

# Visibility-aware Trajectory Optimization with Application to Aerial Tracking

Qianhao Wang, Yuman Gao, Jialin Ji, Chao Xu, and Fei Gao

**Abstract**—The visibility of targets determines performance and even success rate of various applications, such as active slam, exploration, and target tracking. Therefore, it is crucial to take the visibility of targets into explicit account in trajectory planning. In this paper, we propose a general metric for target visibility, considering observation distance and angle as well as occlusion effect. We formulate this metric into a differentiable visibility cost function, with which spatial trajectory and yaw can be jointly optimized. Furthermore, this visibility-aware trajectory optimization handles dynamic feasibility of position and yaw simultaneously. To validate that our method is practical and generic, we integrate it into a customized quadrotor tracking system. The experimental results show that our visibility-aware planner performs more robustly and observes targets better. In order to benefit related researches, we release our code to the public.

## I. INTRODUCTION

In recent years, the progress on various aspects of autonomous robots makes it possible to accomplish complex, systematic tasks. When executing tasks, robots usually equip sensors to obtain external environment information. However, the sensors usually have a limited field of view (FOV), especially for quadrotors under the size, weight, and power (SWaP) constraints. Therefore, the robot's position and orientation should be adjusted to obtain better visibility to the target.

Visibility is of vital importance in a variety of applications. We take three scenarios as instances: **1)** For visual-inertial state estimation, enhancing the visibility of co-visible features can increase the state estimation accuracy significantly [1, 2]. **2)** For exploration tasks, the target's visibility determines the quality and confidence of obtained information [3, 4]. **3)** For tracking, keeping high visibility of target is the key to avoid target loss which is the main reason for mission failure [5, 6]. In conclusion, different circumstances put forward the same requirement: visibility-aware trajectory planning.

However, only a few works take visibility into account in trajectory planning. Most of them use probabilistic or deterministic searching methods with visibility cost to deal with the problem. Whereas these methods suffer from inherent inaccuracy caused by discretization. Besides, when the dimension of state increases, the computational complexity rises exponentially. These adverse properties lead to planning

All authors are with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China, and also with the Huzhou Institute of Zhejiang University, HuZhou 313000, China. E-mails:{qhwangaa, ymgao, jlji, cxu, fgaoaa}@zju.edu.cn

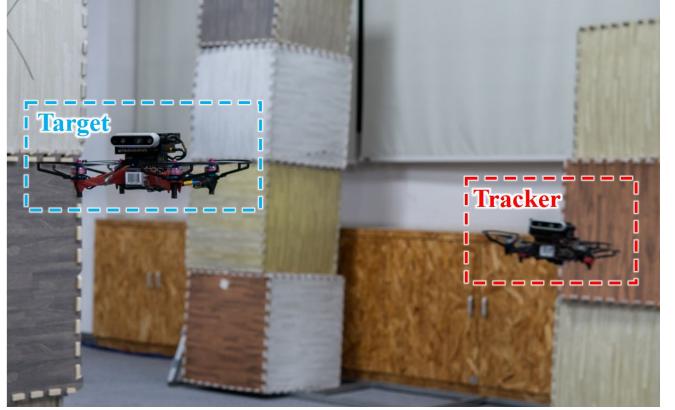


Fig. 1: The real-world experiment of the aerial tracking application using our method.

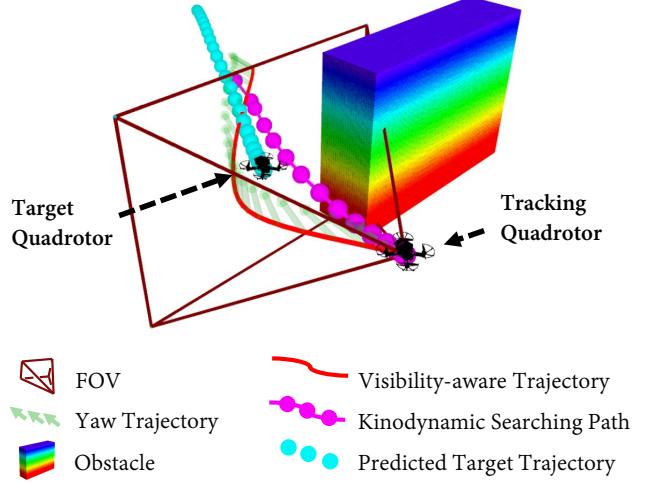


Fig. 2: Illustration of the visibility-aware trajectory result in the application of aerial tracking.

failures, especially with a limited time budget. Additionally, yaw planning approaches for visibility are usually absent. A group of works directly set facing-target yaw as expectation input in control level [7, 8]. However, in practice, the robot's yaw rate, named as yaw feasibility in this paper, can not be changed suddenly, same as the velocity and acceleration of position. (The feasibility mentioned in the following includes yaw feasibility by default) The above control-level strategy fails to reach the expected yaw sometimes due to physical limits. Some other works plan yaw after position, which omits the coupling relationship between them when considering visibility. To summarize, an efficient trajectory planning method that takes visibility into consideration comprehensively is rare.

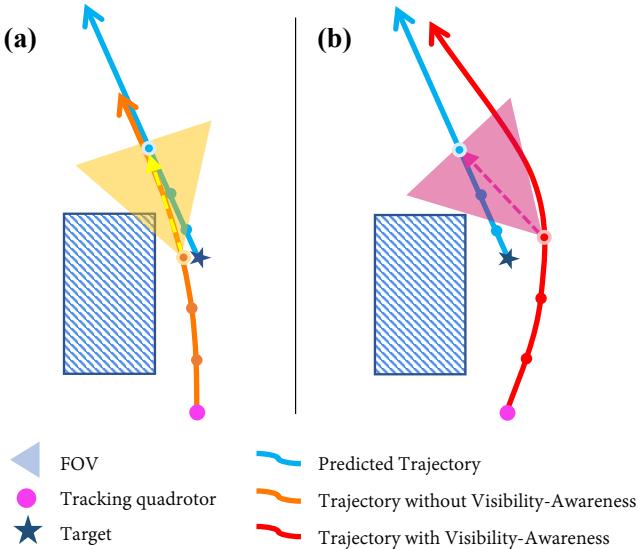


Fig. 3: Comparison of trajectories with and without visibility-awareness. The balls on the trajectory represent the position in time. Without considering visibility, the orange trajectory is easy to lose target because of occlusion, especially when the target makes movements that are hard to predict. (a): the trajectory without visibility-awareness. (b): the trajectory with visibility-awareness.

