Probabilistic Visibility-Aware Trajectory Planning for Target Tracking in Cluttered Environments

Han Gao[1], Pengying Wu[1], Yao Su[2], Kangjie Zhou[1], Ji Ma[1], Hangxin Liu[2] and Chang Liu[1,†]

***Abstract*— Target tracking has numerous significant civilian and military applications, and maintaining the visibility of the target plays a vital role in ensuring the success of the tracking task. Existing visibility-aware planners primarily focus on keeping the target within the limited field of view of an onboard sensor and avoiding obstacle occlusion. However, the negative impact of system uncertainty is often neglected, rendering the planners delicate to uncertainties in practice. To bridge the gap, this work proposes a real-time, non-myopic trajectory planner for visibility-aware and safe target tracking in the presence of system uncertainty. For more accurate target motion prediction, we introduce the concept of belief-space probability of detection (BPOD) to measure the predictive visibility of the target under stochastic robot and target states. An Extended Kalman Filter variant incorporating BPOD is developed to predict target belief state under uncertain visibility within the planning horizon. To reach real-time trajectory planning, we propose a computationally efficient algorithm to uniformly calculate both BPOD and the chance-constrained collision risk by utilizing linearized signed distance function (SDF), and then design a two-stage strategy for lightweight calculation of SDF in sequential convex programming. Extensive simulation results with benchmark comparisons show the capacity of the proposed approach to robustly maintain the visibility of the target under high system uncertainty. The practicality of the proposed trajectory planner is validated by real-world experiments.**

I. Introduction

Target tracking using an autonomous vehicle has garnered widespread utilization in various important applications, such as vehicle tracking [1], cinematography [2], and underwater monitoring [3]. In recent years, visibility-aware motion planning has emerged as a key focus of target-tracking research, where the robot is tasked with generating trajectories to track a mobile target while ensuring continuous visibility, as illustrated in Fig. 1.

Since first characterized by LaValle et al. [4], various motion planning methodologies have been introduced to tackle the visibility-aware tracking task. Some previous works formulate the target tracking problem as a visibility-based pursuer-evader game [5, 6] and focus on proposing the winning strategy for the pursuers to keep uninterrupted visibility of a target. Nevertheless, the adversarial behaviors of the target greatly complicate robot motion planning and lead to high computational costs [6, 7], thus hindering the application to real-time non-adversarial tracking tasks in realistic, complex environments.

*This work was sponsored by Beijing Nova Program (20220484056) and the National Natural Science Foundation of China (62203018).

† Corresponding author. [1] Department of Advanced Manufacturing and Robotics, College of Engineering, Peking University. Emails: {hangaocoe, littlefive, kangjiezhou, maji, changliucoe}@pku.edu.cn. [2] National Key Laboratory of General Artificial Intelligence, Beijing Institute for General Artificial Intelligence (BIGAI). Emails: {suyao, liuhx}@bigai.ai.

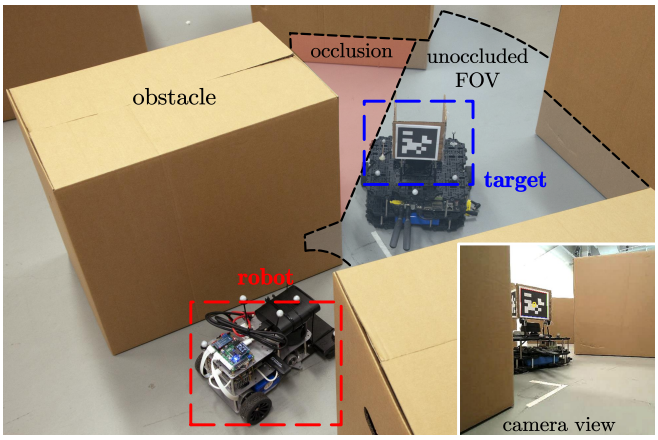


Fig. 1: **Illustration of visibility-aware tracking for a moving target.** The red region represents the part of FOV being occluded, while the light blue region means the unoccluded part. Note that the occluded regions overlapping with obstacles are not visualized.

Trajectory optimization, on the contrary, remains a mainstream methodology for real-time visibility-aware planning. This approach formulates trajectory planning as an optimization problem [8], seeking to maintain the target's visibility by adjusting robot control inputs [9, 10] or predictive trajectory parameters [11]. Our work adopts this methodology due to its scalability and computational efficiency.

Previous works on visibility-aware trajectory planning mainly focus on incorporating explicit considerations for the limited field of view (FOV) and obstacle occlusion, as they will directly impair the visibility of a target. To avoid potential target loss due to a limited FOV, previous works employ geometric costs, such as distance [3, 11] and bearing angle [11, 12] costs between the target and the sensor, to maintain the target within the FOV. To handle possible occlusion by obstacles, previous works propose to apply distance cost between the line of sight (LOS) and obstacles to avoid occlusion [12, 13]. However, most of the abovementioned research overlooks the detrimental effects of system uncertainties arising from imperfect system models, process noise, and measurement noise, which may significantly degrade state estimation and trajectory planning in visibility-aware tracking tasks.

Belief space planning (BSP) [14] provides a systematic approach for handling system uncertainty, and is widely used in target tracking tasks with stochasticity [9, 10]. Specifically, BSP formulates a stochastic optimization problem to generate visibility maintenance trajectories, and the key step lies in the evaluation of predictive target visibility. A commonly employed method for predicting visibility is to evaluate

the probability of detection (POD) by integrating predictive target probability density function (PDF) over the anticipated unoccluded FOV area [15, 16]. However, calculating this integration can be either computationally burdensome when conducted over continuous state spaces [16] or inaccurate due to discretization error when transformed into the sum of probability mass over a grid detectable region [15]. Another method is directly determining target visibility based on the mean positions of the predicted target and the planned FOV [9, 10]. Despite its computational efficiency, such maximum-a-posteriori (MAP) estimation only considers mean positions, neglecting predictive target uncertainty during robot trajectory planning.

This work proposes a model predictive control (MPC)-based non-myopic trajectory planner for mobile target tracking in cluttered environments. The proposed framework systematically considers limited FOV, and obstacle occlusion under a BSP framework to address state uncertainty. The main contributions can be summarized as follows:

- We propose the concept of *belief-space probability of detection* (BPOD) that depends on both robot and target belief states to function as a measure of predicted visibility. We then develop an Extended Kalman Filter (EKF) variant that incorporates BPOD into target state prediction to take into account stochastic visibility in the MPC predictive horizon, which overcomes the deficiency of MAP estimation.
- We present a unified representation for both the BPOD and the collision risk as the *probabilities of stochastic SDF satisfaction* (PoSSDF) via the use of signed distance functions (SDFs), and develop a paradigm to efficiently calculate the PoSSDFs.
- We propose a real-time trajectory planner for visibility-aware target tracking in cluttered environments based on sequential convex programming (SCP). In particular, we propose a two-stage strategy for calculating SDF in SCP to accelerate the planner, reaching a computational speed of $10Hz$. Simulations with benchmark comparisons and real-world experiments demonstrate that our method enables the robot to track a moving target in cluttered environments at high success rates and visible rates, even under high system uncertainty.

The remainder of the article is organized as follows. Sec. II formulates the target tracking problem. Sec. III defines BPOD and proposes a variant of EKF covariance update formulation. Sec. IV approximates BPOD and collision risk by SDF linearization and proposes an SCP framework with a two-stage strategy for online trajectory planning. Simulations and real-world experiments are presented in Secs. V and VI, respectively. Sec. VII presents conclusions and future works.

II. Formulation of Target Tracking

This work considers a mobile target tracking problem in a 2D environment with cluttered obstacles, as shown in Fig. 1. The robot knows its current state but relies on a noisy sensor with limited FOV to detect a stochastically moving target. Therefore, the target state remains partially observable, necessitating estimation by the robot. Besides, the motion model of the robot and the target are both stochastic, adding to the difficulty of target tracking. The goal of the robot is to plan collision-free trajectories to maintain continual detection of the target.

