RINGO: Real-time Locomotion Planning via a Guiding Trajectory for Aerial Manipulators

Zhaopeng Zhang, Shizhen Wu, Chenfeng Guo, Yongchun Fang, Jianda Han, Xiao Liang

***Abstract*—Motion planning for aerial manipulators in constrained environments has typically been limited to rely on pre-built maps or simplified to that of multi-rotors, leading to poor adaptability or overly conservative trajectories. This paper presents RINGO: Real-time Locomotion Planning via a Guiding Trajectory, a planning method that tackles the locomotion problem for aerial manipulators in constrained environments without relying on pre-built maps. A simplified obstacle-avoidance model is established, and its feasibility is theoretically proven. An initial end-effector trajectory is generated via a quadratic Bézier curve and then refined into a B-spline. A gradient-based optimization problem is formulated that integrates smoothness, workspace-feasibility, yaw-rate, and obstacle-avoidance costs. Leveraging the signed distance function (SDF), soft-max technique and the convex hull property of B-spline curves, the trajectory remains theoretically feasible. This paper presents the first work that enables real-time motion planning of aerial manipulators in constrained environments without relying on pre-built maps. The simulation and experimental results show the effectiveness of the proposed method.**

***Note to Practitioners*—This paper addresses the problem of real-time motion planning for aerial manipulators in constrained environments without pre-built maps. We propose RINGO, a motion planning method that integrates obstacle-avoidance modeling, Bézier and B-spline trajectory generation, and gradient-based optimization. By leveraging signed distance functions, the soft-max technique, and the convex hull property of B-spline curves, the method ensures theoretical feasibility while maintaining computational efficiency. From a practical perspective, RINGO can be directly deployed on aerial manipulator platforms to achieve agile and safe locomotion without relying on pre-built maps. This capability significantly improves applicability in real-world tasks such as infrastructure inspection, search and rescue, and payload delivery in constrained environments. Future research will aim to adapt the framework to dynamic environments, where rapidly changing obstacles and uncertain conditions require real-time responsiveness and resilience.**

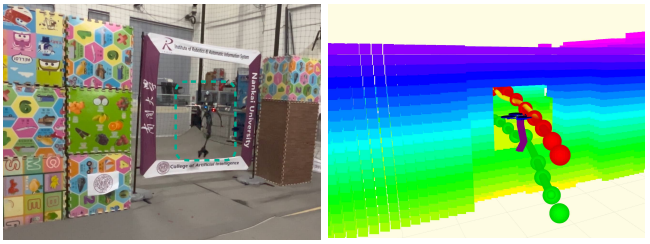
***Index Terms*—Aerial manipulator, motion planning, trajectory optimization.**

I. Introduction

MULTI-ROTOR aerial vehicles, commonly known as drones, have been widely studied in recent years for diverse applications, including autonomous landing [1], [2] and payload delivery [3]–[5]. Meanwhile, robotic manipulators have attracted extensive attention over the past decades owing to their wide applicability in industrial and service domains.

Corresponding author: Xiao Liang

The authors are with the Institute of Robotics and Automatic Information System, College of Artificial Intelligence, Nankai University, Tianjin 300350, China. e-mail: {zhangzp, szwu, guocf}@mail.nankai.edu.cn; {fangyc, hanjianda, liangx}@nankai.edu.cn;



(a) snapshot of the real-world experiment (b) visualization in RViz

Fig. 1: The aerial manipulator successfully passes through a ring-shaped obstacle.

Representative applications range from grasping [6] to complex tasks [7].

Aerial manipulators, typically consisting of a multi-rotor and a robotic arm, integrate the strengths of both components: the robotic arm provides the multi-rotor with manipulation capabilities, while the multi-rotor overcomes the fixed workspace limitation of the robotic arm, enhancing the aerial manipulator's flexibility for large-scale movements. In recent years, aerial manipulators have attracted considerable attention for robust control [8]–[10], contact-based inspection [11] and various practical applications [12], [13].

Motion planning is a fundamental problem in robotics, aiming to generate collision-free and dynamically feasible trajectories for robotic systems. Although some studies have addressed the motion planning problem for aerial manipulators, there are still two key issues that need to be addressed. Firstly, several motion planning methods for aerial manipulators are typically conducted relying on pre-built maps [14]–[20]. However, a critical limitation is that the reliance on pre-built maps significantly restricts the adaptability of aerial manipulators to new environments, while motion planning in constrained spaces further presents significant challenges in terms of computational complexity and real-time performance. Secondly, the motion planning for an aerial manipulator in constrained environments is, in some cases, simplified to that of a multi-rotor [16]. However, this approach, which encloses the entire system within a large bounding ball, tends to yield overly conservative trajectories. While minimizing the ball's radius by retracting the robotic arm can mitigate this issue, it introduces a periodic planning approach, potentially increasing the time required for the arm to reach its goal state [21].

To address the above issues, we propose a novel motion planning method, called **R**eal-t**I**me Locomotion Plan**N**ing via

a **G**uiding trajectO**r**y (**RINGO**) for aerial manipulators in constrained environments without relying on pre-built maps. Our proposed method employs a leader-follower-inspired motion planning framework for aerial manipulators. Based on a previously planned and parameterized B-spline trajectory of the multi-rotor, we then plan the trajectory for the robotic arm. We first establish an obstacle-avoidance model with a simplified treatment of the tilt motion, whose feasibility is theoretically proven. Then, the initial trajectory for the end-effector is generated by a quadratic Bézier curve from the start position to the goal position and refined into a B-spline curve. Finally, a gradient-based optimization problem is formulated by jointly incorporating the workspace-feasibility, smoothness, yaw-rate, and obstacle-avoidance costs, ensuring that the refined end-effector trajectory remains smooth and geometrically valid. Additionally, workspace feasibility is established by constructing a cost function using the soft-max technique and the approximated signed distance functions (SDFs), combined with the convex hull property of the B-spline curve that guarantees the robotic arm motion remains geometrically feasible.

Compared with the existing works, our proposed method is able to generate the trajectory for the aerial manipulator in real-time without reducing the system to a multi-rotor-only model. The simulation and experimental results show that the proposed method can generate a collision-free and workspace-compatible trajectory for the aerial manipulator in real time. We will release the code as open-source for the benefit of the robotics community. The main contributions of this paper are listed as follows:

1) Unlike most existing works [14]–[20] that plan trajectories for aerial manipulators in constrained environments relying on pre-built maps, this paper presents the first planning algorithm capable of navigation in constrained environments without pre-built maps, while ensuring the high real-time performance required for locomotion tasks.
2) Workspace-feasibility is ensured by constructing a cost function with the soft-max technique and approximated SDFs, together with the convex hull property of the B-spline curve that guarantees the robotic arm motion satisfies the geometric feasibility.
3) An obstacle-avoidance model is designed with a simplified treatment of the tilt motion, whose feasibility is theoretically established. This model represents a trade-off between real-time performance and accuracy.

II. Related Work

A. Motion Planning for Multi-rotors

By invoking the differential flatness property [22], the motion planning problem is simplified to consider only the position and the yaw angle of the multi-rotor [23]. In cases where a 360-degree LiDAR, rather than a front-view camera, is mounted on the multi-rotor [24], the yaw angle of the multi-rotor can be disregarded. Consequently, the motion planning problem is further simplified to find a collision-free and dynamically feasible trajectory for a point in $\mathbb{R}^3$, with obstacle avoidance ensured by inflating the obstacles. Alternatively, the multi-rotor can be enclosed within an ellipsoid [25] or a convex polyhedron [26] to reduce the conservatism of the trajectory by formulating the problem in $SE(3)$. Most existing studies on motion planning can be divided into two main stages: path search and trajectory optimization. In [27], the jump point search algorithm is employed to determine the initial waypoints for the multi-rotor, and the trajectory is optimized by parameterizing it as a Bézier curve.

Compared with multi-rotors, aerial manipulators exhibit substantially higher degrees of freedom owing to the integration of robotic arms, thereby making the motion planning problem more complex and challenging.

B. Motion Planning for Aerial Manipulators

Motion planning for aerial manipulators involves generating safe and feasible trajectories that account for both the multi-rotor and the robotic arm. Some studies adopt decoupled motion planning frameworks, in which the multi-rotor and the manipulator are planned in separate stages [21]. For instance, Cao *et al.* [16] propose a two-stage decoupled method for pick-and-place tasks, where the aerial manipulator is enclosed within a large ball to simplify collision avoidance. Zhang *et al.* [28] drive the multi-rotor to a target area and then determines the trajectory for the robotic arm to grasp a target object from a moving platform.

More recent efforts have incorporated whole-body planning strategies. Kim *et al.* [14] integrate informed-RRT* with the local planner in [15] to generate collision-free trajectories in constrained environments. Alvaro *et al.* [17] consider the kinematic model of the aerial robotic system with two arms for long-reach manipulation and use the RRT*-based method to plan the trajectory for the multi-rotor and the robotic arm in a known environment. Deng *et al.* [18] propose a dynamic ellipsoidal approximation method that adapts to varying manipulator configurations for an aerial manipulator with a delta arm. However, this method may not generalize well to serial-link arms. Zhang *et al.* [19] formulated a coupled motion planning method by enclosing the aerial manipulator within a convex polyhedron and optimizing its trajectory with pre-built maps. Lee *et al.* [20] present a whole-body planning and control framework for omnidirectional aerial manipulators, utilizing multiple ellipsoids to enclose the aerial manipulator in known environments.