To bridge this gap, we propose a general optimization-based visibility-aware trajectory planning method. This method uses a continuous and differentiable polynomial that prevents dimension explosion when the planning problem scales up. Furthermore, our method optimizes yaw jointly with position considering visibility, safety, and feasibility. The visibility is defined for a specific target or an area of interest. According to practical experience, we summarize three parts of the visibility criteria:

- Distance of observation (DO):* A moderate distance from target is expected.
- Angle of observation (AO):* The axis of the sensor's FOV is expected to be straight towards the target.
- Occlusion effect against obstacles (OE):* The trajectory prevents the line of sight towards the target from occlusion. Without considering *OE*, occlusion is prone to happen, as in Fig.3, for instance.

To provide a gradient of visibility in trajectory optimization, we design a differentiable visibility cost considering the above parts. To guarantee the joint optimality of position and observation angle, we design a joint optimizer that simultaneously optimizes position and yaw using the visibility cost, safety, and feasibility cost. The planned motions provide the robot with better visibility to a target, as Fig.2 is shown.

Finally, to prove that our method is practical and generic, we integrate the proposed method into a customized quadrotor tracking system, with a kinodynamic occlusion-avoid searching front-end to generate an initial path. We compare the tracking system using our method with the cutting-edge tracking works. The experimental results show that our visibility-aware planner performs more robustly in tracking.

The contributions of this paper are:

- 1) A general differentiable visibility cost taking the aforementioned *DO*, *AO* and *OE* into account.
- 2) Considering visibility, safety and dynamic feasibility, a joint trajectory optimizer optimizes position and yaw of a robot simultaneously.
- 3) Simulations and real-world tests in aerial tracking application validate that our method is practical and generic. Moreover, we make our method open source <sup>1</sup>.

## II. RELATED WORK

The related works of trajectory planning to maximize visibility can be divided into two groups: local control based methods and trajectory generation based methods.

### A. Local Control Based Methods

Several previous works formulate the trajectory planning and control integrally as a local control problem. Penin et al. [9] replan the trajectory online considering the *OE* constraint by solving an optimization problem in the image space. Merging re-active control and planning, this method directly obtains the optimal control command. Lacking the environmental perception, this approach requires complete knowledge of the environment, thus not applicable in real applications. Andersen et al. [10] consider *OE* in trajectory planning to get more information for vehicle overtaking. They generate motions by maximizing the visibility ahead of obstacles with an MPC receding horizon planner. However, this task-specific method is hard to extend to other scenarios. Nägeli et al. [11] also take *OE* and collision simultaneously into account in an MPC planner. A modular cost function based on the re-projection error of targets is designed to account for visibility. While this method fails to maintain its safety against obstacles with arbitrary shape since it assumes that all the obstacles are shaped by ellipsoid.

### B. Trajectory Generation Based Methods

Trajectory generation based methods take the visibility into account and guarantee the smoothness and feasibility simultaneously. Bonatti et al. [12] present a real-time covariant gradient descent method to trade-off smoothness, obstacle avoidance, and *OE* in tracking trajectory planning. Their *OE* cost requires the connection line between the target and the robot to be free of obstacles within a certain range. However, this *OE* cost does not take the conical FOV shape of most sensors into account. Bandyopadhyay et al. [6] define the shortest distance that the target escapes from the chasing robot's visible region as a risk function for *OE*. Nevertheless, the precise geometry of the robot's visible region is required. In contrast to [6], several researches [7, 13] design the minimum value of the Euclidean Signed Distance Field (ESDF) on the line between the target and the robot as visibility metric for *OE*. However, because this metric is not differentiable, it is not capable of being used to optimize visibility directly. To deal with visibility, even if the metric is differentiable, Jeon et al. [7] propose a bi-level tracking planner. They search a graph incorporating safety

<sup>1</sup><https://github.com/ZJU-FAST-Lab/visPlanner>

and *OE* metric to obtain a chasing corridor used for trajectory generation. However, building and traversing a graph is time-consuming. Typically, even with a low speed (1.2m/s) and coarse grid map (0.5 m) resolution, this graph-search based method takes over 200 ms [14]. With similar *OE* metric, Zhou et al. [5] propose a risk-aware trajectory refining strategy. Instead of explicitly optimizing *OE*, to achieve risk-awareness, they use a task-specific method that guarantees sufficient distance to stop if the trajectory passes over a previously unknown obstacle. However, we formulate our general visibility metric into a differentiable cost, enabling us to optimize visibility explicitly.

Most works of trajectory optimization considering visibility only take *OE* into account, while as mentioned in Sec.I, rare works plan a trajectory for *AO* and *DO*. Some works [7, 8] greedily choose the yaw angle facing the target all the time, omitting the yaw feasibility. Other methods such as Zhou et al. [5] optimize the yaw trajectory that passes through a yaw sequence searched by a graph while never joints with the position. There is a strong mutual influence between position and yaw when considering visibility. Our method considers *AO*, with which we are capable of optimizing yaw with position jointly.

### III. PRELIMINARIES

#### A. Problem Statement

Given the sequence of a chosen target position in time, the joint optimizer's goal is to generate a trajectory that maximizes the target's visibility, guaranteeing safety and dynamic feasibility. Here the visibility is defined as the three parts above: *DO*, *AO* and *OE*.

We consider the problem based on the following assumptions:

- 1) Only one target can be chosen at the same time.
- 2) The target is static or has smooth and bounded velocity and acceleration.
- 3) The sensor equipped directly in front of robot has conical limited field of view.

#### B. Analytical Visibility Metric

Notating the position of the robot  $\mathbf{p} = [x_p, y_p, z_p] \in \mathbb{R}^3$ , Euler-yaw angle of the robot  $\psi_p \in SO(2)$  and the position of the target  $\mathbf{c} = [x_c, y_c, z_c] \in \mathbb{R}^3$ , we formulate the details of the three parts of visibility metric as follows.