A. Motion Models and Obstacle Modeling

The robot and target dynamics are defined as follows [17]:

$$\mathbf{z}_{k+1}^r = \mathbf{f}^r(\mathbf{z}_k^r, \mathbf{u}_k^r) + \mathbf{w}^r, \quad \mathbf{w}^r \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^r), \tag{1}$$

$$\mathbf{z}_{k+1}^t = \mathbf{f}^t(\mathbf{z}_k^t, \mathbf{u}_k^t) + \mathbf{w}^t, \quad \mathbf{w}^t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^t), \tag{2}$$

where $\mathbf{z}$ and $\mathbf{u}$ denote the state and control input, respectively. The superscripts r and t denote robot and target, respectively, and the subscript k means time step. The motion function $\mathbf{f}^r$ and $\mathbf{f}^t$ will be specified in Sec. V. The process noise $\mathbf{w}^r$ and $\mathbf{w}^t$ follow Gaussian distributions with zero means and covariance matrix $\mathbf{R}^r$ and $\mathbf{R}^t$, respectively.

This work considers a known continuous map with convex obstacles represented as point sets $\mathcal{O}_i \subset \mathbb{R}^2, i = 1, 2, \cdots, N_o$, where N_o denotes the number of obstacles. Note that the proposed approach can also address nonconvex obstacles by dividing them into multiple convex ones.

B. Sensor Model

The robot's sensor has a limited FOV $\mathcal{V}_k \subseteq \mathbb{R}^2$, which will be specified in Sec. V. The sensor model described in this work considers intermittent measurements due to potential target loss caused by the limited FOV and occlusions, which is defined as

$$\mathbf{y}_k = \begin{cases} \mathbf{f}^s(\mathbf{z}_k^t, \mathbf{z}_k^r) + \mathbf{w}^s, & \mathbf{x}_k^t \in \mathcal{V}_k^v \\ \varnothing, & \mathbf{x}_k^t \notin \mathcal{V}_k^v \end{cases}, \tag{3}$$

where $\mathbf{y}_k \in \mathbb{R}^{d_m}$ is the measurement, the measurement function $\mathbf{f}^s$ will be specified in Sec. V, and $\mathbf{x}_k^t \in \mathbb{R}^2$ denotes target position. The sensor noise $\mathbf{w}^s$ follows a Gaussian distribution with a zero mean and covariance matrix $\mathbf{R}^s$. Here $\mathcal{V}_k^v$ is a subset of $\mathcal{V}_k$ that is not occluded by any obstacles[1].

C. Target State Estimation

The robot needs accurately estimate the target state to make informed tracking behavior. To account for the system nonlinearity and possible target loss, we develop a variant of EKF for target state estimation to take intermittent measurements into account, inspired by [18].

The filtering procedure can be summarized as follows.
Prediction. Predict the prior PDF using target kinematics,

$$\hat{\mathbf{z}}_{k|k-1}^t = \mathbf{f}^t(\hat{\mathbf{z}}_{k-1|k-1}^t, \mathbf{u}_{k-1}^t), \tag{4a}$$

$$\mathbf{P}_{k|k-1} = \mathbf{A}_{k-1}^t \mathbf{P}_{k-1|k-1} (\mathbf{A}_{k-1}^t)^T + \mathbf{R}^t, \tag{4b}$$

where $\hat{\mathbf{z}}_{k-1|k-1}^t$ and $\mathbf{P}_{k-1|k-1}$ represent the mean and covariance of the estimate of $\mathbf{z}_{k-1}^t$, and $\mathbf{A}_{k-1}^t = \nabla_{\mathbf{z}^t} \mathbf{f}^t(\mathbf{z}^t, \mathbf{u}_{k-1}^t)|_{\mathbf{z}^t = \hat{\mathbf{z}}_{k-1|k-1}^t}$ is the Jacobi matrix of the target kinematic model. The $\mathbf{u}_{k-1}^t$ can be estimated by techniques

[1]This work adopts a perfect sensor model for the purpose of simplicity, which assumes that a non-empty measurement is returned if and only if the target is inside the FOV and not occluded. However, it is worth noting that the proposed approach can be easily extended to imperfect sensor models that give false positive or false negative measurements.

such as displacement differentiation, Linear Quadratic Gaussian Controller, etc., and is not the focus of this work.

Update. Use current measurements to update target PDF,

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{C}_k^T(\mathbf{C}_k\mathbf{P}_{k|k-1}\mathbf{C}_k^T + \mathbf{R}^s)^{-1}, \tag{5a}$$

$$\hat{\mathbf{z}}_{k|k}^t = \hat{\mathbf{z}}_{k|k-1}^t + \mu_k\mathbf{K}_k(\mathbf{y}_k - \mathbf{f}^s(\hat{\mathbf{z}}_{k|k-1}^t, \mathbf{z}_k^r)), \tag{5b}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mu_k\mathbf{K}_k\mathbf{C}_k\mathbf{P}_{k|k-1}, \tag{5c}$$

where $\mathbf{C}_k = \nabla_{\mathbf{z}^t}\mathbf{f}^s(\mathbf{z}^t, \mathbf{z}_k^r)|_{\mathbf{z}^t=\hat{\mathbf{z}}_{k|k-1}^t}$ is the Jacobi matrix of measurement model, and μ_k is the *detection variable* (DV) determining whether the target is detected, calculated as

$$\mu_k = \begin{cases} 1, & \mathbf{y}_k \neq \varnothing \\ 0, & \mathbf{y}_k = \varnothing \end{cases}. \tag{6}$$

D. MPC Formulation for Trajectory Planning

The trajectory planner is formulated as an MPC problem

$$\min_{\mathbf{u}_{k:k+N-1}^r} \quad J(\mathbf{b}_{k+1:k+N}^r, \mathbf{b}_{k+1:k+N}^t) \tag{7a}$$

$$s.t. \quad \mathbf{b}_{k+i}^t = \mathbf{g}^t(\mathbf{b}_{k+i-1}^t, \mathbf{b}_{k+i-1}^r), \tag{7b}$$

$$\mathbf{b}_{k+i}^r = \mathbf{g}^r(\mathbf{b}_{k+i-1}^r, \mathbf{u}_{k+i-1}^r), \tag{7c}$$

$$\mathbf{b}_{k+i}^r \in \mathcal{B}^r, \mathbf{b}_{k+i}^t \in \mathcal{B}^t, \mathbf{u}_{k+i-1}^r \in \mathcal{U}, \tag{7d}$$

$$\mathbf{f}^o(\mathbf{b}_{k+i}^r, \mathcal{O}_j) < 0, \tag{7e}$$

$$j = 1, 2, \cdots, N_o, \ i = 1, \cdots, N,$$

where N stands for MPC planning horizon. The *target belief state* $\mathbf{b}_k^t = [\hat{\mathbf{z}}_{k|k}^t, \mathbf{P}_{k|k}]$ encodes the probability distribution of the target state. Due to the existence of process noise, the robot state is stochastic in the predictive horizon. Therefore, we define the *robot belief state* $\mathbf{b}_k^r = [\hat{\mathbf{z}}_k^r, \mathbf{Q}_k]$ to encode mean value $\hat{\mathbf{z}}_k^r$ and covariance matrix $\mathbf{Q}_k$ of robot state. The sets $\mathcal{B}^r$, $\mathcal{B}^t$ and $\mathcal{U}$ are the feasible sets of robot belief state, target belief state and robot control input, respectively. The functions $\mathbf{g}^r$ and $\mathbf{g}^t$ denote belief prediction procedures, which will be described in detail in the next section. The objective function Eq. (7a) and the collision avoidance constraint Eq. (7e) will be further elaborated in Secs. III and IV.