While certain works develop collision-avoidance models that capture the system state with higher accuracy, this increased fidelity often comes at the expense of real-time performance, thus limiting their applicability in constrained environments without pre-built maps. Meanwhile, some works [28]–[31] explore task-constrained planning without addressing collision avoidance issues.

III. Preliminary

A. Aerial Manipulator

In this paper, two coordinate frames are considered: $\{I\} = \{\boldsymbol{i}_1, \boldsymbol{i}_2, \boldsymbol{i}_3\}$ denotes the inertial frame and $\{B\} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3\}$ denotes the body-fixed frame, as illustrated in Fig. 2.

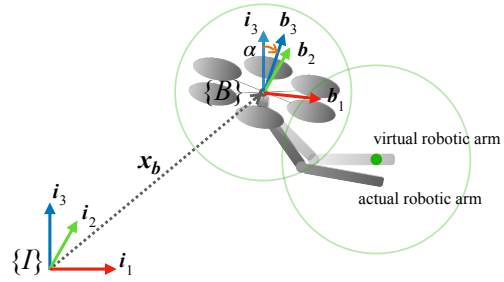


Fig. 2: Aerial Manipulators. The light-colored arm represents the virtual robotic arm without tilt motion, whereas the dark-colored arm corresponds to the actual one.

The aerial manipulator combines a multi-rotor and a robotic arm, as shown in Fig. 2. The state variables of the aerial manipulator are defined as follows:

$$\boldsymbol{q} = \begin{Bmatrix} \boldsymbol{x}_b \\ R \\ \boldsymbol{\theta} \end{Bmatrix} \in \mathbb{R}^3 \times SO(3) \times \Theta,$$

where $\boldsymbol{x}_b \in \mathbb{R}^3$ is the multi-rotor's position with respect to $\{I\}$, $R \in SO(3)$ denotes the rotation matrix from $\{B\}$ to $\{I\}$, and α is the tilt angle between $\boldsymbol{i}_3$ and $\boldsymbol{b}_3$. $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_n\} \in \Theta$ denotes the joint angles of the robotic arm, where $\Theta = \Theta_1 \times \Theta_2 \times \cdots \times \Theta_n$. θ_i $(i = 1, 2, \ldots, n)$ represents the angle of the i-th joint, and $\Theta_i \subset \mathbb{R}$ specifies the allowable range of θ_i. $\boldsymbol{x}_{e,\text{act}} \in \mathbb{R}^3$ is the position of the end-effector with respect to the inertial frame $\{I\}$, which can be expressed as:

$$\boldsymbol{x}_{e,\text{act}} = \boldsymbol{x}_b + R(\boldsymbol{b}_3, \psi)^b\boldsymbol{x}_e(\boldsymbol{\theta}), \tag{1}$$

where $\psi \in \mathbb{S}^1$ represents the yaw angle of the multi-rotor and ${}^b\boldsymbol{x}_e(\boldsymbol{\theta})$ is the position of the end-effector with respect to $\{B\}$. The rotation matrix $R(\boldsymbol{b}_3, \psi)$ can be divided into two parts, including the tilt motion part $H_2(\boldsymbol{b}_3) \in SO(3)$ and the yaw motion part $H_1(\psi) \in SO(3)$, as follows [32]:

$$R(\boldsymbol{b}_3, \psi) = H_2(\boldsymbol{b}_3)H_1(\psi). \tag{2}$$

If the tilt motion is small enough to be neglected, the rotation matrix $R(\boldsymbol{b}_3, \psi)$ is written as

$$R(\boldsymbol{b}_3, \psi) \approx R(\boldsymbol{i}_3, \psi) = H_2(\boldsymbol{i}_3)H_1(\psi) = H_1(\psi). \tag{3}$$

The virtual position of the end-effector can be derived as

$$\boldsymbol{x}_{e,\text{vir}} = \boldsymbol{x}_b + H_1(\psi)^b\boldsymbol{x}_e(\boldsymbol{\theta}) = \boldsymbol{x}_b + \boldsymbol{x}_{ve}(\boldsymbol{\theta}^+), \tag{4}$$

where $\boldsymbol{x}_{ve}(\boldsymbol{\theta}^+) = H_1(\psi)^b\boldsymbol{x}_e(\boldsymbol{\theta})$ and $\boldsymbol{\theta}^+ = \{\psi, \boldsymbol{\theta}\} \in \mathbb{S}^1 \times \Theta$. For the purpose of notational convenience, the virtual end-effector position is denoted by $\boldsymbol{x}_e$ under the simplified kinematics.

B. B-spline Curve

An s-order B-spline curve is determined by a set of $N+1$ control points $\mathcal{X} = \{\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_N\}$ and a knot vector $\mathcal{T} = [t_0, t_1, \ldots, t_M]^\top \in \mathbb{R}_+^{M+1}$, where $\mathcal{X}_i \in \mathbb{R}^3$, $t_i \in \mathbb{R}_+$ and $M = N + s + 1$.

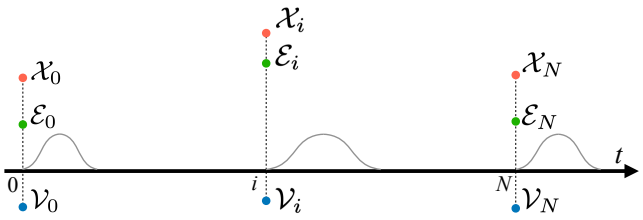


Fig. 3: $\mathcal{X}_i$, $\mathcal{E}_i$, and $\mathcal{V}_i$ belong to three different sets of control points, while the grey curves depict B-spline basis functions.

Property 1. *If two B-spline curves with the same order share the same knot vector, the linear combination of them is still a B-spline curve with the same order.*

It is assumed that the trajectory of the multi-rotor's position $\boldsymbol{x}_b(t)$ is a B-spline curve, determined by one set of control points $\mathcal{X} = \{\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_N\}$, and the trajectory of the virtual end-effector's position $\boldsymbol{x}_e(t)$ is also a B-spline curve, determined by another set of control points $\mathcal{E} = \{\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_N\}$. The two B-spline curves share the same time knot vector $\mathcal{T}$ and are formulated as follows:

$$\boldsymbol{x}_b(t) = \sum_{i=0}^{N} B_i(t)\mathcal{X}_i, \quad \boldsymbol{x}_e(t) = \sum_{i=0}^{N} B_i(t)\mathcal{E}_i,$$

where $B_i(t)$ is the B-spline basis function of order s. Then, $\boldsymbol{x}_{ve}(t)$ in (4) can be derived as

$$\begin{aligned} \boldsymbol{x}_{ve}(t) &= \boldsymbol{x}_e(t) - \boldsymbol{x}_b(t) \\ &= \sum_{i=0}^{N} B_i(t)(\mathcal{E}_i - \mathcal{X}_i) = \sum_{i=0}^{N} B_i(t)\mathcal{V}_i, \end{aligned} \tag{5}$$

where it can be concluded that the trajectory of $\boldsymbol{x}_{ve}(t)$ is also a B-spline curve with the control points $\mathcal{V} = \{\mathcal{V}_0, \mathcal{V}_1, \ldots, \mathcal{V}_N\}$ and share the same time knots vector $\mathcal{T}$ with $\boldsymbol{x}_b(t)$ and $\boldsymbol{x}_e(t)$, as shown in Fig. 3.

IV. Method Overview

A. Collision Model

To ensure real-time efficiency and avoid conservatism, the collision model of the aerial manipulator is represented by a collision model $\mathcal{O} = \{\mathcal{O}_b, \mathcal{O}_e\}$, as depicted in Fig. 2. In this model, $\mathcal{O}_b(\boldsymbol{x}_b, r_b) = \{\boldsymbol{x} \in \mathbb{R}^3 \mid \|\boldsymbol{x} - \boldsymbol{x}_b\| \leq r_b\}$ and $\mathcal{O}_e(\boldsymbol{x}_e, r_e) = \{\boldsymbol{x} \in \mathbb{R}^3 \mid \|\boldsymbol{x} - \boldsymbol{x}_e\| \leq r_e\}$ are two balls centered at the positions of the multi-rotor $\boldsymbol{x}_b$ and the virtual end-effector $\boldsymbol{x}_e$, with radii r_b and r_e, respectively.

In this model, $\mathcal{O}_b$ always encloses the multi-rotor as well as part of the robotic arm, irrespective of the system motion, and the corresponding set of mass points is denoted by $\mathcal{P}_b$. The remaining mass points of the system are denoted by $\mathcal{P}_e$. For any point $p \in \mathcal{P}_e$, its position in the body frame $\{B\}$ is represented as ${}^b\boldsymbol{x}(p)$, and the corresponding virtual and actual positions in the inertial frame $\{I\}$ are denoted by $\boldsymbol{x}_{\text{vir}}(p)$ and $\boldsymbol{x}_{\text{act}}(p)$, respectively.