1) *DO*: The target is expected to be observed in a proper distance range from the robot denoted by

$$od_{min} < \|\mathbf{p} - \mathbf{c}\| < od_{max}, \quad (1)$$

where the  $od_{min}$  and  $od_{max}$  are the lower and upper bounds of the optimal distance of observation.

2) *AO*: In order to keep the axis of the sensor's FOV straight towards the target, the yaw angle is expected to be equal to  $\psi_{best}$  defined as Eq.2.

$$\psi_{best} = \text{atan2}(\mathbf{e}_y^T(\mathbf{p} - \mathbf{c}), \mathbf{e}_x^T(\mathbf{p} - \mathbf{c})), \quad (2)$$

where  $\mathbf{e}_x = [1, 0, 0]^T$  and  $\mathbf{e}_y = [0, 1, 0]^T$ .

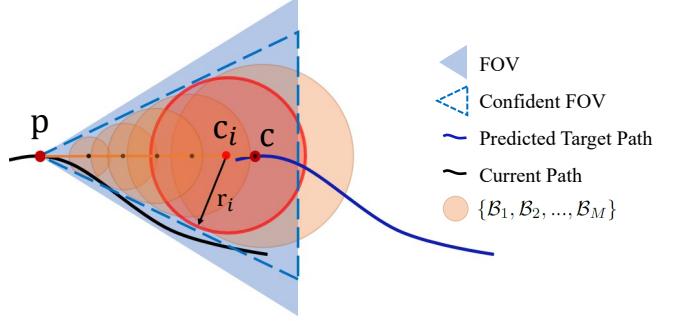


Fig. 4: Illustration of the definition of *OE* metric. A robot at position  $\mathbf{p}$  observes the target  $\mathbf{c}$ . A sequence of ball-shaped areas in orange are used to approximate the confident FOV expected to be obstacle-free.

3) *OE*: As the two-dimension profile illustrates in Fig.4, the blue range represents the FOV of the robot. The blue dashed enveloped area represents the **confident FOV** which we expect the target within and no obstacles included. However, this requirement cannot be represented analytically. Therefore, we approximate the confident FOV with a sequence of ball-shaped areas  $\{\mathcal{B}_1, \mathcal{B}_2, \dots, \mathcal{B}_M\}$ . A ball  $\mathcal{B}_i$  is shown as the red circle with its center  $\mathbf{c}_i$  and radius  $r_i$  calculated by

$$\mathbf{c}_i = \mathbf{p} + \lambda_i(\mathbf{c} - \mathbf{p}), \quad (3)$$

$$r_i = \rho \cdot \lambda_i \cdot \|\mathbf{p} - \mathbf{c}\|, \quad (4)$$

where  $\lambda_i = i/M \in [0, 1]$ , and  $\rho$  is a constant determined by the size of confident FOV. Then we can guarantee *OE* analytically for each ball by

$$r_i < \Xi(\mathbf{c}_i), \quad (5)$$

where  $\Xi(\mathbf{c}_i) : \mathbb{R}^3 \rightarrow \mathbb{R}$  is the distance to the closest obstacle.

## IV. VISIBILITY-AWARE TRAJECTORY OPTIMIZATION

#### A. Trajectory Representation

We require a joint trajectory optimizer, which can generate a dynamic-feasible and collision-free trajectory considering smoothness and visibility simultaneously.

The trajectory is parameterized by a four-dimensional unclamped uniform B-spline curve, which is a piecewise polynomial uniquely determined by its degree  $p_b$ , a knot span  $\Delta t$ , and  $N_c$  control points  $\{\mathbf{Q}_i, \Phi_i\} \in \mathbb{R}^3 \times SO(2)$ . We follow the work of [15] to plan the control points of positions  $\mathbf{Q}_i$  and yaw angles  $\Phi_i$  in a reduced space of the four selected differentially flat outputs  $\{x, y, z, \psi\}$ .

#### B. Objective Functions

In practice, we choose  $p_b = 3$ , then the total duration of the trajectory is  $(N_c - 3) \cdot \Delta t$ . Since some of the objective functions are represented by the expressions of each waypoint  $\{\mathbf{p}_k, \psi_{p,k}\}$  and distance  $d_k$  to the corresponding target  $\mathbf{c}_k$ , they should be calculated in advance:

$$\mathbf{p}_k = \frac{1}{6}(\mathbf{Q}_k + 4\mathbf{Q}_{k+1} + \mathbf{Q}_{k+2}), \quad (6)$$

$$d_k = \|\mathbf{p}_k - \mathbf{c}_k\|, k \in \{1, 2, \dots, N_c - 2\}. \quad (7)$$

Besides,  $\psi_{p,k}$  can be calculated by yaw control point  $\Phi_k$  similarly as Eq.6.

Additionally, we use a  $C^2$  penalty function  $g(x) = \max\{0, x\}^3$  to convert all constraints to penalty terms, and the final unconstrained optimization problem is given by

$$\min_{Q, \psi} J = [J_{DO}, J_{AO}, J_{OE}, J_f, J_{f_\Phi} J_s, J_{s_\Phi}, J_c] \cdot \omega, \quad (8)$$

where  $\omega$  is a weighing vector for trading off each cost:

1) *DO Cost*: To achieve proper observation distance, each waypoint  $\mathbf{p}_k$  is required to satisfy *DO* constraint defined by Eq.1. The *DO* cost is written by:

$$J_{DO} = \sum_{k=1}^{N_c-2} g(od_{min}^2 - d_k^2) + g(d_k^2 - od_{max}^2). \quad (9)$$

2) *AO Cost*: The yaw angle is expected to be close to the best yaw  $\psi_{best}$  defined by Eq.2. The *AO* cost is written by:

$$J_{AO} = \sum_{k=1}^{N_c-2} (\psi_{p,k} - \psi_{best,k})^2. \quad (10)$$

Different from other costs, this gradient affects both position  $\mathbf{p}_k$  and yaw  $\psi_{p,k}$ :

$$\frac{\partial J_{AO}}{\partial \psi_{p,k}} = 2(\psi_{p,k} - \psi_{best,k}), \quad (11)$$

$$\frac{\partial J_{AO}}{\partial \mathbf{p}_k} = \left[ \frac{\partial J_{AO}}{\partial x_{p,k}}, \frac{\partial J_{AO}}{\partial y_{p,k}}, 0 \right]^T, \quad (12)$$

where

$$\frac{\partial J_{AO}}{\partial x_{p,k}} = \frac{2(\psi_{p,k} - \psi_{best,k})}{(\mathbf{e}_x^T \mathbf{L})^2 + (\mathbf{e}_y^T \mathbf{L})^2} \cdot \mathbf{e}_y^T \mathbf{L}, \quad (13)$$

$\partial J_{AO} / \partial y_{p,k}$  can be calculated similarly. With Eq.6, we can propagate the gradient back to control points.

