Ergodic Trajectory Planning with Dynamic Sensor Footprints

Ziyue Zheng[1,2], Yongce Liu[1], Hesheng Wang[3], *Senior Member, IEEE,* and Zhongqiang Ren[1†], *Member, IEEE*

***Abstract*—This paper addresses the problem of trajectory planning for information gathering with a dynamic and resolution-varying sensor footprint. Ergodic planning offers a principled framework that balances exploration (visiting all areas) and exploitation (focusing on high-information regions) by planning trajectories such that the time spent in a region is proportional to the amount of information in that region. Existing ergodic planning often oversimplifies the sensing model by assuming a point sensor or a footprint with constant shape and resolution. In practice, the sensor footprint can drastically change over time as the robot moves, such as aerial robots equipped with downward-facing cameras, whose field of view depends on the orientation and altitude. To overcome this limitation, we propose a new metric that accounts for dynamic sensor footprints, and propose efficient numerical trajectory optimization algorithms with computational reuse based on analytical derivation of sensor footprint models. Experimental results show that the proposed approach can simultaneously optimize both the trajectories and sensor footprints, with up to an order of magnitude better ergodicity than conventional methods. We deploy our approach on real robots for verification.**

***Index Terms*—Motion and Path Planning, Optimization and Optimal Control, Ergodic Search**

I. Introduction

This paper investigates a trajectory planning problem for area search, which arises in applications such as search and rescue [1] and target localization [2]. Given an information map and a probability distribution in the area to be searched, the problem requires planning a trajectory to gather information from this information map efficiently. Existing approaches to solve this problem span a spectrum that ranges from information-theoretic approaches [3], [4], which greedily move the robot to the next location with the highest information gain, to uniform coverage methods [3], [5], which guarantee complete exploration of the entire information map. In the middle of this spectrum lies the ergodic search [6]–[8], which plans trajectories by minimizing an ergodic metric so that, along the planned trajectory, the amount of time spent in a region is proportional to the amount of information in that region. Ergodic search provides a framework to inherently balance between exploration (i.e., covering all possible locations for new information) and exploitation (i.e., myopically searching high-information areas) and is thus able to intelligently plan the robot motion to gather information in the long run.

This work was supported by the Natural Science Foundation of China under Grant 62403313. *(Corresponding author: Zhongqiang Ren. email: zhongqiang.ren@sjtu.edu.cn)*

Ziyue Zheng is with Shanghai Jiao Tong University, Shanghai 200240, China, and also with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218 USA.

Yongce Liu and Zhongqiang Ren are with the Global College, Shanghai Jiao Tong University, Shanghai 200240, China.

Hesheng Wang is with the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai 200240, China.

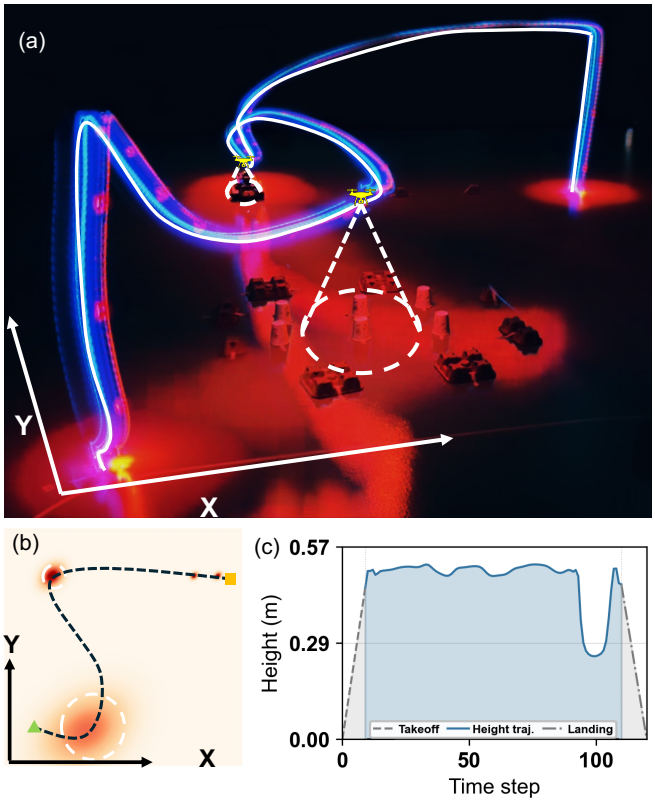


Fig. 1. Ergodic trajectory planning with dynamic sensor footprint. (a) shows our real-robot experiments, where a drone covers an information map under a motion capture system. The drone is equipped with a downward pointing LED to visualize the dynamic sensor footprint whose size depends on the flying height of the robot. With our approach, the robot flies high over regions with widespread information and flies low over regions with concentrated information. (b) shows the information map to be covered, and the top-down view of the trajectory. (c) shows the corresponding height values (z coordinate) of the robot along the trajectory. The orange shadow on the ground in (a) shows the time-averaged statistics of the dynamic sensor footprint, which is similar to the information map to be covered as shown in (b).

Early research on ergodic search simplifies the sensor detection range (i.e., sensor footprint), which describes the footprint as either a single point [6], [7], [9] in the workspace or as constant size geometry such as circles or rectangles [10]. In practice, the sensor footprint often varies its shape and size as the robot moves and such variation is coupled with both the robot states such as position and orientation, and the space to be covered such as non-flat surface or 3D objects. For example, consider a drone with a camera whose optical axis points downward and perpendicular to the ground to collect information (Fig. 1). The field of view (FOV) forms a cone. The sensor footprint projected onto the ground is a circle whose radius changes with the drone's altitude. Moreover, the footprint size correlates negatively with the sensor resolution:

At higher altitudes, the footprint is larger and the resolution is coarser, which collects rough information over a broad area; At lower altitudes, the footprint is smaller and the resolution is finer, which gathers detailed information within small regions. When the ground is a non-flat surface, sensor footprint geometry can be more complex, which plagues the trajectory planning. Ergodic search with varying sensor footprint (or end-effector footprints) on non-flat surfaces is a rising topic recently, for the purpose of surface finishing [11], full-body exploration [12], [13] and 3D object inspection [14], [15], to name a few. However, these approaches usually simplify the dependency of footprint geometry, and the corresponding derivatives, on both robot dynamics and the surface to be covered, which may then affect and burden the underlying trajectory optimization.

To capture these dynamic sensing characteristics and their influence on the trajectory optimization, this paper seeks to integrate a dynamically changing sensor footprint and resolution into the ergodic search framework. In particular, we first focus on 2D non-flat surfaces with bounded curvature (such as hills) covered by a view cone of a robot flying in SE(3). These surfaces can often be described by manifolds that are topologically equivalent to a 2D plane. The major challenge there is deriving the geometry of the sensor footprint when the robot flies in SE(3), which leads to oblique views, and its Jacobian with respect to the robot pose, which is often necessary for efficient trajectory optimization.

To address the challenges, first, we introduce a new footprint ergodic metric which computes the time-averaged statistics of the dynamic sensor footprints and compares it against the information map to be covered. The proposed footprint ergodic metric converges to the existing ergodic metric [6] as the sensor footprint approaches a single point in the limit. Then, with the proposed footprint ergodic metric, we formulate a corresponding ergodic optimal control problem (OCP). Directly solving this OCP is difficult, since the footprint is a set of infinitely many points and the proposed footprint ergodic metric involves an integration over the set which can be inefficient to compute. We thus propose a numerical method to compute it approximately based on a finite number of samples within the set. As the number of samples increases and approaches infinity, our numerical approximation converges to the true footprint ergodic metric. To incorporate more sophisticated constraints from practice, we leverage both the numerical footprint ergodic metric and the Augmented Lagrangian within an iLQR framework to plan the trajectories, and we name this approach Footprint Ergodic Optimization (FEO).

Our FEO involves two novel ideas. First, we approximate smooth surface to be covered with local tangent planes at the intersection points of the sensor optical axis with the surface as the robot moves, and derive the analytical expression of the Jacobian of the position of any point within the sensor footprint on this tangent plane with respect to the robot pose in SE(3). These analytical expressions give us the insight that the sensor footprint with respect to the robot position is inherently *linear*. Intuitively, it aligns with the fact that, the sensor footprint grows proportionally larger or smaller as the robot translates in 3D (e.g. flying lower or higher even with oblique views). Second, this linearity suggests that, parts of these Jacobians remain constant during trajectory optimization, which enables *computational reuse* across iterations and helps reduce the runtime of each iteration during the trajectory optimization. Following this idea, we further look into the computation and leverage computational reuse to speed up our FEO. Besides, in FEO, we also propose a log-spaced line search approach to replace the popular Armijo line search [16] that is widely used in trajectory optimization, which helps reduce the number of iterations of optimization.

For verification, we compare our FEO on various flat and non-flat surfaces in simulation against several baselines, including conventional ergodic search [6], ergodic search with fixed Gaussian footprint [10], extension of fixed Gaussian footprint to meshable surfaces [17], and the recent HEDAC for complex structures [14] and non-planar surfaces [12], [13]. Our approach often achieves up to more than an order of magnitude smaller (better) ergodicity due to the consideration of dynamic sensor footprint, while running fast due to the proposed computational reuse and log-spaced line search, which speeds up our FEO by up to 70%. Intuitively, our FEO allows the drone to intelligently fly further away from the surface in regions with low-density information, and closer to the surface in areas with concentrated information. We deploy our approach with a drone and motion capture in a lab setting for ergodic search with dynamic sensor footprint.

II. Related Work

A. Ergodic Search

Ergodic search plans trajectories by minimizing the difference between the robot's time-averaged statistics and the workspace's information distribution. Several approaches have been proposed to evaluate the difference, including spectral multi-scale coverage using Fourier analysis [6], Kullback–Leibler divergence with Gaussian mixture models [18], kernel ergodic metric for Lie group spaces [19], and maximum mean discrepancy metric [20]. For trajectory optimization, gradient-based methods are widely adopted. Some techniques involve feedback control for dynamic systems [6], receding-horizon planning [2], and LQR-based optimization [7]. Recent developments have incorporated additional considerations like time efficiency [21], multi-objective criteria [22], and inter-robot connectivity [23].

B. Ergodic Coverage of 3D Targets

Although being generic to the general workspace, most experiments on ergodic search are limited to a 2D workspace [6], [9]. Recently, some work investigates coverage in 3D workspaces, including Multi-UAV trajectory planning for complex structures [14] and coverage of non-planar surfaces in object inspection or surface scanning with robotic arms [13], [17]. Some of these methods extend the ergodic metric [6] to handle manifolds, which enables coverage of non-planar surfaces of a 3D target object [17]. Another approach HEDAC adopts a diffusion-based strategy, where ergodic trajectories are generated by solving a heat equation defined over the domain of interest [12]–[14]. Although general and versatile,

this approach requires solving PDE repetitively, which can be computationally expensive [17].

C. Dynamic Sensor Footprints

Information gathering problems were investigated in many ways, such as sensor coverage for planning [24], [25], sensor view points in SLAM, target detection and tracking [26], [27]. However, most of them simplify the sensor as a point in the workspace and ignore the detection range and resolution of the sensor [28], or as a fixed geometry such as rectangles [29], or spheres [30] without considering that the sensor detection range and resolution vary with robot dynamics. Other works consider varying sensor footprints and focus mainly on uniform coverage [31], [32] or environmental monitoring with different resolutions [33]. Recent research considers varying footprints or non-flat surfaces in tactile coverage [13], surface finishing [11], 3D object inspection [14] This paper complements these research by investigating the geometry of sensor footprint in 3D, the resulting gradient and Jacobian, and their use for fast computation for ergodic trajectory optimization.

III. Preliminaries

A. Workspace and Trajectories

Let $\mathcal{W} = [0, L_1] \times \cdots \times [0, L_\nu], \nu = 2, 3$ denote a bounded ν-dimensional workspace, which is to be explored by the robot.[1] The mobile robot (such as a drone) has an n-dimensional state space ($n \geq \nu$), and let $x : [0, T] \to \mathbb{R}^n$ denote a trajectory in the state space with $T \in \mathbb{R}^+$ representing the time horizon. At any time $t \in [0, T]$, the state of the robot $x(t)$ must stay within the set $\mathcal{X} \subseteq \mathbb{R}^n$, which is a set of allowed states. Here, $\mathcal{X}$ describes the constraints on the robot's state, such as avoiding collision with the static obstacles in the workspace or staying within the bounded workspace. The robot has deterministic dynamics given by $\dot{x}(t) = f_d(x(t), u(t))$, where $u(t) \in \mathcal{U}$ is the control input of the robot. Let $f_q : \mathbb{R}^n \to \mathcal{W}$ denote a map that projects a state $x(t)$ to the corresponding point $q(t) \in \mathcal{W}$ in the workspace, i.e., $q(t) = f_q(x(t))$.

B. Ergodic Metric

Let $\phi(w) : \mathcal{W} \to \mathbb{R}_0^+$ denote an information map over the workspace, which is a time-invariant probability distribution function with $\int_{\mathcal{W}_-} \phi(w) dw = 1$, describing the information density at each point in the workspace $\mathcal{W}$. Based on the trajectory $q(t)$, the time-averaged statistics for the robot is defined as follows [6].

$$c(w, x(t)) = c(w, f_q(x(t))) = \frac{1}{T} \int_0^T \delta(w - q(t)) dt \quad (1)$$

[1]The workspace in this paper is simply the space of information to be explored by the robot, as opposed to the space where the robot can move. In other words, the robot can move in a higher-dimensional space than the workspace. For example, the robot can be a drone flying in 3D space while exploring the information in a 2D workspace. The primary focus of this work are 2D workspace, i.e., 2D information maps. Later in Sec. VI about coverage of a general 2D manifold embedded in 3D, the underlying information map is still 2D. We discuss how the proposed techniques can be potentially extended to cover a truly 3D information map in Sec. VIII.

where $\delta(w)$ is the Dirac delta function such that $\delta(\mathbf{0}) = +\infty$ and $\delta(w) = 0, w \neq \mathbf{0}$, satisfying $\int_{\mathcal{W}_-} \delta(w) dw = 1$.

With the information distribution $\phi(w)$ and the time-averaged statistics (1), the ergodic metric (ergodicity) is defined as follows [6].

$$\mathcal{E}(\phi, q(t)) = \sum_{\mathbf{k} \in \mathcal{K}} \Lambda_{\mathbf{k}} (c_{\mathbf{k}} - \phi_{\mathbf{k}})^2 \quad (2)$$

$$= \sum_{\mathbf{k} \in \mathcal{K}} \Lambda_{\mathbf{k}} \left(\frac{1}{T} \int_0^T F_{\mathbf{k}}(q(t)) dt - \int_{\mathcal{W}_-} \phi(w) F_{\mathbf{k}}(w) dw \right)^2.$$

Here, $c_{\mathbf{k}}$ and $\phi_{\mathbf{k}}$ are the Fourier coefficients of $c(w)$ and $\phi(w)$, respectively. $\mathbf{k} = (k_1, \cdots, k_\nu) \in \mathcal{K}$ is the frequency vector of the Fourier coefficients, and $\mathcal{K} \subset \mathbb{N}^\nu$ represents a selected set of frequencies in practical computation. Additionally, $F_{\mathbf{k}}(w) = \frac{1}{h_{\mathbf{k}}} \prod_{o=1}^{\nu} \cos \frac{k_o \pi}{L_o} w_o$ is the cosine basis function with the normalization term $h_{\mathbf{k}}$ [6], [21], [34]. $\Lambda_{\mathbf{k}} = (1 + \| \mathbf{k} \|_2^2)^{-(\nu+1)/2}$ is the weight of each Fourier coefficient.

