-definite matrices, and $v_{\max}$, $a_{\max}$ denote the maximum velocity and acceleration, respectively.

Once the planned trajectory is computed, a lower feedback controller is followed to drive robot i within the time interval $[t, t + h]$, such that $x^i(t + h) = x_1^i(t)$ holds when robot i re-plans at time $t + h$. Furthermore, at each time step t, robot i sends its predetermined trajectory $\overline{\mathcal{P}}^i(t)$ to other robot $j \neq i$, which is constructed directly from the planned trajectory $\mathcal{P}^i(t-h)$ derived at $t-h$. More specifically, $\overline{\mathcal{P}}^i(t) = \{\overline{p}_1^i(t), \overline{p}_2^i(t), \cdots, \overline{p}_K^i(t)\}$, where $\overline{p}_k^i(t) = p_{k+1}^i(t - h)$, $\forall k \in \tilde{\mathcal{K}} \triangleq \{1, 2, \cdots, K-1\}$ and $\overline{p}_K^i(t) \triangleq p_K^i(t-h)$. For the sake of simplicity, the time index t is omitted in the sequel, whenever ambiguity is not caused.

B. Collision Avoidance

1) Inter-Robot Collision Avoidance: The safety area around each robot $i \in \mathcal{N}$ is represented by a ball as $\mathcal{R}^i = \{x + p^i \mid \|x\|_2 \leq r_a\}$, where $r_a > 0$ is a given safety radius. To avoid inter-robot collisions, any pair of robots should satisfy $\|p^i - p^j\|_2 \geq 2r_a$. This constraint can be converted equivalently to linear constraints via the modified buffered Voronoi cells with warning band (MBVC-WB) proposed in our previous work [7] as follows:

$$a_k^{ij\,\mathrm{T}} p_k^i \geq b_k^{ij}, \ \forall j \neq i, \forall k \in \tilde{\mathcal{K}}, \tag{3a}$$

$$a_K^{ij\,\mathrm{T}} p_K^i \geq b_K^{ij} + w^{ij}, \ \forall j \neq i. \tag{3b}$$

where $a_k^{ij} = \frac{\overline{p}_k^i - \overline{p}_k^j}{\|\overline{p}_k^i - \overline{p}_k^j\|_2}$, $b_k^{ij} = a_k^{ij\,\mathrm{T}} \frac{\overline{p}_k^i + \overline{p}_k^j}{2} + \frac{r_{\min}}{2}$, $r_{\min} = \sqrt{4r_a{}^2 + h^2 v_{\max}^2}$ denotes the extended minimum distance, and $w^{ij} \in [0, \epsilon]$ is an additional variable as the warning band with the maximum width $\epsilon > 0$.

2) Obstacle Avoidance: Let $\mathcal{O} \subset \mathbb{R}^d$ denote the set of obstacles' occupied space. Similar to [11], the obstacles are assumed to be convex-shaped. Thus, the constraint of obstacle avoidance requires that the safety area of each robot does not intersect with any obstacle, i.e., $\mathcal{R}^i \cap \mathcal{O} = \emptyset$, $\forall i \in \mathcal{N}$. Namely, a planned trajectory $\mathcal{P}$ is collision-free, if

$$\mathbf{Conv}(\{p_k^i, p_{k+1}^i\}) \cap \tilde{\mathcal{O}} = \emptyset, \ \forall k \in \tilde{\mathcal{K}}, \tag{4}$$

holds, where $\tilde{\mathcal{O}}$ is the set of occupied space after inflating obstacles by r_a and $\mathbf{Conv}(P) \triangleq \{\sum_{i=1}^n \theta_i p_i \mid p_i \in P, \ \theta_i \geq 0, \ \sum_i \theta_i = 1, \ i = 1, 2, \cdots, n\}$ is the convex hull formed by points in the set P.

C. Problem Statement

Assuming that all robots are collision-free initially at time t_0, our goal is to design a distributed trajectory planning algorithm that drives each robot to its target p_{target}^i, $\forall i \in \mathcal{N}$, while respecting the constraints specified in (1)-(4).

III. TRAJECTORY PLANNING METHOD

The overall trajectory planning algorithm will be described in this section, which consists of the construction of safe corridor, the deadlock resolution scheme, the trajectory optimization algorithm and the proof of feasibility guarantee.

A. Safe Corridor Construction

The obstacle avoidance is realized by restricting the planned trajectory to be inside a safe corridor. The corridor is constituted by a sequence of convex polytopes of which the boundaries separate the planned positions p_k^i, $i \in \mathcal{N}$, $k \in \mathcal{K}$ and inflated obstacles. Based on the safe corridor, the obstacle avoidance constraint can be written as

$$a_k^{i,o\mathrm{T}} p_k^i \geq b_k^{i,o}, k \in \mathcal{K}, \tag{5}$$

where $a_k^{i,o}$ and $b_k^{i,o}$ determines the boundary of the corridor for robot i at horizon $k \in \mathcal{K}$ as $\mathcal{B}_k^i = \{p \mid a_k^{i,o\mathrm{T}} p = b_k^{i,o}\}$. As shown in Fig. 2, the safe corridor is constructed in three main steps: path planning, segmentation of EPT and computation of the separating hyperplanes.