3) *OE Cost*: To satisfy the requirement defined by Eq.5, the *OE* cost is written by:

$$J_{OE} = \sum_{k=1}^{N_c-2} \sum_{i=1}^M g(F_{OE}(\mathbf{p}_k)), \quad (14)$$

$$F_{OE}(\mathbf{p}_k) = (r_{k,i}^2 - \Xi^2(\mathbf{c}_{k,i})), \quad (15)$$

where  $\Xi(\mathbf{c}_{k,i})$  is obtained from ESDF, and  $M$  is the number of ball-shaped areas mentioned in Sec.III-B.3.

The gradient of  $F_{OE}$  cost can be written as:

$$\frac{\partial F_{OE}}{\partial \mathbf{p}_k} = 2 \left[ \rho r_{k,i} \lambda_i \cdot \frac{\partial d_k}{\partial \mathbf{p}_k} - (1 - \lambda_i) \Xi(\mathbf{c}_{k,i}) \frac{\partial \Xi(\mathbf{c}_{k,i})}{\partial \mathbf{c}_{k,i}} \right], \quad (16)$$

where  $\partial \Xi(\mathbf{c}_{k,i}) / \partial \mathbf{c}_{k,i}$  is easily obtained from ESDF. The term  $\lambda_i$  acts as a lever for the gradient. Then we use Eq.6 to propagate the gradient back to control points.

4) *Dynamic Feasibility Cost*: The control points of velocity  $\mathbf{V}_i$ , acceleration  $\mathbf{A}_i$  and jerk  $\mathbf{J}_i$  are calculated by

$$\mathbf{V}_i = \frac{\mathbf{Q}_{i+1} - \mathbf{Q}_i}{\Delta t}, \mathbf{A}_i = \frac{\mathbf{V}_{i+1} - \mathbf{V}_i}{\Delta t}, \mathbf{J}_i = \frac{\mathbf{A}_{i+1} - \mathbf{A}_i}{\Delta t}. \quad (17)$$

The control points of yaw velocity  $V_{\Phi,i}$ , yaw acceleration  $A_{\Phi,i}$  and yaw jerk  $J_{\Phi,i}$  can also be calculated by  $\Phi$  similarly. Benefiting from the convex-hull property, dynamic feasibility of the whole trajectory can be guaranteed sufficiently by

$$J_f = \sum_{i=1}^{N_c-1} g(\|\mathbf{V}_i\|^2 - v_m^2) + \sum_{i=1}^{N_c-2} g(\|\mathbf{A}_i\|^2 - a_m^2), \quad (18)$$

where  $v_m, a_m$  are the limits on velocity and acceleration. Similarly, the yaw feasibility cost can be written as:

$$J_{f_\Phi} = \sum_{i=1}^{N_c-1} g(\|V_{\Phi,i}\|^2 - v_{\Phi,m}^2) + \sum_{i=1}^{N_c-2} g(\|A_{\Phi,i}\|^2 - a_{\Phi,m}^2), \quad (19)$$

where  $v_{\Phi,m}, a_{\Phi,m}$  are the limits on velocity and acceleration of yaw.

5) *Smoothness Cost*: Benefiting from the convex hull property, we minimize the control points of high order derivatives of the B-spline trajectory to guarantee the smoothness:

$$J_s = \sum_{i=1}^{N_c-3} \|\mathbf{J}_i\|_2^2, \quad (20)$$

Similarly, we define smoothness cost of yaw  $J_{s_\Phi}$  using  $J_{\Phi,i}$ .

6) *Collision Cost*: To guarantee that the entire trajectory is collision-free, the collision penalty serves as the repulsive force that pushes the control points away from obstacles.

$$J_c = \sum_{i=1}^N g(d_{thr}^2 - \Xi^2(\mathbf{Q}_i)) (\Xi(\mathbf{Q}_i)), \quad (21)$$

where  $d_{thr}$  is the safety threshold distance.

## V. APPLICATION ON QUADROTOR TRACKING

### A. Motivation and Methodology

For tracking, keeping high visibility of target is critical to preventing target loss which is the main reason for failures. Although with a requirement of visibility in the existing tracking systems, they rarely consider the complete metrics we summarize through practical experiments. To validate our method, we integrate the proposed optimizer into our previous work [14], a customized quadrotor tracking system. Its back-end is replaced by the proposed method to generate trajectory considering visibility and yaw feasibility. The predicted target position is generated by the same method based on Bézier regression in our previous work.