III. Belief-Space Probability of Detection-Based State Prediction

A. Probabilistic State Prediction in Predictive Horizon

We employ two different EKF-based approaches for the state prediction of the robot and target, thus specifying Eqs. (7b) and (7c). Note that the lack of observation needs to be properly addressed, which differs from the estimation process Eqs. (4a), (4b) and (5a) to (5c).

Robot state prediction. Like Eqs. (4a) and (4b), we use the prediction step of EKF to predict the robot state in the predictive horizon. Specifically, Eq. (7c) is specified as

$$\hat{\mathbf{z}}_{k+i}^r = \mathbf{f}^r(\hat{\mathbf{z}}_{k+i-1}^r, \mathbf{u}_{k+i-1}^r), \tag{8a}$$

$$\mathbf{Q}_{k+i} = \mathbf{A}_{k+i-1}^r\mathbf{Q}_{k+i-1}(\mathbf{A}_{k+i-1}^r)^T + \mathbf{R}^r, \tag{8b}$$

where $\mathbf{A}_{k+i-1}^r = \nabla_{\mathbf{z}^r}\mathbf{f}^r(\mathbf{z}^r, \mathbf{u}_{k+i-1}^r)|_{\mathbf{z}^r=\hat{\mathbf{z}}_{k+i-1}^r}$ is the Jacobi matrix of the robot kinematic model.

Target state prediction. Within the planning horizon, the target mean is simply propagated using its kinematics with the same control input $\mathbf{u}_{k+i}^t = \mathbf{u}_{k-1}^t, i = 0, \cdots, N-1$, and the covariance $\mathbf{P}_{k+i|k+i-1}$ is predicted by Eq. (4b). However, target visibility is uncertain in the predictive horizon, making it difficult to predict DV and update the covariance following Eqs. (5a) and (5c). To deal with this difficulty, we propose the concept of BPOD to denote the probability that the target is detected, defined as

$$\gamma_k = Pr(\mathbf{y}_k \neq \varnothing | \mathbf{b}_k^r, \mathbf{b}_{k|k-1}^t), \tag{9}$$

where $\mathbf{b}_{k|k-1}^t = [\hat{\mathbf{z}}_{k|k-1}^t, \mathbf{P}_{k|k-1}]$. Compared to traditional PODs that are either determined by the ground truth of robot and target positions [19] or only depend on the predictive target belief under deterministic robot states [15, 16], BPOD is conditioned on the belief states of both the robot and the target, which provides a more precise measurement of visibility under stochastic system states. Next, we will incorporate the BPOD into EKF and constitute the stochastic counterpart of Eq. (5c) to tackle uncertain predictive visibility.

Recall that the DV is determined by the relative pose of the robot and the target, and thus can be reformulated as

$$\mu_k = Pr(\mathbf{y}_k \neq \varnothing | \mathbf{z}_k^r, \mathbf{z}_k^t). \tag{10}$$

Denote $\mathbb{E}_{z|b}(\cdot)$ as the expectation operator with respect to $\mathbf{z}_k^t$ and $\mathbf{z}_k^r$ conditioned on $\mathbf{b}_{k|k-1}^t$ and $\mathbf{b}_k^r$, we can adopt the total probability rule (TPR) and derive $\gamma_k = \mathbb{E}_{z|b}(\mu_k)$. Combining Eq. (10) and the EKF procedures, we can find that $\mathbf{P}_{k|k}$ in Eq. (5c) is conditioned on $\mathbf{z}_k^r$ and $\mathbf{z}_k^t$. Note that accurate estimates of the robot and target states are not available in the predictive horizon. Therefore, the posterior covariance $\tilde{\mathbf{P}}_{k|k}$ in predictive horizon only depends on the predictive robot and target beliefs, and thus can be formulated as the conditional expectation of $\mathbf{P}_{k|k}$, *i.e.*, $\tilde{\mathbf{P}}_{k|k} = \mathbb{E}_{z|b}(\mathbf{P}_{k|k})$ according to TPR. Following this idea, we propose a visibility-aware covariance update scheme, which is formulated as follows:

$$\tilde{\mathbf{P}}_{k|k} = \mathbb{E}_{z|b}(\mathbf{P}_{k|k}) \tag{11a}$$

$$\approx \mathbb{E}_{z|b}(\mathbf{P}_{k|k-1} - \mu_k\tilde{\mathbf{K}}_k\tilde{\mathbf{C}}_k\mathbf{P}_{k|k-1}) \tag{11b}$$

$$= \mathbf{P}_{k|k-1} - \mathbb{E}_{z|b}(\mu_k)\tilde{\mathbf{K}}_k\tilde{\mathbf{C}}_k\mathbf{P}_{k|k-1} \tag{11c}$$

$$= \mathbf{P}_{k|k-1} - \gamma_k\tilde{\mathbf{K}}_k\tilde{\mathbf{C}}_k\mathbf{P}_{k|k-1}, \tag{11d}$$

where covariance $\mathbf{P}_{k|k-1}$ and $\mathbf{P}_{k|k}$ are defined in Eqs. (4b) and (5c), respectively, and $\tilde{\mathbf{C}}_k$, $\tilde{\mathbf{K}}_k$ denote the approximate measurement Jacobi matrix and Kalman gain, respectively. To simplify the computation, we approximate the measurement Jacobi matrix as being equal to the one determined by the mean value of robot belief $\hat{\mathbf{z}}_k^r$, *i.e.*, $\tilde{\mathbf{C}}_k = \nabla_{\mathbf{z}^t}\mathbf{f}^s(\mathbf{z}^t, \hat{\mathbf{z}}_k^r)|_{\mathbf{z}^t=\hat{\mathbf{z}}_{k|k-1}^t}$, and calculate the Kalman gain $\tilde{\mathbf{K}}_k$ from Eq. (5a) by replacing $\mathbf{C}_k$ with $\tilde{\mathbf{C}}_k$, which yields Eq. (11b). Eq. (11c) is derived by the independency of $\tilde{\mathbf{K}}_k$ and $\tilde{\mathbf{C}}_k$ from $\mathbf{z}_k^t$ and $\mathbf{z}_k^r$.

Eq. (11d) is a probabilistic extension of Eq. (5c) that allows us to update the target covariance in Eq. (7b) using only the predicted beliefs of the robot and target. Fig. 2 illustrate the prediction of both robot and target belief states.

B. Objective Functions

There are two mainstream choices of the objective function for target tracking, *i.e.*, to minimize target uncertainty [9, 16],

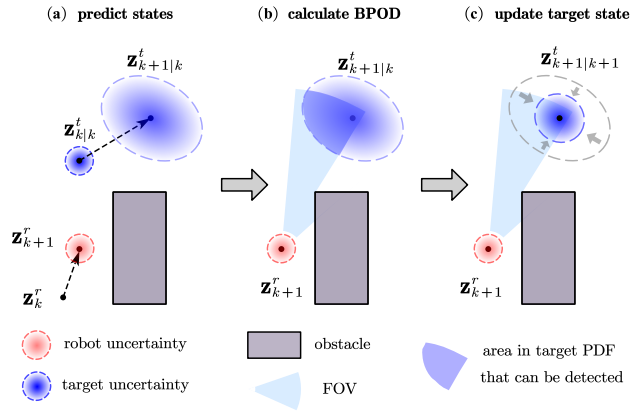


Fig. 2: **Illustration of state prediction process Eqs. (7b) and (7c).** (a) Propagate robot and target prior distribution via Eqs. (4) and (8). (b) Calculate BPOD. (c) Update target covariance using Eq. (11d).

or to maximize the predictive visibility of the target [15]. We correspondingly formulate two objectives as

$$J_1 = \sum_{i=1}^{N} \mathcal{H}(\mathbf{b}_{k+i}^t), \quad J_2 = -\sum_{i=1}^{N} \gamma_{k+i}, \tag{12}$$

where J_1 is the cumulative entropy of target belief, specified as $J_1 = \frac{Nd_t}{2}(\ln(2\pi)+1) + \sum_{i=1}^{N} \frac{1}{2} \ln |\tilde{\mathbf{P}}_{k+i|k+i}|$ with d_t being the dimension of $\mathbf{z}_k^t$, and J_2 is the negation of cumulative BPOD. These two objectives are theoretically proved to be compatible [20], and both J_1 and J_2 are verified to perform well in keeping the target visible and decreasing estimation error in the simulation, as will be presented in Sec. V.