1) Path Planning: To begin with, a reference path $\Gamma^i = \{\overline{p}_K^i, \cdots, p_{\mathrm{target}}^i\}$ is generated for each robot i, which connects the terminal horizon position of $\overline{p}_K^i$ and its target p_{target}^i as shown in Fig 2(a). In this work, the algorithm of Advanced Batch Informed Trees (ABIT*) proposed in [19] is adopted to plan this reference path. Once the path Γ^i is obtained, a tractive point denoted by p_{tractive}^i is found as $p_{\mathrm{tractive}}^i = \Gamma_{m^\star}^i$, where $m^\star$ is the maximum index m such that

$$\mathbf{Conv}(\{\Gamma_m^i, \overline{p}_K^i\}) \cap \tilde{\mathcal{O}} = \emptyset, \tag{6}$$

where Γ_m^i is the m-th point of Γ^i. An illustration is shown in Fig. 2(b). Then, the extended predetermined trajectory (EPT) is defined as $\tilde{\mathcal{P}}^i = \{\overline{p}_1^i, \cdots, \overline{p}_{K+1}^i\}$ where $\overline{p}_{K+1}^i = p_{\mathrm{tractive}}^i$. Notably, it is unnecessary to update the path at every step as tractive point Γ^i can be determined via (6) given the previous path. In other words, the robot only needs to replay its path if its target is not changed.

2) Segmentation of EPT: In order to reduce the computational complexity, the derived EPT $\tilde{\mathcal{P}}^i$ is divided into N_s segments, denoted by $\mathcal{S}_n^i$, $n = 1, 2, \cdots, N_s$, as illustrated in Fig 2(b). These segments are computed in three main steps: First, $\mathcal{S}_1^i$ is initialized as $\mathcal{S}_1^i = \{\overline{p}_{K+1}^i\}$. Then, the point $\overline{p}_k^i$ is added into the current segment $\mathcal{S}_1^i$ consecutively from $k = K$ to $k = 1$, until the convex hull formed by $\mathcal{S}_1^i$ is not collision-free anymore, i.e., $\mathbf{Conv}(\mathcal{S}_1^i) \cap \tilde{\mathcal{O}} \neq \emptyset$. Afterwards, the next segment $\mathcal{S}_2^i$ is initialized as $\mathcal{S}_2^i = \{\overline{p}_k^i\}$ where $\overline{p}_k^i$ is the last point of $\mathcal{S}_1^i$. The above process is repeated until the starting point $\overline{p}_1^i$ is added into the last segment $\mathcal{S}_{N_s}^i$.

3) Computing Separating Hyperplane: After the segmentation of EPT is done, the separating hyperplanes between the n-th segments $\mathcal{S}_n^i$ and obstacles are constructed. Since the convex hull of $\mathcal{S}_n^i$ is obstacle-free and also the obstacles are convex-shaped, a separating hyperplane exists according to separating hyperplane theorem [20]. Then, as shown in Fig. 2(c), an optimization-based method can be provided as follows:

$$\max_{a,b,\gamma} \gamma,$$
$$\mathrm{s.t.,} \ a^\mathrm{T} p^\mathrm{S} \geq \gamma + b,$$
$$a^\mathrm{T} p^O \leq b, \tag{7}$$
$$\|a\|_2 = 1,$$
$$\gamma \geq 0,$$

(a) Path planning.

(b) Segmentation of EPT.

(c) Computation of separating hyperplane.
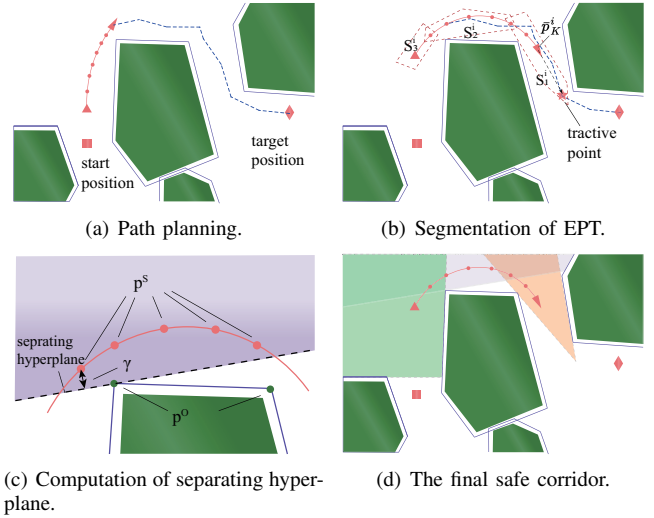
(d) The final safe corridor.

Fig. 2. Illustration of the procedure to construct the safe corridor. Solid lines around the green area are inflated obstacles.

where a and b determine the separating hyperplane; $p^\mathrm{S} \in \mathcal{S}_n^i$ and p^O denote the points of an obstacle; γ is the margin variable. Such an optimization can be further transformed to the following quadratic programming (QP):

$$\min_{a',b'} \ \|a'\|_2^2$$
$$\mathrm{s.t.,} \ a'^\mathrm{T} p^\mathrm{S} \geq 1 + b', \tag{8}$$
$$a'^\mathrm{T} p^O \leq b',$$

where $a' = \frac{a}{\gamma}$ and $b' = \frac{b}{\gamma}$. By solving the QP in (8), the separating hyperplane can be obtained as $a = \frac{a'}{\|a'\|_2}$ and $b = \frac{b'}{\|a'\|_2}$. Moreover, $a_k^{i,o}$ and $b_k^{i,o}$ are chosen as a and b to formulate the constraints in (5).