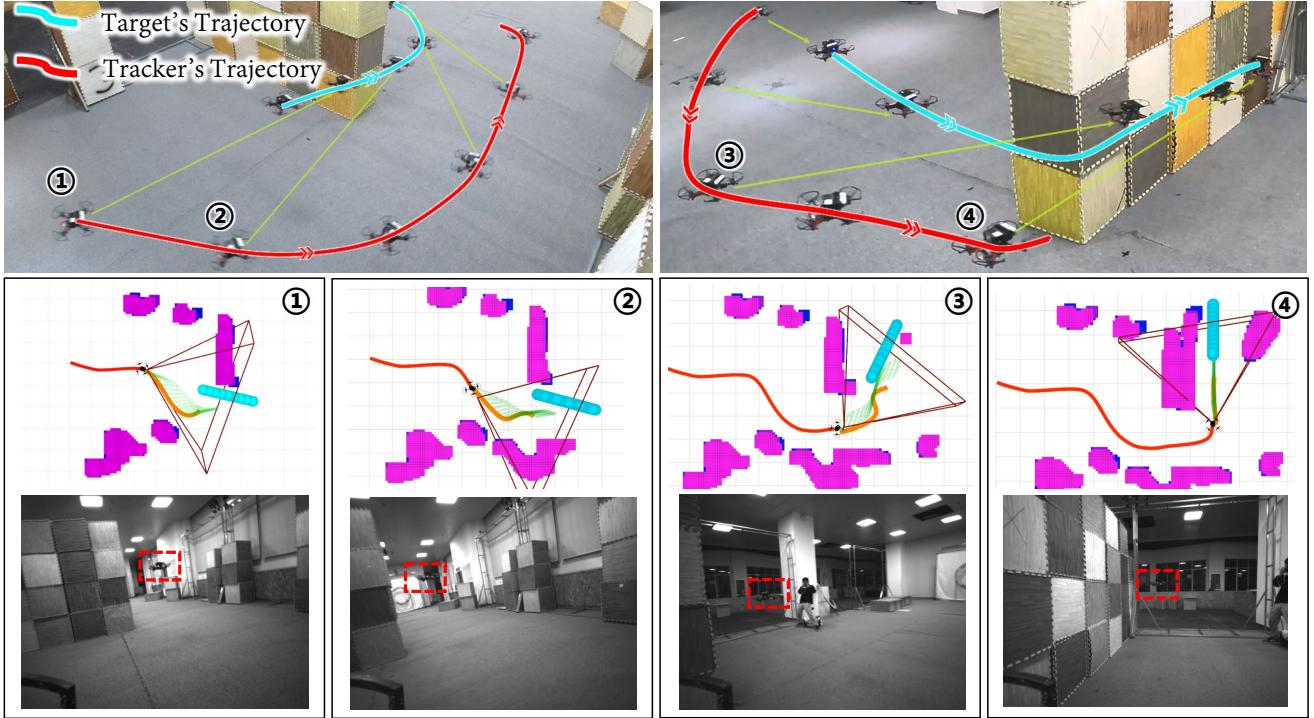


Fig. 5: The real-world experiment. Four moments ①, ②, ③, ④ are extracted from the entire experiment. **Top:** the snapshots of the tracking quadrotor and the target quadrotor. **Middle:** the visualization of the algorithm and the experiment. For the tracking quadrotor, the red curve denotes the executed trajectory, the orange curve denotes the planned trajectory, and the green arrows represent the yaw trajectory. Blue balls depict the predicted trajectory of the target. The red pyramid stands for FOV. **Bottom:** the first-person perspective of the tracking quadrotor. The target is boxed out by the red dotted box.

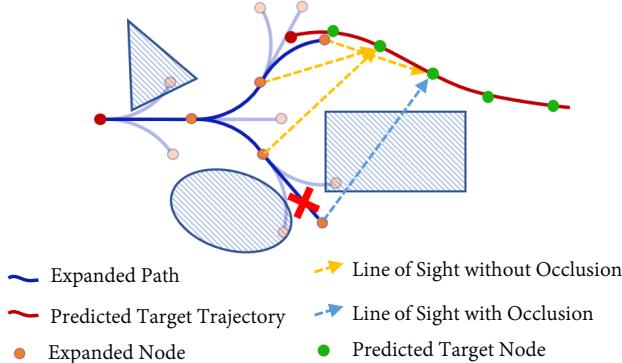


Fig. 6: Illustration of the kinodynamic occlusion-avoid searching method. The expanded node will be rejected if the connection line between the node and the target is occluded, such as the red cross.

### B. Implementation Details

For the front-end, we use the kinodynamic occlusion-avoid searching method based on hybrid-state A\* algorithm [15]. It uses motion primitives to expand nodes and evaluates them for a safe and dynamically feasible trajectory. To guarantee the topological equivalence between the planning path and the target's predicted path, we design a feasibility check function that prevents occlusion between the robot and the target, as Fig.6 is shown.

For the back-end, we apply our visibility-aware joint optimization method. Furthermore, a safe tracking cost  $\omega_v \cdot J_v$  is added to keep the soon-to-arrive environment known. This cost tends to restrict the velocity direction into the

quadrotor's FOV:

$$J_v = \sum_{k=1}^{N_c-2} g \left( (\psi_{v,k} - \psi_{p,k})^2 - \psi_{thr}^2 \right), \quad (22)$$

where  $\psi_v$  is the velocity direction that equals to the tangent direction of the trajectory at the point  $p$ , and  $\psi_{thr}$  is the maximum allowable deviation angle from FOV axis.

## VI. RESULTS

We incorporate our method into a quadrotor tracking system for testing. We set another high-speed quadrotor as the target. Due to the unknown intent of the target, the future position is predicted by history. The quadrotor performs real-time mapping since the environment is unknown. The goal of the tracking system is to follow the target and keep it in FOV. The parameters of optimization are shown in Tab.I.

TABLE I: Parameters of Optimization.

$od_{min}$	$od_{max}$	$\rho$	$M$
2.5m	3.5m	0.8	10

TABLE II: Parameters for Experiments.

	Target Quadrotor Max Velocity	Tracking Quadrotor Max Velocity
Real-world	1.5m/s	2.5m/s
Simulation	2.5m/s	5.0m/s

