Adaptive Tracking and Perching for Quadrotor in Dynamic Scenarios

Yuman Gao ⓘ, Jialin Ji ⓘ, Qianhao Wang ⓘ, *Student Member, IEEE*, Rui Jin ⓘ, Yi Lin ⓘ, Zhimeng Shang ⓘ, Yanjun Cao ⓘ, Shaojie Shen ⓘ, Chao Xu ⓘ, *Senior Member, IEEE*, and Fei Gao ⓘ, *Member, IEEE*

Abstract—**Perching on the moving platforms is a promising solution to enhance the endurance and operational range of quadrotors, which could benefit the efficiency of a variety of air ground cooperative tasks. To ensure robust perching, tracking with a steady relative state and reliable perception is a prerequisite. This paper presents an adaptive dynamic tracking and perching scheme for autonomous quadrotors to achieve tight integration with moving platforms. For reliable perception of dynamic targets, we introduce elastic visibility aware planning to actively avoid occlusion and target loss. Additionally, we propose a flexible terminal adjustment method that adapts the changes in flight duration and the coupled terminal states, ensuring full state synchronization with the time varying perching surface at various angles. A relaxation strategy is developed by optimizing the tangential relative speed to address the dynamics and safety violations brought by hard boundary conditions. Moreover, we take SE(3) motion planning into account to ensure no collision until the contact moment. Furthermore, we propose an efficient spatiotemporal trajectory optimization framework considering full state dynamics The proposed method is extensively tested through benchmark comparisons and ablation studies. To facilitate the application of academic research to industry and to validate the efficiency under strictly limited computational resources, we deploy our system on a commercial drone (DJI MAVIC3) with a full size sport utility vehicle (SUV). We conduct extensive real world experiments, where the drone successfully tracks and perches at 30 km/h (8.3 m/ s) on the top of the SUV, and at 3.5∼m/s with 60° inclined into the trunk of the SUV.**

Index Terms—**Aerial system, motion and path planning, perception and autonomy, trajectory optimization.**

I. INTRODUCTION

UNCREWED aerial vehicles (UAVs) that operate in conjunction with mobile platforms have recently emerged in a variety of applications, such as truck-drone delivery systems [1], [2], [3], car-drone inspection systems [4], [5], [6], and air-ground search and rescue systems [7], [8], [9]. In these applications, due to the short flight duration of UAVs, they need to frequently perch on the mobile platform for resting or recharging, which necessitates the platform to come to a complete stop or interrupt its ongoing operations. For higher automation and task efficiency, it is necessary to let the UAV smoothly attach to moving platforms without interruption. This demands the UAV to track the moving and various angled platforms with steady relative state and reliable perception, and perch on it at the proper moment. For an ideal planner promising successful tracking and perching applied in dynamic real-world scenarios, *adaptability* must be explicitly considered and modeled to cope with several practical challenges analyzed as follows.

The first challenge is how to maintain stable observations during agile flight. For a quadrotor chasing a high-speed ground vehicle, the noncooperative movements of these two agents result in fluctuating relative motions. Such relative motion easily makes the target out of the limited field of view (FoV) of the drone's sensory device, causing target loss. Moreover, irregular obstacle distributions or dynamic objects often block direct observations from the quadrotor to its target, producing occlusion. To avoid the above situations, the trajectory of the drone should be adjusted adaptively according to the target movement and the surrounding environments, summarized as *visibility* requirement.

The second challenge is how to smoothly and quickly attach to the dynamic platform. A desired perching requires the quadrotor to synchronize its position, velocity, and attitude adaptively with the perching surface at a proper contact moment. Since a quadrotor's attitude and motion are coupled, a time-varying terminal state introduces cross-dependent temporal and spatial conditions for the quadrotor's trajectory. To cope with such conditions, the quadrotor should flexibly adjust its flight duration and the coupled terminal states during trajectory generation. Furthermore, strict terminal constraints may contain conflicts

with the quadrotor's safety and dynamics, resulting in no feasible solution. Therefore, a relaxation approach is hoped for reasonably adjusting the final perching state. The above requirements are summarized as *flexibility*.

The third challenge is how to react to sudden changes in complicated situations. In a dynamic scenario without global information, obstacles that appear suddenly, target states that change quickly, and external disturbances that act severely would make the latest generated trajectory deprecated. To let the quadrotor react adaptively in time, high-frequency replanning based on the onboard perception is necessary. However, for a quadrotor with limited onboard resources, it is difficult to satisfy the planning efficiency, the modeling fidelity, and the solving completeness at the same time. Therefore, the *responsiveness* is another critical requirement for planning.

Apart from the above requirements, *safety* and *dynamic feasibility* are also fundamental considerations for conducting an ideal aerial tracking and perching. The generated trajectory should avoid any possible collisions with obstacles, and can only contact the target untill the final moment. Moreover, perching maneuvers with large attitudes often push the actuators toward their physical limits, making dynamic feasibility essential. However, constraints on high-order dynamic states usually demand high computation.

Based on the above observations and analysis, and built upon our previous conference papers [10], [11], we propose a complete aerial system with an adaptive tracking-perching scheme for dynamic targets. In our proposed system, to actively enhance *visibility*, we design a series of differentiable metrics considering the occlusion of obstacles, relative distance, and angle, with the limited 3-D FoV of quadrotors. Based on the surrounding environment, the tracking distance can be elastically adjusted. Moreover, to maintain high-quality observation, the quadrotor's position and attitude are jointly adjusted to lock the target centrally in the image space while perching. Second, to provide *flexibility* in determining the flight duration and the coupled terminal states, we propose a flexible terminal adjustment approach to ensure full-state synchronization with the time-varying perching surface. This approach eliminates the terminal constraints along with reducing optimization variables. Additionally, a relaxation strategy is developed by optimizing the tangential relative speed, addressing the conflict between terminal restrictions, safety, and dynamic feasibility. Third, to ensure *safety* around the contact moment precisely, we geometrically model the quadrotor and platform to prevent collision, and further construct a concise geometric constraint. Furthermore, *dynamic feasibility* is guaranteed by constrained on high-order states, including angular velocity and thrust, with an efficient flatness mapping. To collectively involve the above aspects in trajectory generation, we propose an efficient spatiotemporal trajectory optimization framework. Concise metrics and compact trajectory representation benefit the solving efficiency, ensuring *responsiveness*. Finally, to facilitate the application of academic research to industry and to validate our system in the real world, we deploy our adaptive tracking and perching scheme on a commercial drone (DJI-MAVIC 3) with a sport-utility vehicle (SUV) as the moving platform. Efficient trajectory optimization

Fig. 1. Simulations and real-world experiments of our adaptive tracking and perching system. Please watch our attached videos for more information at: youtu.be/5XKm7qkp2Xs, youtu.be/fBwW93Zq9ss.

enables high-frequency replanning even in embedded processors with severely limited resources. We present experiments in a variety of dynamic scenarios partly shown in Fig. 1, including successful tracking and perching on the SUV with a speed up to 30 km/h (8.3 m/s). Contributions of this article are listed as follows.

1) A series of differentiable planning metrics that enable visibility awareness against occlusion and target loss in aggressive flight efficiently.
2) A concise approach that flexibly adjusts the terminal perching states with safety and feasibility guaranteed by terminal constraint relaxation.
3) An efficient trajectory optimization framework considering full state dynamics and complex collision constraints for tracking and perching.
4) A variety of simulations and real-world tests that validate the proposed methods with a commercialized quadrotor.

This article consolidates the preliminary conference papers presented in [10] and [11] with significant functionality extension, performance improvements, as well as application promotion. Compared to our previous work [10], which relies on external facilities for perception, we build a fully autonomous tracking and perching system with solely onboard sensors. To better exploit the limited sensing capability of onboard vision,

we propose a visibility-aware planner to deform the trajectory for improving the target's observation quality. Besides, our previous tracking method [11] employed a decoupled way to generate a trajectory, where the temporal profile is optimized after its spatial shape. In this article, we jointly optimize the trajectory in space and time and result in much higher optimality. Finally, by deploying our method in a commercial quadrotor and testing it with a full-sized vehicle, we demonstrate that our system can be successfully deployed outside the laboratory. We consider this as an important step forward from our previous works, which are only validated in human-controlled environments.

II. RELATED WORKS

A. Aerial Tracking System

Several previous vision-based tracking works [12], [13] formulate the trajectory planning and control integrally as a reactive control problem, and take the tracking error defined on image space as the feedback. These reactive methods can achieve real-time performance but are too shortsighted to consider safety and occlusion constraints. To overcome these drawbacks, model predictive control (MPC) planners [14], [15] and differential-flatness-based planners [16], [17], [18], [19], [20], [21] with receding horizon formulations emerge. Nageli et al.[14] design a modular visibility cost function based on the reprojection error of targets, and integrate it into an MPC planner. However, this method has a strong assumption that the obstacles are elliptical and thus cannot avoid arbitrarily shaped obstacles well. Penin et al. [15] formulate a nonlinear MPC optimization problem with a visibility term in image space. Lacking environmental perception, this approach requires complete knowledge of the environment. Optimizing trajectories in flat-output space, Bonatti et al. [17] tradeoff shot smoothness, safety, occlusion, and cinematography guidelines. Nevertheless, they avoid occlusions by making the connection line between the target and robot obstacle-free, neglecting the conical FoV shape of sensors. Han et al. [18] propose a safe tracking trajectory planner consisting of a kinodynamic searching front-end and a spatiotemporal optimal trajectory planning back-end. Without visibility considerations, it is prone to fail due to the target being out of the FoV or too close in relative distance. The authors of [19] and [20] propose a graph-search-based preplanning procedure to cover safety and visibility, but carry out path smoothing afterward. Such inconsistency makes the trajectories may not meet all the constraints. Ji et al. [21] use a series of 2-D fans to represent the visible region for tracking planning. But the fans are only generated at the specified height, making this method pseudo-3-D. These recent works address that visibility is significant in determining tracking success rate and robustness. In contrast with the above works, our proposed differentiable metrics comprehensively consider the factors that affect visibility, including the 3-D shape of the conical FoV, the relative observation distance and angle, and obstacle occlusion.

B. Dynamic Landing and Perching

For clarity, we refer to the methods that can merely adjust the end position and velocity as "landing," and methods that can additionally adjust the end attitude as "perching." Dynamic landing on a moving platform has been widely studied in UAV research. Previous studies typically adopt control-level methods, which directly construct a state deviation term based on target observation. Controllers designed with the proportional-integral-derivative method or Lyapunov-function-based methods are widely adopted [22], [23], [24]. Besides, the image-based visual servoing (IBVS) method [25], [26], [27] is another group of control-level methods that define the visual error in image space with less computational complexity. However, the above methods are often shortsighted and face difficulties in introducing state constraints related to interacting objects. To compensate for these drawbacks, most of the recent works address dynamic landing issues with MPC [28], [29], [30] or differential-flatness-based trajectory planning approaches [31], [32], both with a receding planning horizon. However, all these above studies only aim to reach the landing position, lacking the ability to adjust desired attitudes at the contact moment. Furthermore, the coordinated landing of quadrotors with moving platforms is also widely studied [33], [34], where the joint planning and control are designed to accomplish rendezvous and landing. These methods require control of the platform, introducing a higher system complexity, and have limited application scenarios.

Many previous perching works [35], [36], [37], [38], [39], [40], [41] study the problem of perching on stationary inclined surfaces. Most of these works [35], [36], [37], [38] rely on particular mechanisms, such as suction or adhesive grippers, to bypass the terminal attitude requirement. Thomas et al. [39] propose a planning and control strategy fully considering actuator constraints and formulates a quadratic programming (QP) problem using a series of linear approximations. However, this approach cannot adjust the perching duration, and the linearization is oversimplified. Based on this approach, Mao et al. [40] propose a global-bound-checking method to check the dynamic feasibility efficiently, then increase the trajectory duration and recursively solve a QP problem until actuator constraints are satisfied. However, the method ignores that the spatial profile of trajectories also affects dynamic feasibility, which can result in this constraint being unsatisfied. Moreover, Paneque et al. [41] formulate a discrete-time multiple-shooting nonlinear programming (NLP) problem for perching on powerlines, with a task-specific perception-aware term. Nevertheless, this method adjusts terminal states by computationally expensive multiple shooting, resulting in an excessive computation time for orders of magnitude longer than ours.

For dynamic perching, Vlantis et al. [42] study the problem of landing a quadrotor on an inclined moving platform. By solving a discrete time nonlinear MPC, the drone approaches the platform, while maintaining it within the camera's FoV, and finally perches on it. However, such a computationally demanding problem is computed on a ground station. Moreover, the quite small inclined angle (27°) and quite slow speed (0.5 m/s) are inadequate for validation. Hu and Mishra [43] achieve the differentially flat spatiotemporal optimization considering the actuator constraints and collision constraints. But it is only for 3-DOF quadrotors in the 2-D coordinate system. Liu et al. [44] design a suction
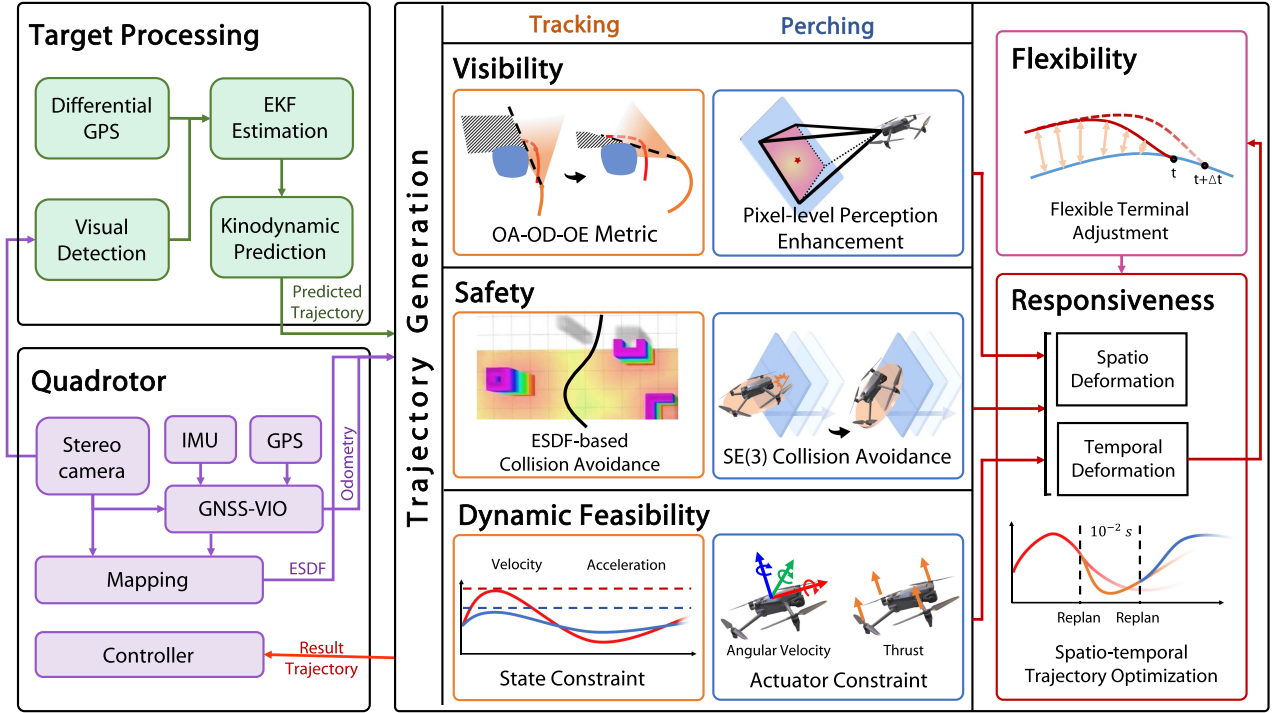
Fig. 2.   Overview of our complete aerial system with dynamic tracking and perching scheme. The trajectory generation module takes all the requirements mentioned in Section I into account and provides spatiotemporal optimal feasible trajectory for stable tracking and dynamic perching.

cup gripper for quadrotors to perch on a moving target, but this work focus more on the novel mechanical design and simply use minimum-jerk-based trajectory generation without safety, actuator limit, and visibility considerations.

III. SYSTEM OVERVIEW AND PRELIMINARIES

A. System Architecture

The overall architecture of our aerial system with the adaptive tracking and perching scheme is illustrated in Fig. 2. The target position is obtained by the fusion of coarser differential-GPS-based relative localization (meter-level error) and finer visual detection (centimeter-level error). After obtaining the target position, we conduct an extended Kalman filter (EKF) based estimation and predict the target trajectory in the future (Section III-C). To meet the five aspects of requirements stated in Section I, we design comprehensive constraints of visibility, safety, and dynamic feasibility, respectively, for tracking (Section IV) and perching (Section V). Collectively considering these constraints, we simultaneously optimize the spatial and temporal profile of the trajectory with high efficiency (Section VI). When trajectory duration changes in temporal deformation, a flexible terminal adjustment approach is adopted to adaptively synchronize the full states of the quadrotor with the time-varying states of the perching surface (Section V-D).