Theorem 1. *By properly choosing the radius r_e and restricting α within the bound $\alpha_{\max}$, i.e. $\|\alpha\| \leq \alpha_{\max}$, it holds that $\boldsymbol{x}_{act}(p) \in \mathcal{O}_e, \forall p \in \mathcal{P}_e$.*

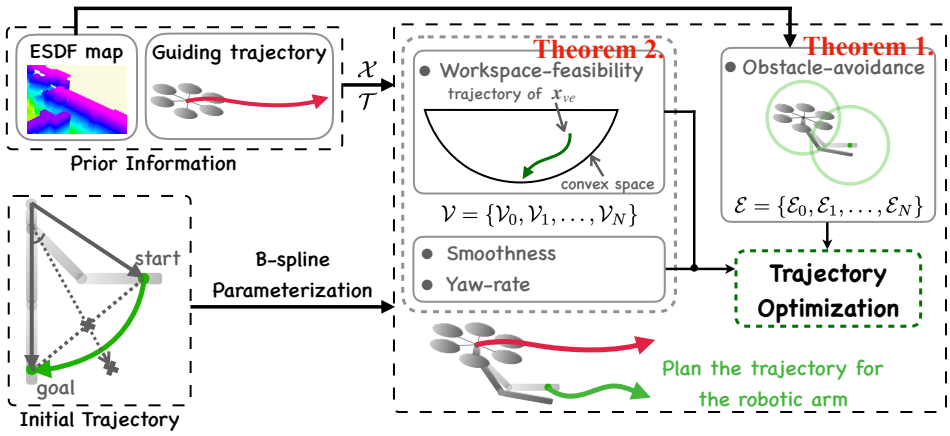


Fig. 4: RINGO framework. The ESDP map and a pre-planned multi-rotor trajectory, parameterized as a B-spline curve with a set of control points $\mathcal{X}$ and a knot vector $\mathcal{T}$, serve as prior information. An initial end-effector trajectory is generated by a Bézier curve and refined through optimization, where workspace-feasibility, smoothness, and yaw-rate costs depends on $\mathcal{V}$, whereas obstacle-avoidance relies on $\mathcal{E}$.

Proof. For any point $p \in \mathcal{P}_e$, the corresponding virtual and actual positions in the inertial frame $\{I\}$ are

$$\boldsymbol{x}_{\text{vir}}(p) = \boldsymbol{x}_b + H_1(\psi)\,{}^b\boldsymbol{x}(p), \tag{6a}$$
$$\boldsymbol{x}_{\text{act}}(p) = \boldsymbol{x}_b + H_2(\boldsymbol{b}_3)H_1(\psi)\,{}^b\boldsymbol{x}(p), \tag{6b}$$

and one can conclude that

$$\boldsymbol{x}_{\text{act}}(p) = \boldsymbol{x}_{\text{vir}}(p) + (H_2(\boldsymbol{b}_3) - \mathcal{I})H_1(\psi)\,{}^b\boldsymbol{x}(p), \tag{7}$$

in which $\mathcal{I}$ is the identity matrix. What the Theorem 1 state is that for any point $p \in \mathcal{P}_e$, the distance $\|\boldsymbol{x}_{\text{act}}(p) - \boldsymbol{x}_e\|$ between the actual position of point p and the center $\boldsymbol{x}_e$ can be bounded by a constant radius.

By the triangle inequality,

$$\begin{aligned}&\|\boldsymbol{x}_{\text{act}}(p) - \boldsymbol{x}_e\| \\ &= \left\|\boldsymbol{x}_{\text{vir}}(p) + (H_2(\boldsymbol{b}_3) - \mathcal{I})H_1(\psi)\,{}^b\boldsymbol{x}(p) - \boldsymbol{x}_e\right\| \\ &\le \underbrace{\|\boldsymbol{x}_{\text{vir}}(p) - \boldsymbol{x}_e\|}_{\text{virtual geometry}} + \underbrace{\|(H_2(\boldsymbol{b}_3) - \mathcal{I})H_1(\psi)\,{}^b\boldsymbol{x}(p)\|}_{\text{tilt motion deviation}}.\end{aligned} \tag{8}$$

Define $r = \sup_{p\in\mathcal{P}_e} \|H_1(\psi)({}^b\boldsymbol{x}(p) - {}^b\boldsymbol{x}_e)\| \in \mathcal{L}_\infty$, then by combining (4) and (6a), we can conclude that

$$\|\boldsymbol{x}_{\text{vir}}(p) - \boldsymbol{x}_e\| = \|H_1(\psi)({}^b\boldsymbol{x}(p) - {}^b\boldsymbol{x}_e)\| \le r. \tag{9}$$

Given the fact that

$$\begin{aligned}\|(H_2(\boldsymbol{b}_3) - \mathcal{I})H_1(\psi)\| &\le \|(H_2(\boldsymbol{b}_3) - \mathcal{I})\|\|H_1(\psi)\| \\ &= 2\sin\frac{\alpha}{2}\cdot 1 \le 2\sin\frac{\alpha_{\max}}{2},\end{aligned} \tag{10}$$

and define $\rho = \sup_{p\in\mathcal{P}_e} \|{}^b\boldsymbol{x}(p)\| \in \mathcal{L}_\infty$, hence

$$\begin{aligned}\|(H_2(\boldsymbol{b}_3) - \mathcal{I})H_1(\psi)\boldsymbol{x}(p)\| &\le \|(H_2(\boldsymbol{b}_3) - \mathcal{I})H_1(\psi)\|\|{}^b\boldsymbol{x}(p)\| \\ &\le 2\sin\frac{\alpha_{\max}}{2}\|{}^b\boldsymbol{x}(p)\| \\ &\le 2\rho\sin\frac{\alpha_{\max}}{2}.\end{aligned} \tag{11}$$

If the radius is chosen as

$$r_e \ge r + 2\rho\sin\frac{\alpha_{\max}}{2}, \tag{12}$$

by combining (8), (9), and (11), the fact that $\alpha \le \alpha_{\max}$, it can be concluded that

$$\begin{aligned}&\|\boldsymbol{x}_{\text{act}}(p) - \boldsymbol{x}_e\| \\ &\quad\le \|\boldsymbol{x}_{\text{vir}}(p) - \boldsymbol{x}_e\| + \|(H_2(\boldsymbol{b}_3) - \mathcal{I})H_1(\psi)\,{}^b\boldsymbol{x}(p)\| \\ &\quad\le r + 2\rho\sin\frac{\alpha_{\max}}{2} \le r_e.\end{aligned} \tag{13}$$

Hence for all points in the set $\mathcal{P}_e$ will be enclosed by the ball $\mathcal{O}_e$ centered at the position of the virtual end-effector $\boldsymbol{x}_e$ with the radius r_e, i.e., $\forall p \in \mathcal{P}_e, \boldsymbol{x}_{\text{act}}(p) \in \mathcal{O}_e$. □

According to the above analysis, the radius r_e can be decomposed into two components. The first term, r, is solely determined by the geometry of the robotic arm and corresponds to the minimum admissible radius in the absence of tilt motion. The second term, $2\rho\sin(\alpha_{\max}/2)$, reflects the additional margin introduced by the arm's motion and the maximum allowable tilt angle. When $\alpha_{\max}$ is small, this contribution becomes negligible, thereby enabling a more compact and less conservative collision model.

B. Problem Statement

To complete locomotion tasks in constrained environments without pre-built maps, aerial manipulators rely exclusively on onboard sensors for environment mapping and online planning to produce smooth and collision-free trajectories, thereby imposing strict real-time requirements on the algorithm.

With the analysis in Section IV-A, we plan the virtual end-effector trajectory $\boldsymbol{x}_e(t)$ jointly with the multi-rotor trajectory $\boldsymbol{x}_b(t)$. From the formulation in (4), we can derive the trajectory of $\boldsymbol{x}_{ve}(t)$. It can be converted, via inverse kinematics, into the generalized joint trajectory $\boldsymbol{\theta}^+(t)$. Executing $\boldsymbol{\theta}^+(t)$ yields the actual end-effector trajectory $\boldsymbol{x}_{e,\text{act}}(t)$ through the kinematics (1), which remains collision-free according to Theorem 1.

The problem that this paper aims to address can be defined as: *Given the start state $\boldsymbol{x}_b(0) = \boldsymbol{x}_{b,0}$ and $\boldsymbol{x}_e(0) = \boldsymbol{x}_{e,0}$ with their time derivatives, and the goal state $\boldsymbol{x}_b(T) = \boldsymbol{x}_{b,d}$ and $\boldsymbol{x}_e(T) = \boldsymbol{x}_{e,d}$ with zero time derivatives, the problem is to*

plan a collision-free trajectory for the aerial manipulator in real time within constrained environments.