IV. Signed Distance Function-Based Online Trajectory Planning

In order to solve the MPC problem Eq. (7), it is crucial to efficiently compute BPOD and the collision constraint Eq. (7e). A main contribution of this work is to represent and compute the BPOD and collision risk in a unified manner that significantly reduces the computational burden of solving the MPC problem, which will be detailed in this section.

A. Unified Expression of Visibility and Collision Risk

Limited FOV and occlusion are two major factors that influence visibility. Following this idea, we factorize the BPOD into two types of probabilities as follows.

Target and FOV. We define the probability of the target being within the FOV area as:

$$\gamma_k^{tf} = Pr(\mathbf{x}_k^t \in \mathcal{V}_k | \mathbf{b}_k^r, \mathbf{b}_{k|k-1}^t). \tag{13}$$

LOS and Obstacle. Likewise, we express the probability that the target is not occluded by obstacle i as:

$$\gamma_{k,i}^{lo} = Pr(\mathcal{L}_k \bigcap \mathcal{O}_i = \varnothing | \mathbf{b}_k^r, \mathbf{b}_{k|k-1}^t), \tag{14}$$

where $\mathcal{L}_k \subset \mathbb{R}^2$ is the line segment between $\mathbf{x}_k^t$ and the robot position $\mathbf{x}_k^r \in \mathbb{R}^2$. Reasonably, we can assume that all variables in $\{\gamma_k^{tf}, \gamma_{k,1}^{lo}, \cdots, \gamma_{k,N_o}^{lo}\}$ are mutually independent. Then the BPOD Eq. (9) can be factorized as

$$\gamma_k = \gamma_k^{tf} \prod_{i=1}^{N_o} \gamma_{k,i}^{lo}. \tag{15}$$

Algorithm 1: Calculation of PoSSDF

1 INPUT: two rigid bodies: $\mathcal{A}_1(\mathbf{x}), \mathcal{A}_2(\mathbf{x})$, $\mathbf{x} \sim \mathcal{N}(\hat{\mathbf{x}}, \Sigma)$,
2 SDF parameters $\hat{p}_1^L, \hat{p}_2^L, \hat{\mathbf{n}}$
3 OUTPUT: $p = Pr(sd(\mathcal{A}_1(\mathbf{x}), \mathcal{A}_2(\mathbf{x})) \leqslant 0)$
4 $\mathbf{p}_i(\mathbf{x}) \leftarrow \mathbf{R}_i(\mathbf{x})\hat{p}_i^L + \mathbf{p}_i^c(\mathbf{x}), i = 1, 2$
5 $sd(\mathbf{x}) \leftarrow \hat{\mathbf{n}}^T(\mathbf{p}_1(\mathbf{x}) - \mathbf{p}_2(\mathbf{x}))$
6 $sd_L(\mathbf{x}) \leftarrow sd(\hat{\mathbf{x}}) + \nabla_{\mathbf{x}} sd(\mathbf{x})|_{\mathbf{x}=\hat{\mathbf{x}}}(\mathbf{x} - \hat{\mathbf{x}})$
7 $p \leftarrow$ Calculate $Pr(sd_L(\mathbf{x}) \leqslant 0)$

To avoid overly conservative trajectory while ensuring safety, we define chance-constrained [21] collision avoidance conditions that bound the collision risk below a user-defined threshold $\delta^s \in \mathbb{R}$. The probability of collision between the robot and obstacle i at time k is defined as:

$$\gamma_{k,i}^{ro} = Pr(\mathbf{x}_k^r \in \mathcal{O}_i | \mathbf{b}_k^r, \mathbf{b}_{k|k-1}^t). \tag{16}$$

Eq. (7e) is then specified as the chance constraints, *i.e.*,

$$\gamma_{k+i,j}^{ro} < \delta_{k+i,j}^s. \tag{17}$$

In the next subsection, a computationally efficient algorithm is designed to calculate γ^{tf}, γ^{lo} and γ^{ro}. For simplicity, we define the set $\Gamma_k = \{\gamma_k^{tf}, \gamma_{k,i}^{lo}, \gamma_{k,i}^{ro}, i = 1, \cdots, N_o\}$.

B. Approximating Γ_k with Linearized SDF

The SDF quantifies the distance between two shapes. Specifically, the SDF of two sets $\mathcal{A}, \mathcal{B} \subset \mathbb{R}^2$ is formulated as the minimum translation distance required to separate or intersect each other, *i.e.*,

$$sd(\mathcal{A}, \mathcal{B}) = \begin{cases} \inf\{\|\mathbf{v}\| \mid (\mathbf{v} + \mathcal{A}) \bigcap \mathcal{B} \neq \varnothing\}, \mathcal{A} \bigcap \mathcal{B} = \varnothing \\ -\inf\{\|\mathbf{v}\| \mid (\mathbf{v} + \mathcal{A}) \bigcap \mathcal{B} = \varnothing\}, \mathcal{A} \bigcap \mathcal{B} \neq \varnothing \end{cases},$$

where $\mathbf{v} \in \mathbb{R}^2$ is the translation vector. Using SDF, we can equivalently reformulate Eqs. (13), (14) and (16) as

$$\gamma_k^{tf} = Pr(sd(\mathbf{x}_k^t, \mathcal{V}_k) \leqslant 0 | \mathbf{b}_k^r, \mathbf{b}_{k|k-1}^t), \tag{18a}$$

$$\gamma_{k,i}^{lo} = Pr(sd(\mathcal{L}_k, \mathcal{O}_i) \geqslant 0 | \mathbf{b}_k^r, \mathbf{b}_{k|k-1}^t), \tag{18b}$$

$$\gamma_{k,i}^{ro} = Pr(sd(\mathbf{x}_k^r, \mathcal{O}_i) \leqslant 0 | \mathbf{b}_k^r, \mathbf{b}_{k|k-1}^t). \tag{18c}$$

Note that Eqs. (18a) to (18c) are all PoSSDFs because $\mathbf{x}_k^t, \mathbf{x}_k^r, \mathcal{L}_k$ and $\mathcal{V}_k$ are all random variables given robot and target beliefs. These probabilities can be evaluated using Monte-Carlo simulation, but at the cost of heavy computational burden. Inspired by [22], we propose a computationally efficient paradigm to evaluate PoSSDF using a linearized SDF expression, as described in Alg. 1.