B. Dynamic Model and Differential Flatness

In this article, we use the simplified dynamics proposed by [45] for a quadrotor, whose configuration is defined by its translation $\mathbf{p} = (p_x, p_y, p_z)^T \in \mathbb{R}^3$ and rotation $\mathbf{R} \in \mathrm{SO}(3)$. Translational motion depends on the gravitational acceleration $\bar{g}$ as well as the thrust $\tilde{f}$. Rotational motion takes the body rate $\boldsymbol{\omega} \in \mathbb{R}^3$ as input. The simplified model is written as

$$\begin{cases} \boldsymbol{\tau} = \tilde{f}\mathbf{R}\mathbf{e}_3/m \\ \ddot{\mathbf{p}} = \boldsymbol{\tau} - \bar{g}\mathbf{e}_3 \\ \dot{\mathbf{R}} = \mathbf{R}\hat{\boldsymbol{\omega}} \end{cases} \tag{1}$$

where $\boldsymbol{\tau}$ denotes the mass-normalized net thrust, $\mathbf{e}_i$ is the ith column of $\mathbf{I}_3$, such as $\mathbf{e}_3 = [0, 0, 1]^T$, and $\hat{\cdot}$ is the skew-symmetric matrix form of the vector cross product.

Exploiting the differential flatness property of the quadrotor, the state and input variables of quadrotors can be parameterized by finite derivatives of flat outputs [45]. The flat output of quadrotors is

$$\mathbf{z} = (p_x, p_y, p_z, \psi)^T \in \mathbb{R}^3 \times \mathrm{SO}(2) \tag{2}$$

where $\psi \in SO(2)$ is the Euler-yaw angle. We further define the flat outputs and their derivatives $\mathbf{z}^{[s-1]} \in \mathbb{R}^{ms}$ as

$$\mathbf{z}^{[s-1]} := (\mathbf{z}^T, \dot{\mathbf{z}}^T, \ldots, \mathbf{z}^{(s-1)T})^T. \tag{3}$$

This makes it possible for us to optimize a trajectory $\mathbf{z}(t): [0, T] \mapsto \mathbb{R}^m$ in the low-dimension flat-output space.

C. Target Prediction Model

Given measurements on the target position $(\varrho_x, \varrho_y, \varrho_z)$, we aim to estimate the full state of the target vehicle. Estimating high-order states will amplify the observation error from the position measurement noise. Therefore, we adopt the constant
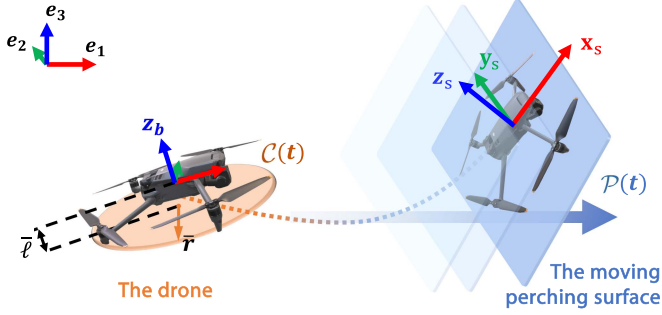
Fig. 3. Illustration of the geometry model. The quadrotor is modeled as a disc $\mathcal{C}$, and the perching surface is modeled as a half-space $\mathcal{P}$.

turn rate and velocity (CTRV) model [46] for state estimation, and the states of the target vehicle are written as

$$\boldsymbol{\chi}(t) = (\varrho_x, \varrho_y, \varrho_z, \theta, v_h, v_v, \omega_\varrho)^T \tag{4}$$

where θ is the heading angle of the vehicle, ω_ϱ is the angular velocity, v_h is the horizontal velocity, and v_v is the vertical velocity, which is usually small and used to estimate uphill and downhill motion. Then we use EKF with this model for the state estimation.

To combine environmental information for target prediction, given the estimated current states, we use a kinodynamic motion primitive method considering collision. Given a prediction duration, by expanding motion primitives using the constant turn rate and acceleration (CTRA) model [46] under different control inputs with a time step Δt, we choose the collision-free one with the least control effort. Expanding until the prediction duration is reached, we obtain the predicted trajectory $\boldsymbol{\varrho}(t) \in \mathbb{R}^3$ with yaw angle $\theta(t) \in SO(2)$ and a series of corresponding discrete target positions denoted as

$$\Psi = \{\boldsymbol{\varrho}_k = \boldsymbol{\varrho}(k\Delta t) \mid k = [0, 1, \ldots, M]\} . \tag{5}$$

D. Geometric Model

To avoid collision with the perching surface, we model the underside of a symmetric quadrotor as a disc, shown in Fig. 3, denoted by

$$\mathcal{C} = \left\{\mathbf{x} = \mathbf{R}\mathbf{B}\bar{r}\mathbf{u} + \mathbf{o} \,\middle|\, \|\mathbf{u}\| \leq 1, \mathbf{u} \in \mathbb{R}_2, \mathbf{x} \in \mathbb{R}_3\right\} \tag{6}$$

where $\mathbf{B} = (\mathbf{e}_1, \mathbf{e}_2) \in \mathbb{R}^{3 \times 2}$ and $\bar{r}$ denotes the radius of the disc. We use $\bar{\ell}$ to denote the thickness of the centroid of the quadrotor to the bottom. Thus, the center of the disc $\mathbf{o} = \mathbf{p} - \bar{\ell}\mathbf{z}_b$, where $\mathbf{z}_b$ is the normal vector corresponding to the z-axis of the body frame.

We assume that the estimated position and yaw angle of the moving platform is denoted as $\boldsymbol{\varrho}$ and θ, and the normal vectors corresponding to the perching surface frame are denoted by $\mathbf{x}_s, \mathbf{y}_s$, and $\mathbf{z}_s$. Therefore, the feasible half-space divided by the perching surface is written as

$$\mathcal{P} = \left\{\mathbf{h}^T\mathbf{x} \leq b \,\middle|\, \mathbf{x} \in \mathbb{R}_3\right\} \tag{7}$$

where $\mathbf{h} = -\mathbf{z}_s$, $b = \mathbf{h}^T\boldsymbol{\varrho}$, and $\mathbf{z}_s = \mathbf{R}_z(\theta)\bar{\mathbf{z}}_s$, where $\mathbf{R}_z(\theta) \in SO(3)$ represents the basic rotation by angle θ around ego z-axis, and $\bar{\mathbf{z}}_s$ is the current normal vector obtained by detection. The other normal vectors can be similarly calculated.

IV. VISIBILITY-AWARE TRACKING PLANNING

As analyzed in Section I, planning with visibility consideration is essential for conducting long-term stable target tracking. To this end, we define several metrics to comprehensively model visibility in this section. Observing typical tracking failures, we can summarize that they are mainly due to observation distance (OD), observation angle (OA), and occlusion effect (OE). In this section, we explicitly model the above OD, OA, and OE metrics, design their penalty functions, and derive corresponding gradients for trajectory optimization. In what follows, the drone's position and yaw angle are denoted as $\mathbf{p} \in \mathbb{R}^3$ and $\psi \in SO(2)$, with the target position $\boldsymbol{\varrho} \in \mathbb{R}^3$.

A. Observation Distance Restriction

A target is expected to be observed within a proper distance range from a drone. Since target ground platforms usually have more aggressive horizontal motion, we separately constrain the horizontal (δ^h) and vertical (δ^v) components of the tracking distance with different margins

$$d_\ell^* \leq \delta^* \leq d_u^*, * = \{h, v\} \tag{8}$$

where the d_ℓ and d_u are the lower and upper bounds of the optimal distance of observation. To make this metric analytically differentiable, we design penalty functions for the OD constraint

$$\mathcal{J}_{OD^v} = g(d_\ell^v - \delta^v) + g(\delta^v - d_u^v) \tag{9a}$$

$$\mathcal{J}_{OD^h} = g\left(d_\ell^h - \delta^h\right) + \mathcal{L}_\mu\left(\delta^h - d_u^h\right) \tag{9b}$$

where $g(x) = \max(x, 0)^3$ and $\mathcal{L}_\mu(\cdot)$ is a C^2-smoothing linear penalty function denoted by

$$\mathcal{L}_\mu(x) = \begin{cases} 0, & x \leq 0 \\ (\mu - x/2)(x/\mu)^3, & 0 < x \leq \mu \\ x - \mu/2, & x > \mu. \end{cases} \tag{10}$$

When the horizontal distance surpasses the upper bound μ, we apply a linearly increasing penalty function $L_\mu(\cdot)$. We choose a more severe penalty $g(\cdot)$ to prevent the drone from getting the target too close and possibly causing a collision.

B. Observation Angle Restriction

In order to keep the observation angle straight toward the target, the expected yaw angle ψ_e is defined as

$$\psi_e(\mathbf{p}, \boldsymbol{\varrho}) = \text{atan2}\left(\mathbf{e}_2^T(\boldsymbol{\varrho} - \mathbf{p}), \mathbf{e}_1^T(\boldsymbol{\varrho} - \mathbf{p})\right) \tag{11}$$

where $\mathbf{e}_i$ is the ith column of $\mathbf{I}_3$. The cost of this term is written as
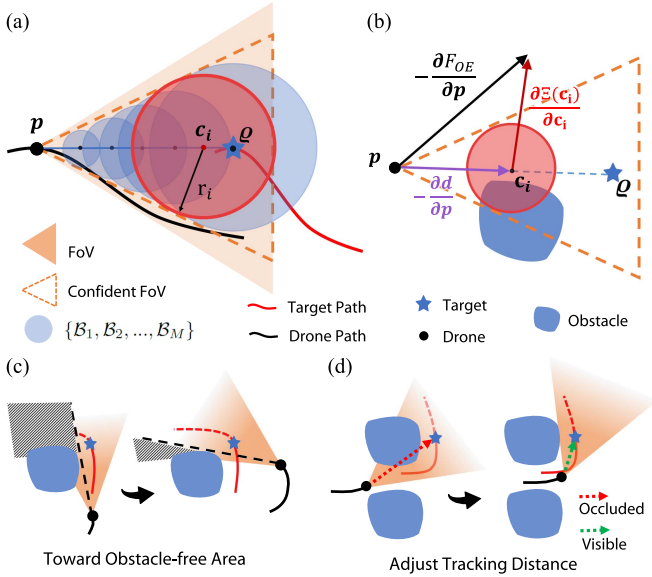
$$\mathcal{J}_{OA} = (\psi - \psi_e)^2. \tag{12}$$

Fig. 4. (a) Illustration of the OE metric. The drone at position $\mathbf{p}$ observes the target ϱ. A sequence of ball-shaped areas in blue are used to approximate the confident FoV expected to be obstacle-free. (b) Example of the two decomposition parts of the negative F_{OE}'s gradient, driving the drone to move toward obstacle-free areas and adjust the tracking distance elastically. (c) Drone moves toward obstacle-free areas for less occlusion probability. (d) Drone adjusts the tracking distance in obstacle-rich areas for less occlusion probability.

Note that the drone could adjust both its position and yaw angle to reach ψ_e. The gradient of $\mathcal{J}_{OA}$ affects both $\mathbf{p}$ and ψ. Here, we present the gradient w.r.t. $\mathbf{p}$ written as [1]

$$\frac{\partial \mathcal{J}_{OA}}{\partial \mathbf{p}} = \left[\frac{\partial \mathcal{J}_{OA}}{\partial p_x}, \frac{\partial \mathcal{J}_{OA}}{\partial p_y}, 0\right]^T \qquad (13)$$

where

$$\frac{\partial \mathcal{J}_{OA}}{\partial p_x} = \frac{2(\psi - \psi_e)}{(e_1^T(\mathbf{p} - \varrho))^2 + (e_2^T(\mathbf{p} - \varrho))^2} \cdot e_2^T(\mathbf{p} - \varrho). \quad (14)$$

$\partial \mathcal{J}_{OA}/\partial p_y$ can be calculated similarly.

C. Elastic Occlusion Effect Avoidance

Complex environments increase the possibility of occlusion during tracking, easily causing target loss. Considering the conical FoV and arbitrarily shaped obstacles, we formulate the OE metric as follows. As shown in Fig. 4(a), the *confident FoV* embedded in the real FoV represents the region that is expected TO BE obstacle-free around the line of sight. To formulate analytically, we approximate the confident FoV with a sequence of ball-shaped areas $\{\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_M\}$, where M is the number of the areas. For each ball, its center $\mathbf{c}_i$ and radius r_i are calculated by

$$\mathbf{c}_i = \mathbf{p} + \lambda_i(\varrho - \mathbf{p}) \qquad (15)$$

$$r_i = \rho \cdot \lambda_i \cdot \|\mathbf{p} - \varrho\| \qquad (16)$$

where $\lambda_i = i/M \in [0, 1]$, and ρ is a constant determined by the size of the confident FoV. Then we guarantee OE analytically

---

[1]Note that we adopt the denominator layout for gradients in this article.

by forcing constraint

$$r_i < \Xi(\mathbf{c}_i) \qquad (17)$$

for each ball-shaped area, where $\Xi(\mathbf{c}_i) : \mathbb{R}^3 \to \mathbb{R}$ is the distance between $\mathbf{c}_i$ and its closest obstacle, which is obtained from Euclidean signed distance field (ESDF). Then, the OE cost is concisely written as

$$\mathcal{J}_{OE} = \sum_{i=1}^{M} g(F_{OE}(\mathbf{p})) \qquad (18a)$$

$$F_{OE}(\mathbf{p}) = r_i - \Xi(\mathbf{c}_i). \qquad (18b)$$

The gradient of F_{OE} can be written as

$$\frac{\partial F_{OE}}{\partial \mathbf{p}} = \rho\lambda_i \cdot \frac{\partial d}{\partial \mathbf{p}} - (1 - \lambda_i)\frac{\partial \Xi(\mathbf{c}_i)}{\partial \mathbf{c}_i} \qquad (19)$$

where $d = \|\mathbf{p} - \varrho\|$, and $\partial \Xi(\mathbf{c}_i)/\partial \mathbf{c}_i$ is obtained from ESDF.

To further analyze the metric, an example of the gradient of F_{OE} is presented in Fig. 4(b). As the two directions of the decomposition parts of $-\partial F_{OE}/\partial \mathbf{p}$ shown, the OE metric can not only make the drone move toward obstacle-free areas [see Fig. 4(c)] but also adjust the tracking distance elastically according to the environment [see Fig. 4(d)]. Both of the two actions can prevent occlusion, especially when a target moves in obstacle-rich regions shown in Fig. 4(d), with a small positional adjustment margin, getting closer to the target is a reasonable way. Such characteristic is further demonstrated in simulation experiments in Section VII-B.

V. AGGRESSIVE AND FLEXIBLE PERCHING PLANNING

To smoothly attach to the perching surface, it is essential to get a full-state alignment at the contact moment, with safety, dynamic feasibility, and continuous target observation guaranteed. To this end, in this section, we detail the specific constraints involving the above factors for aggressive and flexible perching planning. In what follows, the position trajectory of the drone and the target are denoted as $\mathbf{p} \in \mathbb{R}^3$ and $\varrho \in \mathbb{R}^3$, respectively. The attitude quaternion of the quadrotor is denoted as $\mathbf{q} \in \mathcal{S}^3$.