C. Main Method

As shown in Fig. 4, the multi-rotor trajectory is firstly planned as a guiding trajectory and parameterized as a B-spline with control points $\mathcal{X}$ and a knot vector $\mathcal{T}$. An initial end-effector trajectory for $\boldsymbol{x}_{ve}(t)$ is generated as a quadratic Bézier curve, and parameterized as a B-spline curve with control points $\mathcal{V}$, as detailed in Section IV-D. Combining $\boldsymbol{x}_b(t)$ and $\boldsymbol{x}_{ve}(t)$ yields an initial trajectory for $\boldsymbol{x}_e(t)$, which is represented as a B-spline curve sharing the same time knot vector $\mathcal{T}$ and using control points $\mathcal{E}$. Finally, we refine $\boldsymbol{x}_e(t)$ and $\boldsymbol{x}_{ve}(t)$ by minimizing a cost comprising obstacle-avoidance, workspace-compatible, and smoothness terms, as detailed in Section V.

D. Initial Trajectory Generation

As shown in Fig. 4, the initial trajectory for the virtual end-effector $\boldsymbol{x}_e(t)$ is constructed by combining a pre-obtained trajectory $\boldsymbol{x}_b(t)$ with a quadratic Bézier curve for $\boldsymbol{x}_{ve}(t)$. The Bézier curve is defined by $\boldsymbol{x}_{ve}(0)$, $\boldsymbol{P}$, and $\boldsymbol{x}_{ve}(T)$, where the intermediate point $\boldsymbol{P}$ is defined as

$$\boldsymbol{P} = \frac{1}{2}\lambda\left(\boldsymbol{x}_{ve}(0) + \boldsymbol{x}_{ve}(T)\right), \tag{14}$$

with $\lambda = \log\left(\frac{1}{2}\left|\arccos\left(\boldsymbol{x}_{ve}^{\top}(0)\boldsymbol{x}_{ve}(T)\right) + 1\right| + 1\right)$. This formulation effectively pushes the midpoint of the Bézier curve away from the straight line between the start and goal. A larger angle between the start and goal vectors yields a greater λ, which enhances smoothness in the joint space.

The resulting initial trajectory for $\boldsymbol{x}_e(t)$ is obtained by combining $\boldsymbol{x}_b(t)$ with the quadratic Bézier curve of $\boldsymbol{x}_{ve}(t)$. This trajectory is subsequently refined and represented as a B-spline curve parameterized with control points $\mathcal{E} = \{\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_N\}$ and the knot vector $\mathcal{T} = [t_0, t_1, \ldots, t_M]^{\top}$.

V. Trajectory Optimization

Since obstacle avoidance is not explicitly addressed during the initial trajectory generation, the problem is subsequently cast as a trajectory optimization formulation:

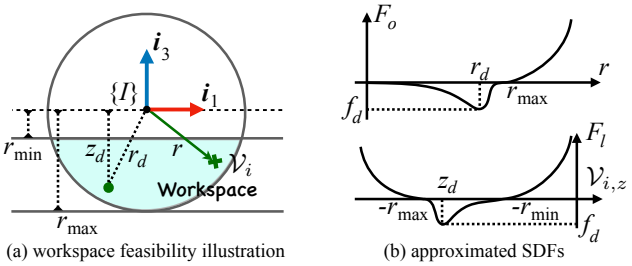
$$\min_{\mathcal{E}} f = \lambda_w f_w + \lambda_s f_s + \lambda_y f_y + \lambda_d f_d, \tag{15}$$

where f denotes the overall cost. f_w, f_s, f_y, and f_d correspond to the workspace-feasibility cost, smoothness cost, yaw-rate cost, and obstacle-avoidance cost, respectively. The scalar coefficients λ_s, λ_w, λ_y, and λ_d serve as the corresponding weighting factors.

In the optimization problem (15), the decision variables are the control points $\mathcal{E} = \{\mathcal{E}_0, \mathcal{E}_1, \ldots, \mathcal{E}_N\}$, which define the B-spline curve of the virtual end-effector trajectory $\boldsymbol{x}_e(t)$ in the inertial frame $\{I\}$. For notational convenience, another set of control points $\mathcal{V}$ is occasionally introduced, with the relationship defined in (5), where $\mathcal{X}$ denotes the control points associated with the multi-rotor trajectory. In the optimization formulation, $\mathcal{X}$ is treated as a fixed constant, while $\mathcal{E}$ (and equivalently $\mathcal{V}$) serve as the optimization variables.

A. Workspace-feasibility Cost

In trajectory planning, it is essential to preserve a reasonable spatial relationship between $\boldsymbol{x}_e(t)$ and $\boldsymbol{x}_b(t)$. Otherwise, the end-effector may deviate excessively from the multi-rotor, resulting in physically infeasible motions. To prevent such cases, a workspace-feasibility cost is introduced to penalize large deviations, thereby constraining $\boldsymbol{x}_{ve}(t)$ to remain within a bounded region relative to the multi-rotor. The auxiliary variable $\boldsymbol{x}_{ve}(t)$ derived in (4) represents the virtual end-effector position originally expressed in $\{B\}$ but transformed into the inertial frame $\{I\}$ through the rotation matrix $H_1(\psi)$. Therefore, $\boldsymbol{x}_{ve}(t)$ is expressed in the inertial frame $\{I\}$.



(a) workspace feasibility illustration (b) approximated SDFs

Fig. 5: (a) The cyan region denotes the feasible workspace in a 2-D illustration. The green point • marks the desired position for $\boldsymbol{x}_{ve}(t)$, while the green cross × indicates the control point $\mathcal{V}_i$. (b) Two approximated SDFs are shown, each expressed in piecewise polynomial form.

As depicted in Fig. 5, the workspace for $\boldsymbol{x}_{ve}(t)$ is formulated as the intersection of a sphere and two half-spaces determined by parallel planes, corresponding to the cyan region, which is the convex workspace $\mathbb{W}$. To ensure that $\boldsymbol{x}_{ve}(t)$ remains within the workspace, the following cost function is introduced:

$$\begin{aligned} f_w &= \sum_{i=s}^{N-s} F_w(\mathcal{V}_i) \\ &= \sum_{i=s}^{N-s} \frac{1}{k}\log\left(e^{kh_o F_o(\mathcal{V}_i)} + e^{kh_l F_l(\mathcal{V}_i)}\right), \end{aligned} \tag{16}$$

where $F_o(\mathcal{V}_i)$ and $F_l(\mathcal{V}_i)$ denote the approximated SDFs associated with the sphere and the limiting planes, respectively. k, h_l and h_o are three positive parameters that regulate the smoothness and relative weight of the cost. It is worth noting that the log-sum-exp term serves as a smooth approximation of the maximum between $h_o F_o(\mathcal{V}_i)$ and $h_l F_l(\mathcal{V}_i)$. The explicit definitions of $F_o(\mathcal{V}_i)$ and $F_l(\mathcal{V}_i)$ are given in the form of

piecewise polynomials as follows:

$$F_o(\mathcal{V}_i) = \begin{cases} b_{o,1}r^2 + a_{o,1}r^3, & 0 \le r \le r_d \\ b_{o,2}(r-r_{\max})^2 + a_{o,2}(r-r_{\max})^3, & r_d \le r \le r_{\max} \\ (r-r_{\max})^2, & r_{\max} \le r \end{cases} \tag{17a}$$

$$F_l(\mathcal{V}_i) = \begin{cases} (\mathcal{V}_{i,z}+r_{\max})^2, & \mathcal{V}_{i,z} \le -r_{\max} \\ b_{l,1}(\mathcal{V}_{i,z}+r_{\max})^2 + a_{l,1}(\mathcal{V}_{i,z}+r_{\max})^3, & -r_{\max} \le \mathcal{V}_{i,z} \le -z_d \\ b_{l,2}(\mathcal{V}_{i,z}+r_{\min})^2 + a_{l,2}(\mathcal{V}_{i,z}+r_{\min})^3, & -z_d \le \mathcal{V}_{i,z} \le -r_{\min} \\ (\mathcal{V}_{i,z}+r_{\min})^2, & -r_{\min} \le \mathcal{V}_{i,z} \end{cases} \tag{17b}$$

where $r_{\max}$ and $r_{\min}$ are the parameters relevant to the convex region $\mathbb{W}$. $\mathcal{V}_i = [\mathcal{V}_{i,x}, \mathcal{V}_{i,y}, \mathcal{V}_{i,z}]^\top \in \mathbb{R}^3$ denotes the control point, with $\mathcal{V}_{i,x}$, $\mathcal{V}_{i,y}$, and $\mathcal{V}_{i,z}$ denoting its Cartesian components. $r = \sqrt{\mathcal{V}_i^\top \mathcal{V}_i}$ is the norm of the vector $\mathcal{V}_i$. By ensuring that $F_o(\mathcal{V}_i)$ and $F_l(\mathcal{V}_i)$ are continuously differentiable and incorporating the given parameters r_d, z_d, and f_d, the coefficients $a_{o,j}$, $b_{o,j}$, $a_{l,j}$, and $b_{l,j}$ $(j = 1, 2)$ can be explicitly computed. Moreover, the use of piecewise polynomials facilitates subsequent trajectory optimization and gradient computation.