C. Ergodic Trajectory Optimization Problem

Given an information map ϕ, the goal of the Ergodic Trajectory Optimization (ETO) problem is to find a dynamically feasible trajectory x of the robot that minimizes the ergodic metric $\mathcal{E}(\phi, q)$ and control effort:

$$\min_{x,u} \ \mathcal{E}(\phi, q) + \int_0^T u^T(t) R u(t) dt \quad (3a)$$

$$\text{s.t. } \dot{x} = f_d(x(t), u) \quad (3b)$$

$$x \in \mathcal{X}, u \in \mathcal{U} \quad (3c)$$

where R is a positive definite matrix describing the weight of the controls. Constraint 3b represents the dynamics of the robot, and 3c defines the constraints of state and control.

IV. Footprint Ergodic Metrics

A. Sensor Footprint and Trajectory

Definition 1 (Sensor Footprint)**.** *Let $\gamma(w, x) \subset \mathcal{W}, w \in \mathcal{W}$ denote the sensor footprint of the robot (over the information map in the workspace) when the robot state is x, satisfying $\int_{\mathcal{W}} \gamma(w, x) dw = 1$ and $\gamma(w, x) > 0, \forall w \in \mathcal{W}$.*

Intuitively, when the robot state is x, for any point in the workspace $w \in \mathcal{W}$, $\gamma(w, x)$ returns the "weight of measurement" at point w, and the weight integrated at all possible points $w \in \mathcal{W}$ should be equal to one. In other words, $\gamma(w, x)$ can be regarded as a probability distribution over the workspace $\mathcal{W}$ that depends on the parameter x, and $\gamma(w = w', x(t_k))$ is the value of the probability density function at a specific point $w' \in \mathcal{W}$.

Example 1. *Consider a 2D workspace ($\nu = 2$) to be explored by a flying robot with a downward pointing camera whose optic axis is always perpendicular to the workspace. The sensor footprint can be described by a uniform circle, whose radius depends on the flying height of the robot. Specifically, for each robot state trajectory x, let $f_h : \mathbb{R}^n \to \mathbb{R}^+$ denote a map that projects a state $x(t)$ to a positive real number $h(t)$*

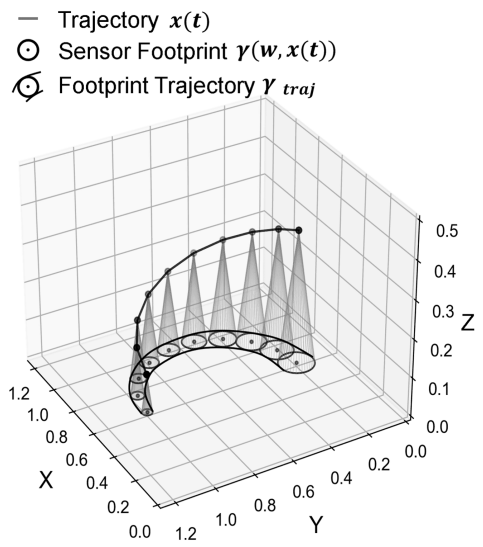


Fig. 2. Illustration of sensor footprint and footprint trajectory of a flying drone with downwards pointing camera over a 2D information map.

representing the height of the robot, i.e., $h(t) = f_h(x(t))$. Then, the sensor footprint $\gamma(w, x)$ can be described by the following uniform distribution within the circle (Fig. 2).

$$\gamma(w, x(t)) = \begin{cases} \frac{1}{\pi k_h^2 f_h(x)^2} & \text{if } ||w - f_q(x)||_2 \leq k_h f_h(x) \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $k_h \in \mathbb{R}^+$ is a known constant related to the downward viewing angle of the robot.[2]

When the robot moves along a trajectory $x(t), t \in [0, T]$, the corresponding sensor footprint varies along $x(t)$.

Definition 2 (Footprint Trajectory (FT))**.** *Let $\gamma(w, x(t))$ denote a* sensor footprint trajectory*, where for each time point $t_k \in [0, T]$, $\gamma(w, x(t_k))$ is a sensor footprint.*

Fig. 2 provides an illustration of footprint trajectory over a 2D information map. Since at any time point $t_k \in [0, T]$, $\gamma(w, x(t_k))$ is a probability distribution over the workspace, an FT $\gamma(w, x(t))$ is actually a *random process*. We will sample from this random process to develop a numerical approach for easy computation as presented later.

B. Footprint Ergodic Metrics and Problems

Similar to the time-averaged statistics of a robot trajectory $c(w, x)$ as mentioned in Eq. 1, we then introduce the time-averaged statistics of an FT $\gamma(w, x(t))$.

Definition 3 (Time-Averaged Statistics of FT)**.** *Let $c_\gamma(w, x(t)), w \in \mathcal{W}, f_q(x(t)) \in \mathcal{W}$ denote the time-averaged statistics of an FT $\gamma(w, x(t))$:*

$$c_\gamma(w, x(t)) = \frac{1}{T} \int_0^T \gamma(w, x(\tau)) d\tau. \quad (5)$$

[2]To simplify the presentation, this work assumes the entire footprint is always inside the workspace, i.e., $f_q(x)$ is inside $\mathcal{W}$ and is at least $k f_h(x)$ away from the nearest boundary of $\mathcal{W}$. In practice, this can be ensured by adding constraints on the robot's state as described by $x \in \mathcal{X}$.

Intuitively, Eq. 5 replaces the delta function δ in Eq. 1 with the footprint trajectory γ. To simplify the notation, we omit t in $x(t)$ and write $c_\gamma(w, x(t))$ as $c_\gamma(w, x)$. An ergodic metric between $c_\gamma(w, x(t))$ and an information map ϕ can then be defined in a similar way as in Eq. 2.

Definition 4 (Footprint Ergodic Metric)**.** *Let $\mathcal{E}_\gamma$ denote the* Footprint Ergodic Metric*:*

$$\mathcal{E}_\gamma(\phi, x) = \sum_{\mathbf{k} \in \mathcal{K}} \Lambda_\mathbf{k} (c_{\gamma,\mathbf{k}} - \phi_\mathbf{k})^2 \quad (6)$$

$$= \sum_{\mathbf{k} \in \mathcal{K}} \Lambda_\mathbf{k} \left(\int_\mathcal{W} c_\gamma(w, x) F_\mathbf{k}(w) dw - \int_\mathcal{W} \phi(w) F_\mathbf{k}(w) dw \right)^2.$$

Here, $c_{\gamma,\mathbf{k}}$ are the Fourier coefficients of $c_\gamma(w, x)$ for $\mathbf{k} \in \mathcal{K}$.

The following theorem shows the relationship between the proposed footprint ergodic metric and the conventional ergodic metric [6]. Let $B(\gamma)$ denote a bounding sphere of $\gamma = \gamma(w, x)$ such that for any point $w' \notin B(\gamma)$, $\gamma(w', x) = 0$. In other words, γ can only take non-zero values within $B(\gamma)$. Let $r_B(\gamma)$ denote the radius of $B(\gamma)$. Intuitively, as $r_B(\gamma)$ approaches zero, since $\int_\mathcal{W} \gamma(w, x) dw = 1$, the footprint $\gamma(w, x)$ becomes a Dirac delta function at $w = q(x)$.

Theorem 1. *As $r_B(\gamma)$ approaches 0:*

$$\lim_{r_B(\gamma) \to 0} c_\gamma(w, x(t)) = c(w, x(t)), \quad (7)$$

$$\lim_{r_B(\gamma) \to 0} \mathcal{E}_\gamma(\phi, x) = \mathcal{E}(\phi, x). \quad (8)$$

Problem 1 (ETO-DF Problem)**.** *Given an information map ϕ, the goal of the Ergodic Trajectory Optimization with Dynamic sensor Footprint (ETO-DF) problem is to find a dynamically feasible trajectory x of the robot that minimizes the footprint ergodic metric $\mathcal{E}_\gamma(\phi, x)$ and the accumulated control effort:*

$$\min_{x,u} \mathcal{E}_\gamma(\phi, x) + \int_0^T u^T(t) R u(t) dt \quad (9)$$

$$\text{s.t. } (3b), (3c). \quad (10)$$

The only difference between ETO-DF and ETO is to replace the ergodic metric $\mathcal{E}$ in the objective function with the footprint ergodic metric $\mathcal{E}_\gamma$. All other constraints remain unchanged. The necessary conditions for local optimality for the ETO-DF problem can be derived in a similar way as in [6], [8], [21], [35], which are omitted here.

V. Numerical Approach

A. Approximation of Footprint Ergodic Metric

To solve ETO-DF numerically, we use direct transcription to convert the continuous-time dynamics and constraints into discrete nonlinear optimization, and then employ an optimization method, such as L-BFGS, to minimize the Lagrange function. The Lagrange function of ETO-DF can be written as follows, with Lagrange multipliers $\lambda_x(t) \in \mathbb{R}^n$:

$$\mathcal{L}(x, u, \lambda_x) = \mathcal{E}_\gamma(\phi, x) + \int_0^T \Big[u^\top(t) R u(t) + \lambda_x^\top(t) \big(\dot{x}(t) - f_d(x(t), u(t)) \big) \Big] dt, \quad (11)$$

where $\mathcal{E}_\gamma(\phi, x)$ is the footprint ergodic metric of Def. 4.

This approach also allows us to handle additional constraints as described in the next section on multi-robot. To optimize the Lagrangian, the gradient of Eq. 11 with respect to the state x and control u is required:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{E}_\gamma(\phi, x)}{\partial x} + \frac{\partial \int_0^T u^T R u + \lambda^T(\dot{x} - f(x,u))dt}{\partial x}. \quad (12)$$

The first term $\frac{\partial \mathcal{E}_\gamma(\phi,x)}{\partial x}$ requires an integration over the sensor footprint to compute $\mathcal{E}_\gamma$ and then taking the derivative of the integrated value with respect to x, which is complicated especially when the sensor footprint $\gamma(w, x)$ corresponds to a complex probability distribution. We therefore propose an alternative way to numerically compute the footprint ergodic metric $\mathcal{E}_\gamma$ approximately.

Let $W_s = \{w_1(t), w_2(t), \cdots, w_M(t)\}$ denote a set of M trajectories in the workspace $\mathcal{W}$ that are realizations of the random process $\gamma(w, x(t))$. Concretely, for any time point $t_k \in [0, T]$, for each realization $w_m \in W_s$, $w_m(t_k)$ is a sampled point from the corresponding random variable $\gamma(w, x(t_k))$ (Fig. 3(c)). Here, each sampled trajectory is continuous with respect to time. Note that, an area with a higher probability value $\gamma(w, x(t_k))$ tends to have more samples $w_m(t_k)$. Then, a footprint trajectory (FT) in Def. 3 can be approximated by the following Approximated FT (A-FT):

$$\gamma'(w, x(t)) = \frac{\sum_{m=1}^M \delta(w - w_m(x(t)))}{M}. \quad (13)$$

The time-average statistics of FT in Def. 3 can be approximated with the help of A-FT:

$$\begin{aligned} c'_\gamma(w, x(t)) &= \frac{1}{T}\int_0^T \gamma'(w, x(\tau))d\tau \\ &= \frac{1}{TM}\int_0^T \sum_{m=1}^M \delta(w - w_m(x(\tau)))d\tau \end{aligned} \quad (14)$$

whose corresponding Fourier coefficients are:

$$c'_{\gamma,\mathbf{k}} = \frac{1}{TM}\int_0^T \sum_{m=1}^M F_\mathbf{k}(w_m(x(\tau)))d\tau, \mathbf{k} \in \mathcal{K}. \quad (15)$$

The footprint ergodic metric (Def. 4) can be approximated by:

$$\mathcal{E}'_\gamma(\phi, x) = \sum_{\mathbf{k}\in\mathcal{K}} \Lambda_\mathbf{k}\left(c'_{\gamma,\mathbf{k}} - \phi_\mathbf{k}\right)^2. \quad (16)$$

Then, the term $\frac{\partial \mathcal{E}_\gamma(\phi,x)}{\partial x}$ required in Eq. 12 can be approximated by $\frac{\partial \mathcal{E}'_\gamma(\phi,x)}{\partial x}$, whose gradient can be derived as follows.

$$\begin{aligned} \frac{\partial \mathcal{E}'_\gamma(\phi, x)}{\partial x} &= 2\sum_{\mathbf{k}\in\mathcal{K}} \Lambda_\mathbf{k}(c'_{\gamma,\mathbf{k}} - \phi_\mathbf{k})\frac{\partial c'_{\gamma,\mathbf{k}}}{\partial x} \\ &= \frac{2}{TM}\sum_{\mathbf{k}\in\mathcal{K}} \Lambda_\mathbf{k}(c'_{\gamma,\mathbf{k}} - \phi_\mathbf{k})\int_0^T \sum_{m=1}^M \frac{\partial F_\mathbf{k}(w_m(x(\tau)))}{\partial x}d\tau \\ &= \frac{2}{TM}\sum_{\mathbf{k}\in\mathcal{K}} \Lambda_\mathbf{k}(c'_{\gamma,\mathbf{k}} - \phi_\mathbf{k}) \\ &\quad \int_0^T \sum_{m=1}^M \nabla_w F_\mathbf{k}(w_m(x(\tau)))^\top \frac{\partial w_m(x(\tau))}{\partial x}d\tau \end{aligned} \quad (17)$$

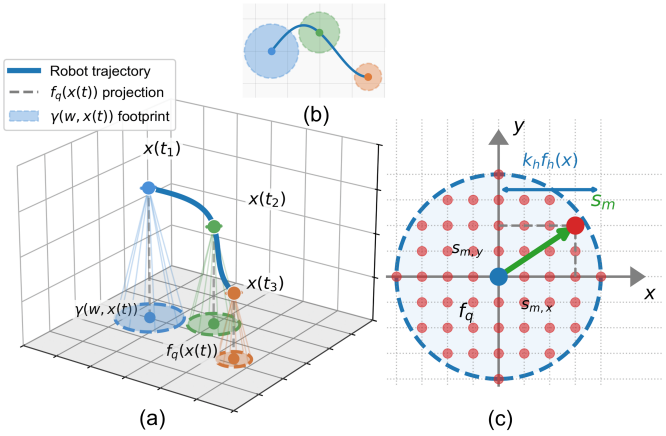


Fig. 3. Sampled trajectory $w_m(x(t))$ within the sensor footprint. (a) Robot trajectory $x(t)$ at three time instances with corresponding footprints $\gamma(w, x(t))$ projected onto the workspace. (b) Top-down view of the footprint trajectory. (c) Sampling geometry: directional vector s_m from projected position $f_q(x)$ to sampled point w_m, scaled by $k_h f_h(x)$.

where $\partial w_m(x(\tau))/\partial x$ is the jacobian of the sampled trajectory $w_m(x)$ (from the sensor footprint) with respect to the robot state x, and $\nabla_w F_\mathbf{k}$ is the gradient of the cosine basis function. Specifically, we use subscript j to denote the j-th component of $\nabla_w F_\mathbf{k}(w)$:

$$\begin{aligned} \left[\nabla_w F_\mathbf{k}(w)\right]_j &= \frac{\partial F_\mathbf{k}(w)}{\partial w_j} \\ &= -\frac{1}{h_\mathbf{k}}\frac{k_j\pi}{L_j}\sin\left(\frac{k_j\pi}{L_j}w_j\right)\prod_{o\neq j}\cos\left(\frac{k_o\pi}{L_o}w_o\right),\ j = 1, \ldots, \nu. \end{aligned}$$

B. Properties of the Gradients

While the gradient $\frac{\partial \mathcal{E}'_\gamma(\phi,x)}{\partial x}$ can be evaluated using Eq. 17 in each iteration of the trajectory optimization, we find a more efficient way to compute Eq. 17 by exploiting the *linearity* in the last term $\frac{\partial w_m(x(\tau))}{\partial x}$ in Eq. 17, which reduces the runtime complexity. We re-explained this linearity at the end of this section in Remark 1.