A. SE(3) Collision Avoidance

Considering the geometry of a quadrotor, adjusting its attitude is critical for collision avoidance with the perching surface, necessitating SE(3) trajectory planning. Given the description of $\mathcal{C}$ (6) and $\mathcal{P}$ (7), to avoid the quadrotor having intersection of the surface, the SE(3) collision avoidance constraint is written as

$$\mathcal{C} \subset \mathcal{P} \qquad (20)$$

which is equivalent to

$$\mathbf{h}^T(\mathbf{R}\mathbf{B}\bar{r}\mathbf{u} + \mathbf{o}) - b \leq 0 \qquad (21a)$$

$$\sup_{\|\mathbf{u}\| \leq 1} \mathbf{h}^T(\mathbf{R}\mathbf{B}\bar{r}\mathbf{u}) + \mathbf{h}^T\mathbf{o} - b \leq 0 \qquad (21b)$$

$$\bar{r}\|\mathbf{B}^T\mathbf{R}^T\mathbf{h}\| + \mathbf{h}^T\mathbf{o} - b \leq 0. \qquad (21c)$$

Since we are utilizing the flat output $\mathbf{z}$ as the state representation, we should first recover $\mathbf{R}$ from $\mathbf{z}$ before computing the constraint

(21c). First, the mass-normalized net thrust can be directly obtained from the trajectory

$$\boldsymbol{\tau} = \mathbf{p}^{(2)} + \bar{g}\mathbf{e}_3. \tag{22}$$

Then, since the z-axis of the body frame $\mathbf{z}_b$ is aligned with the direction of the mass-normalized net thrust according to the simplified dynamics [45], we can obtain

$$\mathbf{z}_b = \boldsymbol{\tau}/\|\boldsymbol{\tau}\|_2. \tag{23}$$

To recover $\mathbf{R}$ from $\mathbf{z}_b$, denoting that $\mathbf{z}_b = [a_z, b_z, c_z]^T$, we utilize an efficient mapping according to Hopf fibration [47], which is written as

$$\mathbf{q}_{abc} = \frac{1}{\sqrt{2(c_z + 1)}} \begin{pmatrix} c_z + 1 \\ -b_z \\ a_z \\ 0 \end{pmatrix} \tag{24}$$

where the unit quaternion $\mathbf{q}_{abc}$ satisfies $\mathbf{R}(\mathbf{q}_{abc})\mathbf{e}_3 = \mathbf{z}_b$. Since this collision avoidance metric is independent of the yaw angle ψ, the rotation of the quadrotor can be obtained by $\mathbf{R} = \mathbf{R}(\mathbf{q}_{abc})$. Adopting this differential flatness with Hopf fibration can reduce computation and reduce the singularities to the best case possible. Finally, we can easily calculate $\mathbf{B}^T\mathbf{R}^T$ in (21c) through $\mathbf{z}_b$

$$\mathbf{B}^T\mathbf{R}^T = \begin{pmatrix} 1 - \frac{a_z^2}{c_z + 1} & -\frac{a_z b_z}{c_z + 1} & -a_z \\ -\frac{a_z b_z}{c_z + 1} & 1 - \frac{b_z^2}{c_z + 1} & -b_z \end{pmatrix}. \tag{25}$$

However, the constraint (21c) should only be activated when $\|\mathbf{p} - \boldsymbol{\varrho}\| \le \bar{d}$, that is, the drone is close to the platform. This constitutes a mixed-integer nonlinear programming problem. Here, we design a smoothed logistic function $\mathcal{L}_\epsilon(\cdot)$ to incorporate the integer variable into our NLP, which is denoted by

$$\mathcal{L}_\epsilon(x) = \begin{cases} 0, & x \le -\epsilon \\ \frac{1}{2\epsilon^4}(x + \epsilon)^3(\epsilon - x), & -\epsilon < x \le 0 \\ \frac{1}{2\epsilon^4}(x - \epsilon)^3(\epsilon + x) + 1, & 0 < x \le \epsilon \\ 1, & x > \epsilon \end{cases} \tag{26}$$

where ϵ is a tunable positive parameter. Then the penalty function of collision avoidance can be written as

$$\mathcal{J}_c = \mathcal{L}_\epsilon(\mathcal{F}_{\bar{d}}) \cdot \mathcal{L}_\mu(\mathcal{F}_c) \tag{27}$$

where $\mathcal{F}_{\bar{d}}$ and $\mathcal{F}_c$ are denoted by

$$\mathcal{F}_{\bar{d}} = \|\mathbf{p} - \boldsymbol{\varrho}\|^2 - \bar{d}^2 \tag{28a}$$

$$\mathcal{F}_c = \bar{r}\|\mathbf{B}^T\mathbf{R}^T\mathbf{h}\| + \mathbf{h}^T\mathbf{o} - b. \tag{28b}$$

Additionally, to limit the drone within a safe relative height range and prevent the drone from hitting the ground, we define a relative height cost $\mathcal{J}_{\Delta z}$ written as

$$\mathcal{J}_{\Delta z} = \mathcal{L}_\mu(z_{\min} - \Delta z) + \mathcal{L}_\mu(\Delta z - z_{\max}) \tag{29}$$

where relative height $\Delta z = \mathbf{e}_3^T(\mathbf{p} - \boldsymbol{\varrho})$, $z_{\min}$ and $z_{\max}$ are the expected minimum and maximum relative heights.

B. Image Space Pixel-Level Perception Enhancement

For accurate perching, high-quality observation is a fundamental factor. With a visual tag that remarks the perching
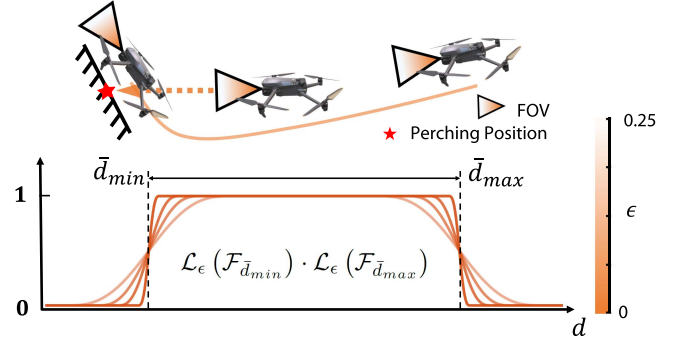

Fig. 5. Illustration of the perception distance range selection function with a front camera. The perception metric is only activated within the setting range $[\bar{d}_{\min}, \bar{d}_{\max}]$. Overly constraining at a close relative distance can lead to violations of terminal attitude constraints.

position, the quadrotor is expected to keep it central in the camera image. Denoting the tag position that coincided with the perching position in the world frame as $^w\boldsymbol{\varrho}$, we first transform it to the camera frame

$$^b\boldsymbol{\varrho} = \mathbf{R}(_w^b\mathbf{q})(^w\boldsymbol{\varrho} - \mathbf{p}) \tag{30a}$$

$$^c\boldsymbol{\varrho} = \mathbf{R}(_b^c\mathbf{q})(^b\boldsymbol{\varrho} - {}_b^c\mathbf{t}) \tag{30b}$$

where we denote index w as world frame, c as camera frame, and b as body frame. $_b^c\mathbf{q}$ and $_b^c\mathbf{t}$ are the rotation quaternion and the translation between the body frame and the camera frame, respectively. $_w^b\mathbf{q}$ is the rotation quaternion between the world frame and the body frame.

Using the pinhole camera model, the tag position in the camera frame, with the center of the image as the origin, can be written as

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} f_x{}^c\boldsymbol{\varrho}_x/{}^c\boldsymbol{\varrho}_z \\ f_y{}^c\boldsymbol{\varrho}_y/{}^c\boldsymbol{\varrho}_z \end{bmatrix} \tag{31}$$

where f_x, f_y are the known camera intrinsics. Excessive constraints at a far distance limit the spatial freedom of trajectories, while overly constraining at a close distance might lead to violations of terminal attitude constraints, as shown in Fig. 5. Therefore, perception constraint should be activated only when the drone is within $[\bar{d}_{\min}, \bar{d}_{\max}]$ distance range from the perching surface, introducing mix-integer programming problem. We still resort to $\mathcal{L}_\epsilon(\cdot)$ defined in (26) to incorporate the integer variable. Then, the penalty function of tag perception is written as

$$\mathcal{J}_p = \mathcal{L}_\epsilon(\mathcal{F}_{\bar{d}_{\min}}) \cdot \mathcal{L}_\epsilon(\mathcal{F}_{\bar{d}_{\max}}) \cdot \mathcal{F}_p \tag{32}$$

where $\mathcal{F}_p$, $\mathcal{F}_{\bar{d}_{\min}}$, and $\mathcal{F}_{\bar{d}_{\max}}$ are denoted by

$$\mathcal{F}_p = u_c^2 + v_c^2 \tag{33a}$$

$$\mathcal{F}_{\bar{d}_{\min}} = \|\mathbf{p} - {}^w\boldsymbol{\varrho}\|^2 - \bar{d}_{\min}^2 \tag{33b}$$

$$\mathcal{F}_{\bar{d}_{\max}} = \bar{d}_{\max}^2 - \|\mathbf{p} - {}^w\boldsymbol{\varrho}\|^2. \tag{33c}$$

As $\mathcal{F}_p$ implies, the drone position $\mathbf{p}$ and attitude $\mathbf{q}$ jointly affect the perception quality. But the coupled attitude and motion of the quadrotor further hinder perception enhancement, as shown in Fig. 6. To address the problem, we provide the gradients of
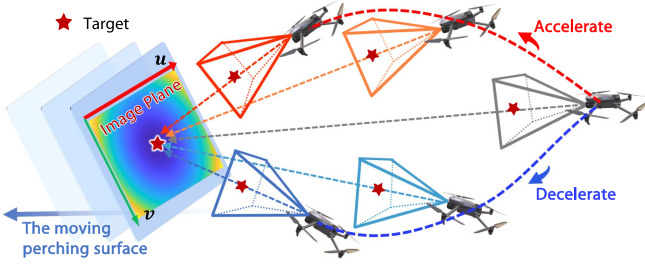
Fig. 6. For perception enhancement, the target is expected to be locked centrally in the image space. Since the motion and view attitude of a quadrotor are coupled, during acceleration or deceleration, the quadrotor needs to jointly adjust its attitude and position as shown to guarantee the perception quality.

$\mathcal{F}_p$ w.r.t. $\mathbf{p}$ and $\mathbf{q}$ here, and optimize the trajectory in quadrotor flat-output space in latter Section VI-B. The gradient w.r.t. $\mathbf{p}$ is written as

$$\frac{\partial \mathcal{F}_p}{\partial \mathbf{p}} = \frac{\partial^c \boldsymbol{\varrho}}{\partial \mathbf{p}} \frac{\partial \mathcal{F}_p}{\partial^c \boldsymbol{\varrho}} \tag{34}$$

where $\frac{\partial^c \boldsymbol{\varrho}}{\partial \mathbf{p}} = -({}_b^c \mathbf{R} \cdot {}_w^b \mathbf{R})^T$. Moreover, the gradient w.r.t. the drone's attitude quaternion can be written as

$$\frac{\partial \mathcal{F}_p}{\partial_b^w \mathbf{q}} = \left(\frac{\partial^b \boldsymbol{\varrho}}{\partial_w^b \mathbf{q}} \frac{\partial^c \boldsymbol{\varrho}}{\partial^b \boldsymbol{\varrho}} \frac{\partial \mathcal{F}_p}{\partial^c \boldsymbol{\varrho}} \right)^{-1} \tag{35}$$

where $\frac{\partial^b \boldsymbol{\varrho}}{\partial_w^b \mathbf{q}}$ is the jacobian matrix of a quaternion rotation w.r.t the quaternion, $\frac{\partial^c \boldsymbol{\varrho}}{\partial^b \boldsymbol{\varrho}} = {}_c^b \mathbf{R}$, and $(\cdot)^{-1}$ is the inversion of quaternion. The gradient w.r.t. attitude quaternion can be further transformed to gradients w.r.t. the flat-output states of the drone according to [47].

C. Actuator Constraints

To ensure dynamic feasibility during agile flight, we constrain the quadrotor's thrust and angular velocity within a reasonable range. First, the mass-normalized net thrust calculated by (22) is bounded as

$$\tau_{\min} \leq \|\boldsymbol{\tau}\| \leq \tau_{\max} \tag{36}$$

which can be constrained by constructing such a penalty function

$$\mathcal{J}_\tau = \mathcal{L}_\mu(\|\boldsymbol{\tau}\|^2 - \tau_{\max}^2) + \mathcal{L}_\mu(\tau_{\min}^2 - \|\boldsymbol{\tau}\|^2). \tag{37}$$

Second, the limitation of body rate $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T = \mathbf{R}^T \dot{\mathbf{R}}$ can also be constrained by penalty function

$$\mathcal{J}_\omega = \mathcal{L}_\mu \left(\|\boldsymbol{\omega}_{xy}\|^2 - \omega_{xy,\max}^2 \right) \tag{38}$$

$$+ \mathcal{L}_\mu \left(\|\omega_z\|^2 - \omega_{z,\max}^2 \right) \tag{39}$$

where $\boldsymbol{\omega}_{xy} = [\omega_x, \omega_y]^T$, $\omega_{xy,\max}$ is the x–y body axis maximum angular velocity and $\omega_{z,\max}$ is the z-axis part. To reduce singularities and simplify calculation, we enforce the constraint with Hopf fibration angle velocity decompose [47]

$$\|\boldsymbol{\omega}_{xy}\|^2 = \omega_x^2 + \omega_y^2 = \|\dot{\mathbf{z}}_b\| \tag{40}$$

where $\dot{\mathbf{z}}_b$ is calculated by

$$\dot{\mathbf{z}}_b = f_{\mathcal{DN}} (\boldsymbol{\tau}) \, \mathbf{p}^{(3)} \tag{41}$$

where $\mathbf{p}^{(3)}$ represents the jerk of the drone on the trajectory, and $f_{\mathcal{DN}}(\cdot)$ is given by

$$f_{\mathcal{DN}}(x) = \left(\mathbf{I}_3 - \frac{xx^T}{x^T x} \right) / \|x\|_2. \tag{42}$$

Since the yaw angle ψ changes a little while perching, we approximately adopt $\omega_z = \dot{\psi}$.

D. Flexible Terminal Adjustment

To attach to the moving perching platform quickly and smoothly, the quadrotor should align its terminal states with the perching surface at a proper moment. During trajectory optimization, the temporal profile is affected by the above dynamics, safety constraints, and the time regularization introduced later in Section VI-B, determining the contact moment. As the flight duration changes, the desired terminal states of the quadrotor vary due to the motion of the perching surface. To flexibly synchronize the quadrotor's position, velocity, and attitude with the time-varying states of the perching surface, we formulate the terminal constraints of the drone trajectory as a function of the predicted target trajectory as follows.

At the contact moment, which corresponds to the end of the perching trajectory, the quadrotor should exactly coincide with the perching pose

$$\mathbf{p}(T) = \boldsymbol{\varrho}(T) - \bar{l}\mathbf{z}_s(T) \tag{43}$$

where T is the trajectory duration. Additionally, we expect the heading of the quadrotor to coincide with the estimated target's heading

$$\psi(T) = \theta(T). \tag{44}$$

Ideally, the drone needs to stay relatively still from the target

$$\mathbf{p}^{(1)}(T) = \boldsymbol{\varrho}^{(1)}(T) - \bar{v}_n \mathbf{z}_s(T) \tag{45}$$

where $\mathbf{p}^{(1)}$ and $\boldsymbol{\varrho}^{(1)}$ are the velocity trajectory of the drone and the target, $\bar{v}_n$ is a preset small normal relative speed that helps the drone to stick to the perching surface at the end. However, the hard constraints of the relative end state may conflict with either the safety or the actuator constraints, as shown in Fig. 7. The conflicting constraints will result in no feasible trajectories. Moreover, considering the observation error, the strict terminal velocity constraint may lead to attitude shaking. Therefore, we relax the end velocity constraint by adding the tangential relative speed $\boldsymbol{\nu} = [\nu_x, \nu_y]^T \in \mathbb{R}^2$ as a new variable to be optimized into (45)

$$\mathbf{p}^{(1)}(T) = \boldsymbol{\varrho}^{(1)}(T) + \boldsymbol{\delta}_v(T) \tag{46a}$$

$$\boldsymbol{\delta}_v(T) = [\nu_x \mathbf{x}_s(T), \nu_y \mathbf{y}_s(T), -\bar{v}_n \mathbf{z}_s(T)]^T \tag{46b}$$

where ν_x and ν_y are the relative speed along the direction of $\mathbf{x}_s$ and $\mathbf{y}_s$, respectively. We minimize the tangential relative speed by introducing a regulation term

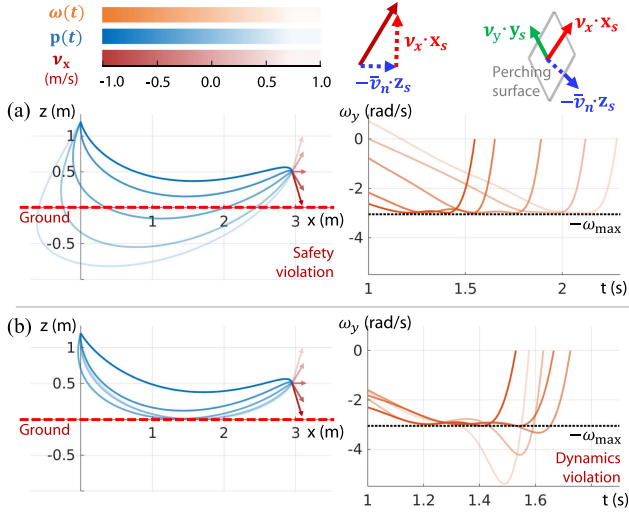$$\mathcal{J}_\nu = \|\boldsymbol{\nu}\|^2. \tag{47}$$

Fig. 7. Optimized perching trajectories for 1.5 rad terminal attitude with different tangential relative speeds along $\mathbf{x}_s$: $\nu_x = \{-1.0, -0.5, 0.0, 0.5, 1.0\}$ (m/s). This instance shows the relative end speed constraint can conflict with safety and actuator constraints. (a) With a hard actuator constraint, trajectories with $\nu_x = \{0.0, 0.5, 1.0\}$ (m/s) violate the safety constraint and hit the ground ($z = 0$). (b) With a hard safety constraint, trajectories with $\nu_x = \{0.0, 0.5, 1.0\}$ (m/s) violate the actuator constraint $\omega_{\max} = 3$ rad/s. Hence, it is necessary to optimize the tangential relative velocity $\boldsymbol{\nu}$ to simultaneously satisfy safety and actuator constraints.