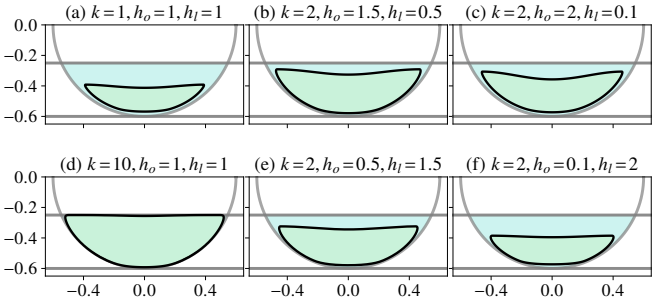


Fig. 6: The cyan regions represent the feasible workspace in a 2-D illustration, while the green areas enclosed by black curves illustrate the zero-level sets of the smooth soft-max function. Different parameter choices of k, h_o, and h_l lead to distinct shapes of the approximated feasible boundary.

Theorem 2. *If the cost $F_w(\mathcal{V}_i)$ associated with each control point is nonpositive, i.e., $F_w(\mathcal{V}_i) \le 0$, then $\boldsymbol{x}_{ve}(t) \in \mathbb{W}, \forall t$, which means that the entire B-spline curve $\boldsymbol{x}_{ve}(t)$ is guaranteed to lie within the convex workspace $\mathbb{W}$.*

Proof. The workspace $\mathbb{W}$, illustrated in Fig. 5, is defined as

$$\mathbb{W} = \{\boldsymbol{x} \in \mathbb{R}^3 \mid F_o(\boldsymbol{x}) \le 0, F_l(\boldsymbol{x}) \le 0\}.$$

Let $\mathbb{F}$ denote the closed set characterized by $F_w(\boldsymbol{x}) \le 0$, i.e.,

$$\mathbb{F} = \{\boldsymbol{x} \in \mathbb{R}^3 \mid F_w(\boldsymbol{x}) \le 0\}.$$

By the log-sum-exp inequality and $k > 0$, which states that for any $a, b \in \mathbb{R}$,

$$\frac{1}{k}\log(e^a + e^b) \le \max(a, b),$$

it follows that

$$F_w(\boldsymbol{x}) \ge \max(h_o F_o(\boldsymbol{x}), h_l F_l(\boldsymbol{x})). \tag{18}$$

If $F_w(\boldsymbol{x}) \le 0$, then (18) implies

$$\max(h_o F_o(\boldsymbol{x}), h_l F_l(\boldsymbol{x})) \le 0 \Rightarrow h_o F_o(\boldsymbol{x}) \le 0,\ h_l F_l(\boldsymbol{x}) \le 0.$$

Since $h_o, h_l > 0$, and recalling the definitions of $F_o(\boldsymbol{x})$ and $F_l(\boldsymbol{x})$ in (17a) and (17b), it can be concluded that $\mathbb{F}$ is a subset of the workspace $\mathbb{W}$, i.e., $\mathbb{F} \subset \mathbb{W}$, as illustrated in Fig. 7.

For illustration, consider a case where $\mathbb{C}_1 \subset \mathbb{R}^3$ denotes the interior of a tetrahedron with vertices $\{\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4\}$ and the corresponding B-spline curve segment $\boldsymbol{x}_{ve1}(t)$ is defined over the time interval $[t_1, t_2]$. By the convex hull property of B-spline curves [33], the trajectory segment satisfies

$$\boldsymbol{x}_{ve1}(t) \in \mathbb{C}_1, t \in [t_1, t_2]. \tag{19}$$

Moreover, since $F_w(\mathcal{V}_i) \le 0$ holds for all control points, it can be derived that

$$\mathcal{V}_i \in \mathbb{F} \subset \mathbb{W},\ i = 1, 2, 3, 4, \tag{20}$$

$$\mathbb{C}_1 \subset \mathbb{F} \subset \mathbb{W}. \tag{21}$$

Combining the above two results (19) and (21), it can be concluded that

$$\boldsymbol{x}_{ve1}(t) \in \mathbb{W}, t \in [t_1, t_2]. \tag{22}$$

For each segment of the B-spline curve, the same conclusion as in (22) holds, i.e., the segment remains confined within the workspace. Therefore, by concatenating all such segments, it follows that the entire B-spline curve $\boldsymbol{x}_{ve}(t)$ is theoretically guaranteed to be in the convex workspace $\mathbb{W}$. □

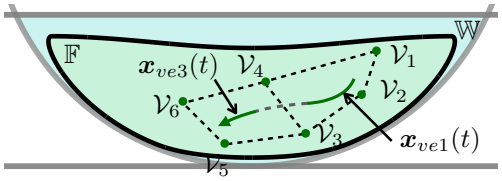


Fig. 7: The green area representing $\mathbb{F}$ is in the cyan area $\mathbb{W}$. Two segments of the B-spline curve $\boldsymbol{x}_{ve1}(t)$ and $\boldsymbol{x}_{ve3}(t)$ are determined by the control points set $\{\mathcal{V}_1, \mathcal{V}_2, \mathcal{V}_3, \mathcal{V}_4\}$ and $\{\mathcal{V}_3, \mathcal{V}_4, \mathcal{V}_5, \mathcal{V}_6\}$, respectively.

B. Smoothness Cost

The smoothness cost is formulated as minimizing the normalization of the acceleration control points. The smoothness cost is formulated as

$$f_s = \sum_{i=s-2}^{N-s} F_s(\mathcal{V}_i, \mathcal{V}_{i+1}, \mathcal{V}_{i+2}) = \sum_{i=s-2}^{N-s} \|F_{s,i}\|^2, \tag{23}$$

where each term corresponds $\|F_{s,i}\|^2$ to the squared norm of the acceleration control point, given by

$$\|F_{s,i}\|^2 = \|m_i\mathcal{V}_i + m_{i+1}\mathcal{V}_{i+1} + m_{i+2}\mathcal{V}_{i+2}\|^2.$$

Here, the coefficients m_i, m_{i+1}, m_{i+2} are determined by the non-uniform knot vector $\mathcal{T}$ and are expressed as

$$m_i = \frac{s(s-1)}{t_{i+2,s-1}} \frac{1}{t_{i+1,s}},$$
$$m_{i+1} = \frac{s(s-1)}{t_{i+2,s-1}} \left(-\frac{1}{t_{i+2,s}} - \frac{1}{t_{i+1,s}} \right),$$
$$m_{i+2} = \frac{s(s-1)}{t_{i+2,s-1}} \frac{1}{t_{i+2,s}},$$

where $t_{i,s}$ denotes the time span from t_i to t_{i+s}, i.e., $t_{i,s} = t_{i+s} - t_i$.

C. Yaw-rate Cost

$\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$ are the projections of the control points $\mathcal{V}_i$ and $\mathcal{V}_{i+1}$ on the x-y plane, expressed as follows:

$$\boldsymbol{v}_i = \begin{bmatrix} \mathcal{V}_{i,x} \\ \mathcal{V}_{i,y} \\ 0 \end{bmatrix}, \ \boldsymbol{v}_{i+1} = \begin{bmatrix} \mathcal{V}_{i+1,x} \\ \mathcal{V}_{i+1,y} \\ 0 \end{bmatrix}.$$

$\boldsymbol{n}_i$ and $\boldsymbol{n}_{i+1}$ are the normalized vector of $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$, respectively, which means that

$$\boldsymbol{n}_i = \begin{bmatrix} \frac{\mathcal{V}_{i,x}}{\sqrt{\mathcal{V}_{i,x}^2+\mathcal{V}_{i,y}^2}} \\ \frac{\mathcal{V}_{i,y}}{\sqrt{\mathcal{V}_{i,x}^2+\mathcal{V}_{i,y}^2}} \\ 0 \end{bmatrix}, \ \boldsymbol{n}_{i+1} = \begin{bmatrix} \frac{\mathcal{V}_{i+1,x}}{\sqrt{\mathcal{V}_{i+1,x}^2+\mathcal{V}_{i+1,y}^2}} \\ \frac{\mathcal{V}_{i+1,y}}{\sqrt{\mathcal{V}_{i+1,x}^2+\mathcal{V}_{i+1,y}^2}} \\ 0 \end{bmatrix}.$$

Then, the yaw-rate cost is formulated as

$$f_y = \sum_{i=s}^{N-s-1} F_y(\mathcal{V}_i, \mathcal{V}_{i+1}) = \sum_{i=s}^{N-s-1} \|\boldsymbol{n}_{i+1} - \boldsymbol{n}_i\|^2. \quad (24)$$

D. Obstacle-avoidance Cost

The obstacle avoidance cost is defined based on the distance between the control point $\mathcal{E}_i$ and its nearest obstacle, and it is formulated as

$$f_d = \sum_{i=s}^{N-s} F_d(d(\mathcal{E}_i)), \quad (25)$$

where $d(\mathcal{E}_i)$ denotes the Euclidean distance from the control point $\mathcal{E}_i$ to the nearest obstacle, which can be effectively extracted from the ESDF map [34]. The cost function on the i-th control point $F_d(d(\mathcal{E}_i))$ is as follows:

$$F_d(d(\mathcal{E}_i)) = \begin{cases} (d(\mathcal{E}_i) - r_e)^2, & d(\mathcal{E}_i) \le r_e \\ 0, & d(\mathcal{E}_i) > r_e \end{cases} \quad (26)$$

where r_e is the radius ensuring the safety, given by (12).