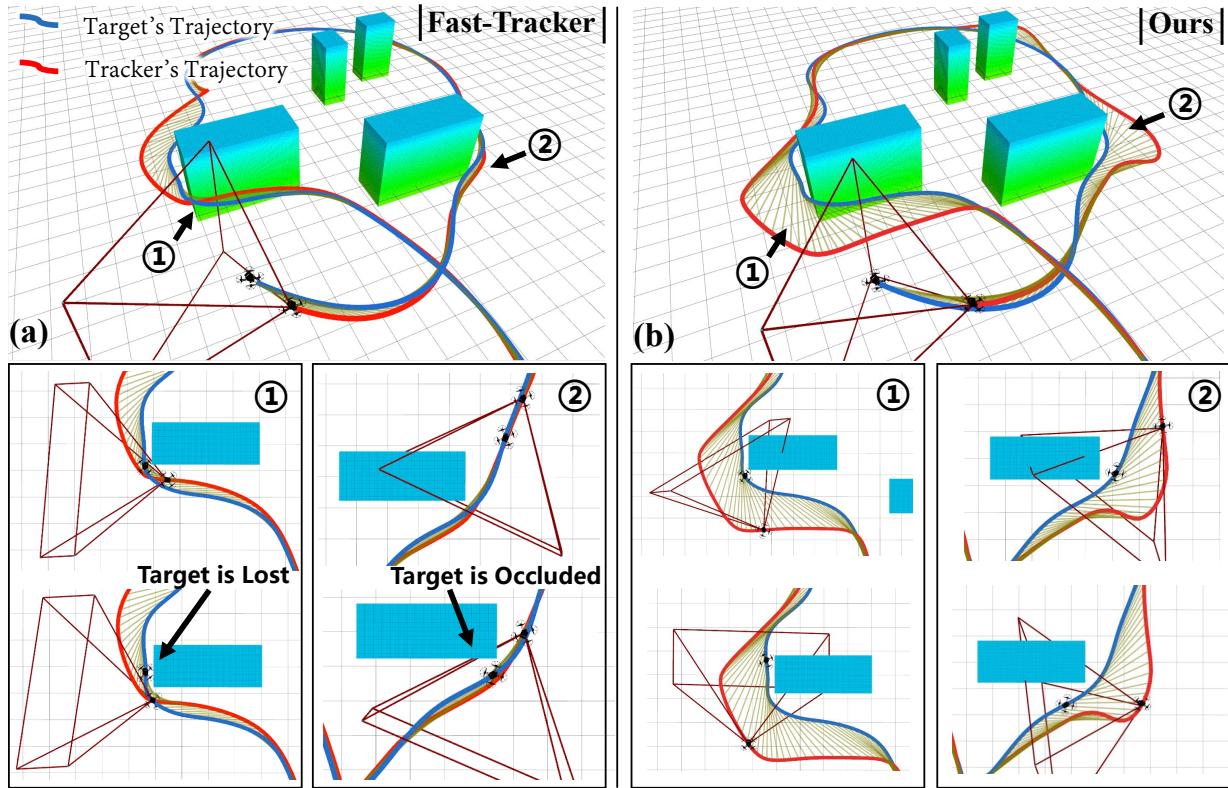


Fig. 7: The simulation results for Case 1. The trajectory of the tracking quadrotor and the target quadrotor are denoted as red and blue, respectively. The yellow lines connect their positions at the corresponding time. **We use the same color scheme in the following figures of simulation experiments.** The bottom snapshots show the situation at two corners where occlusion can easily occur. (a): the result of Fast-Tracker. (b): the result of our method.

We conduct two cases and general tests in the simulation and deploy the system on a quadrotor for real-world experiments. For simulation, we benchmark Fast-Tracker [14] for its cutting-edge performance in tracking. Two special cases are conducted to demonstrate the performance of the proposed method. Case.1 and Case.2 present the validity of our method under *OE* and *AO* metric, respectively. In the general tests, the tracking system runs for a long time in several randomly generated complex environments to comprehensively test our method’s visibility effect and robustness. We further demonstrate the practical performance of the algorithm through real-world experiments.

#### A. Real-World Experiments

Real-world experiments are presented on the same quadrotor platform of [16], which is localized by a robust visual-inertial state estimator [17]. The tracking quadrotor is equipped with an Intel Realsense D435 that has FOV ( $H \times V$ ) =  $86^\circ \times 57^\circ$  and an onboard computer DJI Manifold 2C. The target is an autonomous quadrotor as well broadcasting its location to avoid the impact of the identification, since our proposed method mainly focuses on visibility-aware trajectory generation. In this experiment, the max velocities of the target and the tracking quadrotor are set as Tab.II.

In the experiment scene, the target swerves behind an obstacle, which easily causes target loss. As Fig.5 shows, to avoid the line of sight toward the target being occluded, the tracking quadrotor adjusts its trajectory to keep the confident

FOV obstacle-free. According to the tracking quadrotor camera data, the target is in the the FOV throughout the whole experiment. Furthermore, the tracking trajectory keeps smooth and dynamic feasible. Snapshots and visualization of the experiment can be found in Fig.5. We refer readers to the video for more information<sup>2</sup>.

#### B. Simulation and Benchmark Comparisons

We benchmark our method with Fast-Tracker presented in the [14] in simulation. In Fast-Tracker, they searches for a safe tracking trajectory heuristically by a target informed kinodynamic searching method as the front-end. The back-end optimizer then refines the trajectory into a local spatial-temporal optima trajectory [18]. To compare the tracking performance fairly, we set both of them the same target motion prediction and an appropriate FOV ( $H \times V$ ) =  $80^\circ \times 65^\circ$ . Furthermore, we define the target is lost when occlusion occurs or it is out of the FOV.

1) *Case 1: OE* metric is validated. In this scene, the target makes sharp turns behind obstacles. Fast-Tracker never takes occlusion into account, which causes that the target is easily obstructed by obstacles as shown in Fig.7(a). In contrast, considering *OE*, the planner with our method generates a trajectory that prevents the line of sight towards the target from occlusion, as shown in Fig.7(b).

<sup>2</sup><https://www.youtube.com/watch?v=PhhrOBx54YY>

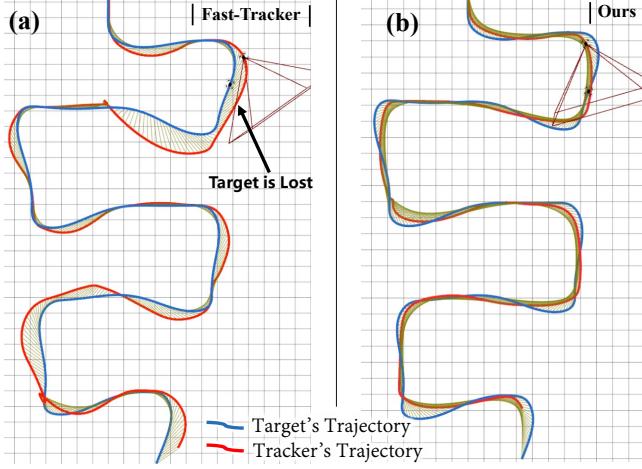


Fig. 8: The test environment of Case 2 for *AO* and *DO*. (a): the result of Fast-Tracker. (b): the result of our method.