For attitude alignment, the z-axis of the drone body frame should coincide with the normal vector of the perching surface

$$\mathbf{z}_b(T) = \mathbf{z}_s(T) \tag{48}$$

which is equivalent to

$$\boldsymbol{\tau}(T)/\|\boldsymbol{\tau}(T)\|_2 = \mathbf{z}_s(T). \tag{49}$$

To deal with the coupled motion and attitude of the quadrotor, we transform the attitude constraint to the final acceleration $\mathbf{p}^{(2)}(T)$ in flat-out space. Considering this constraint and the thrust limit for terminal states, we design a transformation

$$\mathbf{p}^{(2)}(T) = \tau_e \cdot \mathbf{z}_s(T) - \bar{g}\mathbf{e}_3 \tag{50a}$$

$$\tau_e = \tau_m + \tau_r \cdot \sin(\tau_f) \tag{50b}$$

where $\tau_m = (\tau_{\max} + \tau_{\min})/2$ and $\tau_r = (\tau_{\max} - \tau_{\min})/2$. By introducing a new variable $\tau_f \in \mathbb{R}$, the terminal thrust τ_e is limited within $[\tau_{\min}, \tau_{\max}]$ implicitly. The terminal attitude constraint is eliminated by this transformation.

As for terminal jerk, since it is related to the terminal angular velocity according to (41), we should set the final jerk $\mathbf{p}^{(3)}(T) = \mathbf{0}$ to make the relative angular velocity of the robot small when it touches the surface. Finally, the above terminal state constraints are collectively written as

$$\mathbf{z}^{[s-1]}(T) = F(\boldsymbol{\varrho}(T)). \tag{51}$$

VI. SPATIAL-TEMPORAL TRAJECTORY OPTIMIZATION

A. Occlusion-Aware Path Finding

To provide a reasonable initial path for tracking trajectory optimization, we design an occlusion-aware path-finding method
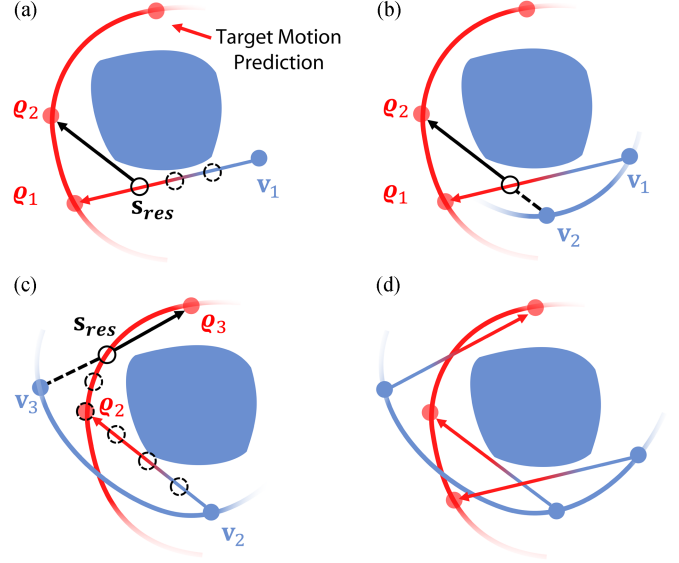

Fig. 8. Illustration of the occlusion-aware multigoal path-finding method.

Algorithm 1: Visibility-aware Tracking Trajectory Generation.

Notation: target trajectory $\boldsymbol{\varrho}(t)$ and corresponding series $\Psi = [\boldsymbol{\varrho}_1, \ldots, \boldsymbol{\varrho}_K]$, current state $\mathbf{z}$, desired tracking distance $\bar{d}$, viewpoint series $\mathbf{V}$, tracking trajectory $\mathfrak{T}$;

1:   $\mathbf{V}$.clear; $Path$.clear;
2:   $\mathbf{v}_1 = \mathbf{z}$;
3:   **for** $i = 2$ to K **do**
4:     $Ray_{i-1} = \boldsymbol{\varrho}_{i-1} - \mathbf{v}_{i-1}$;
5:     $Traj_{i-1} \leftarrow$ **TrajSegment**$(\boldsymbol{\varrho}(t), \boldsymbol{\varrho}_{i-1}, \boldsymbol{\varrho}_i)$;
6:     **for** $\mathbf{s}_j$ on Ray_{i-1} and $Traj_{i-1}$ **do**
7:       **if** **CheckRay**$(\mathbf{s}_j, \boldsymbol{\varrho}_i)$ **then**
8:         $\mathbf{s}_{res} = \mathbf{s}_j$;
9:         **break**;
10:      **end if**
11:     **end for**
12:     $Dir_i = \mathbf{s}_{res} - \boldsymbol{\varrho}_i$;
13:     $\mathbf{v}_i \leftarrow$ **Extend**$(\mathbf{s}_{res}, Dir_i, \bar{d})$;
14:     $\mathbf{V}$.push_back$(\mathbf{v}_i)$;
15:     $Path$.push_back(**SearchPath**$(\mathbf{v}_{i-1}, \mathbf{v}_i)$);
16:   **end for**
17:   $\mathfrak{T} \leftarrow$ **TrackingTrajOpt**$(Path, \Psi, \mathbf{V})$;
18:   **Return** $\mathfrak{T}$;

---

that considers both tracking distance and occlusion. Given the target predicted trajectory $\boldsymbol{\varrho}(t)$ and corresponding target series $\Psi = [\boldsymbol{\varrho}_1, \ldots, \boldsymbol{\varrho}_K]$ by (5), the path-finding method is aimed at offering a viewpoint series $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_K]$ corresponding to Ψ and a path connecting all viewpoints $\mathbf{V}$ for subsequent trajectory optimization. The whole tracking trajectory generation algorithm is listed as Algorithm 1.

Considering efficiency, we use the greedy method, decoupling the whole path-finding problem into smaller multigoal path-searching problems. As an example shown in Fig. 8(a) ($i = 2$

in this example), given the last viewpoint $\mathbf{v}_{i-1}$ and target point $\boldsymbol{\varrho}_{i-1}$, we first traverse the ray from $\mathbf{v}_{i-1}$ to $\boldsymbol{\varrho}_{i-1}$ to get sample points. Then we check the line-of-sight from the sample point $\mathbf{s_j}$ to the current target point $\boldsymbol{\varrho}_i$ in function **CheckRay**(), and select the first no-occlusion one as $\mathbf{s}_{\text{res}}$ (Lines 4–11). To keep the expected tracking distance, as shown in Fig. 8(b), we then extend $\mathbf{s}_{\text{res}}$ along $\overline{\boldsymbol{\varrho}_i \mathbf{s}_{\text{res}}}$ in function **Extend**(), until the length of $\overline{\mathbf{v}_i \boldsymbol{\varrho}_i}$ equals to the expected tracking distance $\bar{d}$ or reach an obstacle (Lines 12–13). If no valid sample point on the ray, we continue sample points on the target trajectory segment between $\boldsymbol{\varrho}_{i-1}$ to $\boldsymbol{\varrho}_i$, which is obtained by the function **TrajSegment**(). An example is shown in Fig. 8(c) ($i = 3$ in this example). Thus, we obtain an occlusion-free viewpoint and simply use $\mathbf{A}^{\star}$ algorithm on the grid map to obtain the path from $\mathbf{v}_{i-1}$ to $\mathbf{v}_i$ in function **SearchPath**() (Line 15). Subsequent viewpoints can be iteratively found. The found path and viewpoints set are used for later tracking trajectory optimization (Line 17).

B. Trajectory Optimization Problem Formulation

Here, we present the general trajectory optimization formulation, and the detailed design for specific stages of tracking and perching is described in Sections VI-E and VI-F, respectively. We use the M-piece polynomial to represent the piecewise flat-output trajectory $\mathbf{z}(t)$. $\mathbf{T} = (T_1, \ldots, T_M)^T \in \mathbb{R}^M_{>0}$ are the durations of each piece, and $\mathbf{c} = (\mathbf{c}_1^T, \ldots, \mathbf{c}_M^T)^T \in \mathbb{R}^{(N+1) \times 4M}$ are the coefficient matrices of each piece. Then, the ith piece is an N-degree polynomial

$$\mathbf{z}_i(t) = \mathbf{c}_i^T \boldsymbol{\beta}(t) \quad \forall t \in [0, T_i] \tag{52}$$

where $\boldsymbol{\beta}(t) = [t^0, t^1, \ldots, t^N]^T$ is the natural basis. The trajectory optimization problem can be formulated as follows:

$$\min_{\mathbf{c}, \mathbf{T}} \mathcal{J}_E = \int_0^T \|\mathbf{z}^{(s)}(t)\|^2 \mathrm{d}t + \rho T \tag{53a}$$

$$\text{s.t. } T \geq T_\ell \tag{53b}$$

$$\mathbf{z}^{[s-1]}(0) = \bar{\mathbf{z}}_0 \tag{53c}$$

$$\mathbf{z}^{[s-1]}(T) = F(\boldsymbol{\varrho}(T)) \tag{53d}$$

$$\mathcal{G}(\mathbf{z}^{[s-1]}(t), \boldsymbol{\varrho}(t)) \preceq \mathbf{0} \quad \forall t \in [0, T] \tag{53e}$$

$$\mathcal{H}(\mathbf{z}^{[s-1]}(t), \boldsymbol{\varrho}(t)) \preceq \mathbf{0} \quad \forall t \in \mathcal{T} \tag{53f}$$

where $T = \sum_{i=1}^M T_i$, cost function (53a) tradeoffs the smoothness and aggressiveness, and ρ is the time regularization parameter. Equation (53b) guarantees the entire duration greater than the lower bound T_ℓ. Equations (53c) and (53d) are the boundary conditions. Equation (53c) guarantees the trajectory starting from initial state $\bar{\mathbf{z}}_0 = \{\mathbf{z}_0, \ldots, \mathbf{z}_0^{(s-1)}\}$. Equation (53d), which is explained in Section V-D, constrains the final state determined by the predicted target trajectory. For different trajectory requirements, we design different additional constraints: (53e) are the inequality constraints that are continuously forced on the entire time, and (53f) are the inequality constraints that are discretely forced on the specific time series $\mathcal{T}$. The discrete-time constraints are designed to ensure consistency with the discrete results of the path-finding method.

To deform the spatial and temporal profiles of the trajectory during optimization efficiently, we adopt $\mathfrak{T}_{\text{MINCO}}$ [48], a state-of-the-art minimum control effort polynomial trajectory class. MINCO efficiently conducts spatiotemporal deformation of the M-piece flat-output trajectory $\mathbf{z}(t)$ by decoupling the space and time parameters with a linear-complexity mapping

$$\mathbf{z}(t) = \mathcal{M}_{\mathbf{m}, \mathbf{T}}(t) \tag{54}$$

where $\mathbf{T} \in \mathbb{R}^M_{>0}$ are the durations defined as aforementioned and $\mathbf{m} = (\mathbf{m}_1, \ldots, \mathbf{m}_{M-1})^T \in \mathbb{R}^{4 \times (M-1)}$ are the adjacent intermediate points between connected pieces. The s-order $\mathfrak{T}^s_{\text{MINCO}}$ consisting of $(2s - 1)$-degree polynomials can represent an s-integrator chain dynamics system. Furthermore, MINCO is advanced in converting $\{\mathbf{m}, \mathbf{T}\}$ to $\{\mathbf{c}, \mathbf{T}\}$ by the parameter mapping $\mathbf{c} = C(\mathbf{m}, \mathbf{T})$ with linear time and space complexity via *banded PLU factorization*. Meanwhile, the gradients for $\{\mathbf{c}, \mathbf{T}\}$ are also propagated to MINCO parameters $\{\mathbf{m}, \mathbf{T}\}$ in linear time. We refer readers to [48] for more details.

To solve the continuous constrained optimization problem conveniently, we convert it to an unconstrained optimization problem. All kinds of equality constraints are implicitly satisfied by using the variable of MINCO for optimization. To eliminate the inequality constraints (53e)–(53f), we use penalty function method introduced in Section VI-C. We adopt appropriate penalty weights and a fast postoptimization check for safety and dynamic feasibility to prevent unreasonable inequality constraint violations. The temporal constraint (53b) is eliminated by diffeomorphic variable substitution, which is introduced in Section VI-D. Finally, (53) is transformed to an unconstrained optimization problem written as

$$\min_{\mathbf{m}, \mathbf{T}} \mathcal{J}_E + \int_0^T \mathcal{J}_\mathcal{G} dt + \sum_{t \in \mathcal{T}} \mathcal{J}_\mathcal{H} \tag{55}$$

where $\mathcal{J}_\mathcal{G}$ is the continuous-time penalty function corresponding to (53e), and $\mathcal{J}_\mathcal{H}$ is the discrete-time penalty function corresponding to (53f).

C. Inequality Constraints Elimination

Inspired by the constraint transcription method [49], the inequality constraints (53e) and (53f) are formulated into penalty functions.

1) Continuous Relative-Time Penalty: For the constraint (53e) forced over the entire continuous trajectory, the derivatives between pieces are independent of each other. We transform them into finite constraints via the integral of constraint violation, which is further transformed into the penalized sampled function $\mathcal{J}_\mathcal{I}$

$$\mathcal{J}_\mathcal{I} = \int_0^T \mathcal{J}_\mathcal{G} dt = \sum_{i=1}^M \mathcal{I}_i \tag{56a}$$

$$\mathcal{I}_i = \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} \bar{\omega}_j \mathcal{J}_\mathcal{G} \left(\mathbf{z}_i^{[s-1]}(t_i), \boldsymbol{\varrho}(t_i) \right) \tag{56b}$$

where κ_i is the sample number, $(\bar{\omega}_0, \bar{\omega}_1, \ldots, \bar{\omega}_{\kappa_i-1}, \bar{\omega}_{\kappa_i}) = (1/2, 1, \ldots, 1, 1/2)$ are the quadrature coefficients following

the trapezoidal rule [50], and $t_i = \frac{j}{\kappa_i} T_i$. Then the gradient of $\mathcal{I}_i$ w.r.t. $\mathbf{c}$ and $\mathbf{T}$ can be easily derived

$$\frac{\partial \mathcal{J}_{\mathcal{I}}}{\partial \mathbf{c}_i} = \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} \left(\bar{\omega}_j \frac{\partial \mathcal{J}_{\mathcal{G}}}{\partial \mathbf{c}_i} \right) \tag{57a}$$

$$\frac{\partial \mathcal{J}_{\mathcal{I}}}{\partial T_i} = \frac{\mathcal{I}_i}{T_i} + \frac{T_i}{\kappa_i} \sum_{j=0}^{\kappa_i} \left(\bar{\omega}_j \frac{j}{\kappa_i} \frac{\partial \mathcal{J}_{\mathcal{G}}}{\partial t_i} \right). \tag{57b}$$

For $\partial \mathcal{J}_{\mathcal{G}} / \partial \mathbf{c}_i$, the total derivative of it is written as

$$\frac{\partial \mathcal{J}_{\mathcal{G}}}{\partial \mathbf{c}_i} = \sum_{k=0}^{s-1} \boldsymbol{\beta}^{(k)} \frac{\partial \mathcal{J}_{\mathcal{G}}}{\partial \mathbf{z}_i^{(k)}}. \tag{58}$$

Similar for $\partial \mathcal{J}_{\mathcal{G}} / \partial t_i$ that can be written as

$$\frac{\partial \mathcal{J}_{\mathcal{G}}}{\partial t_i} = \sum_{k=0}^{s-1} \mathbf{z}_i^{(k+1)} \frac{\partial \mathcal{J}_{\mathcal{G}}}{\partial \mathbf{z}_i^{(k)}} + \dot{\boldsymbol{\varrho}} \frac{\partial \mathcal{J}_{\mathcal{G}}}{\partial \boldsymbol{\varrho}} + \dot{\theta} \frac{\partial \mathcal{J}_{\mathcal{G}}}{\partial \theta}. \tag{59}$$

2) Discrete Absolute-Time Penalty: To ensure consistency with the discrete results of the path-finding method, we formulate the corresponding constraint (53f) forced on the trajectory over specific discrete moments. The cost of a piece of the trajectory at a discrete moment will be affected by the duration changes of the former pieces. For a state $\mathbf{z}(t_k)$, assume that t_k is an absolute moment on the jth piece of the trajectory $\mathbf{z}_j(t)$, the relative time $t_r = (t_k - \sum_{i=1}^{j-1} \mathbf{T}_i)$ on $\mathbf{z}_j(t)$ is within the range

$$\sum_{i=1}^{j-1} T_i \leq t_k \leq \sum_{i=1}^{j} T_i. \tag{60}$$

With the denotation of $\mathcal{J}_{\mathcal{D}} = \sum_{t \in \mathcal{T}} \mathcal{J}_{\mathcal{H}}$ in (55), the gradients of $\mathcal{J}_{\mathcal{D}}$ w.r.t. $\mathbf{c}$ and $\mathbf{T}$ can be evaluated as

$$\frac{\partial \mathcal{J}_{\mathcal{D}}}{\partial \mathbf{c}_i} = \begin{cases} \sum_{k=0}^{s-1} \boldsymbol{\beta}^{(k)}(t_r) \partial \mathcal{J}_{\mathcal{D}} / \partial \mathbf{z}_i^{(k)}, & i = j \\ 0, & i \neq j \end{cases} \tag{61a}$$

$$\frac{\partial \mathcal{J}_{\mathcal{D}}}{\partial T_i} = \begin{cases} \sum_{k=0}^{s-1} -\mathbf{z}_i^{(k+1)}(t_r) \partial \mathcal{J}_{\mathcal{D}} / \partial \mathbf{z}_i^{(k)}, & i < j \\ 0, & i \geq j. \end{cases} \tag{61b}$$

Then, the problem of calculating gradients w.r.t $\mathbf{c}$ and $\mathbf{T}$ is further transformed into taking the gradients w.r.t the drone's flat-output states and their derivatives $\mathbf{z}^{[s-1]}$.