Each sampled trajectory $w_m(t)$ in the sensor footprint can be represented by:

$$w_m(x(t)) = f_q(x(t)) + k_h f_h(x(t))\, s_m, m = 1, 2, \cdots, M \quad (18)$$

where $f_q(x(t))$ is the projected position of the robot in the workspace at time t as described in Sec. III $f_h(x(t))$ is the altitude of the robot at time t as in described in Sec. IV-A, and $s_m = (s_{m,x}, s_{m,y}) \in \mathbb{R}^2$ is the directional (column) vector that points from the robot position to the position of the sampled point when the robot's altitude is one unit (Fig. 3). Note that, when a deterministic sampling is used, as we do in this work, the vector s_m is a *constant vector* over time, and the distance between the robot position $f_q(x(t))$ and the sampled trajectory point (at time t) $w_m(x(t))$ is equal to vector s_m scaled by the altitude of the robot $k_h f_h(x(t))$ (Fig. 3).

As a result, the last term $\frac{\partial w_m(x(\tau))}{\partial x}$ in Eq. 17 has the following form

$$\frac{\partial w_m(x)}{\partial x} = \frac{\partial f_q(x)}{\partial x} + k_h\, s_m \left(\frac{\partial f_h(x)}{\partial x}\right). \quad (19)$$

Note that the map $f_q(x)$ extracts the 2D position vector of the robot from the robot state x. In practice, the position vector of a mobile robot are usually the first two components of the robot state x. As a result, the first term $\frac{\partial f_q(x)}{\partial x} \in \mathbb{R}^{2\times n}$ in Eq. 19 has the following form

$$\frac{\partial f_q(x)}{\partial x} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \end{pmatrix}, \tag{20}$$

which is a constant matrix.

Similarly, the map $f_h(x)$ extracts the altitude, which is for example the third component of the robot state x, and the term $\frac{\partial f_h(x)}{\partial x} \in \mathbb{R}^{1\times n}$ in Eq. 19 has the following form

$$\frac{\partial f_h(x)}{\partial x} = \begin{pmatrix} 0 & 0 & 1 & 0 & \cdots & 0 \end{pmatrix}, \tag{21}$$

which is a constant row vector. As a result, the second term in Eq. 19 is:

$$k_h\, s_m \left(\frac{\partial f_h(x)}{\partial x}\right) = \begin{pmatrix} 0 & 0 & k_h s_{m,x} & 0 & \cdots & 0 \\ 0 & 0 & k_h s_{m,y} & 0 & \cdots & 0 \end{pmatrix}. \tag{22}$$

Eq. 19 can be written as:

$$\frac{\partial w_m(x)}{\partial x} = \begin{pmatrix} 1 & 0 & k_h s_{m,x} & 0 & \cdots & 0 \\ 0 & 1 & k_h s_{m,y} & 0 & \cdots & 0 \end{pmatrix}, \tag{23}$$

which depends purely on s_m.

The set of all $s_m, m = 1, 2\cdots, M$ can be pre-computed, and as a result, the gradient term $\frac{\partial w_m(x)}{\partial x}$ can be pre-computed and reused in all iterations during the trajectory optimization.

Example 2 (Uniform Sampling in the Euclidean Space)**.** *Let $s_m = (s_{m,x}, s_{m,y}) = (k_x\Delta, k_y\Delta)$, where Δ is a user-specified sampling step sizes along the x, y axes, with $k_x, k_y \in \mathbb{Z}, k_x, k_y \in [\frac{-1}{\Delta}, \frac{1}{\Delta}]$ being the indices of the samples. Substituting this expression into Eq. 23 yields*

$$\frac{\partial w_m(x)}{\partial x} = \begin{pmatrix} 1 & 0 & k_h\, k_x\Delta & 0 & \cdots & 0 \\ 0 & 1 & k_h\, k_y\Delta & 0 & \cdots & 0 \end{pmatrix}. \tag{24}$$

Example 3 (Uniform Sampling in the Polar Coordinates)**.** *Consider a polar grid inside the unit disk with d_1 radial levels and d_2 angular samples, and let $M = d_1 d_2$. For indices $i \in \{1, \ldots, d_1\}$ and $j \in \{0, \ldots, d_2 - 1\}$, define*

$$r_i = \frac{i}{d_1}, \qquad \theta_j = \frac{2\pi j}{d_2}, \tag{25}$$

and set the corresponding samples as

$$s_m = (s_{m,x}, s_{m,y})^\top = \big(r_i \cos\theta_j,\ r_i \sin\theta_j\big)^\top. \tag{26}$$

Substituting this expression into Eq. 23 yields

$$\frac{\partial w_m(x)}{\partial x} = \begin{pmatrix} 1 & 0 & k_h\, r_i \cos\theta_j & 0 & \cdots & 0 \\ 0 & 1 & k_h\, r_i \sin\theta_j & 0 & \cdots & 0 \end{pmatrix}. \tag{27}$$

Note that this sampling in polar coordinates corresponds to a non-uniform circular sensor footprint with more weights near the center and with less weights near the boundary, as opposed to the uniform circular sensor footprint as in Example 1.

Remark 1. *The derivative $\frac{\partial w_m(x(\tau))}{\partial x}$ is constant, which is caused by the fact that the sampled point $w_m(x(t))$ at any time point t is a linear function with respect to $x(t)$ (Eq. 19). Intuitively, if the robot flies higher (or lower), the sampled point $w_m(x(t))$ moves further away from (or closer to) the center of the sensor footprint proportionally, and if the robot moves along the* X,Y*-axes horizontally, the sampled point $w_m(x(t))$ moves along the* X,Y*-axes horizontally correspondingly.*

C. Augmented Lagrangian and iLQR

This section presents our FEO, which is based on Augmented Lagrangian and iLQR numerical optimization approach (Alg. 1) to solve the ETO-DF problem. FEO leverages that $\frac{\partial w_m}{\partial x}$ is a constant and can be pre-computed and re-used during the iLQR iterations to save computational burden.

1) Discrete Problem Formulation: Let $x_{0:N}, u_{0:N-1}$ denote the discretized state and control trajectories and let $x_{k+1} = f_d(x_k, u_k)$ denote the discrete dynamics of the system. The approximated footprint ergodic metric can be rewritten as:

$$\mathcal{E}'_\gamma(\phi, x_{0:N}) = \sum_{\mathbf{k}\in\mathcal{K}} \Lambda_{\mathbf{k}} \big(c'_{\gamma,\mathbf{k}}(x_{0:N}) - \phi_{\mathbf{k}}\big)^2. \tag{28}$$

The ETO-DF problem can be rewritten into discrete version as follows.

Problem 2 (ETO-DF via iLQR Problem)**.** *Based on the approximate footprint ergodic metric $\mathcal{E}'_\gamma$, we formulate the following discrete-time optimal control problem:*

$$\min_{x_{0:N},\, u_{0:N-1}} \ \mathcal{E}'_\gamma(\phi, x_{0:N}) + \sum_{k=0}^{N-1} u_k^\top R u_k \tag{29}$$

$$\text{s.t. } x_{k+1} = f_d(x_k, u_k), \quad k = 0, \ldots, N-1 \tag{30}$$

$$x_k \in \mathcal{X}, u_k \in \mathcal{U} \tag{31}$$

2) Augmented Lagrangian and Gradients: To solve this problem, we incorporate these constraints via an augmented Lagrangian $\mathcal{L}_{\mathrm{AL}}$ of the form

$$\mathcal{L}_{\mathrm{AL}}(x_{0:N}, u_{0:N-1}, \boldsymbol{\mu}, r) = \mathcal{E}'_\gamma(\phi, x_{0:N}) + \sum_{k=0}^{N-1} u_k^\top R u_k + \Psi(g(x_{0:N}, u_{0:N-1}); \boldsymbol{\mu}, r), \tag{32}$$

where $g(x_{0:N}, u_{0:N-1})$ represents the aggregate constraint violation, $\boldsymbol{\mu}$ denotes the vector of Lagrange multipliers (dual variables), $r > 0$ is the penalty parameter, and $\Psi(\cdot)$ is the augmented penalty term defined as:

$$\Psi(g; \boldsymbol{\mu}, r) = \frac{1}{2r} \sum_{i=1}^{m} \left[\max(0, \mu_i + r g_i)^2 - \mu_i^2\right]. \tag{33}$$

Unlike indirect methods (Eq. 11) where co-state variables enforce equality dynamics constraints in the Lagrangian, we use direct transcription: dynamics are satisfied explicitly through forward integration. The $\mathcal{L}_{\mathrm{AL}}$ handles only the inequality constraints via the penalty term $\Psi(\cdot; \boldsymbol{\mu}, r)$. The gradient of $\mathcal{L}_{\mathrm{AL}}$ with respect to each state and control sample (x_k, u_k) can be written as

$$\nabla_{x_k} \mathcal{L}_{\mathrm{AL}} = \nabla_{x_k} \mathcal{E}'_\gamma(\phi, x_{0:N}) + \nabla_{x_k} \Psi \tag{34}$$

$$\nabla_{u_k} \mathcal{L}_{\mathrm{AL}} = 2 R u_k + \nabla_{u_k} \Psi \tag{35}$$

where $\nabla_{x_k} \mathcal{E}'_\gamma$ is obtained from the sampling-based expression in Eq. 17.

Algorithm 1 FEO

Require: Initial guess x_0, $\mathbf{u}_{0:N-1}$, penalty r, multipliers $\boldsymbol{\mu}$
Ensure: Optimized $\mathbf{x}^*_{0:N}$, $\mathbf{u}^*_{0:N-1}$
// Precompute constant Jacobians (reused across all iterations)
1: Precompute $\{\partial w_m/\partial x\}_{m=1}^{M}$ via Eq. 23
2: **while** not converged **do**
3: **for** $i = 1$ **to** N_{inner} **do**
4: Forward rollout: compute $\mathbf{x}_{0:N}$ from $\mathbf{u}_{0:N-1}$
5: Linearize dynamics: A_k, B_k at each (x_k, u_k)
// Gradient computation reuses $\{\partial w_m/\partial x\}$ from Line 1
6: Compute $q_k \leftarrow \nabla_{x_k}\mathcal{L}_{\text{AL}}$, $r_k \leftarrow \nabla_{u_k}\mathcal{L}_{\text{AL}}$
7: Backward pass: compute K_k, d_k via Riccati recursion
8: Forward pass: $\delta u_k \leftarrow K_k\delta x_k + d_k$, $\forall k$
9: $\alpha^* \leftarrow$ LINESEARCH$(\mathbf{u}, \boldsymbol{\delta u}, \mathcal{L}_{\text{AL}})$
10: $u_k \leftarrow \text{clip}(u_k + \alpha^*\delta u_k, u_{\min}, u_{\max})$, $\forall k$
11: Update $\boldsymbol{\mu}$ or increase r based on constraint satisfaction
12: **return** $\mathbf{x}^*_{0:N}$, $\mathbf{u}^*_{0:N-1}$

3) FEO Summary: Our FEO (Alg. 1) is similar to the regular iLQR [7], [35], [36], which employs a double-loop structure. The inner loop (Lines 3–10) performs iLQR updates: given the current trajectory, we linearize dynamics and compute the augmented Lagrangian gradients (q_k, r_k). The backward Riccati recursion computes P_k (quadratic cost-to-go) and r_k (linear term), followed by a forward pass that yields the descent direction $\{\delta u_k\}$. The outer loop (Lines 11–12) updates dual multipliers $\boldsymbol{\mu}$ when sufficient decrease is achieved, or increases penalty r otherwise.

A computational advantage of Alg. 1 is that the terms $\{\frac{\partial w_m}{\partial x}\}$ are precomputed at the beginning (Line 1), and are reused when computing $\nabla_{x_k}\mathcal{E}'_\gamma$ in Eq. 34 (Line 6), which speeds up the algorithm. We can do so because $\{\frac{\partial w_m}{\partial x}\}$ depends only on the fixed sample set rather than the evolving trajectory as aforementioned.

4) Accelerated Log-Spaced Line Search: At each FEO iteration (Section V-C), the backward pass yields a descent direction $\Delta\mathbf{u}_{0:N-1} := \{\Delta\mathbf{u}_k\}_{k=0}^{N-1}$ for the current control sequence $\mathbf{u}_{0:N-1} := \{\mathbf{u}_k\}_{k=0}^{N-1}$. For a scalar step size $\alpha > 0$, we form the candidate update

$$\mathbf{u}_k(\alpha) = \Pi_{\mathcal{U}}(\mathbf{u}_k + \alpha\,\Delta\mathbf{u}_k) \tag{36}$$

where $\mathcal{U}$ is the admissible control set and $\Pi_{\mathcal{U}}$ means the projection onto the this control set. The corresponding state trajectory $\mathbf{x}_{0:N}(\alpha)$ is obtained via a forward rollout.

To select the best step size α, a line search LINESEARCH is needed in each iteration of FEO, which selects α by minimizing the augmented-Lagrangian merit $\mathcal{L}_{\text{AL}}$ defined in Eq. 32, evaluated on the rolled-out trajectories $\big(\mathbf{x}_{0:N}(\alpha), \mathbf{u}_{0:N-1}(\alpha)\big)$. In this LINESEARCH, multiple candidate values of α are evaluated, each requiring a rollout and recomputation of (i) the sampling-based footprint ergodic objective and (ii) the inequality penalties, making step-size selection expensive if many α are tried.

Classical Armijo backtracking line search accepts the first step size α satisfying the condition $\mathcal{L}(\alpha) \le \mathcal{L}(0) - c\alpha\nabla\mathcal{L}^\top\Delta\mathbf{u}$ during the iterative updates of α [7], [16], [35].[3] If no candidate yields improvement, a predefined smallest step size is used. Such a process can be inefficient when each merit evaluation requires costly computation.

[3]For example, in the i-th iteration, $\alpha_i = \alpha_0\beta^i$, we use $\beta = 0.5$ in our implementation.

Algorithm 2 LINESEARCH: Accelerated Log-Spaced Search

Require: current $\mathbf{u}_{0:N-1}$, direction $\Delta\mathbf{u}_{0:N-1}$, $(\boldsymbol{\mu}, r)$, $(\alpha_0, \alpha_{\min}, N_\alpha)$
1: Construct candidate set $\mathcal{A}$ using 37
2: **for all** $\alpha \in \mathcal{A}$ **do**
3: $\mathbf{u}_{0:N-1}(\alpha) \leftarrow \Pi_{\mathcal{U}}(\mathbf{u}_{0:N-1} + \alpha\,\Delta\mathbf{u}_{0:N-1})$
4: Roll out $\mathbf{x}_{0:N}(\alpha)$
5: $J(\alpha) \leftarrow \mathcal{L}_{\text{AL}}(\mathbf{x}_{0:N}(\alpha), \mathbf{u}_{0:N-1}(\alpha), \boldsymbol{\mu}, r)$ using 32
6: $\alpha^\star \leftarrow \arg\min_{\alpha\in\mathcal{A}} J(\alpha)$
7: **if** $J(\alpha^\star) \le J(0)$ **then**
8: **return** $\mathbf{u}_{0:N-1}(\alpha^\star)$
9: **else**
10: **return** $\mathbf{u}_{0:N-1}$

In Alg. 1, instead of iterative backtracking, we speed up this LINESEARCH by evaluating a fixed set $\mathcal{A} = \{\alpha_i\}_{i=0}^{N_\alpha-1}$ of N_α log-spaced step sizes between α_0 and $\alpha_{\min}$, where

$$\alpha_i = \alpha_0\left(\frac{\alpha_{\min}}{\alpha_0}\right)^{\frac{i}{N_\alpha-1}}. \tag{37}$$

The best α^* in this set is then selected

$$\alpha^\star = \arg\min_{\alpha\in\mathcal{A}}\ \mathcal{L}_{\text{AL}}(\mathbf{x}_{0:N}(\alpha), \mathbf{u}_{0:N-1}(\alpha), \boldsymbol{\mu}, r)\,. \tag{38}$$

We accept this α^* if it decreases the current cost $\mathcal{L}_{\text{AL}}$. Otherwise, we keep the current controls unchanged. This strategy offers two advantages: (i) the fixed evaluation count enables parallelization, and (ii) selecting the best rather than the first acceptable step often yields larger cost reductions per iteration. Experimental results show that this log-spaced line search speeds up the computation obviously as detailed later.