The inputs of Alg. 1 include two rigid bodies $\mathcal{A}_1, \mathcal{A}_2 \subset \mathbb{R}^2$, whose poses are determined by an arbitrary random Gaussian variable $\mathbf{x}$ with dimension d_x. This algorithm also needs to precalculate the signed distance $\hat{d} \in \mathbb{R}$ between $\mathcal{A}_1(\hat{\mathbf{x}})$ and $\mathcal{A}_2(\hat{\mathbf{x}})$, along with the closest points from each set, $\hat{\mathbf{p}}_1 \in \mathcal{A}_1(\hat{\mathbf{x}}), \hat{\mathbf{p}}_2 \in \mathcal{A}_2(\hat{\mathbf{x}})$, as illustrated in Fig. 3(a). This operation can be efficiently carried out by using Gilbert–Johnson–Keerthi (GJK) algorithm [23] and Expanding Polytope Algorithm (EPA) [24]. Then we obtain the *SDF parameters* including the contact normal $\hat{\mathbf{n}} = sgn(\hat{d}) \cdot (\hat{\mathbf{p}}_1 - \hat{\mathbf{p}}_2)/\|\hat{\mathbf{p}}_1 - \hat{\mathbf{p}}_2\|$, and the local coordinates of $\hat{\mathbf{p}}_1$ and $\hat{\mathbf{p}}_2$ relative to $\mathcal{A}_1(\hat{\mathbf{x}})$ and $\mathcal{A}_2(\hat{\mathbf{x}})$ respectively, noted as $\hat{p}_1^L$ and $\hat{p}_2^L$. Line 4 provides an analytical approximation

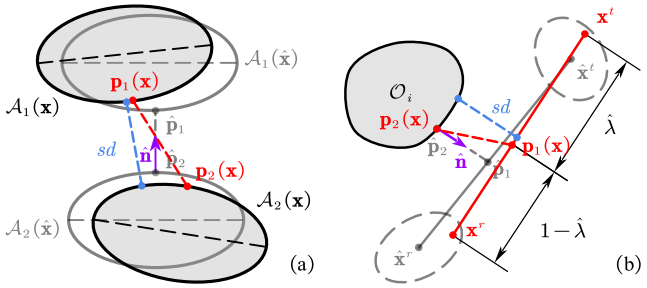


Fig. 3: **Illustration of SDF approximation in Alg. 1.** The blue segment represents the true SDF value of two sets $\mathcal{A}_1(\mathbf{x})$ and $\mathcal{A}_2(\mathbf{x})$, while the approximate SDF is calculated by projecting $\mathbf{p}_1(\mathbf{x})-\mathbf{p}_2(\mathbf{x})$ (the red dashed segment) to $\hat{\mathbf{n}}$ (the purple vector). (a) A general case. (b) Application of Alg. 1 to calculate γ^{lo} with the dashed ellipses denoting the uncertainties of robot and target.

of the closest points $\mathbf{p}_1(\mathbf{x})$ and $\mathbf{p}_2(\mathbf{x})$ in the world frame. Here we assume that the local coordinates of $\mathbf{p}_1(\mathbf{x})$ and $\mathbf{p}_2(\mathbf{x})$, relative to $\mathcal{A}_1(\mathbf{x})$ and $\mathcal{A}_2(\mathbf{x})$ respectively, are fixed and equal to $\hat{p}_1^L$ and $\hat{p}_2^L$. The approximate closest point $\mathbf{p}_i(\mathbf{x})$ in the world frame can then be obtained by using $\mathbf{R}_i(\mathbf{x})$, the rotation matrix of $\mathcal{A}_i(\mathbf{x})$, and $\mathbf{p}_i^c(\mathbf{x})$, the origin of the local coordinate system attached to $\mathcal{A}_i(\mathbf{x})$. In Line 5, the SDF between $\mathcal{A}_1(\mathbf{x})$ and $\mathcal{A}_2(\mathbf{x})$ is approximated by projecting the distance between $\mathbf{p}_1(\mathbf{x})$ and $\mathbf{p}_2(\mathbf{x})$ onto the contact normal $\hat{\mathbf{n}}$. Fig. 3(a) illustrates the SDF approximation in Lines 4 and 5. In Lines 6 and 7, we linearize the SDF around the Gaussian mean and analytically calculate the PoSSDF using the fact that the probability of a linear inequality for a Gaussian variable can be explicitly expressed as follows [25],

$$Pr(\mathbf{a}^T\cdot\mathbf{x}\leqslant b)=\frac{1}{2}(1-\text{erf}(\frac{\mathbf{a}^T\cdot\hat{\mathbf{x}}-b}{\sqrt{2\mathbf{a}^T\Sigma\mathbf{a}}})), \tag{19}$$

where $\text{erf}(\cdot)$ represents the Gauss error function, and $\mathbf{a}\in\mathbb{R}^{d_x}, b\in\mathbb{R}$ form the linear constraint of $\mathbf{x}$.

Alg. 1 provides a general procedure to calculate PoSSDF. Denote $\mathbf{x}=\left[\mathbf{z}_k^{r\,T},\mathbf{z}_k^{t\,T}\right]^T$, we can make the following adjustment and apply Alg. 1 to efficiently calculate Γ_k. To take the variable length of the LOS $\mathcal{L}_k(\mathbf{x})$ into account when calculating γ^{lo}, we precalculate the separative ratio $\hat{\lambda}$ of $\mathcal{L}_k(\hat{\mathbf{x}})$ before carrying out Alg. 1, which is formulated as $\hat{\lambda}=\|\hat{\mathbf{p}}_1-\hat{\mathbf{x}}_k^t\|/\|\hat{\mathbf{x}}_k^r-\hat{\mathbf{x}}_k^t\|$, where $\hat{\mathbf{p}}_1$ is the nearest point on $\mathcal{L}_k(\hat{\mathbf{x}})$. We then replace the SDF parameter $\hat{p}_1^L$ with $\hat{\lambda}$ and reformulate the approximated nearest point (Line 4) on $\mathcal{L}_k(\mathbf{x})$ in the world frame as

$$\mathbf{p}_1(\mathbf{x})=\hat{\lambda}\mathbf{x}_k^r(\mathbf{x})+(1-\hat{\lambda})\mathbf{x}_k^t(\mathbf{x}). \tag{20}$$

This adjustment for calculating γ^{lo} is illustrated in Fig. 3(b).

Note that Eq. (17) is not violated even though γ^{ro} is approximated with Alg. 1. This is because in Alg. 1, the obstacle is expanded to a half space with the closest point on its boundary and $\hat{\mathbf{n}}$ being its normal vector, which overestimates γ^{ro} thus ensuring Eq. (17) to be strictly enforced.

C. Sequential Convex Optimization

After being specified in Secs. IV-A and IV-B, problem Eq. (7) can be solved using the SCP algorithm, where the nonlinear constraints are converted into l_1 penalty functions,

$$J_m=J+\eta(\sum_i|g_i^n|^+ +\sum_i|h_i^n|). \tag{21}$$

Here J denotes the objective function Eq. (7a), and g_i^n, h_i^n represent the nonlinear inequality and equality constraints, respectively. Here η is the penalty coefficient and $|x|^+=\max(x,0)$. The SCP consists of two loops: The outer loop progressively increases η to drive the nonlinear constraint violation to zero, while in the inner loop, the trust region method is implemented to minimize the new objective function J_m. Interested readers can refer to [22] for details.

Directly adopting this framework turns out to be slow due to the frequent calls of GJK algorithms and EPA when calculating the gradient of Eq. (21). To overcome this limitation, we use a fixed SDF parameter set that encodes all the SDF parameters in the MPC horizon to calculate $\Gamma_{k+1:k+N}$ in the gradient, and update the SDF parameter set after obtaining a new solution. This two-stage strategy significantly speeds up the planner with minor accuracy loss, and is further described in the trajectory planning framework presented in Alg. 2.

The variable x_0 is initialized with zero control inputs at the first step, and is extrapolated from its value at the previous step for all subsequent steps. This is followed by precalculating an SDF parameter set $\mathcal{C}_0$ at the mean value of the initial robot and target beliefs, which is propagated by x_0 (Line 7). Using a fixed $\mathcal{C}_0$, we can fastly obtain the gradient (Line 10) and linearize the objective function (Line 11). After solving the linearized problem, the SDF parameter set is updated (Line 13) and utilized to update the trust region size (Line 14). The inner loop ends when the improvement is small (Line 15). The outer loop checks the terminal condition for the solution x^* (Line 18) and increases the penalty coefficient η (Line 19).