D. Temporal Constraints Elimination

Time slack constraint (53b) can be written as

$$\sum_{i=1}^{M} T_i \geq T_\ell. \tag{62}$$

We denote $\boldsymbol{\varsigma} = (\varsigma_1, \ldots, \varsigma_M) \in \mathbb{R}^M$ as new variables to be optimized and use the transformation

$$T_\Sigma = T_l + \varsigma_M^2 \tag{63a}$$

$$T_i = \frac{e^{\varsigma_i}}{1 + \sum_{j=1}^{M-1} e^{\varsigma_j}} T_\Sigma, 1 \leq i < M \tag{63b}$$

$$T_M = T_\Sigma - \sum_{i=1}^{M-1} T_i. \tag{63c}$$

The temporal constraints are eliminated by such substitution. Especially, when $T_\ell = 0$, we use a simpler transformation $T = e^\varsigma$ to eliminate the constraint (53b). Since the zero duration is invalid, here, constraint (53b) is changed to $T > T_\ell$.

E. Tracking Constraints Formulation

For trajectory optimization for tracking, we adopt $\mathfrak{T}_{\text{MINCO}}^3$ to represent position trajectory and $\mathfrak{T}_{\text{MINCO}}^2$ to represent yaw trajectory, guaranteeing sufficient optimization freedom. The differentiable visibility metrics have been presented in Section IV, and the constraints for tracking are further clarified here. Since the front end offers discrete drone and target series, the corresponding visibility costs forced at specific times are discrete absolute-time penalties. Then, the continuous relative-time and discrete absolute-time penalties are listed as

$$\mathcal{J}_{\mathcal{G}} = \lambda_{\mathcal{G}} \left[\mathcal{J}_o, \mathcal{J}_v, \mathcal{J}_a \right]^T \tag{64a}$$

$$\mathcal{J}_{\mathcal{H}} = \lambda_{\mathcal{H}} \left[\mathcal{J}_{OD}, \mathcal{J}_{OA}, \mathcal{J}_{OE} \right]^T \tag{64b}$$

where $\lambda_{\mathcal{G}}$ and $\lambda_{\mathcal{H}}$ are preset weight vectors. $\mathcal{J}_{OD}$ is the observation distance penalty function defined as (9), $\mathcal{J}_{OA}$ is the observation angle part defined as (12), and $\mathcal{J}_{OE}$ is the occlusion effect avoidance part defined as (18). In addition, $\mathcal{J}_o$ is obstacle avoidance cost defined as

$$\mathcal{J}_o(t) = g(d_{\text{thr}}^2 - \Xi(\mathbf{p}(t))^2) \tag{65}$$

where d_{thr} is the safety threshold, and $\Xi(\cdot)$ is the distance to the closest obstacle obtained from ESDF. $\mathcal{J}_v, \mathcal{J}_a$ are dynamic feasibility costs

$$\mathcal{J}_v(t) = g(\|\mathbf{p}^{(1)}(t)\|^2 - v_{\max}^2) \tag{66}$$

$$\mathcal{J}_a(t) = g(\|\mathbf{p}^{(2)}(t)\|^2 - a_{\max}^2) \tag{67}$$

where $v_{\max}$ and $a_{\max}$ are the maximum velocity and acceleration, respectively. For temporal constraint (53b), the duration of tracking trajectory should be equal to the prediction duration of the target motion T_p, ideally. However, enforcing the drone to reach the final states in a fixed duration may cause dynamic infeasibility, for instance, when the target moves faster than the drone. Therefore, we make a time slack $T \geq T_p$. For the final state constraint (53d), we fix it to the last viewpoint $\mathbf{v}_K$. For trajectory initialization, the initial guess of $\mathbf{m}$ is obtained by sampling on the path provided by the occlusion-aware path-finding method, and the initial $T_i = T_p/M$.

F. Perching Constraints Formulation

For trajectory optimization for perching, we adopt $\mathfrak{T}_{\text{MINCO}}^4$ to represent position trajectory and $\mathfrak{T}_{\text{MINCO}}^2$ to represent yaw trajectory. When perching, it is necessary to force the terminal angular velocity to be 0, and $\mathfrak{T}_{\text{MINCO}}^4$ is the lowest order MINCO supporting to constrain the terminal angular velocity [48]. With the initial time set as $T_i = \|\mathbf{p} - \boldsymbol{\varrho}\|/v_{\max}$, a boundary value problem is solved to obtain the initial guess of $\mathbf{m}$. Then, we summarize the penalty functions as

$$\mathcal{J}_{\mathcal{G}} = \lambda_{\mathcal{G}} \left[\mathcal{J}_c, \mathcal{J}_p, \mathcal{J}_\omega, \mathcal{J}_\tau, \mathcal{J}_{\Delta z}, \mathcal{J}_\nu \right]^T \tag{68a}$$

TABLE I
PARAMETERS SETTING FOR TRACKING SIMULATIONS

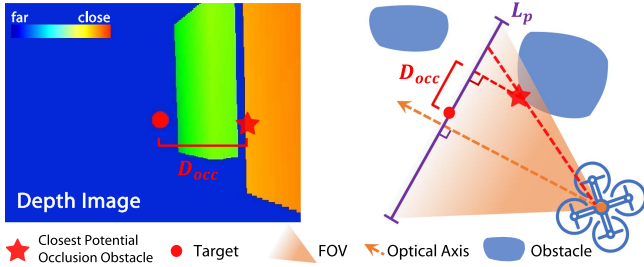| v_{max}	a_{max}	$\|\omega_{max}\|$	FoV	Image resolution	Tracking distance
Drone: 3.0 m Target: 1.5 m	Drone: 6.0 m/s^2 Target: 2.0 m/s^2	3.0 rad/s	80°× 65°	640× 480 px^2	5.5 m

Fig. 9. Illustration of the occlusion criterion D_{occ}, which denotes the closest possible occluding obstacle. L_p is the projected plane crossing the target. Only obstacles on the camera image between the drone and L_p are recorded.

where $\mathcal{J}_c$ is collision avoidance cost defined in (27), $\mathcal{J}_p$ is perception cost defined in (32). $\mathcal{J}_\omega, \mathcal{J}_\tau$ are dynamic feasibility cost defined in (38) and (37), $\mathcal{J}_{\Delta z}$ is relative height cost defined in (29), $\mathcal{J}_\nu$ is the regulation for tangential relative speed defined in (47). For temporal constraint (53b), the low bound is set to $T_\ell = 0$.

VII. SIMULATION AND BENCHMARK

In this section, we conduct extensive simulations to validate the performance of our tracking and perching planning methods. As for benchmarks, cutting-edge works [18], [20], [21], [40], [41] in the tracking and perching fields are selected. All the simulations run on an Intel Core i5-9400F CPU with GeForce GTX 2060 GPU.

A. General Comparison of Visibility-Aware Tracking

We benchmark our method with the work of Jeon et al. [20] and Ji et al.'s Elastic-Tracker [21], which both consider the target's visibility. To further demonstrate the importance of visibility awareness, we also compare with Han's Fast-Tracker [18] without visibility consideration. These works are all open source and have not been modified. To compare the tracking performance fairly, we set the same parameters as listed in Table I with the same target trajectory. To compare the capability of occlusion avoidance, since the drone obtains target information directly from camera images, we record the projected distance between the target and the closest possible occluding obstacle (D_{occ}). The occlusion criterion is shown in Fig. 9.

We test all four methods in the same simulation environment as Elastic-Tracker. Since obstacle density has a significant impact on tracking performance, we test in scenarios with different obstacle densities. The sparse, medium, and dense density correspond to 6, 5, and 4 m average obstacle spacing, respectively. During tracking, we count the target positions projected to x–y plane in the tracking quadrotor's FoV. In Fig. 10(a), the heat map shows the distribution of the target positions, relative to

TABLE II
COMPUTATION TIME COMPARISON BETWEEN TRACKING METHODS

Methods	Average time (ms)		
	t_{path}	t_{opt}	t_{total}
Fast	9.56	0.38	9.94
Jeon's	11.32	3.20	14.32
Elastic	1.24	2.34	3.58
Ours	0.96	1.85	**2.81**

The boldface values indicate the optimal cases in the corresponding test scenario.

the tracking drone. Our method and Elastic-Tracker can both keep the target in the proper position of the FoV. Fig. 10(b) shows the distribution of D_{occ}. Since the main failures for Fast-Tracker are caused by tracking too close to the target or letting the target out of FoV, making the D_{occ} criterion meaningless, we do not include it in Fig. 10(b). Our method results in larger D_{occ} in all obstacle densities and is more effective at avoiding potential occlusion.

Furthermore, we record the failure reasons during tracking. Once the target is out of FoV, too near to the target, or occluded by obstacles, a failure is recorded. The failure reasons in the dense obstacle scene are counted in Fig. 10(c). We also benchmark the computation time, including path finding and trajectory optimization parts, as shown in Table II. Fast-Tracker consumes less time for trajectory optimization since it ignores visibility, resulting in more failures. In comparison, our proposed method consumes a much lower total computation budget, due to our lightweight path-finding method and concise metrics for trajectory optimization.

B. Case Study of Visibility-Aware Tracking

To better demonstrate and compare the tracking performance of different methods, we set up a representative scenario and present the whole process of tracking in Fig. 11. The parameter setting is the same as in Section VII-A. D_{occ} and the tracking distance are shown for all four methods. The occluded moments are indicated by the red lines connecting the drone and the target, and also correspond to the red background in the D_{occ} curve. For Fast-Tracker, due to the lack of visibility consideration, the quadrotor consistently follows the target closely. In the event of a sudden target turn, the drone often fails to adjust quickly, resulting in the target escaping from the FoV, as the moment shown in Fig. 11(a). As shown in Fig. 11(b), Jeon's method results in multiple occlusion moments and a less smooth trajectory, since the sequential graph-search-based path planner and smooth planner fail to jointly optimize visibility and smoothness and lack temporal optimization. As shown in Fig. 11(c), because Elastic-Tracker handles visibility by generating 2-D visible fans at a certain height, it fails to adjust visibility against the 3-D obstacle, which in this case is a two-layer instance. More importantly, when passing obstacle-rich areas, our method can adjust tracking distance elastically to reduce occlusion probability, as highlighted in green in Fig. 11(d). Especially when tracking the target passing the narrow gap, the drone with our method
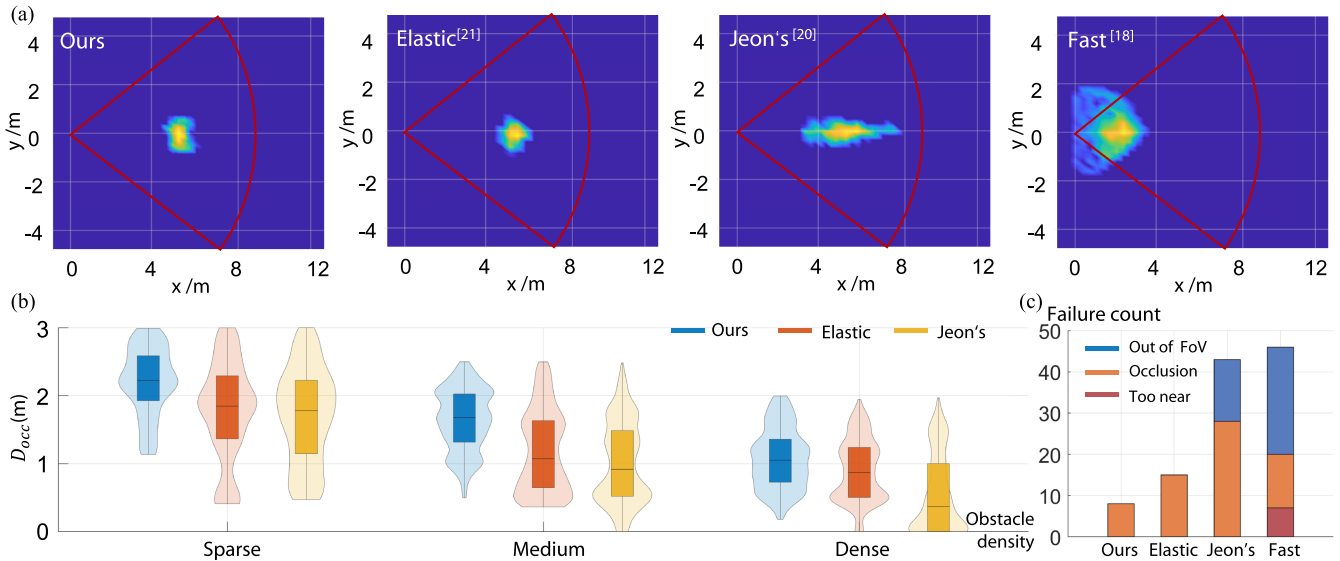
Fig. 10. (a) Comparison of the distribution heat map of the target position relative to the tracking quadrotor. The red sector represents the FoV of the drone. (b) Violin graph presenting a comparison of the distribution of the closest possible occluding obstacle on the image plane (D_{occ}) under different obstacle densities. (c) Failure reasons including out of view, too near (distance less than $1\,m$), occluded by obstacles.

TABLE III
PARAMETERS SETTING FOR PERCHING SIMULATIONS

τ_{min}	τ_{max}	$\|\omega_{max}\|$	FoV	Image resolution
$5.0\,\text{m/s}^2$	$17.0\,\text{m/s}^2$	$3.0\,\text{rad/s}$	$80° \times 65°$	$640 \times 480\,\text{px}^2$

TABLE IV
COMPUTATION TIME COMPARISON BETWEEN PERCHING METHODS

Methods	Average time (ms)		
	$30°$	$60°$	$90°$
Paneque's	239.59	308.05	208.81
Mao's (A)	3.42	2.36	✗
Mao's (B)	16.65	13.73	✗
Ours	3.76	4.31	16.68

deployed elastically reduced the tracking distance to 4 m and avoid occlusion successfully. Consequently, our method shows reasonable visibility adjustment and has better performance at avoiding occlusion, due to the comprehensive and effective 3-D visibility metrics.

C. Perching Benchmark

We benchmark our perching method with the state-of-the-art perching methods proposed by Mao et al. [40] and Paneque et al. [41], which both consider visibility during perching. Paneque formulates a constrained discrete-time NLP problem and solves it using ForcesPRO [51]. They model the inputs of the system as the desired constant thrust derivatives and control each rotor thrust directly. To compare fairly, we constrain the thrust to be the same as ours so that the dynamics model of Paneque's and ours are similar. Since this work focuses on the perception of powerlines, we replace its perception cost with ours to compare fairly. Mao formulates a QP problem to constrain the terminal state and velocity bound. They increase the trajectories' time iteratively and recursively solve the QP problem until the thrust constraint is satisfied. We use the open-source code of Paneque's work and reimplement Mao's work, which is not open-source but not difficult to implement with the QP formulation. In the comparison, the terminal pitch angle is set to $\pi/2$ rad, and the trajectories are required to be rest-to-rest. The parameters of the drone are listed in Table III.