VI. EXTENSIONS

While the previous sections assume the robot's FOV is represented by a view cone whose optical axis is always perpendicular to the flat 2D ground, this section allows the robot to fly in SE(3), and the sensor footprint can be projected onto a general surface that is non-flat. Correspondingly, the view cone may not be perpendicular to the ground and can be oriented in any direction.

For presentation purposes, we begin with the case that the information map remains a flat 2D plane (Fig. 5) while the robot flies in SE(3). At the end of the section, we present how to generalize the proposed approach to surfaces in 3D by approximating the surface with local tangent planes computed by local Taylor expansions. When the information map is a 2D flat ground, the local tangent plane at any point is always the same as the information map.

Assumption 1. *The derivation in this section assumes that the surface to be covered is smooth, has bounded curvature, and is topologically equivalent to a 2D plane.*

The purpose of having this assumption is that the surface can then be represented by a single flat information map with little distortion, and that the sensor does not view the surface

at near-grazing incidence. The error introduced by the tangent-plane approximation, as we will introduce soon, is bounded as analyzed in Sec. VI-G.

We derive both the resulting footprint on the local tangent plane at the point where the optical axis intersects the surface and the Jacobians required for gradient-based trajectory optimization. In contrast to the 2D case in the previous section, here, only part of the gradient $\frac{\partial w_m(x)}{\partial x}$ remains constant in 3D, while the remaining terms become pose-dependent. Computational reuse is still possible for those constant parts within our FEO framework.

For complex surfaces that violate this assumption, such as closed surfaces that are topologically different from a plane, we briefly discuss how to adapt our approach and showcase its use in Sec. VII-F in the experimental results, as it is not the focus of our approach.

A. Overview

We begin with an overview of our approach for general surfaces (Fig. 4). Let $\mathcal{S}$ denote a 2D manifold embedded in 3D, and let $\mathcal{X}$ denote the full pose state of the robot including both position and orientation. Let a denote a ray corresponding to the optical axis of the FOV cone of the robot sensor, and let p_0 denote the first intersection point of this optical axis with the surface $\mathcal{S}$. Let $\mathcal{T}$ denote the tangent plane of $\mathcal{S}$ at point p_0. Within the FOV cone, a set of rays are sampled (indexed by $m = 1, 2, \cdots, M$) and the m-th ray has an intersection point $\mathbf{q}_m \in \mathcal{T}$, which we referred to as a footprint point. This work always uses local tangent planes to approximate the surface during the motion of the robot, correspondingly, the footprint point on the tangent plane is also regarded as approximately located on the surface $\mathbf{q}_m \in \mathcal{S}$.

Besides, let $\Pi_{\mathcal{W}} : \mathcal{S} \to \mathcal{W}$ denote a map from any point on the surface to the underlying information map ϕ embedded in the workspace $\mathcal{W}$. In other words, any footprint point $\mathbf{q}_m \in \mathcal{S}$ on the surface can be mapped to a workspace point on the information map $w_m = \Pi_{\mathcal{W}}(\mathbf{q}_m)$. The associated Jacobian is $\mathbf{J}_{\mathcal{W}}(\mathbf{q}_m) = \frac{\partial \Pi_{\mathcal{W}}(\mathbf{q}_m)}{\partial \mathbf{q}_m}$ By the chain rule, the Jacobian of a footprint sample with respect to the robot state x is

$$\frac{\partial w_m(x)}{\partial x} = \mathbf{J}_{\mathcal{W}}(\mathbf{q}_m) \frac{\partial \mathbf{q}_m}{\partial \boldsymbol{\mathcal{X}}} \frac{\partial \boldsymbol{\mathcal{X}}}{\partial x}. \tag{39}$$

The rest of this section details these concepts, derives the analytical form of the terms in Eq. 39, and show how the derived results are leveraged in our FEO framework.

B. 2D Manifold, View Cone and Tangent Planes

1) 2D Manifold and Tangent Planes: We represent the 2D surfaces to be covered implicitly as a manifold $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^3 \mid f(\mathbf{x}) = 0\}$, where $f : \mathbb{R}^3 \to \mathbb{R}$ is a smooth function with $\nabla f \neq \mathbf{0}$ on $\mathcal{S}$. Let $\mathbf{p}_0$ denote the intersection point of the optical axis and the surface:

Definition 5 (Intersection Point)**.** *The intersection point is the first intersection of the optical axis with the surface:*

$$\mathbf{p}_0 = \mathbf{p} + l_0 \mathbf{a}, \qquad l_0 = \min_{l>0} \{l \mid f(\mathbf{p} + l\mathbf{a}) = 0\}. \tag{40}$$

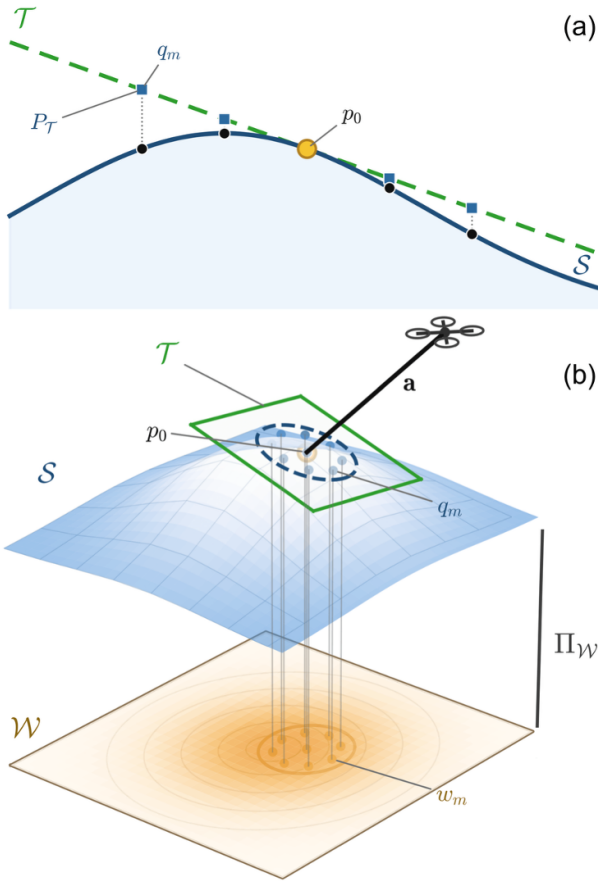


Fig. 4. Illustration of concepts and maps. The surface $\mathcal{S}$ (a 2D manifold) is locally approximated by the tangent plane $\mathcal{T}$ at the intersection point of the sensor optical axis $\mathbf{a}$ and the surface $\mathcal{S}$. Within the FOV cone, a set of sampled footprint points $q_m \in \mathcal{T}$ are computed. The workspace mapping $\Pi_{\mathcal{W}}$ maps any footprint point q_m to a point $w_m \in \mathcal{W}$ on the information map ϕ, where the information density $\phi(w_m)$ can be evaluated. Note that although the terrain to be covered by the robot is a 2D manifold which is non-flat, the underlying information map is still 2D and can be represented by a 2D flat information map due to the map $\Pi_{\mathcal{W}}$.

Then, the local tangent plane at point $\mathbf{p}_0$ is:

Definition 6 (Tangent Plane)**.** *Let the outward unit normal at* $\mathbf{p}_0$ *be*

$$\mathbf{n} = \frac{\nabla f(\mathbf{p}_0)}{\|\nabla f(\mathbf{p}_0)\|}. \tag{41}$$

The tangent plane at $\mathbf{p}_0$ *is*

$$\mathcal{T} = \left\{\mathbf{x} \in \mathbb{R}^3 \mid \mathbf{n}^\top (\mathbf{x} - \mathbf{p}_0) = 0\right\}. \tag{42}$$

2) View Cone: When the robot position is at point $\mathbf{p}$ in the world frame, the Robot's FOV is described by a view cone.

Definition 7 (View Cone)**.** *Let* $\mathbf{a} \in \mathbb{S}^2$ *denote the unit vector along the optical axis, and* $\theta_c \in (0, \pi/2)$ *denote the cone's half-angle. The rays within the cone that point outwards are*

$$\mathcal{D}(\theta_c) = \left\{\mathbf{d} \in \mathbb{S}^2 \mid \mathbf{d}^\top \mathbf{a} \geq \cos\theta_c\right\}. \tag{43}$$

The view cone emanating from the robot position $\mathbf{p} \in \mathbb{R}^3$ *is then defined as*

$$\mathcal{C} = \left\{\mathbf{r} \in \mathbb{R}^3 \mid \exists l > 0,\ \exists \mathbf{d} \in \mathcal{D}(\theta_c) : \mathbf{r} = \mathbf{p} + l\mathbf{d}\right\}. \tag{44}$$

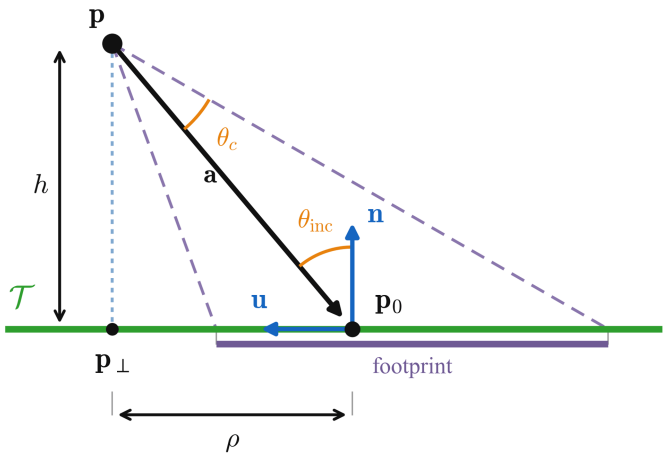


Fig. 5. Geometric setup for the 3D footprint model. The view cone with half-angle θ_c originates from the robot position $\mathbf{p}$ with optical axis $\mathbf{a}$, intersecting the terrain at the contact point $\mathbf{p}_0$. The local tangent frame $\mathcal{L}_T = \{\mathbf{u}, \mathbf{v}, \mathbf{n}\}$ is an orthonormal basis with origin at $\mathbf{p}_0$, where $\mathbf{n}$ is the surface normal, $\mathbf{u}$ points toward the projected robot position $\mathbf{p}_\perp$, and $\mathbf{v} = \mathbf{n} \times \mathbf{u}$ points out of the plane of the figure. The incidence angle θ_{inc} is the angle between the optical axis and the surface normal.

3) Local Tangent Frame: The intersection point $\mathbf{p}_0$ is the intersection of the optical axis of the sensor with the ground (information map). The point $\mathbf{p}_0$ is the "center" of the sensor footprint on the local tangent plane (Fig. 5). In the *perpendicular case* considered in Sec. IV-A, $\mathbf{p}_0$ is same as the projected position $f_q(x)$.

Let $\mathbf{n}$ denote the normal vector of the local tangent plane.

Definition 8 (Local Tangent Frame)**.** *To construct a local coordinate frame on the tangent plane $\mathcal{T}$, we first project the robot position onto the tangent plane:*

$$\mathbf{p}_\perp = \mathbf{p} - \big(\mathbf{n}^\top(\mathbf{p} - \mathbf{p}_0)\big)\mathbf{n}. \tag{45}$$

We then define the local tangent frame $\mathcal{L}_T = \{\mathbf{u}, \mathbf{v}, \mathbf{n}\}$ as an orthonormal basis with origin at $\mathbf{p}_0$:

$$\mathbf{u} = \frac{\mathbf{p}_\perp - \mathbf{p}_0}{\|\mathbf{p}_\perp - \mathbf{p}_0\|}, \qquad \mathbf{v} = \mathbf{n} \times \mathbf{u}, \qquad \mathbf{n} = \frac{\nabla f(\mathbf{p}_0)}{\|\nabla f(\mathbf{p}_0)\|}. \tag{46}$$

Note that when the optical axis is perpendicular to the tangent plane, the projection satisfies $\mathbf{p}_\perp = \mathbf{p}_0$, resulting in an undefined frame $\mathcal{L}_T$. This special case corresponds to the circular footprint model that is already discussed in the previous Sec. IV-A. The subsequent derivations assume the non-perpendicular case ($\|\mathbf{p}_\perp - \mathbf{p}_0\| > 0$).

These basis vectors $\{\mathbf{u}, \mathbf{v}, \mathbf{n}\}$ form a rotation matrix $\mathbf{R}_\mathcal{T} = [\mathbf{u}, \mathbf{v}, \mathbf{n}] \in \mathrm{SO}(3)$ that describes the orientation of this local tangent frame with respect to the world frame. Together with the origin $\mathbf{p}_0$, the pair $(\mathbf{R}_\mathcal{T}, \mathbf{p}_0) \in \mathrm{SE}(3)$ defines the transformation of the local tangent frame with respect to the world frame. Besides, a 3D point $\mathbf{q} \in \mathbb{R}^3$ on the tangent plane represented in the world frame can be transformed into its coordinates (u, v) on the tangent plane with respect to the local tangent frame via a projection matrix

$$\mathbf{P}_\mathcal{T} = \begin{bmatrix} \mathbf{u}^\top \\ \mathbf{v}^\top \end{bmatrix} \in \mathbb{R}^{2\times 3}, \qquad \begin{pmatrix} u \\ v \end{pmatrix} = \mathbf{P}_\mathcal{T}(\mathbf{q} - \mathbf{p}_0). \tag{47}$$

In other words, the matrix $\mathbf{P}_\mathcal{T}$ extracts the local coordinates of a 3D point on the tangent plane.

Definition 9 (Geometric Parameters)**.** *We define the following geometric quantities to help the derivation later: the height h, the horizontal offset ρ, and the incidence angle θ_{inc}:*

$$h = \mathbf{n}^\top(\mathbf{p} - \mathbf{p}_0), \ \rho = \|\mathbf{p}_\perp - \mathbf{p}_0\|, \ \theta_{\text{inc}} = \arctan\Big(\frac{\rho}{h}\Big). \tag{48}$$

These quantities satisfy $\mathbf{p} - \mathbf{p}_0 = \rho\mathbf{u} + h\mathbf{n}$ and $\rho = h\tan\theta_{\text{inc}}$.

If $\mathbf{p}_0 = \mathbf{p} + l_0\mathbf{a}$ with $l_0 > 0$ and $\mathbf{n}$ points outward from the surface, then the incidence angle satisfies

$$\cos\theta_{\text{inc}} = -\mathbf{n}^\top\mathbf{a}, \qquad \sin\theta_{\text{inc}} = \sqrt{1 - (\mathbf{n}^\top\mathbf{a})^2}. \tag{49}$$

Intuitively, h is the height of the robot above the tangent plane, ρ is the horizontal offset from $\mathbf{p}_0$ to the robot's projection $\mathbf{p}_\perp$, and θ_{inc} measures the incidence angle between the optical axis and the surface normal (Fig. 5). Larger values of θ_{inc} correspond to more oblique viewing angles, and when $\theta_{\text{inc}} = 0$, the optical axis is perpendicular to the tangent plane.