In practice, the separating hyperplanes between different segments and obstacles are constructed in a sequential manner. Specifically, starting from the nearest obstacle to the farthest one, a separating hyperplane is built if this obstacle has a contact with the convex polytopes formed by the existing hyperplanes. Otherwise, such an obstacle is omitted since it has been separated by the existing hyperplanes.

Lemma 1. *The proposed safe corridor has the following three properties:*

1) *If the predetermined trajectory $\overline{\mathcal{P}}^i$ is obstacle-free, then a safe corridor can be generated.*
2) *The predetermined trajectory $\overline{\mathcal{P}}^i$ satisfies the constraints in (5), i.e., $a_k^{i,o\mathrm{T}} \overline{p}_k^i \geq b_k^{i,o}$, $k \in \mathcal{K}$.*
3) *If the constraints in (5) are satisfied, then the planned trajectory $\mathcal{P}^i$ is obstacle-free.*

Proof. First, since $\mathbf{Conv}(\{p_{\mathrm{tractive}}^i, \overline{p}_K^i\}) \cap \tilde{\mathcal{O}} = \emptyset$, a collision-free $\tilde{\mathcal{P}}^i$ can be found. Consider the most conservative segmentation that the segments are formed by every adjacent two points of $\tilde{\mathcal{P}}^i$. Since there always exists separating hyperplanes between the points of segment and any of obstacle, the convex polytope can be formed. Consequently, the safe corridor can be constructed as a sequence of polytopes. Second, since the

constraints in (5) are obtained from the optimization in (7) with p^S chosen from the predetermined trajectory $\overline{\mathcal{P}}^i$, it is clear that $\overline{\mathcal{P}}^i$ satisfies these constraints. Lastly, due to the given segmentation method, $\overline{p}_k^i$ and $\overline{p}_{k+1}^i$, $k \in \tilde{\mathcal{K}}$ belong to the same segment. Thus, if the constraints in (5) are enforced, the planned trajectory p_k^i and p_{k+1}^i are restricted to a common collision-free polytope. Consequently, the original constraints in (4) hold and the planned trajectory is collision-free. $\square$

B. Deadlock Resolution

After the constraints for obstacle avoidance is transformed into (5), the overall trajectory optimization at each time step is formulated as follows:

$$\min_{\{u_{k-1}^i, x_k^i, w^{ij}\}} C^i \tag{9a}$$
$$\text{s.t.} \ (1), (2), (3), (5), \ k \in \mathcal{K}, \ \forall j \neq i,$$
$$v_K^i = \mathbf{0}_d, \tag{9b}$$

where C^i is the objective function defined in the sequel; constraints in (1) and (2) stem from the complex dynamic model and constraints; constraints in (3) and (5) ensure the inter-robot and robot-obstacle collision avoidance; constraints in (9b) is imposed to ensure the feasibility of underlying optimization, as done similarly in [7], [3].

The objective function in (9a) is defined as $C^i = C_p^i + C_w^i$, where C_p^i is the balanced term between the distance to the tractive point and the consecutive velocities:

$$C_p^i = \frac{1}{2} Q_K \| p_K^i - p_{\text{tractive}}^i \|_2^2 + \frac{1}{2} \sum_{k=1}^{K-1} Q_k \| p_{k+1}^i - p_k^i \|_2^2,$$

where $Q_k > 0$, $\forall k \in \mathcal{K}$ are weight parameters. In addition, C_w^i is the penalty term designed for deadlock resolution as follows:

$$C_w^i = \sum_{j \neq i} \rho^{ij} (\epsilon - w^{ij})^2, \tag{10}$$

where $\rho^{ij} > 0$ is the designed parameter to adjust the repulsive force from robot j to robot i.

The process of updating ρ^{ij} online is summarized in Algorithm 1. To begin with, the deadlock detection mechanism is introduced in Line 1, where a boolean signal is introduced as $flag^i = True$. A deadlock is detected if the following two conditions hold: i) $p_K^i(t) \neq p_{\text{target}}^i$, $p_K^i(t) = p_K^i(t - h)$, and $p_K^i(t) = p_{K-1}^i(t)$; and ii) there exists another robot j, such that $w^{ij} < \epsilon$ holds. The reason is that when a potential deadlock happens, the planned terminal position $p_K^i(t)$ of robot i remains static and the second last planned position $p_{K-1}^i(t)$ approaches $p_K^i(t)$. Moreover, in obstacle-dense environments, deadlocks may be induced by static obstacles instead of the blocking of other robots. To resolve this situation, an additional condition is added to the detection process. Namely, if there exists robot j such that $w^{ij} < \epsilon$ holds, it indicates that robot i is blocked by at least one another robot.