Algorithm 2: Trajectory Planning in SCP Framework with Two-Stage Strategy for Efficient SDF Calculation

1 PARAMETERS:
2 η : initial penalty coefficient
3 d : initial trust region size
4 β : increase ratio of the penalty coefficient
5 τ_c, τ_p, τ_f : convergence tolerances
6 OUTPUT: $x^*=\mathbf{u}_{k:k+N-1}^r$: the optimal solution

7 $x_0, \mathcal{C}_0 \leftarrow$ Initialize the solution and SDF parameter set
8 **while** *TRUE* **do**
9 **while** *TRUE* **do**
10 $\nabla J_x \leftarrow$ Get the gradient of J_m on x_0 using $\mathcal{C}_0$
11 $\tilde{J}(x,\eta,\mathcal{C}_0) \leftarrow J_m(x_0,\eta,\mathcal{C}_0)+\nabla J_x\cdot(x-x_0)$
12 $x^* \leftarrow \textbf{argmin}_x \tilde{J}(x,\eta,\mathcal{C}_0)$ subject to trust region, linear constraints, and semidefinite constraints of covariance matrices.
13 $\mathcal{C}^* \leftarrow$ Update SDF parameter set from x^*
14 $d \leftarrow$ Update trust region size by improvement ratio $\frac{J_m(x_0,\eta,\mathcal{C}_0)-J_m(x^*,\eta,\mathcal{C}^*)}{J_m(x_0,\eta,\mathcal{C}_0)-\tilde{J}(x^*,\eta,\mathcal{C}_0)}$
15 **If** $\|x^*-x_0\|\leqslant\tau_c$ **or** $|\nabla J_x\cdot(x^*-x_0)|\leqslant\tau_f$ **break**
16 $x_0\leftarrow x^*$, $\mathcal{C}_0\leftarrow\mathcal{C}^*$ If trust region is expanded
17 **end**
18 **If** $\sum_i|g_i^n(x^*,\mathcal{C}^*)|^+ +\sum_j|h_j^n(x^*,\mathcal{C}^*)|\leqslant\tau_p$ **break**
19 $\eta\leftarrow\beta\cdot\eta$
20 **end**

V. Simulations

The proposed method is validated through multiple simulations in MATLAB using a desktop (12th Intel(R) i7 CPU@2.10GHz), and the MOSEK solver is adopted to optimize the trajectory according to the SCP routine. The robot takes a unicycle model, where the robot state $\mathbf{z}_k^r = [\mathbf{x}_k^{r\,T}, \theta_k^r, v_k^r]^T \in \mathbb{R}^4$ contains position $\mathbf{x}_k^r \in \mathbb{R}^2$, orientation $\theta_k^r \in (-\pi, \pi]$, and velocity $v_k^r \in \mathbb{R}_+$, and the robot control input $\mathbf{u}_k^r = [\omega_k^r, a_k^r]^T \in \mathbb{R}^2$ is composed of angular velocity $\omega_k^r \in \mathbb{R}$ and acceleration $a_k^r \in \mathbb{R}$. The robot dynamics follow the following motion model:

$$\mathbf{f}^r(\mathbf{z}_k^r, \mathbf{u}_k^r) = \mathbf{z}_k^r + [v_k^r \cos\theta_k^r,\ v_k^r \sin\theta_k^r,\ \omega_k^r,\ a_k^r]^T \cdot \Delta t, \quad (22)$$

with $\Delta t = 0.5s$ representing our sampling interval. We set the range of robot acceleration (m/s^2) as $-4 \leqslant a_k^r \leqslant 2$, angular velocity (rad/s) as $-\pi/3 \leqslant \omega_k^r \leqslant \pi/3$ and speed limit as $4m/s$. The motion noise is set as $\mathbf{R}^r = 10^{-3} \cdot diag(4, 4, 0.4, 0.4)$. The robot's sensor FOV is modeled as an annular sector, with minimal detection distance $r_1 = 2m$, maximal distance $r_2 = 10m$ and the sensing angle ψ^s is $2\pi/3$. We shrink the FOV to its convex subset when calculating Eq. (13) to fit the inputs of GJK algorithm and EPA. The predictive horizon is set as $N = 4$. A $60m \times 50m$ map with cluttered polygon obstacles is designed in our simulation tests (Fig. 4(a)). The proposed method only considers obstacles near the LOS, which we note as "valid obstacles". The initial state of the robot is designated as $[32, 7, \frac{3}{4}\pi, 0]^T$.

To quantitatively evaluate the performance of our method, several metrics are computed for a tracking simulation with a total step T and target trajectory $\{\tilde{\mathbf{x}}_k^t, k = 1, \cdots, T\}$, including computing time t_{cal}, loss rate r_{los} that denotes the percentage of time the robot cannot see the target, and the estimation error e_{est} that is defined as the mean absolute error (MAE) of the target's estimated position, *i.e.*, $e_{est} = \frac{1}{T}\sum_{k=1}^{T} \|\hat{\mathbf{x}}_k^t - \tilde{\mathbf{x}}_k^t\|$. Besides, we claim a *tracking failure* if the robot collides with an obstacle or loses sight of the target in 15 consecutive steps, and we define the success rate r_{suc} over multiple tracking experiments as the proportion that the robot finishes the tracking task without failure.

A. Performance Analysis in a Challenging Scenario

We design a target trajectory with several sharp turns near obstacles to evaluate the performance of the tracker under challenging scenarios, as shown in Fig. 4(b). We adopt a single integrator to describe the target motion model, which is formulated as $\mathbf{z}_{k+1}^t = \mathbf{z}_k^t + \mathbf{u}_k^t \cdot \Delta t$, where the target state consists of its position, *i.e.*, $\mathbf{z}_k^t = \mathbf{x}_k^t$, and target control $\mathbf{u}_k^t \in \mathbb{R}^2$ is known to the robot. Covariance $\mathbf{R}^t$ is set as $0.01\mathbf{I}_2$, where $\mathbf{I}_2$ denotes the 2×2 identity matrix. A range-bearing sensor model is adopted to acquire the distance and bearing angle of the target relative to the robot, which is written as:

$$\mathbf{f}^s(\mathbf{z}_k^t, \mathbf{z}_k^r) = [\|\mathbf{x}_k^t - \mathbf{x}_k^r\|, \angle(\mathbf{x}_k^t - \mathbf{x}_k^r) - \theta_k^r]^T. \quad (23)$$

The covariance $\mathbf{R}^s$ is set as $diag(0.3, 0.05)$. The target position is initialized at $[28, 9]^T$. We use cumulative entropy J_1 in Eq. (12) as our objective function.

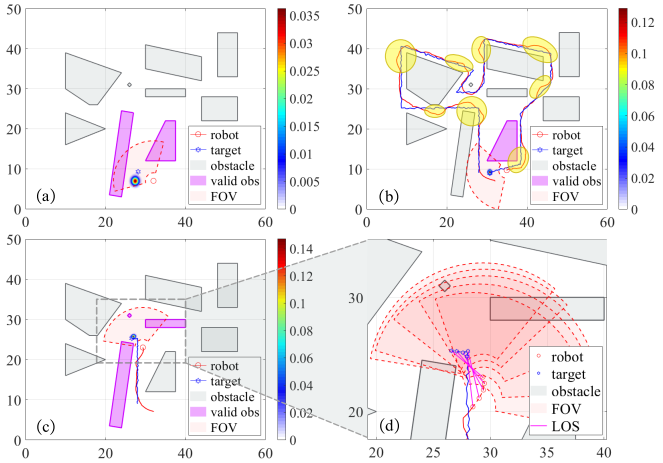


Fig. 4: **Visibility-aware tracking of the target with linear motion model.** The colorful background represents the target's PDF, and the color bar shows the colormap of probability. The "valid obs" denotes the valid obstacles. (a) Initial simulation scenario. (b) Entire tracking trajectory. (c) and (d) highlight a sharp turn of the target and display the robot's trajectory with high visibility.