C. Sensor Footprint Geometry for Oblique Views

We now investigate the geometry of the sensor footprint on the tangent plane.

Definition 10 (Local Sensor Frame)**.** *As aforementioned, assuming $\sin\theta_{\text{inc}} = \|\mathbf{a} \times \mathbf{n}\| > 0$ (i.e., the non-perpendicular case), we define a local frame attached to the sensor by orthonormal vectors $\mathcal{L}_S = \{\mathbf{a}, \mathbf{e}_1, \mathbf{e}_2\}$, where*

$$\mathbf{e}_2 = -\frac{\mathbf{a} \times \mathbf{n}}{\|\mathbf{a} \times \mathbf{n}\|}, \qquad \mathbf{e}_1 = \mathbf{e}_2 \times \mathbf{a}. \tag{50}$$

When expanded in the local tangent frame basis $\{\mathbf{u}, \mathbf{v}, \mathbf{n}\}$, these vectors take the form

$$\begin{aligned} \mathbf{a} &= -\sin\theta_{\text{inc}}\,\mathbf{u} - \cos\theta_{\text{inc}}\,\mathbf{n}, \\ \mathbf{e}_1 &= \cos\theta_{\text{inc}}\,\mathbf{u} - \sin\theta_{\text{inc}}\,\mathbf{n}, \\ \mathbf{e}_2 &= -\mathbf{v}. \end{aligned} \tag{51}$$

We refer to this frame $\mathcal{L}_S$ as the local sensor frame.

Within the local sensor frame $\mathcal{L}_S$, any ray within the cone can be described by two angles (β, φ) with respect to the optical axis: For any angle $\beta \in [0, \theta_c]$ from the optical axis and azimuth $\varphi \in [0, 2\pi)$, the corresponding ray direction within the cone can be represented in the world frame as:

$$\mathbf{d}(\beta, \varphi) = \cos\beta\,\mathbf{a} + \sin\beta\cos\varphi\,\mathbf{e}_1 + \sin\beta\sin\varphi\,\mathbf{e}_2. \tag{52}$$

Define an auxiliary variable $D(\beta, \varphi)$ as

$$D(\beta, \varphi) = \cos\theta_{\text{inc}}\cos\beta + \sin\theta_{\text{inc}}\sin\beta\cos\varphi. \tag{53}$$

Note that $D(\beta, \varphi) = -\mathbf{n}^\top\mathbf{d}(\beta, \varphi)$, which means:

- $D < 0$: the ray points away from the tangent plane;
- $D = 0$: the ray is parallel to the tangent plane;
- $D > 0$: the ray points towards the tangent plane.

We only need to consider $D > 0$ in this paper, which is guaranteed for all rays within the view cone when $\theta_{\text{inc}} + \theta_c < \pi/2$.

Definition 11 (Sensor Footprint Geometry)**.** *For any ray (β, φ) within the view cone, let*

$$\mathbf{q}(\beta,\varphi) = \mathbf{p} + l^*(\beta,\varphi)\mathbf{d}(\beta,\varphi) \tag{54}$$

denote the corresponding footprint point, i.e., the intersection of the ray with the tangent plane $\mathcal{T}$ expressed in the world frame, where

$$l^*(\beta,\varphi) = \frac{h}{D(\beta,\varphi)}. \tag{55}$$

The footprint boundary can be obtained by setting $\beta = \theta_c$ and varying $\varphi \in [0, 2\pi)$.

As a special case, when $\theta_{\text{inc}} = 0$ (i.e., the optical axis is perpendicular to the tangent plane $\mathcal{T}$), we have $D(\beta,\varphi) = \cos\beta$, and the footprint point $\mathbf{q}(\beta,\varphi)$ reduces to a circular footprint centered at $\mathbf{p}_0$ with radius $h\tan\theta_c$ as aforementioned in Sec. IV-A.

The footprint geometry in Def. 11 provides a way to describe the sensor footprint distribution $\gamma(w,x)$ (Sec. IV-A): Given a point $w \in \mathcal{W}$ on the tangent plane, the value $\gamma(w,x)$ can be computed by checking whether w lies within the boundary of the footprint on the tangent plane.

D. Pose Jacobians

To incorporate the derivation into gradient-based trajectory optimization, we need to compute the gradient of the Lagrangian with respect to the robot state, as shown in Eq. (12). In particular, the term $\frac{\partial \mathcal{E}'_\gamma(\phi,x)}{\partial x}$ in Eq. (17) requires the Jacobian $\frac{\partial w_m(x)}{\partial x}$ for each sampled point within the footprint. We derive the Jacobians of these sampled points with respect to the robot pose, and examine whether these Jacobians can be reused across optimization iterations. We derive the Jacobians by fixing the local tangent frame at each time instant; that is, we treat $\mathbf{p}_0$ and $\mathbf{n}$ as constants in the derivation.

Definition 12 (Robot Pose State)**.** *Let $\boldsymbol{\mathcal{X}} = [\mathbf{p}^\top, \boldsymbol{\Theta}^\top]^\top \in \mathbb{R}^6$ denote the full pose state, where $\mathbf{p} \in \mathbb{R}^3$ is the position and $\boldsymbol{\Theta} = [\phi_r, \theta_p, \psi_y]^\top$ are the roll, pitch, and yaw Euler angles, respectively.*

Note that the pose state $\boldsymbol{\mathcal{X}}$ is a subvector of the full robot state x, and its Jacobian with respect to x, $\frac{\partial \boldsymbol{\mathcal{X}}}{\partial x}$ is therefore a constant matrix that selects the corresponding entries from x.

To simplify the derivation, we set the optical axis $\mathbf{a}$ to align with the x-axis of the local sensor frame $\mathcal{L}_S$.[4] Note that, the direction of the optical axis in the world frame is determined by pitch and yaw only. In other words, the roll angle does not affect the orientation and shape of the view cone.

For the rest of this section, we list the key results to show the idea, and provide the detailed proofs in Appendix.

Lemma 1 (Position Jacobian)**.** *For any ray (β,φ) with $D(\beta,\varphi) > 0$, let $\mathbf{q}(\beta,\varphi) = \mathbf{p} + l^*\mathbf{d}$ denote the corresponding* footprint point*, i.e., the point where the ray intersects with the tangent plane, represented in the world frame. The Jacobian of $\mathbf{q}$ with respect to the robot position $\mathbf{p}$ is*

$$\mathbf{J}_p(\beta,\varphi) = \frac{\partial \mathbf{q}}{\partial \mathbf{p}} = \mathbf{I}_3 + \frac{1}{D(\beta,\varphi)}\,\mathbf{d}(\beta,\varphi)\,\mathbf{n}^\top \in \mathbb{R}^{3\times 3}. \tag{56}$$

Remark 2. *This lemma indicates that, for a fixed tangent plane (i.e., both (β,φ) and $\mathbf{n}$ are constant), the position Jacobian $\mathbf{J}_p$ is then a constant matrix which suggests the linearity between the sensor footprint and the robot position: when the robot translates, the footprint size grows larger or smaller proportionally with respect to the translation. Note that, this fixed tangent can have any orientation and does not have to be horizontal. When the tangent plane varies, (i.e., (β,φ) and $\mathbf{n}$ are varying), the position Jacobian $\mathbf{J}_p$ is no more a constant matrix: the first term I_3 remains constant while the second term $\frac{1}{D(\beta,\varphi)}\,\mathbf{d}(\beta,\varphi)\,\mathbf{n}^\top \in \mathbb{R}^{3\times 3}$ depends on the orientation of the tangent plane. In practice, to compute $\mathbf{J}_p$, we only need to evaluate the second term there.*

For the orientation component, we first derive the Jacobian of $\mathbf{q}$ with respect to the optical axis $\mathbf{a}$, then relate $\mathbf{a}$ to the Euler angles $\boldsymbol{\Theta}$ via the chain rule. Derivation is in the Appendix.

Lemma 2 (Orientation Jacobian)**.** *For any ray (β,φ) with $D(\beta,\varphi) > 0$, the Jacobian of the footprint point $\mathbf{q}$ with respect to the optical axis of the robot is*

$$\mathbf{J}_a(\beta,\varphi) = \frac{\partial \mathbf{q}}{\partial \mathbf{a}} = \frac{h}{D}\frac{\partial \mathbf{d}}{\partial \mathbf{a}} - \frac{h}{D^2}\,\mathbf{d}\left(\frac{\partial D}{\partial \mathbf{a}}\right)^\top \in \mathbb{R}^{3\times 3}, \tag{57}$$

where

$$\frac{\partial \mathbf{d}}{\partial \mathbf{a}} = \cos\beta\,\mathbf{I}_3 + \sin\beta\cos\varphi\,\frac{\partial \mathbf{e}_1}{\partial \mathbf{a}} + \sin\beta\sin\varphi\,\frac{\partial \mathbf{e}_2}{\partial \mathbf{a}}, \tag{58}$$

$$\frac{\partial D}{\partial \mathbf{a}} = \frac{-\sin\theta_{\text{inc}}\cos\beta + \cos\theta_{\text{inc}}\sin\beta\cos\varphi}{\sin\theta_{\text{inc}}}\mathbf{n}. \tag{59}$$

To relate the optical axis $\mathbf{a}$ to the robot's orientation, we introduce the Jacobian $\mathbf{J}_{a/\Theta} = \frac{\partial \mathbf{a}}{\partial \boldsymbol{\Theta}}$, which we refer to as the axis-to-Euler Jacobian.

Lemma 3 (Axis-to-Euler Jacobian)**.** *The optical axis in the world frame is*

$$\mathbf{a} = \begin{bmatrix} \cos\theta_p\cos\psi_y \\ \cos\theta_p\sin\psi_y \\ -\sin\theta_p \end{bmatrix} \tag{60}$$

and $\mathbf{J}_{a/\Theta}$ is

$$\mathbf{J}_{a/\Theta} = \frac{\partial \mathbf{a}}{\partial \boldsymbol{\Theta}} = \begin{bmatrix} 0 & -\sin\theta_p\cos\psi_y & -\cos\theta_p\sin\psi_y \\ 0 & -\sin\theta_p\sin\psi_y & \cos\theta_p\cos\psi_y \\ 0 & -\cos\theta_p & 0 \end{bmatrix}. \tag{61}$$

Combining the position Jacobian (Lemma 1) and the orientation Jacobian (Lemmas 2–3), we obtain the Jacobian with respect to the full pose.

Lemma 4 (Full Pose Jacobian)**.** *For a ray (β,φ), the Jacobian of a footprint point $\mathbf{q}$ with respect to the full pose state $\boldsymbol{\mathcal{X}}$ is*

$$\frac{\partial \mathbf{q}}{\partial \boldsymbol{\mathcal{X}}} = \begin{bmatrix} \mathbf{J}_p(\beta,\varphi) & \mathbf{J}_a(\beta,\varphi)\,\mathbf{J}_{a/\Theta} \end{bmatrix} \in \mathbb{R}^{3\times 6} \tag{62}$$

.

[4] If not, one can re-define the local sensor frame so that they align.

The above Jacobians describe how a footprint point moves with respect to the robot pose.

E. Extract Information

We now discuss the map $\Pi_{\mathcal{W}}$ that maps from any point on the tangent plane to the underlying information map ϕ embedded in the workspace $\mathcal{W}$. Depending on the representation of the information map, $\Pi_{\mathcal{W}}$ takes different forms. This work considers the following two cases.

1) Case I: The information map ϕ aligns with the manifold $\mathcal{S}$: When the information distribution is defined over on the 2D manifold $\mathcal{S}$, the map $\Pi_{\mathcal{W}}$ is an identity map and the point on the information map is same as the point on the manifold $w_m = \mathbf{q}_m$. Correspondingly, $\mathbf{J}_{\mathcal{W}}$ is an identity matrix:

$$\mathbf{J}_{\mathcal{W}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{63}$$

In practice, the information map on a manifold can be represented in different ways, such as using a mesh [17]. This work uses a point cloud to represent the information map, which samples a finite number of points $\mathcal{W}_S$ from the manifold and assigns each point $w_s \in \mathcal{W}_S$ a probability value while ensuring that all these values sum to one. When computing $\phi(w)$ for any point $w \in \mathbb{W}$, we find the nearest sampled point in $w_s \in \mathcal{W}_S$ and use the value $\phi_(w_s)$ for $\phi(w)$.

2) Case II: The information map ϕ is defined over a flat 2D plane.: When the surface $\mathcal{S}$ is relatively simple, the information map related to $\mathcal{S}$ can be possibly described by a 2D flat information map. In this case, $\Pi_{\mathcal{W}}$ projects each sampled footprint point $\mathbf{q}_m$ to a point w_m on the 2D information map. Correspondingly, $\mathbf{J}_{\mathcal{W}}$ is a constant matrix that describes the projection. For instance, when ϕ is defined over the xy flat plane, the map $\Pi_{\mathcal{W}}$ extracts the (x, y) components from $\mathbf{q}_m \in \mathbb{R}^3$, and

$$\mathbf{J}_{\mathcal{W}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{64}$$

F. Jacobians and Computational Reuse

Now we have all we need to obtain the analytical form of the Jacobian in Eq. 39, which is stated in the following theorem.

Theorem 2 (Footprint Point Jacobian)**.** *For each sampled footprint point q_m with parameters (β_m, φ_m) defined in Definition 11, as overviewed in Sec. VI-A, the Jacobian of a footprint point with respect to the robot state x is:*

$$\frac{\partial w_m(x)}{\partial x} = \mathbf{J}_{\mathcal{W}}(\mathbf{q}_m) \frac{\partial \mathbf{q}_m}{\partial \boldsymbol{\mathcal{X}}} \frac{\partial \boldsymbol{\mathcal{X}}}{\partial x}, \tag{65}$$

where $\mathbf{J}_{\mathcal{W}}(\mathbf{q}_m)$ is presented in Sec. VI-E, $\frac{\partial \mathbf{q}_m}{\partial \boldsymbol{\mathcal{X}}}$ is the full pose Jacobian as derived in Lemma 4, and $\frac{\partial \boldsymbol{\mathcal{X}}}{\partial x}$ is a matrix in $\mathbb{R}^{6\times n}$ that extracts the corresponding position and orientation components from the robot state vector x.

While the entire Jacobian in Theorem 2 is pose-dependent, some components in its computation can be reused. This computational saving arises in several parts. First, as discussed in Remark 2, the evaluation of $\mathbf{J}_p(\beta, \varphi)$ requires only computing the second term that is pose dependent. Second, the matrices $\mathbf{J}_{a/\Theta}$ and $\frac{\partial \boldsymbol{\mathcal{X}}}{\partial x}$ do not depend on the footprint index m. They are therefore computed once (Line 1 in Alg. 1) and reused for all sampled footprint points during gradient computation (Line 6). Third, for any sampled ray with parameters $\{(\beta_m, \varphi_m)\}_{m=1}^{M}$, these parameters are fixed, and the associated trigonometric terms $\{\cos\beta_m, \sin\beta_m \cos\varphi_m, \sin\beta_m \sin\varphi_m\}$ remain constant during optimization, which can be computed once and reused. Finally, in both cases described in Sec. VI-E, the Jacobian $\mathbf{J}_{\mathcal{W}}$ reduces to a constant matrix, which can be computed once (Line 1 in Alg. 1) and reused across all iterations.