Algorithm 1: DeadlockResolution()

Input : $\overline{\mathcal{P}}^j$, p_{target}^j, ψ^j, ζ^j

1 Obtain $flag^i$ via deadlock detection ;
2 Obtain η^i from (11) ;
3 **if** $\exists t > t_0, \ p_K^i(t) = p_{\text{target}}^i$ **then**
4     $\psi^i = \mathbf{1}$;
5 **else**
6   **if** $\eta^i \geq \eta_{\max}$ **then**
7   **if** $(\zeta^i = True)$ **and** $(\psi^j \neq \mathbf{3}, \forall j)$ **and** $(\forall j,$ $\zeta^j = True$ implies $\| p_K^i - p_{\text{target}}^i \|_2 < \| p_K^j - p_{\text{target}}^j \|_2)$ **then**
8          $\psi^i = \mathbf{3}$;
9          $\zeta^i = True$;
10 **if** $(\psi^i = \mathbf{3})$ **and** $(w^{ij} = \epsilon, \forall j)$ **then**
11     $\psi^i = \mathbf{2}$;
12 **return** ρ^{ij} as (12);

Afterwards, it is followed by the update of a key parameter $\eta^i(t)$ as in Line 2, which represents the magnitude of the deadlock. And it is updated as follows:

$$\eta^i(t) = \begin{cases} \eta^i(t - h) + \Delta\eta, & \text{if } flag^i = True; \\ \eta_{\max}, & \text{if } (flag^i = True \\ & \text{and } \eta^i(t - h) \geq \eta_{\max}); \\ 0, & \text{if } \forall j \neq i, w^{ij}(t - h) = \epsilon; \\ \eta^i(t - h), & \text{else}, \end{cases} \tag{11}$$

where $\Delta\eta > 0$ is a design parameter and $\eta_{\max}$ is the chosen upper bound of $\eta^i(t)$.

Based on these parameters, the dynamic priority of robot i denoted by ψ^i is determined. A higher priority leads to more egoistic behaviors when avoiding the others. Initially, all robots have the medium priority as $\psi^i = \mathbf{2}$. Furthermore, if a robot has reached its target, it obtains the lowest priority $\psi^i = \mathbf{1}$ as in Line 4. Moreover, if $\eta^i \geq \eta_{\max}$, i.e., the deadlock measurement has reached the highest magnitude, robot i is chosen as the candidate that obtains the Priority $\mathbf{3}$ and consequently $\zeta^i = True$. More importantly, any robot i actually obtains Priority $\mathbf{3}$ as in Line 8 if the following three conditions are all satisfied: i) it is the one of the candidates, i.e., $\zeta^i = True$; ii) the highest priority is still vacant; and iii) compared with other candidates, robot i has the shortest distance to the target. By this mechanism, the highest priority is assigned in a distributed way and only one robot is entitled the highest priority. Notably, the highest priority of one particular robot is not permanent as once it has escaped from the deadlock and has no contact with any others, its highest priority is deprived as in Line 11. In other words, the priorities are dynamically changing based on the status of the robots.

After the priorities are determined, the parameter ρ^{ij} in (10) is computed as follows:

$$\rho^{ij} = \begin{cases} \frac{\rho_0}{\gamma^{ij}} e^{(\eta^i \sin\theta^{ij})}, & \text{if } \psi^i = \psi^j; \\ \rho_{\min}, & \text{if } \psi^i > \psi^j; \\ \rho_{\max}, & \text{if } \psi^i < \psi^j, \end{cases} \tag{12}$$

where the θ^{ij} is the angle in $x-y$ plane between the projection of the line $\overline{p}_K^i$ to the point p_{tractive}^i and to the point $\overline{p}_K^j$; γ^{ij} is
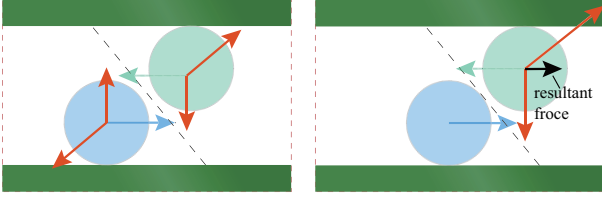
Fig. 3. **Left**: the existence of static obstacles can invalidate the right-hand rule. **Right**: after obtaining the highest priority, the blue robot ignores the repulsive effect from the green robot and continues its motion.

Algorithm 2: The Complete Algorithm

Input: $p^i(t_0)$, p^i_{target}, $\mathcal{O}$

1   $\overline{\mathcal{P}}^i(t_0) = \{p^i(t_0), \ldots, p^i(t_0)\}$;
2   **while** not all robots at target **do**
3   **for** $i \in \mathcal{N}$ concurrently **do**
4   obtain $Data^j$, $j \neq i$ via communication;
5   obtain $a_k^{i,o}$, $b_k^{i,o}$ via constructing safe corrdior;
6       $\rho^{ij} \leftarrow$ DeadlockResolution($Data^j$);
7   obtain $\mathcal{P}^i(t)$ from optimization (9);
8   send $\mathcal{P}^i(t)$ to lower-level controller;
9     $t \leftarrow t + h$;

---

used to adjust the penalty in a proper range, which is updated as $\gamma^{ij}(t) = (1-\beta)\gamma^{ij}(t-h) + \beta\,\epsilon\,w^{ij}(t-h)$, where $\beta \in [0.5, 1)$ and $\gamma^{ij}(t_0) = \epsilon^2$; ρ_0, $\rho_{\min}$ and $\rho_{\max}$ are coefficients which satisfy $\rho_{\max} \gg \frac{\rho_0}{\gamma^{ij}} \gg \rho_{\min} > 0$.