We compare both the trajectory quality and computation time of these methods, the results are presented in Fig. 12 and Table IV. Paneque's and our method successfully generate perching trajectories satisfying actuator and visibility constraints. Nevertheless, Paneque's method consumes much more computation time than ours, since a complex discrete-time multiple-shooting NLP problem formulation. Adopting MINCO, our method can optimize trajectories in the low-dimension flat-output space with high efficiency and can achieve real-time replanning during agile flight. Moreover, due to the finite discrete resolution, its thrusts and body rates are less smooth than ours. As shown in Fig. 13, Mao's method fails to guarantee dynamic feasibility even after sufficient iterations, which double the duration without providing any relief. Mao's method ignores that dynamic feasibility is affected by not only the temporal profile but also the spatial profile. As a result, its strategy narrows the solution space, leading to no solutions under such tight actuator constraints. Moreover, Mao's computation time depends on both initial duration and time sampling resolution. Thus, for Mao's (A), we set a better initial duration of 1.5 s and a coarser resolution 0.1 s, while for Mao's (B), a worse initial duration 1.0 s and a finer resolution 0.05 s are set. The results indicate that the computation time of Mao's method greatly increases with a
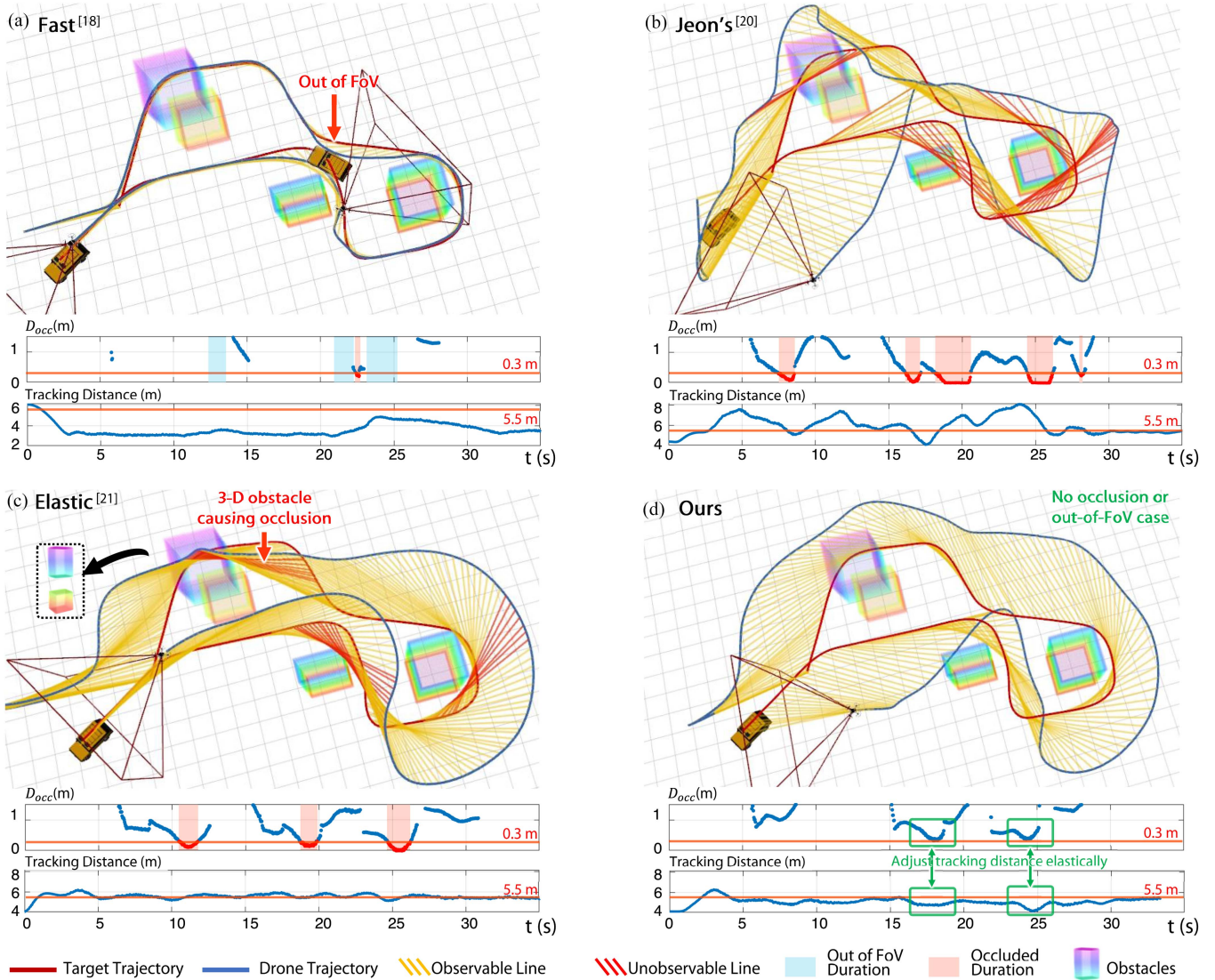
Fig. 11. Case study of visibility-aware tracking. Trajectories of the drone and the target, and the lines connecting their positions at the corresponding time are presented. The red lines show the occlusion moments. The D_{occ} and the tracking distance are shown for all four methods. We choose 0.3 m as the threshold at which the occlusion is likely to occur. An out-of-FoV failure of Fast-Tracker is shown in (a). (c) Shows that elastic-tracker fails to adjust visibility against the 3-D obstacle due to its pseudo-3-D visibility metric. (d) Shows that our method has no occlusion and out-of-FoV situations, and can adjust the relative distance elastically according to the environment.

worse initial value and a finer resolution. In comparison, our proposed method demonstrates high efficiency without greatly sacrificing the optimality of solutions.

Furthermore, we compare the capability of our perching planner with Paneuqe's planner, Mao's planner, and our previous planner [10]. The result is summarized in Table V. Both Paneuqe's and Mao's methods can only handle static targets. While ours can simultaneously handle collision avoidance, dynamic feasibility, and visibility constraints, and generate perching trajectories toward a moving target with a low computation budget. To further demonstrate our capability for adapting dynamic targets, Fig. 14 presents the generated trajectories for perching on moving targets with different speeds. Balanced aggressiveness and dynamic feasibility, our method drives the quadrotor perch on the target at proper contact moments. With different contact

TABLE V
CAPABILITY COMPARISON BETWEEN PERCHING METHODS

Capability Method	Collision Avoidance	Dynamic Feasibility	Visibility Awareness	Dynamic Target
Paneque's	✓	✓	✓	✗
Mao's	✗	✓	✓	✗
Ji's	✓	✓	✗	✓
Ours	✓	✓	✓	✓

The boldface values indicate the optimal cases in the corresponding test scenario.

moments and different terminal states, the quadrotor could still get full-state synchronization with the perching surface, presenting flexibility.
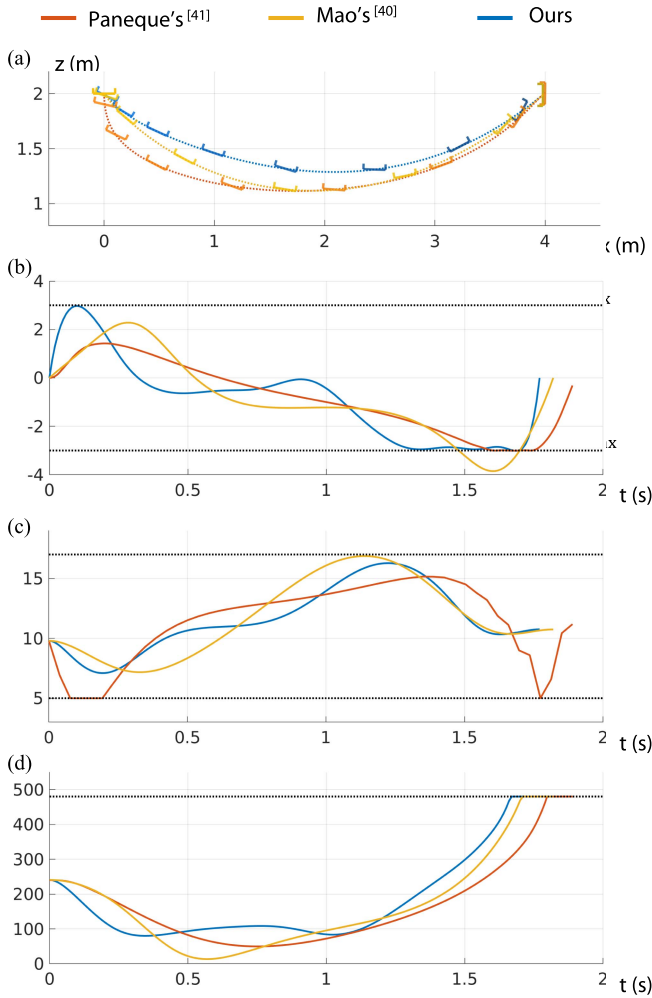
Fig. 12. Comparison of the trajectories generated by Paneuqe's, Mao's, and our planner for perching on vertical surfaces. (a) x–z trajectory shape. (b) Angular velocity. Mao's method exceeds the limitation. (c) Mass-normalized net thrust. Due to the finite discrete resolution of Paneuqe's method, its thrusts are less smooth than the others. (d) Target position on the u-v image space. All the methods guarantee that the target is visible.
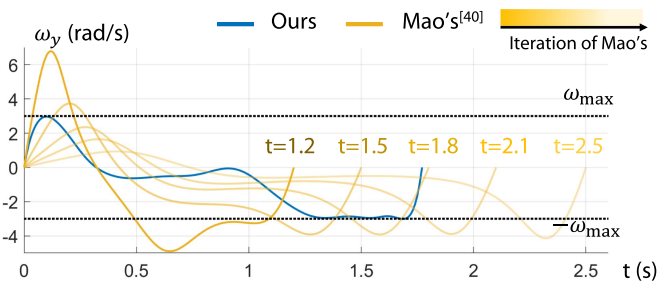

Fig. 13. Comparison of the angular velocity of the trajectories generated by our method and Mao's method. Mao's method fails to guarantee dynamic feasibility even after sufficient iterations to increase trajectory duration.

D. Ablation Study of the Visibility Awareness for Perching

Continuous and stable perception of targets is the foundation for perching planning. We validate our visibility-aware approach by comparing the perching trajectories with and without the visibility term. We use AprilTag [52] to mark the precise perching
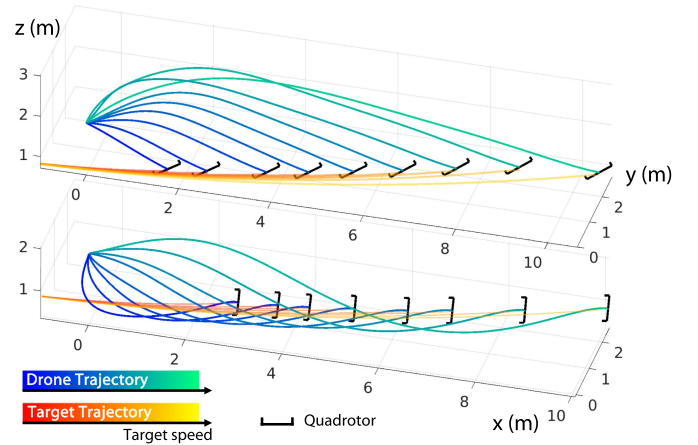

Fig. 14. Illustration of the 3-D trajectories generated for perching on a moving plane with different speeds. Top: 0.5 rad terminal pitch angle. Bottom: 1.5 rad terminal pitch angle. The target's velocity is ranged from 1.5 to 3.0 m/s. The kinematics of the target conforms to the CTRV model with $\omega = 0.2$ rad/s.

TABLE VI
ABLATION STUDY OF THE VISIBILITY AWARENESS FOR PERCHING

Terminal pitch angle	Visibility awareness	Observable distance (m)	Observable duration (s)
0.0 rd	w/o	0.64 (↑203%)	0.60 (↑152%)
	w/	**1.96**	**1.51**
0.5 rd	w/o	0.44 (↑352%)	0.36 (↑367%)
	w/	**1.99**	**1.68**
1.5 rd	w/o	3.61 (↑15%)	2.2 (↑35%)
	w/	**4.15**	**2.96**

The boldface values indicate the optimal cases in the corresponding test scenario.

position. As shown in Fig. 16, localization error increases with the observation angle θ. The quadrotor we used has front and down cameras for observation. Therefore, we choose the down camera for continuous detection when the terminal attitude is below $30°$ and the front camera when it is above $30°$. Moreover, the decreasing error with decreasing distance also presents the necessity of replanning.

We conduct tests under different terminal attitudes, as shown in Fig. 15. The parameters are the same as in Section VII-C. The expected observation range is set to $\bar{d}_{\min} = 0.2$ m and $\bar{d}_{\max} = 2.0$ m. The orange and blue trajectories show quadrotor maneuvers with and without visibility considerations, respectively. For 0.0 and 0.5 rd terminal pitch angles, without visibility consideration, the drone can only observe the target at a short distance and a short duration. While with the perception cost, the quadrotor can keep continuous observation within the preset observe distance $\bar{d}_{\max} = 2.0$ m. For 1.5 rad pitch angle, the visibility-aware perching can keep the target at the center of the image space as long as possible. The observable distance and duration before touching the target plane are summarized in Table VI. Consequently, with our visibility-aware perching planning, the target visibility in terms of both space and time is dramatically improved.
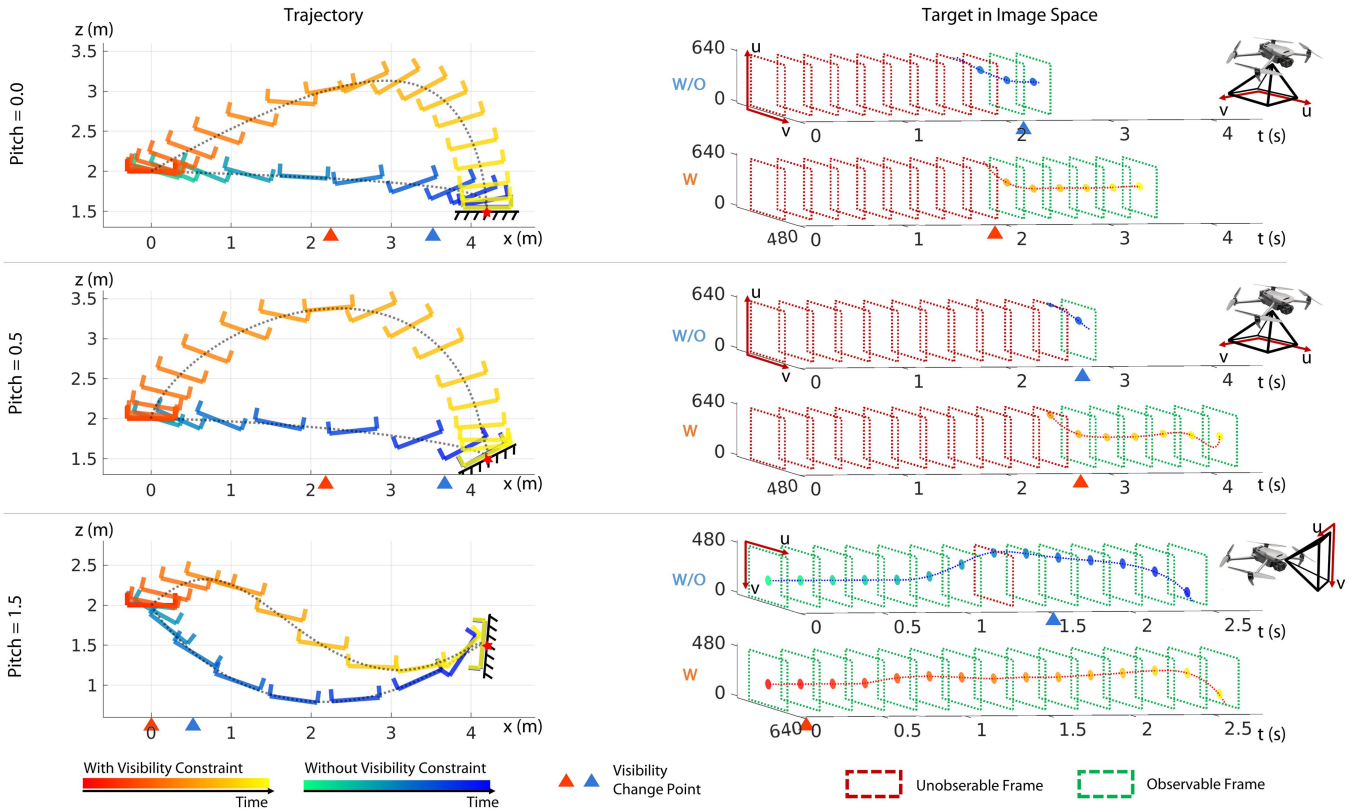
Fig. 15. Ablation study of the visibility awareness at different perching terminal angles. The quadrotor uses the down camera to observe the target at 0.0 and 0.5 rad terminal pitch angles, and the front camera for 1.5 rad terminal pitch angles. The expected observe distance is set to $\bar{d}_{\min} = 0.2$ m and $\bar{d}_{\max} = 2.0$ m. Left: The x–z trajectory w/ and w/o perception constraint. The triangles on the x-axis represent the position when the spatial visibility changes. The target is visible on the right side of the point. Right: The target position in the u–v image space. The target position is expected to be as close to the image's center as possible. The triangles on the x-axis represent the moment when the temporal visibility changes.