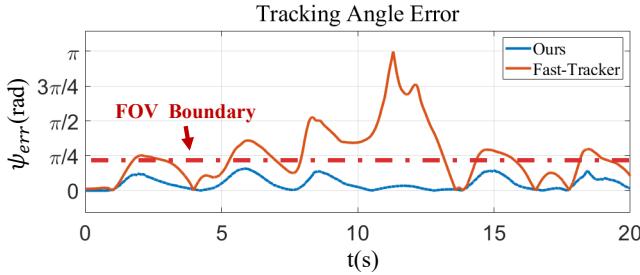


Fig. 9: Comparison of the tracking angle error . The tracking angle error of our method is remarkably less than Fast-Tracker. For our method,  $\psi_{err}$  is below FOV boundary for the whole time. In contrast, in Fast-Tracker, the target is always out of FOV bounds, which leads to target loss.

2) *Case 2: AO, DO* metric are validated. In this scene, the target makes large turns that are always more than  $90^\circ$ . Omitting a moderate distance and yaw angle to observe the target, Fast-Tracker loses it easily when turning, as shown in Fig.8. In contrast, taking *AO* and *DO* into account, our method jointly optimizes position and yaw, which makes the quadrotor quickly turn toward the target even if there are many big turns. For comparison, we define tracking angle error  $\psi_{err} = |\psi_p - \psi_{best}|$ . Fig.9 shows the comparison of  $\psi_{err}$  in the whole time. The FOV boundary equals half of the FOV angle. Consequently, our method ensures the target in FOV. However, Fast-Tracker almost has the target out of FOV at every turn.

Furthermore, as is shown in Fig.10, we count the target positions projected to x-y plane in the tracking quadrotor's FOV. The heat map shows the distribution of the target positions, relative to the tracking quadrotor. For better visualization, we replace the count number  $N$  in each heat map grid by  $\log(N + 1)$ .

Consequently, our method keeps the target in a moderate area of FOV, which is conducive for observation, while Fast-Tracker always loses the target.

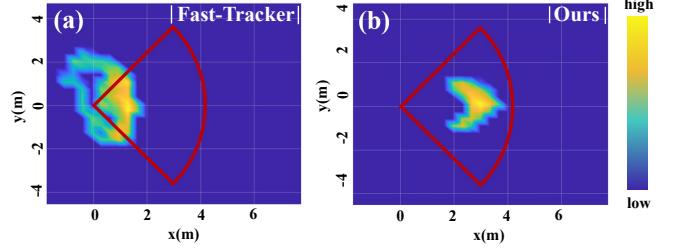


Fig. 10: Heat map comparison. The heat map shows the distribution of the target positions relative to the tracking quadrotor on x-y plane. The red sector represents the FOV of the tracking quadrotor. The closer it is to yellow, the more frequently the target appears. Our approach always restricts the target to the central area. (a): heat map of Fast-Tracker. (b): heat map of our method.

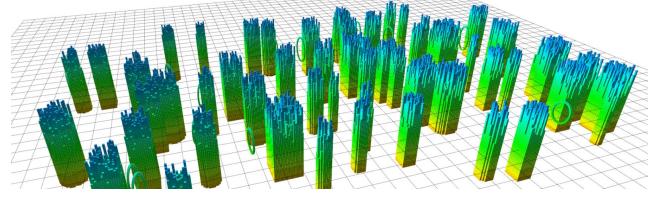


Fig. 11: The random environment in general test.

3) *General Test*: To prove that our method is generic, we compare both methods in an environment generated with randomly deployed obstacles as shown in Fig.11.

In a variety of applications, we may be incapable to obtain the target location directly, which requires the quadrotor to identify the target's location. As a result, we define the tracking mission as a failure when the tracking quadrotor loses the target. In tests, the tracking quadrotor follows the target moving along a random trajectory that lasts for 100s. For comparison, we further define failure time  $T_{fast}$  and  $T_{our}$  are the time when the mission fails. The result is shown in Tab.III.  $V_{max}$  is the max velocity set to the target, and we set double  $V_{max}$  to the tracking quadrotor. The failure time are compared in 10 tracking missions for each scenario. Consequently, in general environment, our method maintains a higher success rate, while Fast-Tracker fails easily.

TABLE III: Comparision of Failure Time in General Tests.

$V_{max}$ (m/s)	0.5	1.5	2.5
$T_{fast}$ (s)	39.1	66.25	11.0
$T_{our}$ (s)	100	100	85.7

## VII. CONCLUSION

In this paper, we summarize three metrics *DO*, *AO*, *OE* to evaluate the visibility according to practical experience. Under these metrics, we design differentiable visibility costs and propose a general visibility-aware trajectory optimization method. An joint optimizer is proposed to adjust the position and yaw simultaneously. To validate the visibility effect of our method, we integrate the optimizer into the back-end of a quadrotor tracking system and benchmark it against a state-of-the-art tracking planner. Simulation comparisons and real-world experiments validate that it is robust and efficient.