In general, since most agents have Priority 2, they adopt a right-hand rule as $\rho^{ij} = \frac{\rho_0}{\gamma^{ij}} e^{(\eta^i \sin \theta^{ij})}$ to coordinate with each other. More specifically, when the deadlock is detected as described above and $\eta^{ij} > 0$, the right-hand rule is activated. For the robot j on the left, i.e., $\theta^{ij} > 0$, it holds that $e^{(\eta^i \sin \theta^{ij})} > 1$ which indicates that the repulsive effect from robot j and robot i is increased, which keeps robot i away from robot j. On the contrary, if robot i approaches robot j on its right hand, the coordination strategy is different from the previous case. More specifically, when the deadlock magnitude η^i exceeds the designated upper bound $\eta_{\max}$, its priority is changed as described earlier. Thus, the robot with a higher priority has a much lower penalty coefficient as $\rho^{ij} = \rho_{\min}$, which means that it can approach and expel other robots that have lower priorities. This mechanism is critical in obstacle-dense environments since the right-hand rule may be invalidated when obstacles block the way along with other robots. An example of such situation is shown in Fig. 3. However, the robot with the highest priority can neglect the repulsive effect from other robots and continue moving in order to break this logjam.

It should be mentioned that similar design of ρ^{ij} is also adopted in [7], which however is restricted to the obstacle-free scenario. The main differences are that a dynamic priority mechanism is needed in this paper and the magnitude of deadlock, $\eta^i(t)$, now has an upper bound.

C. Overall Trajectory Optimization Algorithm

Given the optimization problem in (9) and the deadlock resolution scheme described in the previous section, the overall trajectory optimization is summarized in Algorithm 2.

To begin with, the predetermined trajectory is initialized in Line 1. After initialization, in the main loop, each robot runs the same planning algorithm in a parallel and distributed way as in Line 3. Therein, the packed data $Data^i = \{\overline{\mathcal{P}}^i, p^i_{\text{target}}, \psi^i, \zeta^i\}$ is exchanged among the robots, which are essential for formulating and solving the final optimization and enforcing the deadlock resolution scheme, as in Line 4. In particular, it is followed by the safe corridor construction as descried in Sec. III-A and Line 5. Thereafter, Algorithm 1 is followed to resolve potential deadlocks and further obtain the penalty parameter ρ^{ij} in Line 6. Thus, the final optimization

in (9) is formulated and solved in Line 7. The resulting trajectory is executed via sending it to the lower-level feedback controller (Line 8). This procedure is repeated until all robots have reached their target positions.

The overall computational complexity of the proposed method is analyzed as follows. The computational cost consists of four main parts: i) finding the tractive points, ii) segmentation of the EPT, iii) computing the separating hyperplanes and iv) solving the final optimization. To obtain a tractive point, the collision detection between line-polytopes is carried out less than $n_{\text{path}} \cdot n_{\text{proximal}}$ times, where n_{path} is the number of points in Γ^i and n_{proximal} is the number of proximal obstacles around the robot. In general, the collision detection between line-polytopes is a special case of the collision detection between polytopes, which can be done with complexity $\mathcal{O}(n_1 + n_2)$ where n_1 and n_2 are the numbers of vertices in these two polytopes [21]. The segmentation of the EPT is finished after checking the collision between two polytopes for $2Kn_{\text{proximal}}$ times. To compute the separating hyperplanes, the worst situation induces K segments and consequently hyperplanes with n_{proximal} obstacles are obtained after solving Kn_{proximal} QPs. Each of these QPs has $d + 1$ variables and $d + n + 1$ constraints where n is the number of vertices belonging to the underlying obstacles. In the final optimization, a quadratically constrained quadratic programming (QCQP) with $Kd + N - 1$ variables and $(N + n_{\text{priximal}} + 1)K + 2N - 1$ constraints is formulated. Owing to mature optimization tools, e.g., [22], the underlying optimization can be resolved within tens of milliseconds.

D. Feasibility Guarantee

Different from most existing optimization-based methods, e.g., [3], [10], the proposed planning algorithm guarantees the recursive feasibility of the optimization problem, as proven in the following theorem.

Theorem 1. *If all robots are collision-free with each other and any obstacle initially, they remain so under Algorithm 2.*

Proof. At the beginning time t_0, it is clear that the predetermined trajectory $\overline{\mathcal{P}}^i(t_0)$ is a feasible solution of optimization (9) for all robots.

As stated in Lemma 1, if trajectory planning at $t - h$ is feasible, the planned trajectory at $t - h$ is obstacle-free,
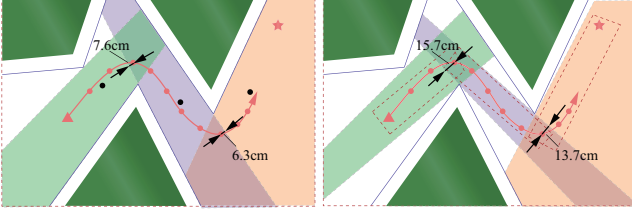
Fig. 4. **Left**: the safe corridor generated by IRIS [8]. **Right**: ours safe corridor. In comparison, points of predetermined trajectory generated by our method are further away from the obstacles boundary, yielding a larger safety margin.
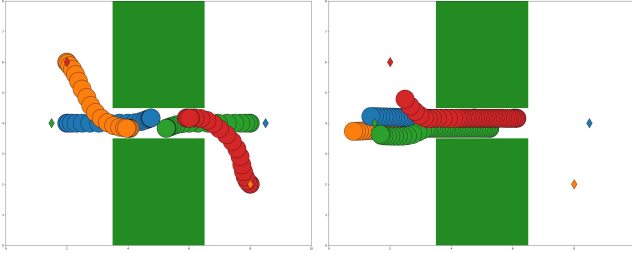

Fig. 5. **Left**: four robots remain being blocked by each other after only the right-hand rule is activated. **Right**: the green robot navigates through the corridor safely by expelling the blue and orange robots, under the proposed dynamic-priority scheme.