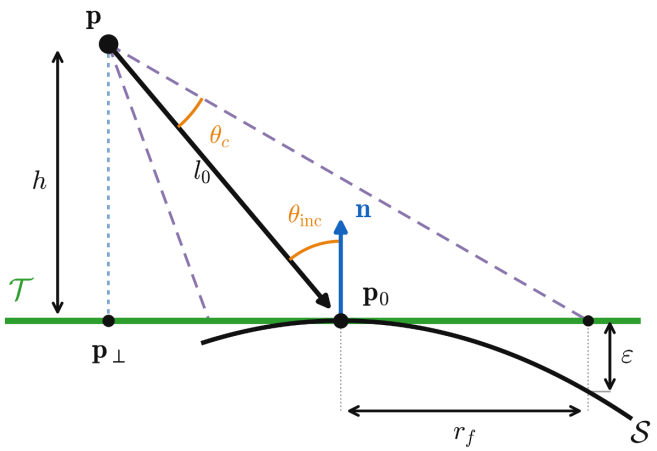


Fig. 6. Geometry of the local tangent-plane approximation error. The robot at $\mathbf{p}$ views the surface $\mathcal{S}$ along its optical axis, which reaches the contact point $\mathbf{p}_0$ at distance l_0; the view cone (dashed) has half-angle θ_c. $\mathcal{T}$ is the tangent plane of $\mathcal{S}$ at $\mathbf{p}_0$, $\mathbf{n}$ its outward normal, and θ_{inc} the incidence angle between the optical axis and $\mathbf{n}$. $\mathbf{p}_\perp$ is the projection of $\mathbf{p}$ onto $\mathcal{T}$ and $h = l_0 \cos\theta_{\text{inc}}$ the height of $\mathbf{p}$ above it. The footprint radius r_f is the largest in-plane distance of a footprint point from $\mathbf{p}_0$, and ε is the deviation of $\mathcal{T}$ from $\mathcal{S}$ at that point, bounded by $\varepsilon \le \frac{1}{2}\kappa_{\max} r_f^2$ (Theorem 3). This figure helps illustrate the derivation of approximation errors.

Remark 3. *The Jacobians become singular when $D(\beta, \varphi) \to 0$, i.e., when a sampled ray $\mathbf{d}(\beta, \varphi)$ becomes parallel to the tangent plane ($\mathbf{n}^\top \mathbf{d} \to 0$). This singularity can be avoided by skipping this sampled ray since it is parallel to and has no intersection point with the tangent plane. In addition, the orientation Jacobian Eq. 57 contains the factor $1/\sin\theta_{\text{inc}}$ and therefore becomes singular when $\sin\theta_{\text{inc}} \to 0$, i.e., when the optical axis $\mathbf{a}$ is perpendicular to the tangent plane. As aforementioned, this case degenerates to the simpler case presented in Sec. V.*

G. Local Tangent Plane Approximation Errors

Under the Assumption 1, we bound the resulting approximation error and its effect on the footprint ergodic metric $\mathcal{E}_\gamma$.

1) Geometric Error: As illustrated in Fig. 6, consider an optical axis of the sensor intersecting a surface $\mathcal{S}$ at point p_0 with $\mathcal{T}$ denoting the local tangent plane at p_0. Let $\kappa_{\max}$ denote the largest absolute value of the principal curvatures of $\mathcal{S}$ within the FOV, which is finite due to Assumption 1.

For any footprint point p within the FOV in the local tangent plane, let q denote the corresponding point on the surface and let r denote the in-plane distance of that footprint point p from p_0. Based on differential geometry, with a second-order Taylor expansion of the surface $\mathcal{S}$ at p_0, the distance $\varepsilon(r)$ of q from the point p in the tangent plane $\mathcal{T}$ is bounded by

$$\varepsilon(r) \leq \tfrac{1}{2}\,\kappa_{\max}\, r^2 + O(r^3). \tag{66}$$

Theorem 3 (Approximation error)**.** *Let r_f be the footprint radius, i.e., the largest r within the FOV. Under Assumption 1, every footprint point on $\mathcal{T}$ deviates from $\mathcal{S}$ by at most*

$$\varepsilon \leq \tfrac{1}{2}\,\kappa_{\max}\, r_f^2, \tag{67}$$

$$r_f = l_0 \sin\theta_c / \cos(\theta_{\text{inc}} + \theta_c). \tag{68}$$

Proof. Note that, r_f is finite for $\theta_{\text{inc}} + \theta_c < \pi/2$. Ignoring the higher-order term in (66), $\varepsilon(r) \leq \frac{1}{2}\kappa_{\max} r^2$, so it suffices to bound r by the footprint radius r_f. Let $\mathbf{p}_\perp$ be the projection of $\mathbf{p}$ onto $\mathcal{T}$ (Def. 8), at height $h = \mathbf{n}^\top(\mathbf{p} - \mathbf{p}_0) = l_0 \cos\theta_{\text{inc}}$. The distance between $\mathbf{p}_\perp$ and $\mathbf{p}_0$ is $\rho = h\tan\theta_{\text{inc}}$. Following the frame defined in Def. 10, in the plane containing $\mathbf{a}$ and $\mathbf{n}$, the angle between the outermost cone ray ($\beta = \theta_c$, $\varphi = \pi$) and the line between $\mathbf{p}$ and $\mathbf{p}_\perp$ is $\theta_{\text{inc}} + \theta_c$ with $D(\theta_c, \pi) = \cos(\theta_{\text{inc}} + \theta_c)$ by (53). The outermost cone ray meets $\mathcal{T}$ at a distance of $h\tan(\theta_{\text{inc}} + \theta_c)$ from $\mathbf{p}_\perp$. With $h = l_0\cos\theta_{\text{inc}}$:

$$\begin{aligned} r_f &= h\tan(\theta_{\text{inc}} + \theta_c) - h\tan\theta_{\text{inc}} \\ &= \frac{h\sin\theta_c}{\cos(\theta_{\text{inc}} + \theta_c)\cos\theta_{\text{inc}}} = \frac{l_0\sin\theta_c}{\cos(\theta_{\text{inc}} + \theta_c)}, \end{aligned} \tag{69}$$

where the second equality uses $\tan(\theta_{\text{inc}} + \theta_c) - \tan\theta_{\text{inc}} = \sin\theta_c / [\cos(\theta_{\text{inc}} + \theta_c)\cos\theta_{\text{inc}}]$. Substituting $r \leq r_f$ into $\varepsilon(r) \leq \frac{1}{2}\kappa_{\max} r^2$ yields (68). □

It shows that ε grows linearly with the curvature and quadratically with the footprint radius, and the maximum footprint radius grows linearly with l_0, the distance between $\mathbf{p}$ and $\mathbf{p}_0$. Intuitively, it indicates that the approximation error is large when the robot is far away from the surface, or when the robot has a very oblique view of the surface.

2) Effect on the Footprint Ergodic Metric: The footprint ergodic metric $\mathcal{E}_\gamma$ is a nonlinear function of the trajectory, which makes it challenging to derive a closed-form representation of how ε propagates to $\mathcal{E}_\gamma$. We only provide an intuitive discussion. The metric does not use the footprint points directly. It uses only a finite set of low-frequency Fourier coefficients of the time-averaged statistics of the footprint trajectory (Def. 3). The geometric approximation error (Eq. (68)) ε perturbs each coefficient bounded by an amount proportional to ε, which is often a small amount and usually affects the high frequency components of the Fourier coefficients. Since in practice $\mathcal{E}_\gamma$ is computed by using only a finite number of low-frequency modes with decaying weights Λ_k, the resulting ergodic metrics are less sensitive to such geometric approximation error.

Finally, for surfaces with large curvatures, or surfaces that violate Assumption 1 (e.g., 3D objects), a promising future direction is to iteratively apply our approach to a local region on such surfaces as the robot moves. Besides, similar to regular ergodic metric, the proposed footprint ergodic metric (Def. 4) also suffers from distortion in the metrics, especially when the surface has large curvature. Such distortion can be handled by using the eigenfunctions of the Laplace-Beltrami operator on the surface mesh, which form an intrinsic Fourier-like basis and reduce to the cosine modes on a flat domain [13], [17], [37], [38]. As it is not the major focus of this paper, we briefly showcase its usage in the experimental results (Sec. VII-F).

VII. Experimental Results

We first run experiments with 2D information maps (Sec. VII-B1-VII-C), then with 3D complex terrains and 3D Objects (Sec. VII-E-VII-F), and finally with real robots (Sec. VII-G).

A. Experimental Settings

1) Dynamics and Parameters: For experiments with 2D information maps, the robot dynamics is double integrators with the robot state including (x, y, z) positions and (v_x, v_y, v_z) velocities and the robot control including (a_x, a_y, a_z) accelerations. The position of the robot is limited to a bounded space $[0, 1]^3$ and the controls are bounded component-wise within $[-0.5, 0.5]$. We adopt the uniform circular footprint model in Example 1, where the footprint radius scales with altitude as $r(t) = k_h f_h(x(t))$ and $k_h = 0.25$. The footprint ergodic metric is approximated using deterministic uniform sampling inside the unit disk with Cartesian grid resolution $\Delta = 0.35$, which corresponds to $M = 25$ sampled points within the footprint. The number of cosine Fourier basis for computing the ergodicity is 10×10.

2) Baseline Methods: We consider the following baselines.

- Baseline 1 Regular Ergodic Metric: It optimizes the original ergodic metric $\mathcal{E}(\phi, x)$ with Dirac delta function [6].
- Baseline 2 Fixed Sampled Footprint: It optimizes the proposed footprint ergodic metric yet with a fixed footprint, which is equivalent to that the robot flies with a fixed altitude, resulting in a constant footprint radius.
- Baseline 3 Fixed Gaussian Footprint: Following the literature [10], [17], the uniform footprint in Eq. 4 is replaced by a Gaussian distribution $\mathcal{N}(\mu(t), \sigma)$ whose mean value $\mu(t)$ is same as the robot x, y position, and its variance σ is constant.
- Baseline 4 HEDAC: It is based on the heat equation driven area coverage method [12], [13], which guides robot motion by the gradient of the evolving heat field instead of trajectory optimization. We implement this approach with a fixed size sensor footprint in our setting as a baseline for comparison.

Note that, all baselines have a fixed sensor footprint, while our approach has a dynamic footprint and the trajectory is optimized in 3D space, including altitude, without orientation. Fig. 7 shows the planned trajectories by our FEO.

B. Footprint Ergodic Metrics in Different Maps

We run the planners in four different maps as shown in Fig. 8 and compare their runtime and the corresponding ergodic metrics. This section discusses the ergodicity, and we discuss the runtime in Sec. VII-C later.

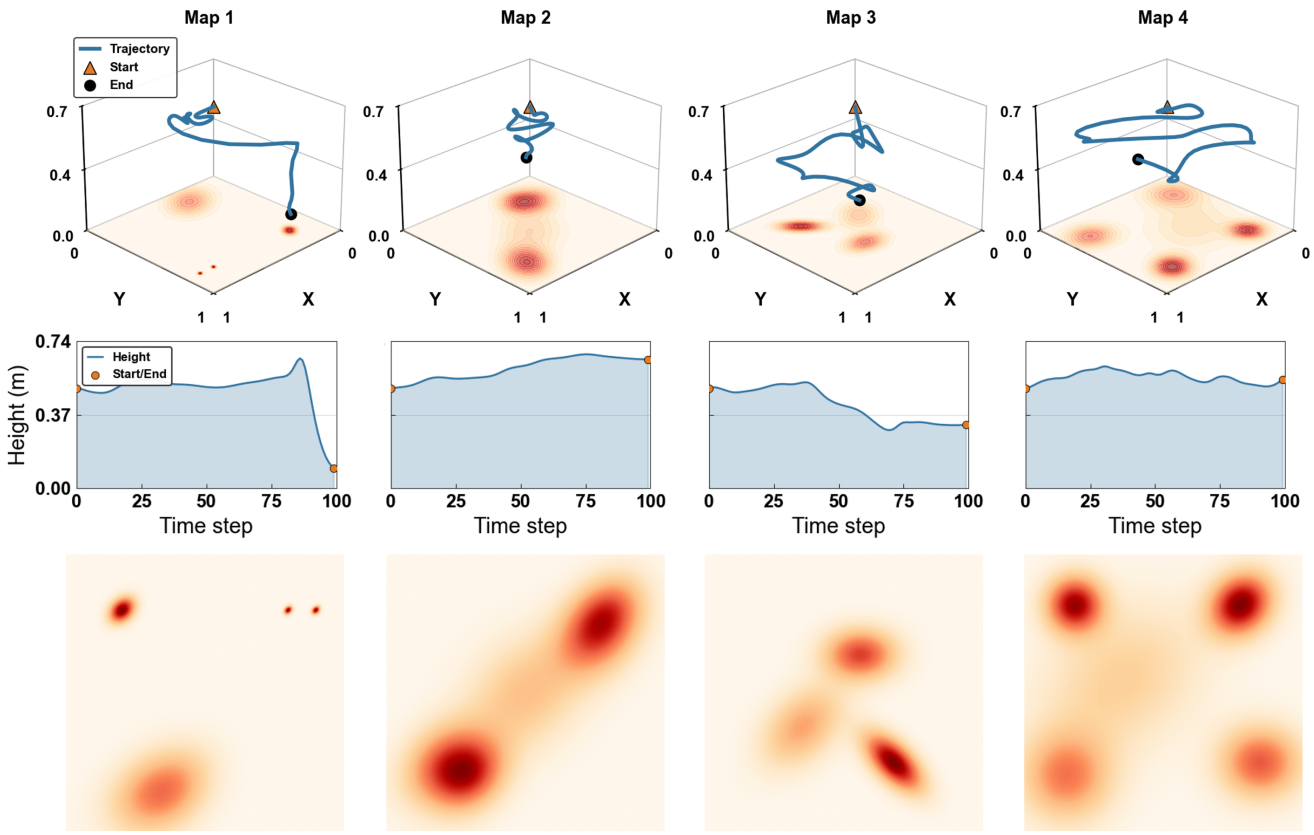


Fig. 7. The experimental results in four different information maps. The first row (I) visualizes the planned trajectories by our FEO. The middle row (II) shows the corresponding height value (z coordinates) of the robot along the planned trajectories by our FEO. The last row (III) shows four information maps with various regions with widespread and concentrated information. Our FEO can adapt the flying height of the robot and the sensor footprint based on the information map.

1) Against Regular Ergodic Metric: The results are shown in the first column in Fig. 8. Our FEO achieves better (lower) ergodicity than the baselines with or without log-spaced search when using the regular ergodic metric. In particular, the ergodicity is reduced by approximately 57% on Map 1, 76% on Map 2, 15% on Map 3, and 62% on Map 4, which verifies the benefits of considering dynamic footprint. In addition to the improved ergodicity, our FEO consistently reduces the total runtime across all maps. A detailed discussion on runtime is provided in Sec. VII-C.

2) Against Fixed Sampled Footprint: The radius of the fixed footprint is selected as the one corresponding to the midpoint of the allowable altitude range. As shown in the middle column in Fig. 8, our FEO achieves lower ergodicity than the baseline on all four maps, where the ergodicity is reduced by approximately 57% on Map 1, 74% on Map 2, and 66% on Map 4. In particular, on Map 3, where the information distribution is relatively smooth and has a single concentrated area, both methods achieve comparable performance (with only 3.7% difference). On other maps, the differences in the ergodicity are larger. When the information map has both concentrated and wide-spread information, a single fixed-size footprint (as in the baseline) covers the map poorly, while our FEO covers well due to the dynamic footprint.