The gradients of the above-mentioned cost functions with respect to the control points are derived in Appendix A.

VI. Implementation and Results

In order to verify the effectiveness of the proposed method, the implementation details, simulation results and experimental results are introduced in this section.

A. Algorithm Implementation and Experiment Setup

1) Algorithm Implementation: The order s of the B-spline curve is set to 3, and the optimization problem in (15) is solved using NLopt Library[1]. The guiding trajectory for the multi-rotor is generated following the method of Fast-Planner [24]. To remain consistent with the collision model design in Section IV-A, the multi-rotor's maximum acceleration is bounded to ensure that the resulting tilt angle remains within the prescribed $\alpha_{\max}$.

The simulation platform is adapted from Fast-Planner, including the multi-rotor dynamics model, random map generator, and point cloud rendering module. All simulations are conducted on an Intel Core i7−13700 CPU and GeForce GTX 3070 Ti GPU. To ensure a fair comparison, all computations are conducted with the same aforementioned computation capability. In real world experiments, all the state estimation, mapping and motion planning modules run on an Intel Core i7−13700 CPU with 16GB RAM.

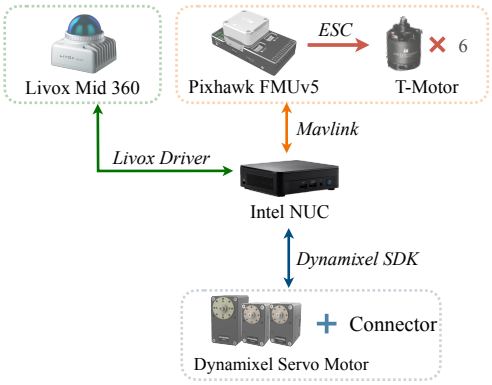


Fig. 8: Experimental platform.

2) Experiment Setup: The aerial manipulator system consists of an F550 hexrotor and a custom-built robotic arm. Specifically, the hexrotor is equipped with a Pixhawk FMUv5 flight controller, which connects to the onboard computer via the Mavlink communication protocol. The flight controller employs the cascaded P-PID controller incorporating the feed-forward term from the robotic arm, which is embedded within the Pixhawk FMUv5[2]. The flight controller powers six pairs of T-Motors and 10-inch propellers through the Electronic Speed Controllers (ESCs). The robotic arm is conducted by Dynamixel servo motors and a few self-designed connectors.

The Livox Mid 360 LiDAR is mounted on the multi-rotor and integrated using the Livox Driver[3]. Fast-lio2 [35] is employed to estimate the odometry of the multi-rotor and to generate a dense point cloud map. The state estimation for the robotic arm is handled by the Dynamixel SDK, while the end-effector's position is computed using the robotic arm's forward kinematics.

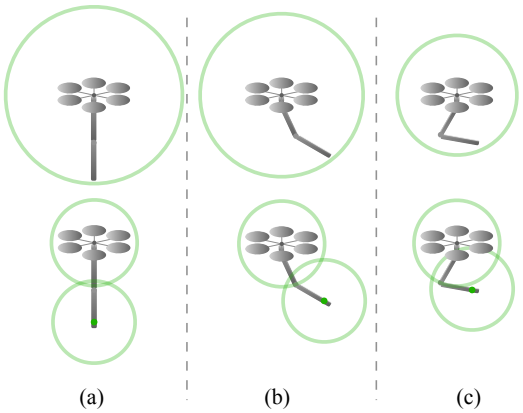


Fig. 9: Illustration of simulation scenarios.

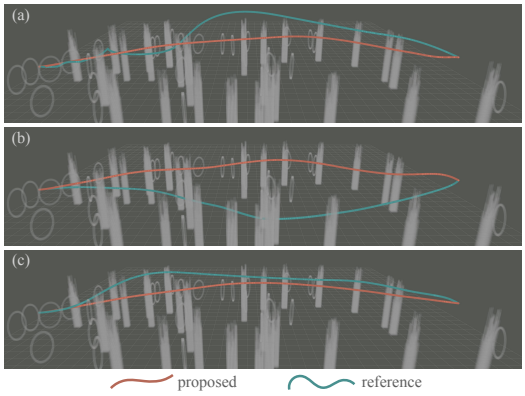


Fig. 10: Multi-rotor's travel trajectories in different scenarios.

B. Simulation Results

Since most existing works on motion planning for aerial manipulators rely on pre-built maps [14]–[20], a direct comparison is not feasible in our online planning setting. To provide a fair comparison, a multi-rotor-only planning is implemented where the robotic arm remains fixed throughout the entire trajectory and only the multi-rotor's motion is planned, which serves as a *reference* method. Although it is not an existing method from the literature, it highlights the necessity of considering the robotic arm's motion in the planning process. In the simulation study, three representative scenarios are considered, as illustrated in Fig. 9. In each scenario, the aerial manipulator starts from different initial configurations, where both the initial and goal states of the multi-rotor and the robotic arm are explicitly shown.

The quantitative results in TABLE I show that the proposed method consistently achieves shorter trajectory length and flight time compared to the reference multi-rotor-only method. This improvement is attributed to the reduced conservatism when the manipulator's motion is explicitly considered, rather than conservatively enclosing the entire aerial manipulator within a bounding ball. The average computation time for each planning process is also reported. Although the proposed method additionally accounts for the robotic arm's motion, its overall computation time is lower than that of the reference method in all scenarios. This is because the reference method requires planning for the multi-rotor within a more constrained configuration space, which increases both path searching and trajectory optimization computation cost.

[1]https://nlopt.readthedocs.io/en/latest/
[2]https://ardupilot.org/copter/docs/common-cuav-v5plus-overview.html
[3]https://github.com/Livox-SDK/Livox-SDK

TABLE I: Quantitative Comparison.

	Method	Travel Trajectory		Computation Time (ms)		
		Length (m)	Time (s)	multi-rotor	robotic arm	total
(a)	proposed	**42.52**	**15.79**	0.79	0.22	**1.01**
	reference	49.63	22.87	1.73	/	1.73
(b)	proposed	**42.58**	**16.03**	1.00	0.21	**1.21**
	reference	44.08	17.72	1.55	/	1.55
(c)	proposed	**42.39**	**15.88**	1.05	0.12	**1.17**
	reference	43.82	16.65	1.22	/	1.22

For the purpose of further exploring the impact of the proposed method on the computation time of the robotic arm, we present a detailed boxplot in Fig. 11. The boxplot provides a detailed illustration of the computation time for the robotic arm across different scenarios. Even in the most extreme case for scenario (a), the time remains *less than 0.6 ms*.

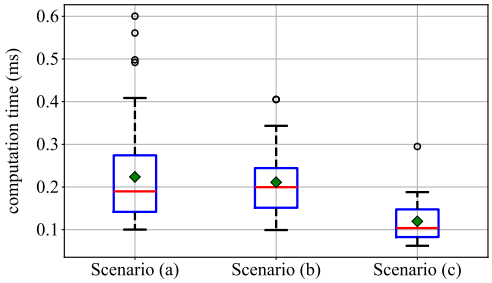


Fig. 11: Distribution of the robotic arm's computation time across different scenarios.

C. Experimental Results

The experiments rely solely on onboard localization and mapping, without requiring any pre-built maps or external positioning systems. In this paper, we present two fully autonomous flight experiments conducted in constrained environments, as illustrated in Fig. 12 and Fig. 13.

In Experiment 1, the aerial manipulator successfully plans a collision-free trajectory for both the multi-rotor and the end-effector through a ring-shaped obstacle. In Experiment 2, the aerial manipulator starts behind vertical obstacles, and it successfully plans a collision-free trajectory that involves hurdling over horizontal obstacles and navigating through a ring-shaped obstacle.

To the best of our knowledge, this is the first instance of an aerial manipulator achieving autonomous flight in constrained environments without pre-built maps. The experimental results demonstrate the effectiveness of the proposed method in real-world applications.

VII. Conclusion

In this paper, we proposed RINGO, a real-time motion planning method for aerial manipulators in constrained environ-

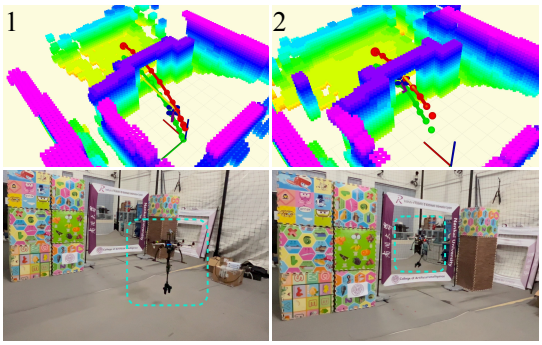


Fig. 12: The snapshots and visualization of Experiment 1.