TABLE I
COMPARISON WITH STATE-OF-ART METHODS. (SAFETY: NO COLLISION OCCURS. T_t[s]: TRANSITION TIME. L_t[m]: THE LENGTH OF TRANSITION. T_c[ms]: MEAN COMPUTATION TIME PER REPLANNING.)

	method	Safety	T_t	L_t	T_c
"Forest"	Ego-swarm [6]	No	9.2	105.8	9.6
	MADER [3]	No	22.3	111.1	104.0
	LSC [13]	Yes	22.3	114.3	53.2
	Ours	Yes	8.5	102.2	85.6
"H"	Ego-swarm [6]	No	7.5	66.6	10.2
	MADER [3]	No	14.2	71.5	116.7
	LSC [13]	Yes	-	-	62.3
	Ours	Yes	7.5	65.8	75.2

multiprocessing. CVXOPT [22] is used for solving the trajectory optimization, and OMPL [23] for ABIT* path planning. The simulation and experiment videos can be found at https://youtu.be/Er7v9zQx784.

A. Numerical Simulations and Comparison

In the following simulations, the main parameters of robots are chosen as follows: $r_a = 0.3$m; $a_{\max} = 2$m/s^2; $v_{\max} = 3$m/s; $h = 0.15$s.

To begin with, the safe corridor generation is compared with IRIS [8]. For IRIS, the centers of segmented points (shown as black points in Fig. 4) are initialized to generate the corridor. On average, IRIS takes 21ms to compute a safe corridor in contrast to 4ms via our method. As shown in Fig. 4, IRIS generates a larger interaction area than ours. However, the trajectories generated by our method have a larger distance to the boundary of the safe corridors, yielding a larger safety area around critical regions. This can be beneficial for ensuring the feasibility of underlying trajectory optimization.

To further validate our method, a rather extreme scenario is considered where four robots encounter in a narrow passage which only accommodates one robot. As shown in Fig. 5, even with the activated right-hand rule in [7], four robots are stuck in this passage as they all have the same priority. However, since under the proposed algorithm the green robot obtains the highest priority after its parameter η^i reaches $\eta_{\max}$, it can expel the blue and orange robots to continue its motion through the passage and reach the target. Afterwards, the red robot follows the blue robot through the passage.

Furthermore, comparisons with Ego-swarm [6], MADER [3] and LSC [13] are made within the "Forest" and "H" environments. Sampled trajectories of these scenarios are illustrated in Figs. 6 and 7, respectively. The results are summarized in Table I. Ego-swarm has an impressive computation time in addition to a relatively smooth and fast trajectories. Unfortunately, it cannot guarantee the collision avoidance, as it adopts an unconstrained optimization in planning. For MADER, the optimization under the strict constraints guarantees the avoidance between robots. However, regarding obstacle avoidance, MADER is not well performed in an obstacle-dense environment as several collisions appear. LSC has the feasibility guarantee which ensures the safety of robots, but its trajectory is not soft since the cubic safe
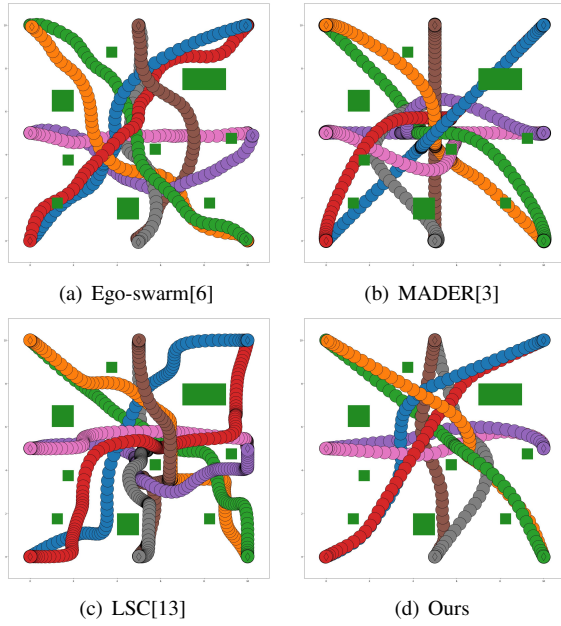
based on which a safe corridor can be derived. Thus, the final optimization (9) can be formulated. Given the feasible solution at the previous time step, i.e., $u_{k-1}^i(t-h)$ and $x_k^i(t-h)$ for $k \in \mathcal{K}$, a feasible solution $x_k^i(t) = x_{k+1}^i(t-h)$, $u_k^i(t) = u_{k+1}^i(t-h)$ can be derived where we enforce $x_K^i(t) = x_K^i(t-h)$ and $u_{K-1}^i(t) = \mathbf{0}_d$. More specifically, as the result of optimization at time step $t - h$, $x_{k+1}^i(t-h)$ and $u_k^i(t-h)$ with $k \in \tilde{\mathcal{K}}$, satisfy the constraints in (1) and (2) naturally. In addition, since $x_K^i(t) = x_K^i(t-h)$, $v_{K-1}^i(t) = v_K^i(t-h) = \mathbf{0}_d$ and $u_{K-1}^i(t) = \mathbf{0}_d$ hold, $x_K^i(t)$ and $u_{K-1}^i(t)$ also satisfy these constraints. Meanwhile, as $x_K^i(t) = x_K^i(t-h) = x_{K-1}^i(t)$ holds, it is evident that the constraint (9b) holds as well. Then, as proven similarly in Theorem 3 of [7], the constraints in (3) are satisfied as well. Lastly, as stated in the first and third properties of Lemma 1, the feasible solution at time $t - h$ is obstacle-free and the associated safe corridor can be determined. Thus, the constraint in (5) is feasible and the optimization in (9) is feasible in a recursive way.