3) Against Fixed Gaussian Footprint: When compared to the baseline [10], [17] in the last column in Fig. 8. Our FEO reduces the ergodicity by approximately $18\times$ on Map 1, $61\times$ on Map 2, $40\times$ on Map 3, and $25\times$ on Map 4. The reason is that a Gaussian footprint with fixed covariance better covers low-frequency components in the Fourier spectrum and cannot ergodically cover both concentrated and wide-spread information simultaneously, while our FEO can adjust the footprint size when needed.

4) Against HEDAC: As shown in Fig. 9, baseline 4 HEDAC returns the first solution very quickly and is able to improve the solution quality as the runtime increases. However, this improvement plateaus after a while. In contrast, FEO achieves a two-order-of-magnitude lower ergodicity than HEDAC within 2 seconds across all four maps. Specifically, in Maps $2, 3, 4$, our FEO reaches the 10^{-3} ergodicity, which is the termination threshold for the optimization, within 2 seconds. In Map 1, while both methods show slower convergence, FEO still reduces the ergodicity by about two orders of magnitude within 2 seconds. Our observation aligns with the results reported in the literature [17], which verifies the benefits of trajectory optimization over PDE-driven solution.

C. Runtime Comparison and Ablation Study

1) Runtime Comparison against Baselines: Although our method optimizes an additional state dimension (i.e., altitude in 3D), it often requires less runtime than all baselines across the four information maps (Fig. 8(b)(I), 8(b)(II), and 8(b)(III)).

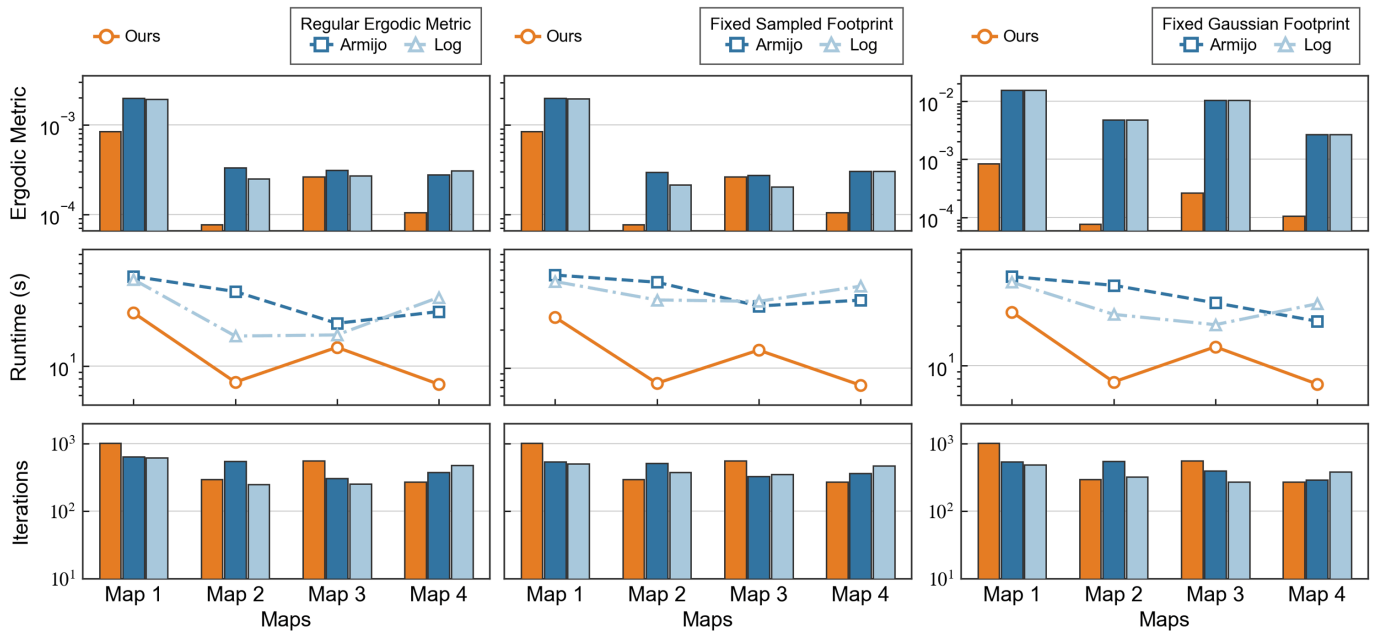


Fig. 8. We conducted a comparison using three baselines to evaluate the trajectory optimization over 100 time steps across four different information maps shown in Fig 7. In the first row (a), we present the comparison of the ergodic metric. The second row (b) displays the time cost for convergence, while the third row (c) presents the number of iterations required. Columns (I), (II), and (III) correspond to the results from Baselines 1, 2, and 3, respectively.

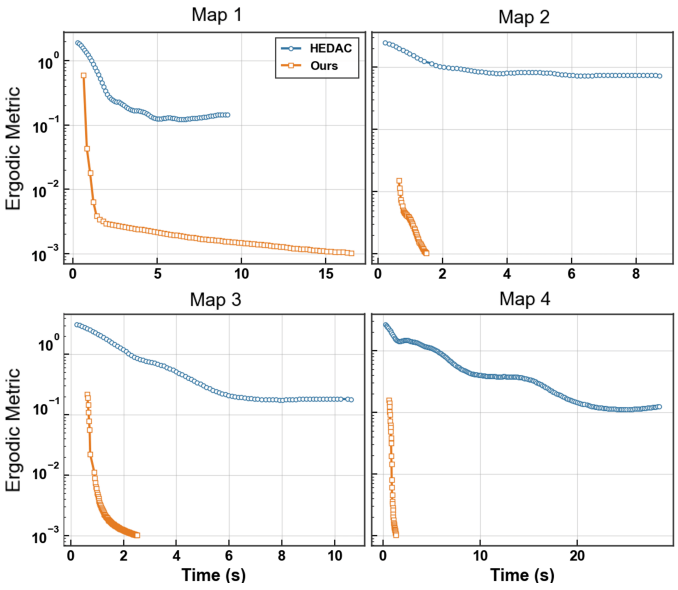


Fig. 9. The iterations of HEDAC optimized over four maps shown in Fig. 7, the footprint size is similar to the Baseline Fixed Footprint VII-A2.

Note that, the number of iterations is not always smaller. In certain cases (e.g., Map 1), ours requires more iterations than the baselines (Fig. 8(c)). However, the total runtime remains much lower, indicating that per-iteration computational runtime is an important factor. The reason for such computational efficiency of our FEO is due to (i) the proposed reuse of Jacobians and (ii) the proposed log-spaced line search, during the trajectory optimization. We verify the benefits of these two proposed techniques in the following ablation study.

2) Ablation Study: Within our FEO framework, we consider the following variants for comparison.

- Armijo-NoReuse: This variant uses the popular Armijo backtracking line search approach and use the auto-differentiation provided by JAX rather than using our proposed Jacobian reuse techniques.
- Log-NoReuse: This variant uses our log-spaced line search and use the auto differentiation provided by JAX rather than using our proposed Jacobian reuse techniques.
- Armijo-Reuse: This variant uses the popular Armijo backtracking line search approach and use our Jacobian reuse techniques.
- Log-Reuse: This variant is the full version of our approach, including both our log-spaced line search and our proposed Jacobian reuse techniques.

Fig. 10 shows the per-iteration runtime on four maps. First, by comparing Armijo-NoReuse and Log-NoReuse, we find that the proposed log-spaced line search itself can obviously speed up the computation independently. Second, by comparing Armijo-NoReuse and Armijo-Reuse, we find that the proposed Jacobian reuse technique itself can also obviously speed up the computation independently. Finally, when applying both proposed techniques together, further improvement in runtime can be observed sometimes, but it is often tiny. A possible reason is that, after the log-spaced line search reduces the number of iterations, using Jacobian reuse to further save the computation per iteration become less important since the number of iterations is already very small.

D. Selection of Parameters

This section studies the effect of the number of footprint samples M and the number of Fourier bases $\mathbf{k}$, using Map 2

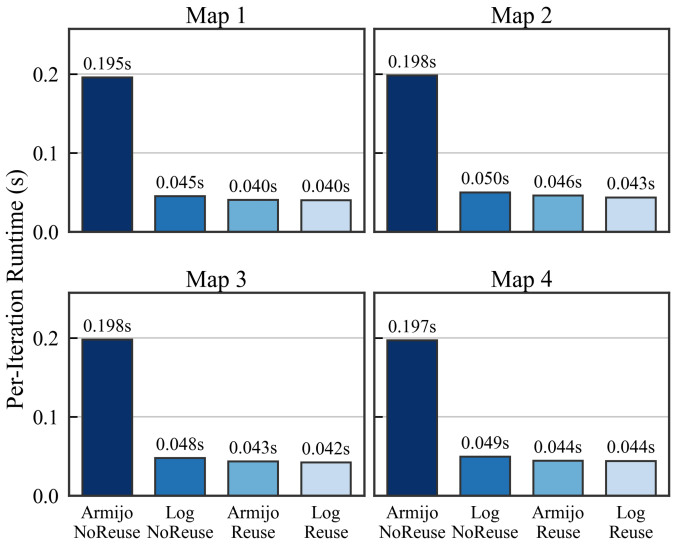


Fig. 10. Per-iteration runtime comparison of different line-search strategies across four information maps shown in Fig 7. The proposed gradient reuse method consistently reduces computational cost.

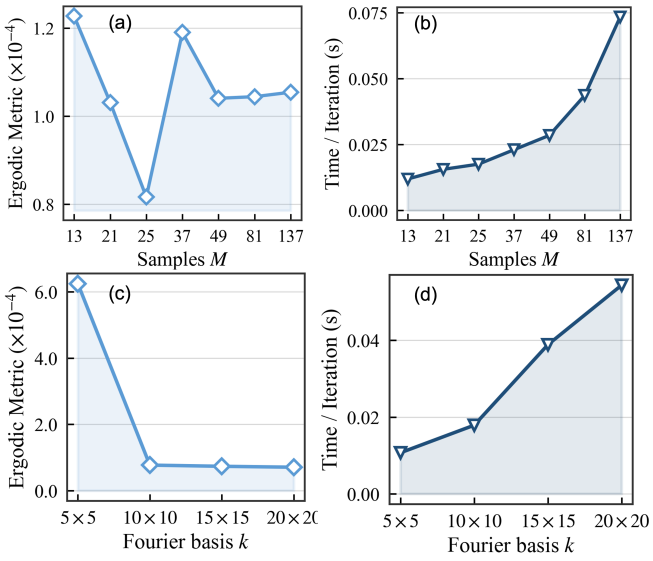


Fig. 11. Ergodicity and runtime with varying M (number of sampled points in the sensor footprint), and k (number of Fourier basis). All tests are run on Map 2 in Fig. 7.

as an example. As shown in Fig. 11, increasing M does not consistently improve the ergodic metric while it does increase runtime, whereas increasing $\mathbf{k}$ improves the metric with diminishing returns.

We find that $\mathcal{E}_\gamma$ has non-monotonic dependence on M, which is caused by the sampling of footprint points: As shown in Fig. 13, the footprint is sampled on a Cartesian grid of step Δ, and M is the number of grid nodes falling inside the circular footprint. As Δ decreases, M changes with discrete jumps $(13, 21, 25, 37, \dots)$, each with a different layout rather than a refinement of the previous one. Consequently, for each specific M, $\mathcal{E}_\gamma$ varies smoothly with Δ, while $\mathcal{E}_\gamma$ jumps when M changes (Fig. 12). For example, the reconstructed map using Fourier coefficients for $M = 25$ is circular, while varies with protruding corners at $M = 37$. Despite these jumps, note that, this variation is about 10^{-5}, which is orders of magnitude smaller than the differences in $\mathcal{E}_\gamma$ returned by different algorithms (Fig. 8). Based on these results, we set $M = 25$ and $\mathbf{k} = 10\times10$ in all experiments, balancing solution quality and per-iteration runtime cost.

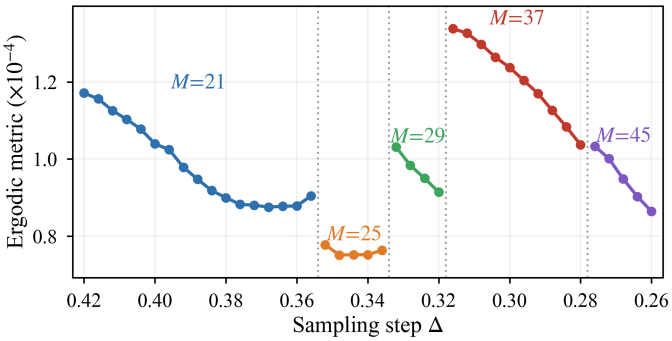


Fig. 12. Footprint ergodic metric $\mathcal{E}_\gamma$ versus the footprint sampling step size Δ on Map 2. $\mathcal{E}_\gamma$ varies "smoothly" with Δ for each M, but jumps when M changes. The reason is illustrated with Fig. 13.

E. Extensions to Complex Terrains

While the previous sections focused on planar settings, this section evaluates our FEO on diverse terrains to investigate:

- the benefits of pose-aware planning;
- the computational saving of Jacobian reuse in 3D;
- the generalization to various 3D terrain and objects.

1) Experiment Settings: For experiments with 3D terrain information maps, the robot state including positions (x,y,z), velocities (v_x,v_y,v_z), and pitch and yaw (θ_p, θ_i). The robot control includes accelerations (a_x, a_y, a_z) and angular velocity of pitch and yaw (w_p, w_i). The position dynamics is double integrator and the angles follow a single integrator. The position of the robot is limited to a bounded space $[0, 10]^2 \times [2, 8]$, with the additional constraint that the altitude must maintain at least 0.8 m clearance above the terrain surface. The controls are bounded as $a_x, a_y \in [-1.5, 1.5]$ m/s^2, $a_z \in [-1.0, 1.0]$ m/s^2, and $\omega_p, \omega_y \in [-0.3, 0.3]$ rad/s. The velocity magnitude is constrained to $\|\mathbf{v}\| \leq 3.0$ m/s, and the pitch angle is limited to $\theta_p \in [-45, 45]$. We adopt the view cone defined in Sec. 7 with half FOV angle $\alpha = 15°$. The footprint are computed with $N_{\text{radial}} \times N_{\text{azimuthal}} = 3 \times 12 = 36$ sampled rays plus the optical axis itself, yielding 37 sample points. The information maps ϕ are defined on the 2D terrain manifold as mentioned in Sec. VI-E1. The terrain is defined as a height function $H : [0, 10]^2 \to \mathbb{R}$. As shown in the first row of Fig. 15, the terrains are generated by mixtures of Gaussians (a-d) and a real-world terrain (e) constructed from publicly available elevation data of the Los Angeles region. The information map are also generated with mixtures of Gaussians randomly sampled, which are then projected onto the terrain, with the red areas indicating the high-information regions as shown in the figure. We refer to these five maps as (a) Double-Peak, (b) Ridge, (c) Valley, (d) Multi-Peak, and (e) Real-World.

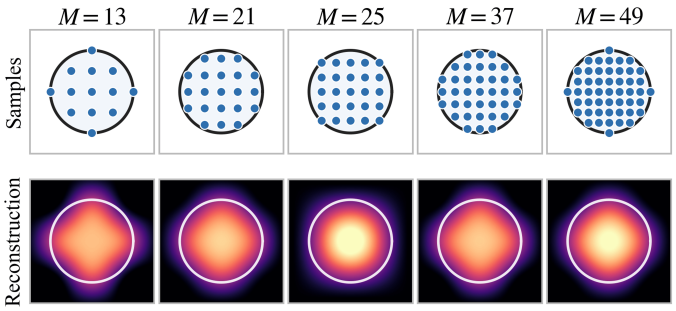


Fig. 13. Reconstructed footprints for different number of sampled footprint points M. The top row shows the Cartesian grid samples retained inside the circular footprint. The bottom row shows the footprint reconstructed from its Fourier coefficients, where the white circle is the boundary of the footprint. It help illustrate the reason behind the jump in $\mathcal{E}_\gamma$ in Fig. 12.