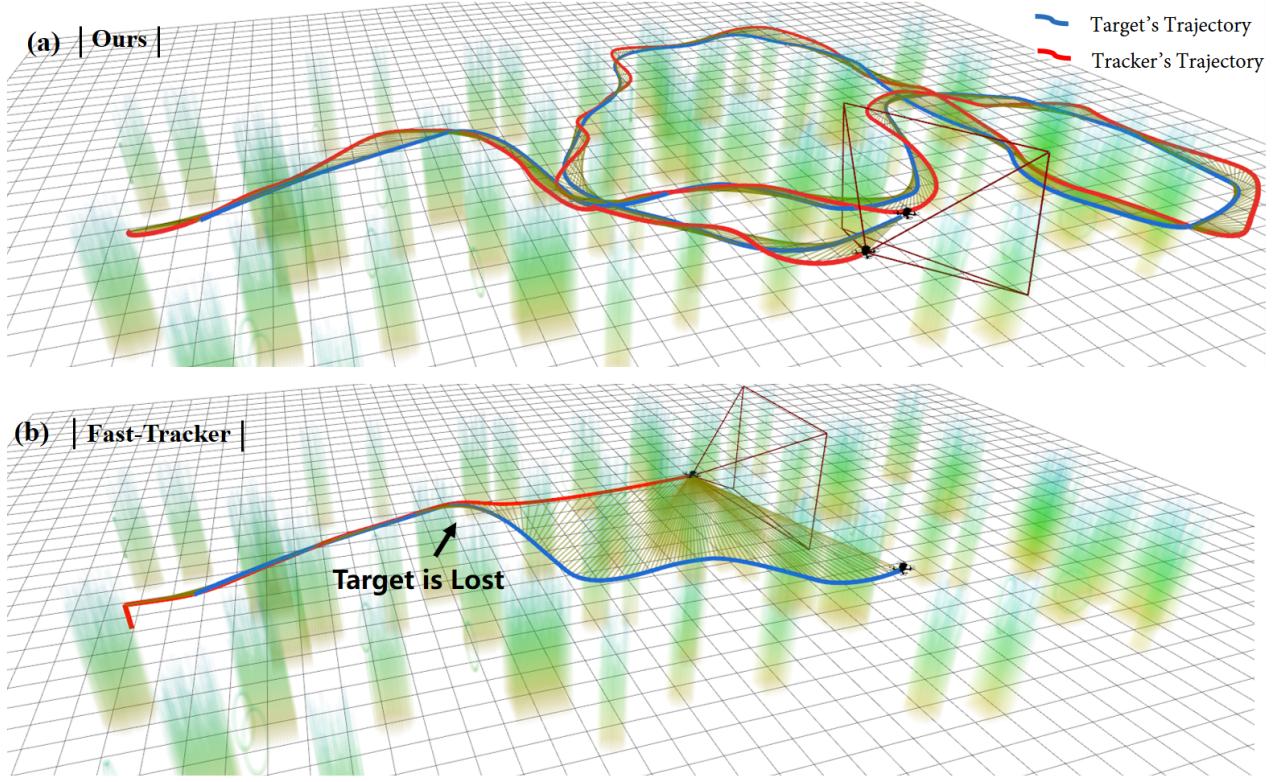


Fig. 12: Comparision of the trajectories in one of the general tests. (a): the result of our method. (b): the result of Fast-Tracker.

In the future, our visibility-aware trajectory optimization method will be extended to active slam and exploration areas. As for aerial tracking, we will continue to improve the prediction method to make it a more robust system.

#### REFERENCES

- [1] V. Murali, I. Spasojevic, W. Guerra, and S. Karaman, “Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness,” in *Proceedings of the American Control Conference*, vol. 2019-July. IEEE, 2019, pp. 3936–3943.
- [2] Z. Zhang and D. Scaramuzza, “Perception-aware receding horizon navigation for MAVs,” in *Proceedings - IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 2534–2541.
- [3] L. Ly and Y.-H. R. Tsai, “Autonomous exploration, reconstruction, and surveillance of 3d environments aided by deep learning,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5467–5473.
- [4] S. Freitag, B. Weyers, and T. W. Kuhlen, “Interactive Exploration Assistance for Immersive Virtual Environments Based on Object Visibility and Viewpoint Quality,” in *25th IEEE Conference on Virtual Reality and 3D User Interfaces, VR 2018 - Proceedings*. IEEE, 2018, pp. 355–362.
- [5] B. Zhou, J. Pan, F. Gao, and S. Shen, “Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight,” *arXiv preprint arXiv:2007.03465*, 2020.
- [6] T. Bandyopadhyay, D. Hsu, and M. H. Ang, “Motion Strategies for People Tracking in Cluttered and Dynamic Environments,” in *Springer Tracts in Advanced Robotics*, vol. 54, 2009, pp. 463–472.
- [7] B. Jeon, Y. Lee, and H. J. Kim, “Integrated Motion Planner for Real-time Aerial Videography with a Drone in a Dense Environment,” in *Proceedings - IEEE International Conference on Robotics and Automation*, 2020, pp. 1243–1249.
- [8] J. Chen, T. Liu, and S. Shen, “Tracking a moving target in cluttered environments using a quadrotor,” in *IEEE International Conference on Intelligent Robots and Systems*, vol. 2016-November, 2016, pp. 446–453.
- [9] B. Penin, P. R. Giordano, and F. Chaumette, “Vision-Based Reactive Planning for Aggressive Target Tracking while Avoiding Collisions and Occlusions,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3725–3732, 2018.
- [10] H. Andersen, J. Alonso-Mora, Y. H. Eng, D. Rus, and M. H. Ang, “Trajectory Optimization and Situational Analysis Framework for Autonomous Overtaking with Visibility Maximization,” in *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 1. IEEE, 2020, pp. 7–20.
- [11] T. Nageli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, “Real-time motion planning for aerial videography with real-time with dynamic obstacle avoidance and viewpoint optimization,” *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, 2017.
- [12] 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 *International Symposium on Experimental Robotics*. Springer, 2018, pp. 119–129.
- [13] B. F. Jeon and H. J. Kim, “Online Trajectory Generation of a MAV for Chasing a Moving Target in 3D Dense Environments,” in *IEEE International Conference on Intelligent Robots and Systems*, 2019, pp. 1115–1121.
- [14] Z. Han, R. Zhang, N. Pan, C. Xu, and F. Gao, “Fast-tracker: A robust aerial system for tracking agile target in cluttered environments,” *arXiv preprint arXiv:2011.03968*, 2020.
- [15] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Path planning for autonomous vehicles in unknown semi-structured environments,” *International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.
- [16] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, “Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments,” *arXiv preprint arXiv:2011.04183*, 2020.
- [17] T. Qin, P. Li, and S. Shen, “Vins-mono: A robust and versatile monocular visual-inertial state estimator,” *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.
- [18] Z. Wang, H. Ye, C. Xu, and F. Gao, “Generating large-scale trajectories efficiently using descriptions of polynomials,” *arXiv preprint arXiv:2011.02662*, 2020.