Since the initial optimizations as well as these successive ones are feasible, the enforced constraints are satisfied. Due to the third property in Lemma 1 and Theorem 1 in [7], the collision avoidance among robots and between robots and obstacles are ensured. This completes the proof. □

IV. NUMERICAL SIMULATIONS AND EXPERIMENTS

This section validates the performance of the proposed algorithm via numerical simulations and hardware experiments. The algorithm is implemented in Python3 and publicly available at https://github.com/PKU-MACDLab/IMPC-OB. All tests are performed on a computer with Intel Core i9 @3.2GHz and the team of robots is simulated using

(a) Ego-swarm[6]   (b) MADER[3]

(c) LSC[13]   (d) Ours

Fig. 6.   Comparison of four planners in the "Forest" scenario.

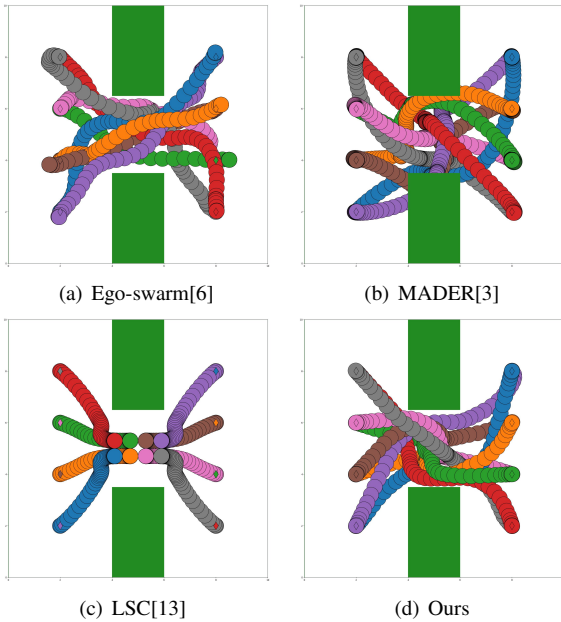(a) Ego-swarm[6]   (b) MADER[3]

(c) LSC[13]   (d) Ours

Fig. 7.   Comparison of four planners in the "H" scenario.

corridor. Moreover, deadlocks occurs in "H"-transition as the heuristic deadlock resolution method in LSC is ineffective in this scenario. In contrast, the adaptive right-hand rule is leveraged by the proposed method, resulting in the right-hand rotations in the bottleneck of the "H". Moreover, it can be seen that the proposed planner generates a much faster and smoother trajectories as the result of high-efficiency safe corridor.

Last but not the least, the time cost distribution of the four main steps: finding the tractive point, segmentation of the EPT, computing the separating planes and solving the final optimization (9) is shown in Fig. 8. The result is collected from the "Forest" scenario shown in Fig. 6. It can be seen that
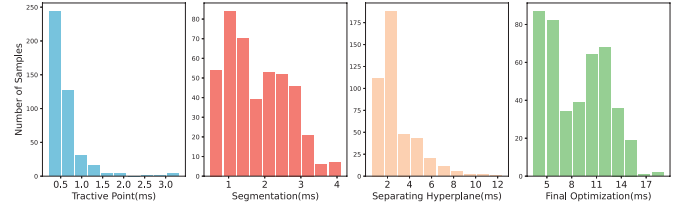

Fig. 8.   The distribution of computation cost in four main steps of the replanning.
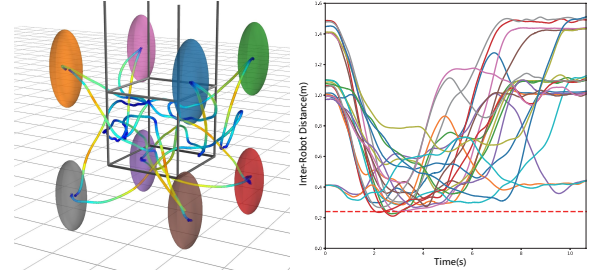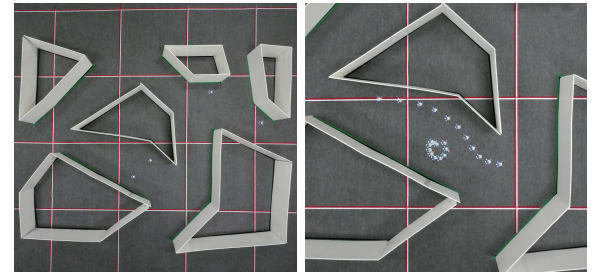

(a) **Left**: final trajectories of the 8-robot swarm where the ellipsoids indicate the safety area around robots. **Right**: the inter-robot distances over all time.