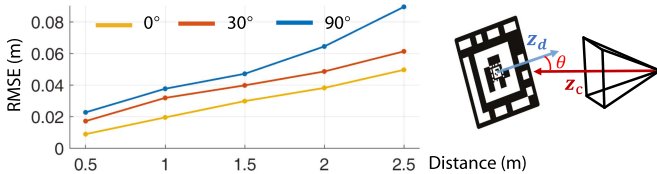

Fig. 16. Visual tag localization error at different distances and observation angles θ. $\mathbf{z}_d$ is the normal vector of the tag plane and $\mathbf{z}_c$ is the optical axis.

VIII. REAL WORLD EXPERIMENTS

A. System Configuration

We deploy our adaptive tracking and perching scheme on a commercial drone (DJI MAVIC3) shown in Fig. 18, with a full-size SUV as the moving platform. Our system operates outside the laboratory and is tested on the road, bringing it closer to real-world applications. The drone will attempt to track and perch on the SUV. The perching stage starts when receiving external perching order.

The target position is obtained by the fusion of finer visual detection and coarser nonstation differential-GPS-based relative localization. We use the recursive visual fiducial tags [52] to mark the precise perching position. The software implementation of our scheme is integrated into the drone's embedded

processors, and interfaces with the existing modules to form a complete system. The localization and mapping functionality of the DJI MAVIC3 is retained as the foundation of the system. After the generation of the trajectory, control instructions are derived and transmitted to the flight controller for execution. Software modules, estimation, perception, planning, and control are all running onboard in real time. On the severely limited computing platform ($8\times$ slower than simulation settings), our planning scheme can still achieve 20 Hz replanning.

B. Visibility-Aware Tracking Validation

We select two common real-world scenarios to validate the visibility awareness of our tracking scheme. Static and moving impediments will appear in the path of the vehicle when it moves, stops, or turns. The drone is expected to track the vehicle without occlusion or target loss.

1) Scenario A: L-Shape Turn: The occlusion frequently happens when the target makes turns around obstacles. In this case, the target vehicle makes an L-shape turn, and two trucks are parked at the bend, which could easily block the view of the tracking drone. As shown in Fig. 17(a), due to our visibility-aware tracking scheme, the drone actively adjusts its relative position and keeps its FoV free of occlusion even when the vehicle makes sudden turns near obstacles.
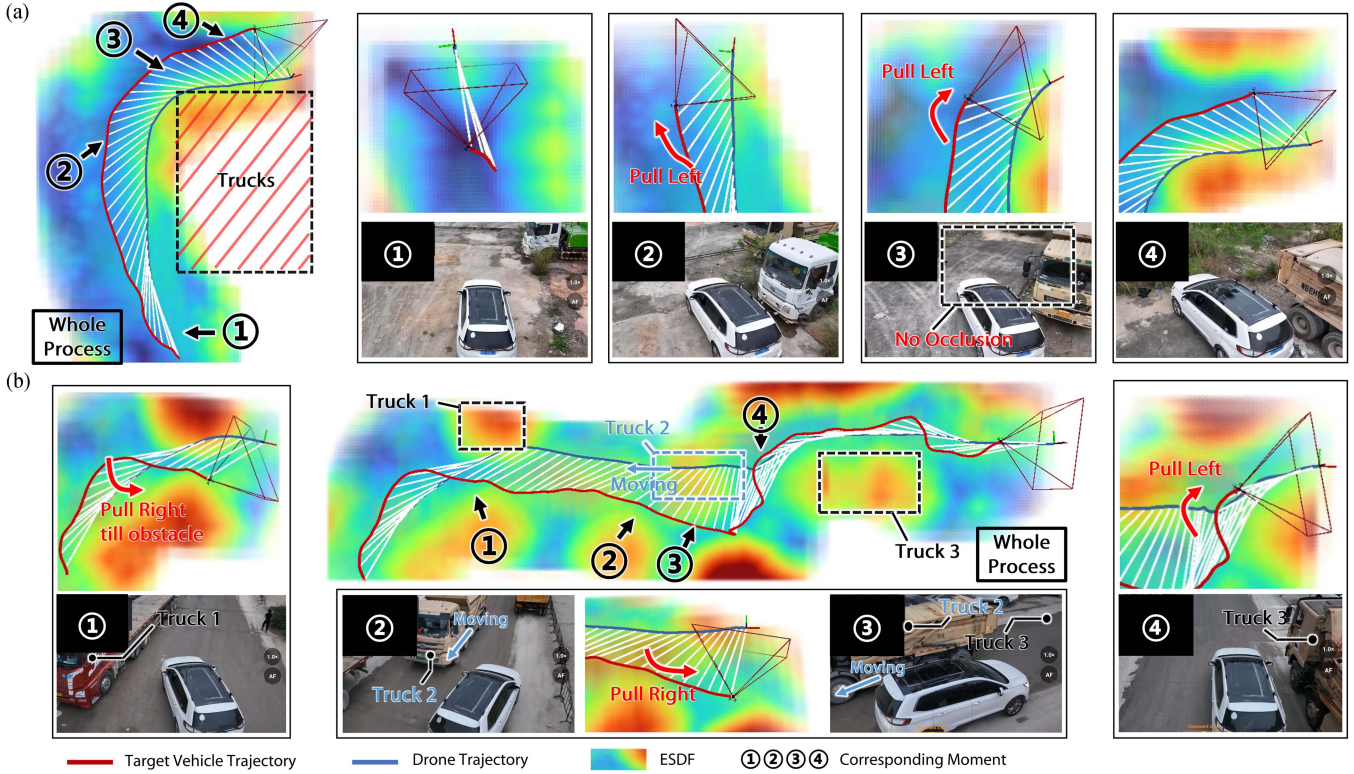
Fig. 17. Real-world visibility-aware tracking validation. (a) L-shape Turn: ① The drone tracks the SUV behind. ② A trunk appears closely on the right. The drone pulls left to avoid occlusion. ③ The SUV turns right along the trunk, easily causing occlusion. The drone pulls left to avoid it. ④ Constantly tracking on the left for better visibility. (b) Bilateral staggered obstacles: ① The SUV heading straight, truck 1 appears on the left, and the drone pulls right. ② Trunk 2 is approaching the target SUV from the left. ③ The SUV stops to make way for truck 2. The tracking drone keeps pulling right as truck 2 gets closer. ④ The SUV passes truck 3, and the drone pulls left for less occlusion.

Fig. 18. Illustration of the quadrotor platform (DJI MAVIC3). The front and bottom cameras are used for detection. The gimbal camera was not selected for detection since its dedicated designed for photography and with high link delay. The drone and the surface are secured with velcro after perching.

Fig. 19. Comparison with the origin DJI tracking under scenario 2. (a) *Active t* function of DJI MAVIC loses the target car at the turning. (b) Our tracking scheme enhances visibility actively, tracking successfully.

2) Scenario B: Bilateral Staggered Obstacles: While the SUV is moving, static or dynamic trucks will appear alternately on the left and right sides. The target vehicle will either stop or move forward. In this experiment, not only the active adjustment of the drone to the static surroundings is demonstrated, but also the ability to avoid dynamic obstacles affecting the observation line of sight. As shown in Fig. 17(b), the tracking drone maintains high visibility of the target SUV throughout the process.

3) Comparative Experiment: We compare our tracking scheme with the mature commercial *Active Tracking* function of DJI MAVIC3 under Scenario A, as shown in Fig. 19. When the target SUV turns right near the truck, the drone with an *active tracking* is unaware of the potential occlusion and loses the target at the turning. While our method drives the drone to avoid the occlusion effect of the truck, and allows the drone to continuously receive visual information of the target.

In these experiments, the drone observes the target SUV at a certain distance with a straight-forward angle and effectively avoids the occlusion of obstacles. Consequently, our scheme drives the drone to maintain a stable relative state and high-quality observation with the target.

C. High-Speed Dynamic Tracking and Perching Validation

Extensive experiments are conducted to validate dynamic tracking and perching in multiple scenarios. The system is tested
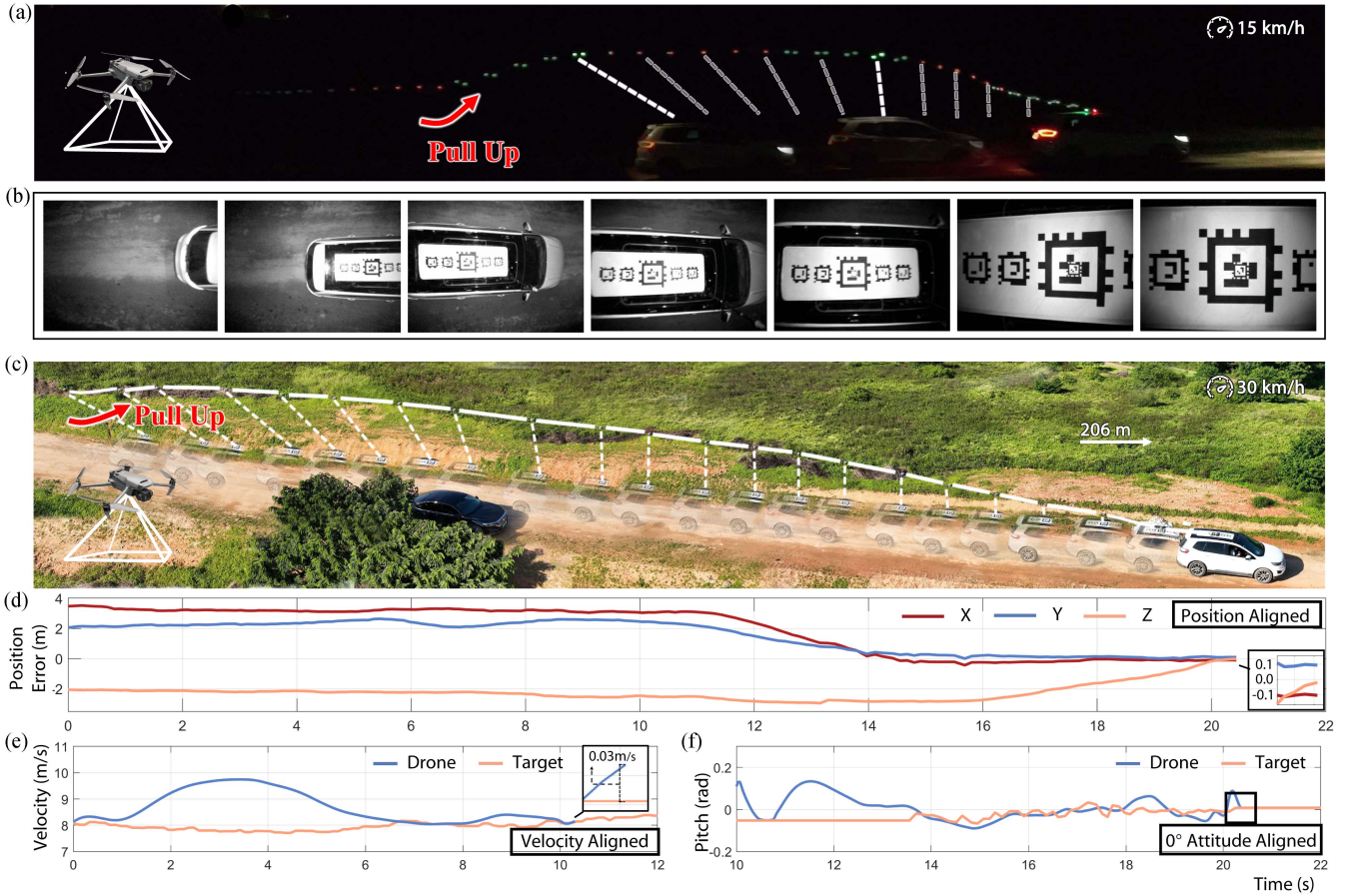
Fig. 20. (a) and (b) Illustration of the 15 km/h (4.2 m/s) tracking and horizontal perching real-world experiment in the night scene. (b) Images obtained by the down camera of the drone. (c)–(f) Illustration of the 30 km/s (8.3 m/s) high-speed tracking and horizontal perching. (d) Position error between the drone and the desired perching point. (e) Velocity curve of the drone and the SUV. (f) Pitch angle curve of the drone and the perching surface on the SUV.

TABLE VII
MULTIPLE SCENES REAL-WORLD EXPERIMENTS

Scene	Target Vel. (m/s)	Drone Vel. (m/s) (Max. / Avg.)	Target Acc. (m/s^2)	Error (cm)
Indoor	1.5	2.65 / 2.21	0.65	3.31
Road	4.0	5.13 / 4.76	1.01	4.47
Field Night	4.0	5.22 / 4.47	1.67	8.36
Field	4.0	5.37 / 4.69	1.12	5.87
	6.0	7.53 / 6.87	1.18	10.92
	8.0	9.51 / 8.44	1.31	14.19

The boldface values indicate the optimal cases in the corresponding test scenario.

TABLE VIII
MULTIPLE ATTITUDE DYNAMIC PERCHING EXPERIMENTS

Pitch angle	Target Vel. (m/s)	Drone Vel. (m/s) (Max. / Avg.)	Target Acc. (m/s^2)	Error (cm)
0.5 rd	0.0	1.63 / 0.82	0.27	2.05
	1.0	1.73 / 1.51	0.49	4.85
	2.0	3.04 / 2.33	0.60	6.29
1.0 rd	0.0	2.37 / 1.43	0.26	2.86
	1.0	2.84 / 1.91	0.42	7.01
	2.0	3.87 / 2.86	0.41	9.33

in high-speed situations that require an efficient, robust, and accurate tracking and perching scheme. The following tests present the system's high precision, system-level robustness, and active visibility enhancement. We first test horizontal perching in multiple scenarios. The results of the experiments are summarized in Table VII. Indoor and road scenarios have good road conditions, yet the field road provides for safe high-speed tests but with more uneven ground causing target shaking. These abundant scenes also demonstrate the application value. And the multiple desire terminal attitudes with different target speeds are also tested and summarized in Table VIII. For both tables, the maximum and average velocity of the drone during perching are

recorded. The average target acceleration estimation shows both real velocity changes and perception disturbance. Moreover, the final perching errors are listed in the table. We present three cases for more details as follows.

1) Case 1: 15 km/h Tracking and Perching at Night: In this case, we present the visibility-aware behavior resulting from our approach. Thanks to the spotlight on the bottom of the drone, the bottom detection camera is adaptable to night scenes. The visual perception range is set to $[0.0\,\text{m}, 2.0\,\text{m}]$.

As shown in Fig. 20(a), the initial relative height is less than 2.0 m, so the drone pulls itself up to ensure the observation of the target within the set range. Such behavior is not intentionally designed but rather emerges automatically due to the perception
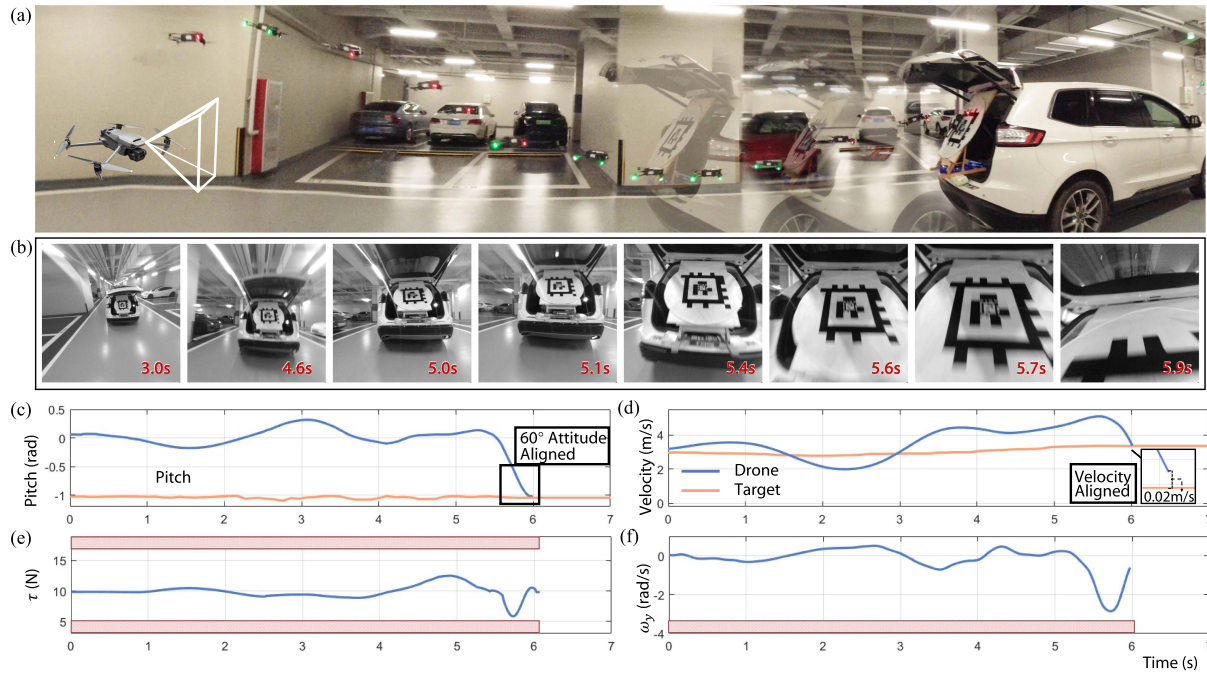
Fig. 21. Illustration of the large attitude dynamic tracking and perching real-world experiment. (a) Snapshot of the whole process. The vehicle moves at approximately 3.5 m/s, and the perching surface is $60°$ inclined. (b) Images obtained by the front camera of the quadrotor. (c) Pitch angle of the drone and the moving plane. (d) Velocity curve of the drone and the moving plane. (e) Thrust curve of the drone. (f) Angular velocity curve.

metric. The target tag is kept in the center of the image space as much as possible since its initial detection, which can enhance the target state estimation.