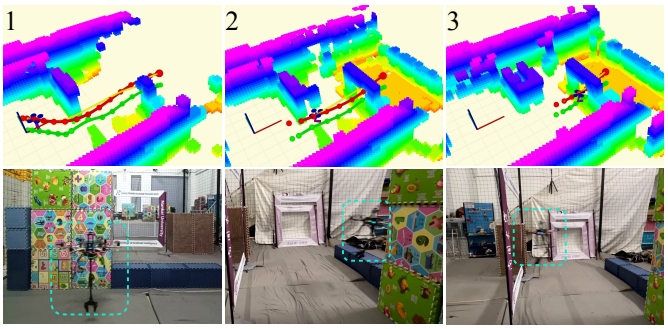


Fig. 13: The snapshots and visualization of Experiment 2.

ments without relying on pre-built maps. The pre-planned and parameterized trajectory of the multi-rotor serves as a guiding reference. An initial trajectory for the end-effector is generated by a quadratic Bézier curve. Then, the gradient-based optimization method incorporating workspace-feasibility, smoothness, yaw-rate, and obstacle-avoidance costs is utilized to refine the trajectory of the end-effector. By jointly considering the multi-rotor and the robotic arm, RINGO reduces unnecessary conservatism in trajectory planning and achieves high real-time performance required for locomotion tasks. The incorporation of a simplified obstacle-avoidance model and theoretically guaranteed workspace-feasibility constraints enables the system to generate smooth and geometrically valid trajectories in real time. Simulation and experimental results validated that the proposed method can efficiently generate collision-free and workspace-feasible trajectories in real time.

APPENDIX A
THE GRADIENT OF THE COST FUNCTION

The workspace-feasibility, smoothness, and yaw-rate are relevant to the control points $\mathcal{V}$. Owing to the derivation in (5), the gradient of them with respect to $\mathcal{E}_i$ is derived as

$$\begin{aligned}\frac{\partial f_w}{\partial \mathcal{E}_i} &= \frac{\partial f_w}{\partial \mathcal{V}_i}\frac{\partial \mathcal{V}_i}{\partial \mathcal{E}_i} = \frac{\partial f_w}{\partial \mathcal{V}_i},\\ \frac{\partial f_s}{\partial \mathcal{E}_i} &= \frac{\partial f_s}{\partial \mathcal{V}_i}\frac{\partial \mathcal{V}_i}{\partial \mathcal{E}_i} = \frac{\partial f_s}{\partial \mathcal{V}_i},\\ \frac{\partial f_y}{\partial \mathcal{E}_i} &= \frac{\partial f_y}{\partial \mathcal{V}_i}\frac{\partial \mathcal{V}_i}{\partial \mathcal{E}_i} = \frac{\partial f_y}{\partial \mathcal{V}_i}.\end{aligned}$$

A. Workspace-feasibility cost

The gradient of the workspace-feasibility cost with respect to the control point $\mathcal{V}_i$ is expressed as

$$\begin{aligned}\frac{\partial f_w}{\partial \mathcal{V}_i} = &\frac{h_o e^{h_o k F_o(\mathcal{V}_i)}}{e^{h_o k F_o(\mathcal{V}_i)} + e^{h_l k F_l(\mathcal{V}_i)}}\frac{\partial F_o(\mathcal{V}_i)}{\partial \mathcal{V}_i}\\ &+ \frac{h_l e^{h_l k F_l(\mathcal{V}_i)}}{e^{h_o k F_o(\mathcal{V}_i)} + e^{h_l k F_l(\mathcal{V}_i)}}\frac{\partial F_l(\mathcal{V}_i)}{\partial \mathcal{V}_i}.\end{aligned}$$

The gradient of r with respect to $\mathcal{V}_i$ is obtained as

$$\frac{\partial r^2}{\partial r}\cdot\frac{\partial r}{\partial \mathcal{V}_i} = 2r\cdot\frac{\partial r}{\partial \mathcal{V}_i} = 2\mathcal{V}_i \;\Rightarrow\; \frac{\partial r}{\partial \mathcal{V}_i} = \frac{\mathcal{V}_i}{r}.$$

The gradient of $F_o(\mathcal{V}_i)$ with respect to $\mathcal{V}_i$ is given by

$$\begin{aligned}&\frac{\partial F_o(\mathcal{V}_i)}{\partial \mathcal{V}_i} = \frac{\partial F_o(\mathcal{V}_i)}{\partial r}\frac{\partial r}{\partial \mathcal{V}_i} = \frac{\partial F_o(\mathcal{V}_i)}{\partial r}\frac{\mathcal{V}_i}{r}.\\ &\frac{\partial F_o(\mathcal{V}_i)}{\partial r} =\\ &\begin{cases} 2b_{o,1}r + 3a_{o,1}r^2, & 0 \le r \le r_d\\ 2b_{o,2}(r - r_{\max}) + 3a_{o,2}(r - r_{\max})^2, & r_d \le r \le r_{\max}\\ 2(r - r_{\max}). & r_{\max} \le r \end{cases}\end{aligned}$$

The gradient of the line cost $F_l(\mathcal{V}_i)$ is calculated as

$$\begin{aligned}&\frac{\partial F_l(\mathcal{V}_i)}{\partial \mathcal{V}_i} = \begin{bmatrix}0 & 0 & \frac{\partial F_l(\mathcal{V}_i)}{\partial \mathcal{V}_{i,z}}\end{bmatrix}^\top.\\ &\frac{\partial F_l(\mathcal{V}_i)}{\partial \mathcal{V}_{i,z}} =\\ &\begin{cases} 2(\mathcal{V}_{i,z} + r_{\max}), & -r_{\max} \le \mathcal{V}_{i,z},\\ 2b_{l,1}(\mathcal{V}_{i,z} + r_{\max}) + & \\ \quad 3a_{l,1}(\mathcal{V}_{i,z} + r_{\max})^2, & -r_{\max} \le \mathcal{V}_{i,z} \le -z_d\\ 2b_{l,2}(\mathcal{V}_{i,z} + r_{\min}) + & \\ \quad 3a_{l,2}(\mathcal{V}_{i,z} + r_{\min})^2, & -z_d \le \mathcal{V}_{i,z} \le -r_{\min}\\ 2(\mathcal{V}_{i,z} + r_{\min}). & -r_{\min} \le \mathcal{V}_{i,z} \end{cases}\end{aligned}$$

B. Smoothness cost

The gradient of the smoothness cost f_s with respect to the control point $\mathcal{V}_i$ is calculated as

$$\begin{aligned}\frac{\partial f_s}{\partial \mathcal{V}_i} &= \left(2F_{s,i}^\top\frac{\partial F_{s,i}}{\partial \mathcal{V}_i}\right)^\top + \left(2F_{s,i-1}^\top\frac{\partial F_{s,i-1}}{\partial \mathcal{V}_i}\right)^\top\\ &\quad + \left(2F_{s,i-2}^\top\frac{\partial F_{s,i-2}}{\partial \mathcal{V}_i}\right)^\top\\ &= 2\left(\frac{\partial F_{s,i}}{\partial \mathcal{V}_i}\right)^\top F_{s,i} + 2\left(\frac{\partial F_{s,i-1}}{\partial \mathcal{V}_i}\right)^\top F_{s,i-1}\\ &\quad + 2\left(\frac{\partial F_{s,i-2}}{\partial \mathcal{V}_i}\right)^\top F_{s,i-2}.\\ &= 2m_{i,0}F_{s,i} + 2m_{i-1,1}F_{s,i-1} + 2m_{i-2,2}F_{s,i-2}.\end{aligned}$$