(b) **Left**: four nano-quadrotors are deployed in a polygon-shape environment. **Right**: Zoomed-in snapshot when one quadrotor navigates through the narrow passage, while another quadrotor makes room via the proposed deadlock resolution scheme.
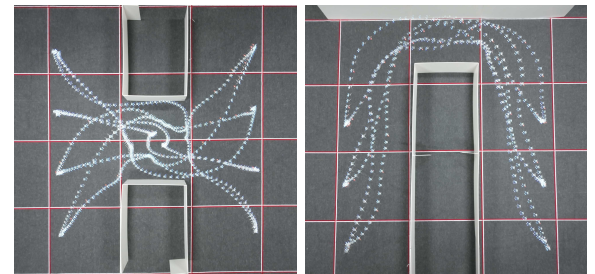
(c) **Left**: experiment results of 8 robots navigating in the "H" scenario. **Right**: experiment results of 6 robots switching positions in the "n" scenario.

Fig. 9.   Different test environments in the hardware experiments.

finding the tractive point takes a negligible time and computing separating planes consumes more time than the segmentation of EPT. In contrast, most of the planning time is spent on solving the final optimization for the desired trajectory.

B. Hardware Experiments

Hardware experiments are executed on the platform of crazyswarm [24], where multiple nano-quadrotors are tracked by the motion capture system OptiTrack. The computation of all robots takes place on a central computer with multiple

processes at 5Hz frequency to comply with the sampling time step $h = 0.2$s. For each crazyfile, a feedback controller [25] is adopted to track the planned trajectory.

The first experiment is shown in Fig. 9(a), where 8 crazyfiles fly through a 0.6m cubic framework. Considering the air turbulence, a crazyfile in the inter-robot avoidance is represented as an ellipsoid with diameter 0.24m in the $x - y$ plane and 0.6m in the z axis. Owing to the deformation of robots, the inter-robot constraints are accordingly adjusted by modifying a_k^{ij} and b_k^{ij} as

$$a_k^{ij} = E \frac{E(\overline{p}_k^i - \overline{p}_k^j)}{\|E(\overline{p}_k^i - \overline{p}_k^j)\|_2}, \quad b_k^{ij} = a_k^{ij\,\mathrm{T}} \frac{E(\overline{p}_k^i + \overline{p}_k^j)}{2} + \frac{r_{\min}}{2},$$

where $E = \mathrm{diag}(1.0, 1.0, 0.4)$, $r_{\min} = \sqrt{4r_a{}^2 + v_{\max}^2}$ and $r_a = 0.12$m. From the result given in Fig 9(a), it is apparent that the crazyfiles can achieve this transition.

In addition, four crazyfiles moving in a polygonal environment as shown in Fig 9(b). Given initial positions, the targets are randomly chosen. After arriving at the targets, the new one are published immediately and this process is repeated 5 times. In this scenario, the feasible space is the irregular-shaped passage at the interval of polygon-shaped obstacles where the width of these passages ranges from 0.4m to 0.7m. Via the proposed deadlock resolution scheme, particularly the dynamic-priority mechanism, a robot can smoothly squeeze out a way between the obstacle and another lower-priority robot as shown in Fig. 9(b).

At last, other two experiments as shown in Fig. 9(c) are carried out, where the scenarios are "H" and "n" environments respectively. Therein, both the safety and deadlock resolution are achieved in narrow passages. In "H"-transition, the potential deadlock is resolved via right-hand rule and the transition is completed within 9s. Regarding the "n"-transition, the intersection of two groups of quadrotors at the top passage is the main challenge for this mission. It can be seen that the proposed method resolves it as the quadrotors fly through the passage without sacrificing any speed. As a result, the task is finished within 8s.

V. Conclusion

This work has proposed a novel multi-robot trajectory planning method for obstacle-dense environments, where the collision avoidance is guaranteed and potential deadlocks are resolved online. Comprehensive simulations and hardware experiments have been conducted to validate the proposed method. However, a theoretical guarantee on the resolution of deadlocks or even livelocks remains open, which is part of our ongoing work.

References

[1] S.-J. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[2] L. C. E. and A. P. Schoellig, "Trajectory generation for multiagent point-to-point transitions via distributed model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 375–382, 2019.

[3] J. Tordesillas and J. P. How, "Mader: Trajectory planner in multiagent and dynamic environments," *IEEE Transactions on Robotics*, pp. 1–14, 2021.

[4] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 1917–1922.

[5] J. Alonso-Mora, P. Beardsley, and R. Siegwart, "Cooperative collision avoidance for nonholonomic robots," *IEEE Transactions on Robotics*, vol. 34, no. 2, pp. 404–420, 2018.

[6] 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)*, 2021, pp. 4101–4107.

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

[8] R. Deits and R. Tedrake, *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*. Springer International Publishing, 2015, pp. 109–124.

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

[10] C. Toumieh and A. Lambert, "Decentralized multi-agent planning using model predictive control and time-aware safe corridors," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 110–11 117, 2022.

[11] B. Şenbaşlar, W. Hönig, and N. Ayanian, "Rlss: Real-time multi-robot trajectory replanning using linear spatial separations," *arXiv preprint arXiv:2103.07588*, 2021.

[12] 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.

[13] 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.

[14] W. Hönig, J. A. Preiss, T. K. 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.

[15] F. Gao, L. Wang, B. Zhou, X. Zhou, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1526–1545, 2020.

[16] 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.

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

[18] 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)*, 2021, pp. 429–436.

[19] M. P. Strub and J. D. Gammell, "Advanced BIT (ABIT): Sampling-based planning with advanced graph-search techniques," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 130–136.

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

[21] E. Gilbert, D. Johnson, and S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[22] A. Martin, D. Joachim, and V. Lieven, "Cvxopt," Website, http://cvxopt.org/.

[23] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.

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

[25] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 2520–2525.