2) Case 2: 30 km/h High-Speed Tracking and Perching: In this case, we conduct high-speed tracking and perching on the top of the SUV. During the perching process, as shown in Fig. 20(b), the SUV's speed is 30 km/h (8.3 m/s), while the maximum speed of the drone reaches about 10 m/s. The final perching error on the $x - y$ plane is 10.7 cm. The inaccuracy is mostly caused by poor target estimating (due to detection and system link delays) and trajectory tracking errors. The wind and the ground effect are countered by the integrator control. Due to the turbulence of the vehicle, the attitude of the perching surface is shaking. The drone can obtain the plane's attitude during continuous observation and constantly replan, and finally ensure the attitude alignment at the end.

3) Case 3: Large Attitude Dynamic Tracking and Perching: In this case, we set a $60°$ inclined plane in the trunk of the SUV as the perching surface to validate the large attitude perching ability. Fig. 21 shows the snapshot of the whole process. For the sake of perception enhancement, the trajectory shape is similar to the simulated one with 1.5 rad pitch angle shown in Fig. 15. During the time interval from 3.0 to 5.7 s, the drone keeps the tag in the center of the image space as much as possible. The angular velocity and the thrust are both within the dynamic feasible region.

IX. CONCLUSION

In this article, we analyzed the core dilemmas to achieve tracking and perching in the dynamic scenario in detail and summarized five aspects of requirement accurately to solve the above problems. Then we, respectively, designed metrics to enhance visibility, guarantee safety, and dynamic feasibility in both tracking and perching stages. To adapt to the dynamic target state, we adjusted the terminal state of the drone accordingly to achieve state alignment. High-frequency spatiotemporal SE(3) trajectory optimization provided feasible tracking and perching trajectories. Finally, we deployed the adaptive tracking and perching scheme systematically on the commercial drone with a full-size SUV as the moving platform for real-world experiments. The results verified that our scheme was effective and had great application value.

REFERENCES

[1] D. N. Das, R. Sewani, J. Wang, and M. K. Tiwari, "Synchronized truck and drone routing in package delivery logistics," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 5772–5782, Sep. 2021.

[2] G. Wu, N. Mao, Q. Luo, B. Xu, J. Shi, and P. N. Suganthan, "Collaborative truck-drone routing for contactless parcel delivery during the epidemic," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 25077–25091, Dec. 2022.

[3] Y. Liu, Z. Liu, J. Shi, G. Wu, and W. Pedrycz, "Two-echelon routing problem for parcel delivery by cooperated truck and drone," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 51, no. 12, pp. 7450–7465, Dec. 2021.

[4] G. Wu, W. Pedrycz, H. Li, M. Ma, and J. Liu, "Coordinated planning of heterogeneous earth observation resources," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 46, no. 1, pp. 109–125, Jan. 2016.

[5] P. Tokekar, J. Vander, D. H. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1498–1511, Dec. 2016.

[6] C. Shen, Y. Zhang, Z. Li, F. Gao, and S. Shen, "Collaborative air-ground target searching in complex environments," in *Proc. IEEE Int. Symp. Saf. Secur. Rescue Robot.*, 2017, pp. 230–237.

[7] J. Delmerico, E. Mueggler, J. Nitsch, and D. Scaramuzza, "Active autonomous aerial exploration for ground robot path planning," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 664–671, Apr. 2017.

[8] S. Wu, W. Xu, F. Wang, G. Li, and M. Pan, "Distributed federated deep reinforcement learning based trajectory optimization for air-ground cooperative emergency networks," *IEEE Trans. Veh. Tech.*, vol. 71, no. 8, pp. 9107–9112, Aug. 2022.

[9] M. Mittal, R. Mohan, W. Burgard, and A. Valada, "Vision-based autonomous UAV navigation and landing for urban search and rescue," in *Proc. Int. Symp. Robot. Res.*, 2022, pp. 575–592.

[10] J. Ji, T. Yang, C. Xu, and F. Gao, "Real-time trajectory planning for aerial perching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 10516–10522.

[11] Q. Wang, Y. Gao, J. Ji, C. Xu, and F. Gao, "Visibility-aware trajectory optimization with application to aerial tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5249–5256.

[12] H. Cheng, L. Lin, Z. Zheng, Y. Guan, and Z. Liu, "An autonomous vision-based target tracking system for rotorcraft unmanned aerial vehicles," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1732–1738.

[13] A. G. Kendall, N. N. Salvapantula, and K. A. Stol, "On-board object tracking control of a quadcopter with monocular vision," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2014, pp. 404–411.

[14] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1696–1703, Jul. 2017.

[15] B. Penin, P. R. Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3725–3732, Oct. 2018.

[16] J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, "Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1762–1769, Jul. 2017.

[17] R. Bonatti, Y. Zhang, S. Choudhury, W. Wang, and S. Scherer, "Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming," in *Proc. Int. Symp. Exp. Robot*, 2018, pp. 119–129.

[18] Z. Han, R. Zhang, N. Pan, C. Xu, and F. Gao, "Fast-tracker: A robust aerial system for tracking agile target in cluttered environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2021, pp. 328–334.

[19] B. F. Jeon and H. J. Kim, "Online trajectory generation of a MAV for chasing a moving target in 3D dense environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 1115–1121.

[20] B. Jeon, Y. Lee, and H. J. Kim, "Integrated motion planner for real-time aerial videography with a drone in a dense environment," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 1243–1249.

[21] J. Ji, N. Pan, C. Xu, and F. Gao, "Elastic tracker: A spatio-temporal trajectory planner for flexible aerial tracking," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2022, pp. 47–53.

[22] A. Borowczyk, D.-T. Nguyen, A. P.-V. Nguyen, D. Q. Nguyen, D. Saussié, and J. L. Ny, "Autonomous landing of a multirotor micro air vehicle on a high velocity ground vehicle," *IFAC-PapersOnline*, vol. 50, no. 1, pp. 10488–10494, 2017.

[23] Y. Qi, J. Jiang, J. Wu, J. Wang, C. Wang, and J. Shan, "Autonomous landing solution of low-cost quadrotor on a moving platform," *Robot. Auton. Syst.*, vol. 119, pp. 64–76, 2019.

[24] J. Ghommam and M. Saad, "Autonomous landing of a quadrotor on a moving platform," *IEEE Trans. Aerosp. Elec. Syst.*, vol. 53, no. 3, pp. 1504–1519, Jun. 2017.

[25] P. Serra, R. Cunha, T. Hamel, D. Cabecinhas, and C. Silvestre, "Landing of a quadrotor on a moving target using dynamic image-based visual servo control," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1524–1535, Dec. 2016.

[26] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward image based visual servoing for aerial grasping and perching," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2014, pp. 2113–2118.

[27] D. Lee, T. Ryan, and H. J. Kim, "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 971–976.

[28] J. A. Macés-Hernández, F. Defaÿ, and C. Chauffaut, "Autonomous landing of an UAV on a moving platform using model predictive control," in *Proc. 11th Asian Control Conf.*, 2017, pp. 2298–2303.

[29] A. Mohammadi, Y. Feng, C. Zhang, S. Rawashdeh, and S. Baek, "Vision-based autonomous landing using an MPC-controlled micro UAV on a moving platform," in *Proc. Int. Conf. Unmanned Aircr. Syst.*, 2020, pp. 771–780.

[30] A. Paris, B. T. Lopez, and J. P. How, "Dynamic landing of an autonomous quadrotor on a moving platform in turbulent wind conditions," in *Proc. IEEE Int. Conf. Robot. Autom*, 2020, pp. 9577–9583.

[31] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza, "Vision-based autonomous quadrotor landing on a moving platform," in *Proc. IEEE Int. Symp. Saf. Secur. Rescue Robot.*, 2017, pp. 200–207.

[32] P. Wang, C. Wang, J. Wang, and M. Q.-H. Meng, "Quadrotor autonomous landing on moving platform," *Procedia Comput. Sci.*, vol. 209, pp. 40–49, 2022.

[33] J. M. Daly, Y. Ma, and S. L. Waslander, "Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays," *Auton. Robots*, vol. 38, pp. 179–191, 2015.

[34] T. Muskardin, G. Balmer, S. Wlach, K. Kondak, M. Laiacker, and A. Ollero, "Landing of a fixed-wing UAV on a mobile ground vehicle," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 1237–1242.

[35] H. Tsukagoshi, M. Watanabe, T. Hamada, D. Ashlih, and R. Iizuka, "Aerial manipulator with perching and door-opening capability," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4663–4668.

[36] L. Daler, A. Klaptocz, A. Briod, M. Sitti, and D. Floreano, "A perching mechanism for flying robots using a fibre-based adhesive," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 4433–4438.

[37] A. Kalantari, K. Mahajan, D. Ruffatto, and M. Spenko, "Autonomous perching and take-off on vertical walls for a quadrotor micro air vehicle," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 4669–4674.

[38] E. Barrett et al., "Autonomous battery exchange of UAVs with a mobile ground base," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 699–705.

[39] J. Thomas et al., "Aggressive flight with quadrotors for perching on inclined surfaces," *J. Mechanisms Robot.*, vol. 8, no. 5, 2016, Art. no. 051007.

[40] J. Mao, S. Nogar, C. M. Kroninger, and G. Loianno, "Robust active visual perching with quadrotors on inclined surfaces," *IEEE Trans. Robot.*, vol. 39, no. 3, pp. 1836–1852, Jun. 2023.

[41] J. L. Paneque et al., "Perception-aware perching on powerlines with multirotors," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 3077–3084, Apr. 2022.

[42] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Quadrotor landing on an inclined platform of a moving ground vehicle," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 2202–2207.

[43] B. Hu and S. Mishra, "Time-optimal trajectory generation for landing a quadrotor onto a moving platform," *IEEE/ASME Trans. Mechatronics*, vol. 24, no. 2, pp. 585–596, Apr. 2019.

[44] S. Liu, Z. Wang, X. Sheng, and W. Dong, "Hitchhiker: A quadrotor aggressively perching on a moving inclined surface using compliant suction cup gripper," *IEEE Trans. Autom. Sci. Eng.*, to be published, doi: 10.1109/TASE.2023.3263558.

[45] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1294–1310, Dec. 2015.

[46] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 7, no. 3, pp. 652–674, Sep. 2022.

[47] M. Watterson and V. Kumar, "Control of quadrotors using the Hopf fibration on SO(3)," in *Robotics Research.* Berlin, Germany: Springer, 2020, pp. 199–215.

[48] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 3259–3278, Oct. 2022.

[49] L. S. Jennings and K. L. Teo, "A computational algorithm for functional inequality constrained optimization problems," *Automatica*, vol. 26, no. 2, pp. 371–375, 1990.

[50] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3rd ed. New York, NY, USA: Cambridge Univ. Press, 2007.

[51] A. Domahidi and J. Jerez, "Forces professional," Embotech AG, 2014. [Online]. Available: https://embotech.com/FORCES-Pro

[52] M. Krogius, A. Haggenmiller, and E. Olson, "Flexible layouts for fiducial tags," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 1898–1903.

Yuman Gao received the graduation degree in control science and engineering in 2021 from Zhejiang University, Hangzhou, China, where he is concurrently pursuing a minor in advanced honor class of engineering education (ACEE), with Chu Kochen Honors College. He is currently working toward the Ph.D. degree in automation under the supervision of Fei Gao, with the FAST Lab, Zhejiang University.

His research interests include motion planning and autonomous navigation for multirobot systems.

Zhimeng Shang received the master's degree in mechanical engineering and automation from Beihang University, Beijing, China, in 2015.

Since 2015, he has been with DJI, Nanshan, China, focusing on multirotor control and planning. He participated in the development of DJI's Mavic/Phantom/Inspire/FPV series, as well as flight controllers like A3/N3. His research interests include classic and optimal control, motion planning, and UAV design and control.

Jialin Ji received the graduation degree in mechatronic engineering in 2019 from the School of Mechanical Engineering, Zhejiang University, Hangzhou, China, and a minor in advanced honor class of engineering education (ACEE) from Chu Kochen Honors College, Hangzhou, China. He is currently working toward the Ph.D. degree in automation under the supervision of Fei Gao, with FAST Lab, Zhejiang University, working on motion planning and autonomous navigation for aerial robotics.

Yanjun Cao received the Ph.D. degree in computer and software engineering from the University of Montreal, Montreal, Canada, in 2020.

He is currently an Associate Researcher with the Center of Swarm Navigation, Huzhou Institute, Zhejiang University, Hangzhou, China. He leads the Field Intelligent Robotics Engineering (FIRE) group of the Field Autonomous System and Computing Lab (FAST Lab). His research interests include key challenges in multirobot systems, such as collaborative localization, autonomous navigation, perception, and communication.

Qianhao Wang (Student Member, IEEE) received the M.Eng. degree in 2022 in control science and engineering from Zhejiang University, Hangzhou, China, where he is currently working toward the Ph.D. degree in control science and engineering.

His research interests include motion planning, computational geometry, LiDAR SLAM, and autonomous navigation for aerial robotics.

Shaojie Shen received the B.Eng. degree in electronic engineering from the Hong Kong University of Science and Technology, Hong Kong, (HKUST) in 2009, the M.S. degree in robotics and the Ph.D. degree in electrical and systems engineering from the University of Pennsylvania, PA, USA, in 2011 and 2014, respectively.

In 2014, he joined the Department of Electronic and Computer Engineering, HKUST as an Assistant Professor, and was promoted to Associate Professor in 2020. His research interests include robotics and unmanned aerial vehicles, with focus on state estimation, sensor fusion, computer vision, localization and mapping, and autonomous navigation in complex environments.

Rui Jin received the B.E. degree in mechanical design & manufacturing and their automation from Northwestern Polytechnical University, Xi'an, China, in 2021. He is currently working toward the M.Eng. degree in robotics with the FAST Lab, Zhejiang University, Hangzhou, China.

Her research interests include the design, control, and motion planning of coaxial helicopters.

Chao Xu (Senior Member, IEEE) received the Ph.D. degree in mechanical engineering from Lehigh University, Bethlehem, PA, USA, in 2010.

He is currently an Associate Dean and Professor with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China. He is also the inaugural Dean of ZJU Huzhou Institute, and plays the role of a Managing Editor for *IET Cyber-Systems and Robotics*. He has authored or coauthored more that 100 papers in international journals, including *Science Robotics*, *Nature Machine Intelligence*, etc. His research interests include flying robotics and control-theoretic learning.

Yi Lin received the B.Eng. degree in software engineering from the Harbin Institute of Technology, Harbin, China, in 2015, and the M.Phil. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2017.

He is currently a Senior Engineer with DJI Technology Company, Ltd., Nanshan, China. His research interests include computer vision, simultaneous localization and mapping, 3-D reconstruction, autonomous navigation, unmanned aerial vehicles, motion planning, and optimization.

Fei Gao (Member, IEEE) received the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2019.

He is currently a tenured Associate Professor with the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, where he leads the Flying Autonomous Robotics (FAR) group affiliated with the Field Autonomous System and Computing (FAST) Laboratory. His research interests include aerial robots, autonomous navigation, motion planning, optimization, and localization and mapping.