C. Yaw-rate cost

The gradient of the yaw-rate cost f_y with respect to the control point $\mathcal{V}_i$ is given as

$$\begin{aligned}\frac{\partial f_y}{\partial \mathcal{V}_i} &= \frac{\partial F_y(\mathcal{V}_{i-1},\mathcal{V}_i)}{\partial \mathcal{V}_{i-1}} + \frac{\partial F_y(\mathcal{V}_{i-1},\mathcal{V}_i)}{\partial \mathcal{V}_i} \\ &= \left(\frac{\partial \boldsymbol{n}_i}{\partial \mathcal{V}_i}\right)^\top 2(\boldsymbol{n}_i - \boldsymbol{n}_{i-1}) - \left(\frac{\partial \boldsymbol{n}_i}{\partial \mathcal{V}_i}\right)^\top 2(\boldsymbol{n}_{i+1} - \boldsymbol{n}_i) \\ &= \left(\frac{\partial \boldsymbol{n}_i}{\partial \mathcal{V}_i}\right)^\top 2(\boldsymbol{n}_i - \boldsymbol{n}_{i-1} - \boldsymbol{n}_{i+1} + \boldsymbol{n}_i),\end{aligned}$$

where,

$$\begin{aligned}\frac{\partial \boldsymbol{n}_i}{\partial \mathcal{V}_i} &= \begin{bmatrix} \frac{\partial \boldsymbol{n}_{i,x}}{\partial \mathcal{V}_{i,x}} & \frac{\partial \boldsymbol{n}_{i,x}}{\partial \mathcal{V}_{i,y}} & 0 \\ \frac{\partial \boldsymbol{n}_{i,y}}{\partial \mathcal{V}_{i,x}} & \frac{\partial \boldsymbol{n}_{i,y}}{\partial \mathcal{V}_{i,y}} & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \frac{\mathcal{V}_{i,y}^2}{(\mathcal{V}_{i,x}^2+\mathcal{V}_{i,y}^2)^{\frac{3}{2}}} & \frac{-\mathcal{V}_{i,x}\mathcal{V}_{i,y}}{(\mathcal{V}_{i,x}^2+\mathcal{V}_{i,y}^2)^{\frac{3}{2}}} & 0 \\ \frac{-\mathcal{V}_{i,x}\mathcal{V}_{i,y}}{(\mathcal{V}_{i,x}^2+\mathcal{V}_{i,y}^2)^{\frac{3}{2}}} & \frac{\mathcal{V}_{i,x}^2}{(\mathcal{V}_{i,x}^2+\mathcal{V}_{i,y}^2)^{\frac{3}{2}}} & 0 \\ 0 & 0 & 0 \end{bmatrix} = \left(\frac{\partial \boldsymbol{n}_i}{\partial \mathcal{V}_i}\right)^\top.\end{aligned}$$

D. Obstacle-avoidance cost

The gradient of the obstacle avoidance cost f_d with respect to $\mathcal{E}_i$ is calculated as

$$\frac{f_d}{\partial \mathcal{E}_i} = \begin{cases} 2\left(d(\mathcal{E}_i) - r_e\right)\frac{\partial d(\mathcal{E}_i)}{\partial \mathcal{E}_i}, & d(\mathcal{E}_i) \le r_e \\ 0, & d(\mathcal{E}_i) > r_e \end{cases}$$

where the gradient of the distance $d(\mathcal{E}_i)$ with respect to the control point $\mathcal{E}_i$ can be obtained in the ESDF map [34] directly.

REFERENCES

[1] J. Lin, Y. Wang, Z. Miao, H. Wang, and R. Fierro, "Robust image-based landing control of a quadrotor on an unpredictable moving vehicle using circle features," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1429–1440, 2023.

[2] Q. Cao, Z. Liu, H. Yu, X. Liang, and Y. Fang, "Autonomous landing of the quadrotor on the mobile platform via meta reinforcement learning," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 2269–2280, 2025.

[3] Y. Wu, Y. Li, and J. Dong, "A nonlinear trajectory tracking approach of double-pendulum aerial transportation systems: Transient performance improvement and saturated inputs," *IEEE Transactions on Automation Science and Engineering*, 2025. DOI:10.1109/TASE.2025.3604026.

[4] X. Peng, Z. Zhang, A. Zhang, and K.-V. Yuen, "Antisaturation backstepping control for quadrotor slung load system with fixed-time prescribed performance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 60, no. 4, pp. 4170–4181, 2024.

[5] Y. Chai, Z. Zhang, H. Yu, J. Han, Y. Fang, and X. Liang, "A trajectory planning scheme for collaborative aerial transportation systems by graph-based searching and cable tension optimization," *IEEE/ASME Transactions on Mechatronics*, 2025. DOI:10.1109/TMECH.2025.3588341.

[6] Z. Liu and Y. Fang, "A novel dmps framework for robot skill generalizing with obstacle avoidance: Taking volume and orientation into consideration," *IEEE/ASME Transactions on Mechatronics*, pp. 1–11, 2025. DOI:10.1109/TMECH.2024.3524207.

[7] W. Wang, C. Zeng, H. Zhan, and C. Yang, "A novel robust imitation learning framework for complex skills with limited demonstrations," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 3947–3959, 2025.

[8] M. Wang, Z. Chen, K. Guo, X. Yu, Y. Zhang, L. Guo, and W. Wang, "Millimeter-level pick and peg-in-hole task achieved by aerial manipulator," *IEEE Transactions on Robotics*, vol. 40, pp. 1242–1260, 2024.

[9] K. Cai, H. Yu, Z. Zhang, X. Liang, Y. Fang, and J. Han, "An experiment study for unmanned aerial manipulator systems with l1 adaptive augmentation of geometric control," *Control Engineering Practice*, vol. 164, p. 106418, 2025.

[10] Z. Li, H. Li, Q. Xu, X. Yu, and M. V. Basin, "Coupling disturbance modeling and compensation for aerial manipulator in highly dynamic motion," *IEEE Transactions on Cybernetics*, vol. 55, no. 1, pp. 124–135, 2025.

[11] K. Bodie, M. Brunner, M. Pantic, S. Walser, P. Pfändler, U. Angst, R. Siegwart, and J. Nieto, "Active interaction force control for contact-based inspection with a fully actuated aerial vehicle," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 709–722, 2021.

[12] D. Lee, H. Seo, D. Kim, and H. J. Kim, "Aerial manipulation using model predictive control for opening a hinged door," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1237–1242, IEEE, 2020.

[13] A. González-Morgado, E. Cuniato, M. Tognon, G. Heredia, R. Siegwart, and A. Ollero, "Controlled shaking of trees with an aerial manipulator," *IEEE/ASME Transactions on Mechatronics*, 2024. doi:10.1109/TMECH.2024.3410167.

[14] H. Kim, H. Seo, J. Kim, and H. J. Kim, "Sampling-based motion planning for aerial pick-and-place," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7402–7408, IEEE, 2019.

[15] H. Seo, S. Kim, and H. J. Kim, "Locally optimal trajectory planning for aerial manipulation in constrained environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1719–1724, IEEE, 2017.

[16] H. Cao, J. Shen, C. Liu, B. Zhu, and S. Zhao, "Motion planning for aerial pick-and-place with geometric feasibility constraints," *IEEE Transactions on Automation Science and Engineering*, 2024.

[17] A. Caballero, A. Suarez, F. Real, V. M. Vega, M. Bejar, A. Rodriguez-Castaño, and A. Ollero, "First experimental results on motion planning for transportation in aerial long-reach manipulators with two arms," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8471–8477, IEEE, 2018.

[18] W. Deng, H. Chen, B. Ye, H. Chen, and X. Lyu, "Whole-body integrated motion planning for aerial manipulators," *arXiv preprint arXiv:2501.06493*, 2025.

[19] Z. Zhang, H. Yu, Y. Chai, Z. Yang, X. Liang, Y. Fang, and J. Han, "An end-effector-oriented coupled motion planning method for aerial manipulators in constrained environments," *IEEE/ASME Transactions on Mechatronics*, 2025. DOI:10.1109/TMECH.2025.3550562.

[20] D. Lee, B. Kim, and H. J. Kim, "Autonomous aerial manipulation at arbitrary pose in se(3) with robust control and whole-body planning," *arXiv preprint arXiv:2508.19608*, 2025.

[21] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, present, and future of aerial robotic manipulators," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 626–645, 2021.

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

[23] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[24] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[25] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se(3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, 2018.

[26] Z. Han, Z. Wang, N. Pan, Y. Lin, C. Xu, and F. Gao, "Fast-racing: An open-source strong baseline for se(3) planning in autonomous drone racing," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8631–8638, 2021.

[27] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "Faster: Fast and safe trajectory planner for navigation in unknown environments," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 922–938, 2021.

[28] G. Zhang, Y. He, B. Dai, F. Gu, L. Yang, J. Han, G. Liu, and J. Qi, "Grasp a moving target from the air: System & control of an aerial manipulator," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1681–1687, IEEE, 2018.

[29] M. Tognon, E. Cataldi, H. A. T. Chavez, G. Antonelli, J. Cortés, and A. Franchi, "Control-aware motion planning for task-constrained aerial manipulation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2478–2484, 2018.

[30] W. Luo, J. Chen, H. Ebel, and P. Eberhard, "Time-optimal handover trajectory planning for aerial manipulators based on discrete mechanics and complementarity constraints," *IEEE Transactions on Robotics*, vol. 39, no. 6, pp. 4332–4349, 2023.

[31] J. Welde, J. Paulos, and V. Kumar, "Dynamically feasible task space planning for underactuated aerial manipulators," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3232–3239, 2021.

[32] M. Watterson and V. Kumar, "Control of quadrotors using the hopf fibration on so(3)," in *Proceedings of the Robotics Research: The 18th International Symposium (ISRR)*, pp. 199–215, Springer, 2020.

[33] C. De Boor, *A practical guide to splines*, vol. 27. Springer New York, 1978.

[34] L. Han, F. Gao, B. Zhou, and S. Shen, "Fiesta: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4423–4430, 2019.

[35] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.