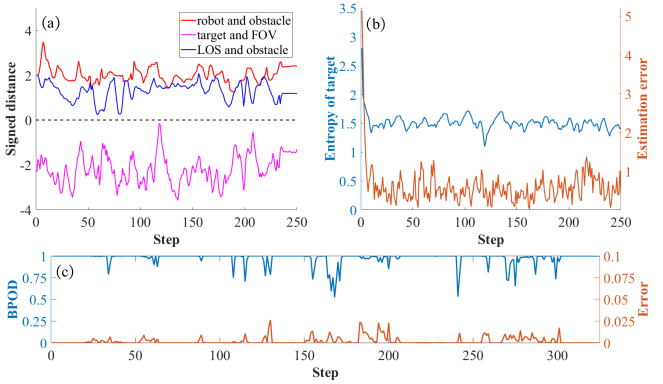


Fig. 5: **Trajectory performance.** (a) History of signed distances. (b) Target entropy and estimation error. (c) The BPOD value and the approximation error of the BPOD compared to the ground truth.

Fig. 4 shows the tracking process. The target uncertainty is initialized to be very large (Fig. 4(a)). As the tracking progresses, the robot plans a visibility-aware trajectory to track the target (Fig. 4(b)). Specifically, when the target takes a sharp turn near an obstacle, the robot moves away from the obstacle in advance to reduce the likelihood of target loss (Fig. 4(c) and (d)). Note that this roundabout path frequently appears in the tracking process (highlighted by yellow ellipses in Fig. 4(b)) and is a characteristic of visibility-aware trajectories.

Fig. 5 displays the performance data. The robot can maintain the visibility of the target throughout the simulation without any target loss. To see this, Fig. 5(a) shows the purple curve that indicates the target is inside our convexified FOV most of the time, and the blue curve that indicates LOS and obstacles never intersect. Due to uninterrupted measurements, target entropy rapidly converges and the estimation error is kept small (Fig. 5(b)). We also record the BPOD value and evaluate the accuracy of the BPOD approximation against the Monte-Carlo simulation as the ground truth, as shown in Fig. 5(c). The subfigure shows that the BPOD is

kept near 1 most of the time, and simulation results also demonstrate that we can calculate one BPOD within $1ms$ with an MAE of 2.5×10^{-3}.

The experiment is conducted 30 times for quantitative performance evaluation. The robot completes the tracking task in 28 cases without failure, and the results show that the robot can track the target with low mean estimation error $\bar{e}_{est}=0.308m$, low mean loss rate $\bar{r}_{los}=4.1\%$ with the mean computing time $\bar{t}_{cal}=0.176s/step$.

B. Evaluation with Stochastic Target Trajectories

We benchmark our method against a representative visibility-aware trajectory planning method based on [11]. To ensure a fair comparison, both methods utilize an MPC framework with identical robot kinematics constraints and SCP solver. The main distinction lies in that we adapt the deterministic visibility cost and the collision cost in [11] to the baseline MPC framework as the objective function.

The target takes a unicycle model [26] with the target state represented as $\mathbf{z}_k^t=[\mathbf{x}_k^{t\,T},\theta_k^t]^T$, where θ_k^t stands for target orientation. In addition, the target controls $\mathbf{u}_k^t=[v_k^t,\omega_k^t]^T$ that encode the speed $v_k^t\in\mathbb{R}$ and angular velocity $\omega_k^t\in\mathbb{R}$ remain unknown to the robot. To estimate and predict target belief according to the EKF fashion, the robot estimates $\mathbf{u}_k^t$ by differentiating target displacement and rotation. We choose the cumulated BPOD J_2 in Eq. (12) as the objective function. Besides, we adopt a camera sensor model to detect target distance, bearing angle and orientation:

$$\mathbf{f}^s(\mathbf{z}_k^t,\mathbf{z}_k^r)=[\|\mathbf{x}_k^t-\mathbf{x}_k^r\|,\angle(\mathbf{x}_k^t-\mathbf{x}_k^r)-\theta_k^r,\theta_k^t-\theta_k^r]^T. \quad (24)$$

We test both our algorithm and the baseline algorithm under different scales of measurement noise. Specifically, the measurement noise is set as $10^{-2}\beta\cdot diag(1,0.5,1)$, with $\beta\in\{0.5,2,5\}$ corresponding to different uncertainty levels. For each level, 50 randomly generated target trajectories are tested to compare the tracking performance of the two planners. The target speed limit is set as $3m/s$ and each trajectory has 400 time steps. Target motion noise is set as $0.5\mathbf{I}_3$ to simulate stochastic target movements.

A pair of comparative tracking experiments is presented in Fig. 6 to highlight the negative impact of system uncertainty on a deterministic trajectory planner and to showcase the robustness of our approach. Simulation metrics comparisons are depicted in Fig. 7. The results show that in low-noise scenarios, both planners provide precise estimations for target states and effectively keep the target visible. However, in the face of higher system uncertainty, the proposed planner consistently maintains low estimation error, high visible rate, and high success rates when tracking a target, while the performance of the baseline deteriorates significantly. It is important to note that although our method requires additional time for trajectory planning due to extra computational cost for considering the system uncertainty, the overall duration remains below 0.1 seconds, which is sufficient for real-time planning. The overall metrics show that our method achieves a better balance between efficiency and effectiveness in the face of high uncertainty.

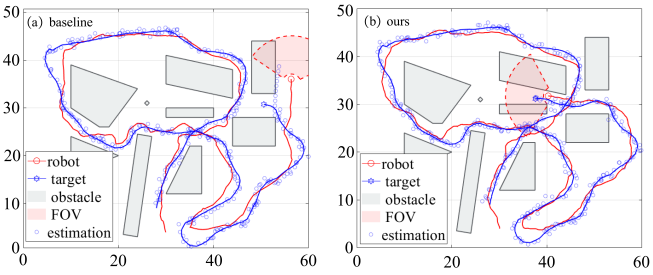


Fig. 6: **Trajectory comparisons between baseline method (a) and ours (b).** In the face of stochastic target trajectories and high system noise ($\beta=5$), the baseline method generates a more tortuous trajectory and finally loses the target due to inaccurate estimations. In contrast, our method generates a smoother trajectory and succeeds in completing the tracking task.

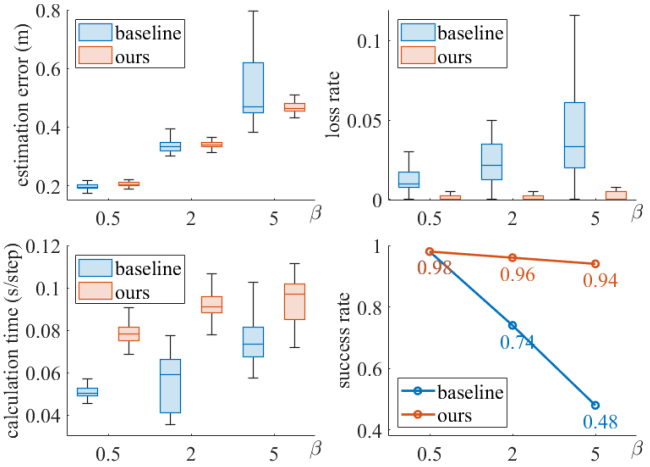
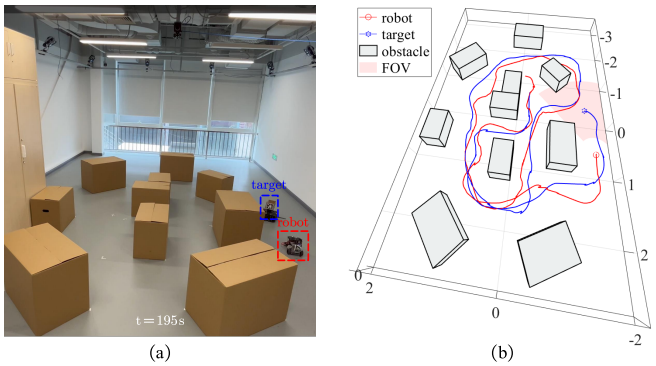


Fig. 7: **Box graphs of the performance metrics under different uncertainty levels β.** The estimation error, loss time and calculation time are all averaged over 50 tracking experiments.

VI. Real-World Experiments

The proposed approach has been tested by using a Wheeltec ground robot equipped with an ORBBEC Femto W camera to keep track of a moving Turtlebot3, on which three interlinked Apriltags are affixed. The speed limit of the tracker and the target are $0.4m/s$ and $0.31m/s$, respectively. The onboard processor (NVIDIA Jetson TX1) on the Wheeltec robot calculates the distance, bearing angle, and orientation of the detected Apriltag from the camera images and transmits the messages through ROS to a laptop (11th Intel(R) i7 CPU@2.30GHz) that performs the planning algorithm. The parameters of the sensor model are calibrated as $r_1=0.3m$, $r_2=1.5m$, $\theta=100^\circ$, and $\mathbf{R}^s=diag(0.069,8.3\times10^{-4},0.0055)$. The covariance of robot motion noise is set as $\mathbf{R}^r=diag(0.08\mathbf{I}_2,0.055\mathbf{I}_2)$ to handle model discretization error and mechanical error. Other parameters are kept the same as Sec. V-B except for the distance parameter that has been scaled down by a factor of 10. The ground truth of the robot and target poses are obtained by a Vicon motion-capture system.

We conduct three tracking experiments by remotely controlling the target to randomly traverse an indoor map with cluttered obstacles (Fig. 8(a)), and the duration of all three experiments exceed 2 minutes. For each experiment, we record the total planning steps T, estimation error e_{est},



(a) (b)

Fig. 8: **Real-world target tracking.** (a) Map configuration and one tracking frame. (b) Simulation scene corresponding to (a) and the trajectories of the target and the robot.

TABLE I: Real-World Experiment Results

	T	e_{est}(m)	t_{cal}(s/step)	d_{min}(m)
1	538	0.072	0.087	0.247
2	497	0.078	0.089	0.210
3	471	0.071	0.090	0.232

calculation time t_{cal} and minimum distance to obstacles d_{min}, as shown in Tab. I. Both Fig. 8(b) and Tab. I show that our planner can generate safe trajectories for the robot to track a moving target while reducing target uncertainty in real-world scenarios.

VII. Conclusion

We propose a target-tracking approach that systematically accounts for the limited FOV, obstacle occlusion, and state uncertainty. In particular, the concept of BPOD is proposed and incorporated into the EKF framework to predict target uncertainty in systems subject to measurement noise and imperfect motion models. We subsequently develop an SDF-based method to efficiently calculate the BPOD and collision risk to solve the trajectory optimization problem in real time. Both simulations and real-world experiments validate the effectiveness and efficiency of our approach.

Future work includes mainly two aspects. First, we will investigate the properties and applications of BPOD in non-Gaussian belief space planning. Second, we will extend the proposed method to unknown and dynamic environments.

Acknowledgement: We thank Dr. Meng Wang at BIGAI for his help with photography.

References

[1] G. Punyavathi, M. Neeladri, and M. K. Singh, "Vehicle tracking and detection techniques using iot," *Materials Today: Proceedings*, vol. 51, pp. 909–913, 2022.

[2] X. Xu, G. Shi, P. Tokekar, and Y. Diaz-Mercado, "Interactive multi-robot aerial cinematography through hemispherical manifold coverage," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[3] M. Xanthidis, M. Kalaitzakis, N. Karapetyan, J. Johnson, N. Vitzilaios, J. M. O'Kane, and I. Rekleitis, "Aquavis: A perception-aware autonomous navigation framework for underwater vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[4] S. M. LaValle, H. H. González-Banos, C. Becker, and J.-C. Latombe, "Motion strategies for maintaining visibility of a moving target," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1997.

[5] R. Zou and S. Bhattacharya, "On optimal pursuit trajectories for visibility-based target-tracking game," *IEEE Transactions on Robotics (T-RO)*, vol. 35, no. 2, pp. 449–465, 2018.

[6] E. Lozano, I. Becerra, U. Ruiz, L. Bravo, and R. Murrieta-Cid, "A visibility-based pursuit-evasion game between two nonholonomic robots in environments with obstacles," *Autonomous Robots*, vol. 46, no. 2, pp. 349–371, 2022.

[7] E. Lozano, U. Ruiz, I. Becerra, and R. Murrieta-Cid, "Surveillance and collision-free tracking of an aggressive evader with an actuated sensor pursuer," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 3, pp. 6854–6861, 2022.

[8] Y. Su, Y. Jiang, Y. Zhu, and H. Liu, "Object gathering with a tethered robot duo," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 2132–2139, 2022.

[9] C. Liu and J. K. Hedrick, "Model predictive control-based target search and tracking using autonomous mobile robot with limited sensing domain," in *American Control Conference (ACC)*, 2017.

[10] T. Li, L. W. Krakow, and S. Gopalswamy, "Sma-nbo: A sequential multi-agent planning with nominal belief-state optimization in target tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[11] Q. Wang, Y. Gao, J. Ji, C. Xu, and F. Gao, "Visibility-aware trajectory optimization with application to aerial tracking," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[12] 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 (RA-L)*, vol. 3, no. 4, pp. 3725–3732, 2018.

[13] H. Masnavi, J. Shrestha, M. Mishra, P. Sujit, K. Kruusamäe, and A. K. Singh, "Visibility-aware navigation with batch projection augmented cross-entropy method over a learned occlusion cost," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 4, pp. 9366–9373, 2022.

[14] S. Patil, Y. Duan, J. Schulman, K. Goldberg, and P. Abbeel, "Gaussian belief space planning with discontinuities in sensing domains," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[15] H. Yu, K. Meier, M. Argyle, and R. W. Beard, "Cooperative path planning for target tracking in urban environments using unmanned air and ground vehicles," *IEEE/ASME Transactions on Mechatronics (TMECH)*, vol. 20, no. 2, pp. 541–552, 2014.

[16] C. Zhang and I. Hwang, "Multiple maneuvering target tracking using a single unmanned aerial vehicle," *Journal of guidance, control, and dynamics*, vol. 42, no. 1, pp. 78–90, 2019.

[17] K. Zhou, P. Wu, Y. Su, H. Gao, J. Ma, H. Liu, and C. Liu, "Aspire: An informative trajectory planner with mutual information approximation for target search and tracking," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[18] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, 2004.

[19] S. Papaioannou, P. Kolios, T. Theocharides, C. G. Panayiotou, and M. M. Polycarpou, "Jointly-optimized searching and tracking with random finite sets," *IEEE Transactions on Mobile Computing*, vol. 19, no. 10, pp. 2374–2391, 2019.

[20] H. Gao, P. Wu, Y. Su, K. Zhou, J. Ma, H. Liu, and C. Liu, "Probabilistic visibility-aware trajectory planning for target tracking in cluttered environments," *arXiv preprint arXiv:2306.06363*, 2023.

[21] L. Blackmore, M. Ono, and B. C. Williams, "Chance-constrained optimal path planning with obstacles," *IEEE Transactions on Robotics (T-RO)*, vol. 27, no. 6, pp. 1080–1094, 2011.

[22] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 9, pp. 1251–1270, 2014.

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

[24] G. Van Den Bergen, "Proximity queries and penetration depth computation on 3d game objects," in *Game developers conference*, vol. 170, 2001.

[25] A. Prékopa, *Stochastic programming*, vol. 324. Springer Science & Business Media, 2013.

[26] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.