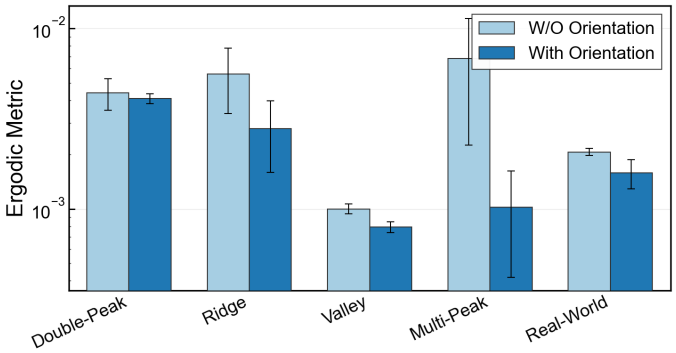


Fig. 14. The comparison between position control and altitude control in different terrains.

2) Pose and Position Planning: Fig. 15 shows the optimized SE(3) trajectories over five different 3D terrains. The first row displays the trajectories to cover the terrain. The middle row shows the altitude over time, including terrain height and safety bounds. The last row illustrates the poses, where arrows indicate the sensor orientation. Intuitively, the robot tends to fly with lower altitude to move closer to the terrain in high-information region, while flying further away with a large FOV for low-information regions. However, this is not always the case since the robot FOV has orientation and it is possible that, the robot is at a high altitude and the FOV is oblique to better cover the nearby terrains as opposed to covering the terrains beneath the robot.

To better understand the benefits of planning with orientation, we compare against a variant where the robot orientation is fixed to let the sensor always point downwards (i.e., w/o orientation). Fig. 14 shows that, our method achieves nearly 40% on average smaller (better) ergodicity by planning with orientation than w/o orientation among five different terrains.

3) Jacobian Reuse in 3D: To investigate the benefits of computational reuse in 3D, we compare three variants: (i) Log-NoReuse-JAX uses the automatic differentiation toolbox to compute the Jacobians; (ii) Log-NoReuse-Analytic uses the analytical Jacobian we derived without any computational reuse; and (iii) Log-Reuse-Analytic uses the analytical Jacobian we derived while leverage our proposed computational reuse as in Sec. VI-F. Note that all three variants compared here employ the same log-spaced line search to eliminate the influence of step-size selection, and the only difference lies in their gradient evaluation.

Fig. 16 shows the average per-iteration runtime in 3D terrains. First, Log-NoReuse-Analytic is slower than Log-NoReuse-JAX, since JAX is a fast auto-diff library with implementation optimization, while Log-NoReuse-Analytic is implemented in Python and may run slow. Second, although our Log-Reuse-Analytic is also implemented in Python directly, compared with Log-NoReuse-JAX, our Log-Reuse-Analytic reduces the per-iteration runtime by approximately 25%–40%. For example, in Real-World map, the per-iteration runtime is reduced from $152.3\,\text{ms}$ (JAX) to $91.6\,\text{ms}$ (ours), corresponding to a reduction of approximately 40%.

F. Generalization to 3D Objects

While our main focus is to cover 2D manifolds, such as uneven terrains, this section evaluates the extension of our approach to cover complex objects in 3D. As aforementioned in Sec. VI-G, to avoid metric distortion, we follow the intrinsic surface representation of [17], where the ergodic metric is computed using Laplace-Beltrami eigenfunctions [37], [38] defined on the object surface mesh, rather than the flat 2D Fourier basis. We use [17] as the baseline and follow its 3D object coverage optimization setting such as the point-mass single-integrator dynamics.

For our method, we introduce two variants (ours-1, ours-2): Ours-1 is same as the baseline with the only difference on its sensor footprint, which is no longer a fixed Gaussian. Instead, the sensor footprint is a dynamic cone, whose optical axis points from the point-mass robot to the nearest point on the object, and the size of the cone varies based on the distance to the object. Ours-2 includes orientations into the robot state and dynamics, as opposed to treating the robot as a point mass as the baseline and ours-1 do. Ours-2 aligns the optical axis with the x-axis of the robot. We set a terminating threshold of 2×10^{-6} for ergodicity.

Fig. 17 shows that, ours-1 and the baseline [17] achieve similar performance and trajectories. Specifically, while the baseline [17] converges faster than ours-1 at the beginning (e.g. < 300 s), ours-1 eventually converges to the lowest ergodic metric very quickly among the three planners (1.99×10^{-6} in about 500 s), roughly 1500 s faster than the baseline (2.10×10^{-6} in more than 2000 s). In contrast, ours-2 converges slower and yields higher ergodicity (3.08×10^{-6} in 2500 s) due to the inclusion of orientation into the robot dynamics, which makes the optimization harder. For a more rigorous discussion of ergodicity on curved surfaces, please refer to [13], [17].

G. Real Robots Experiments

We deploy our planner on quadrotors with dynamics [39]. We use Crazyflie 2.1 micro aerial vehicles [40] within a motion capture system, which are controlled using the Crazyswarm2 framework [41] integrated with ROS2 [42]. Each quadrotor is equipped with a downward-facing LED to visualize its sensor footprint during the robot motion.

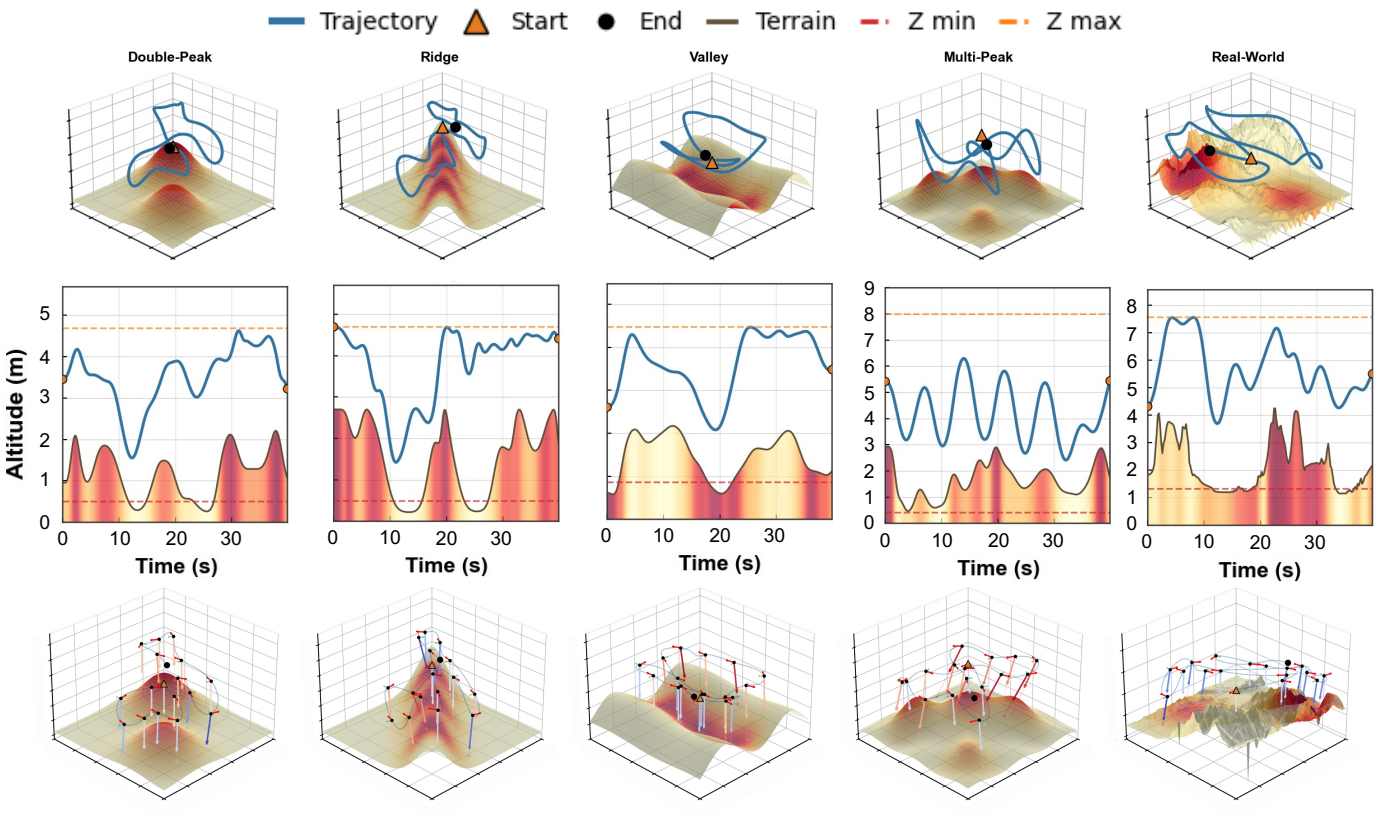


Fig. 15. SE(3) trajectories planned by our FEO over various complex terrains. Each row corresponds to a different terrain map: (a) Double-Peak, (b) Ridge, (c) Valley, (d) Multi-Peak, and (e) Real-World. Among them, (e) is constructed from publicly available elevation data of the Los Angeles region. First Row: spatial trajectories over terrain surfaces. Middle: altitude profiles over time, including terrain height and safety bounds. Last Row: full SE(3) pose evolution with orientation visualization. Our FEO adjusts both position and orientation to conform to terrain geometry using its dynamic sensor footprint.

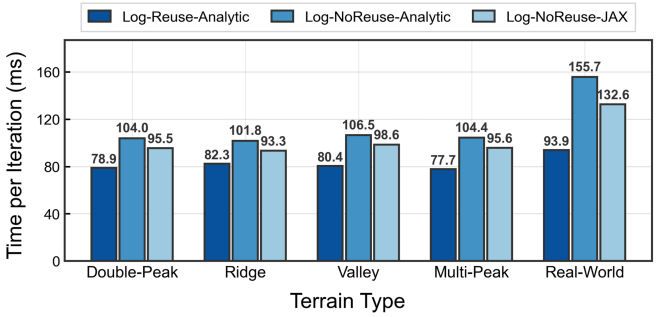


Fig. 16. Per-iteration runtime for complex terrains. All planners use log-spaced line search. Our method with analytical gradient reuse can consistently reduce runtime in all terrains.

The test environment is a 2D workspace of size 4.0×4.0 meters with a single drone that seeks to ergodically cover a 2D information map. The flying height of the drone is limited to $[0, 1]$ meters. The planning horizon is 10 seconds with each time step being 0.1. Fig. 1 and 18 show the test setting and some snapshots of the robot motion. We place a few items on the ground in the test area to indicate the information distribution. The robot first flies high to cover the wide-spread information with a coarse sensor footprint. After a while, the robot flies to the information peak and lowers its altitude, as shown in Fig. 1(b), in order to cover the dense information with a fine sensor footprint.

VIII. Conclusions and Future Work

This paper explores Ergodic Trajectory Optimization with Dynamic sensor Footprints (ETO-DF) and introduces a new footprint ergodic metric that considers the dynamic sensor footprint. We formulate a corresponding Optimal Control Problem (OCP), and develop a numerical approach FEO to compute the proposed footprint ergodic metric and leverage Lagrangian optimization to solve the problem. The key idea in our FEO is the derivation of analytical gradient of the sampled points in the sensor footprint with respect to the robot pose in SE(3), for both flat ground and uneven terrain represented by manifolds. We evaluate our FEO and compare it against several recent baselines in various maps, with both simulation and real quadrotor experiments. The results verify that, our FEO can plan trajectories for the robots by considering their varying sensor footprints so as to better cover the information map ergodically.

For future work, one can consider dynamic sensor footprints in dynamically changing information maps by integrating the Bayesian filtering framework [34], or consider connectivity maintenance and communication among the robots [35].

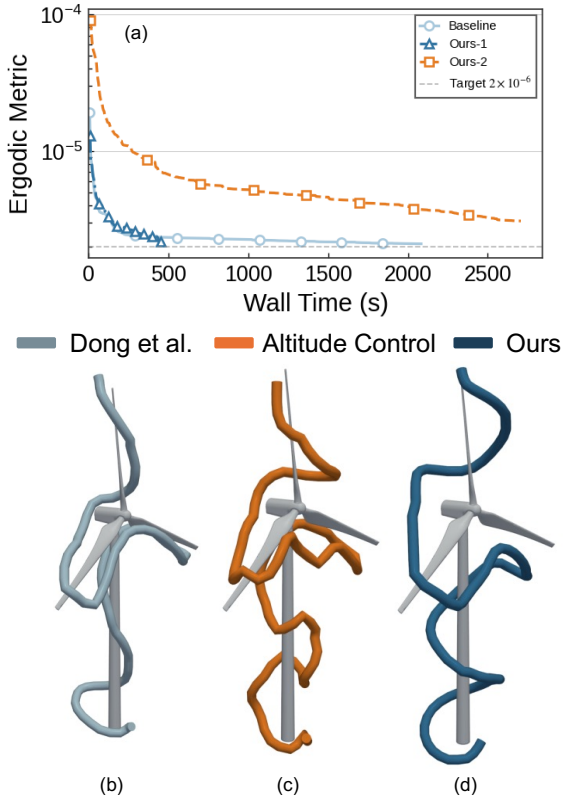


Fig. 17. Tests on 3D objects. (a) Ergodic metric versus wall time. Our FEO achieves the fastest convergence and the lowest final metric. (b)-(d) Trajectory of three different methods.

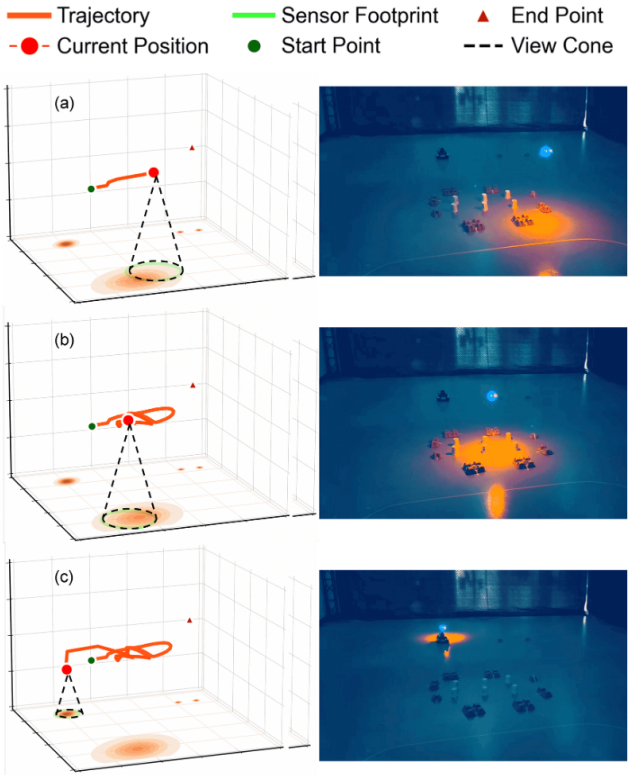


Fig. 18. Snapshots of a drone executing the trajectory planned by FEO in Map 1 shown in Fig. 7. The drone is equipped with a downward-pointing LED to illustrate the sensor footprint. (a)-(c) show the snapshot of the system at time $1.7, 8.3, 9.4$ seconds.

REFERENCES

[1] Y. Liu and G. Nejat, "Robotic urban search and rescue: A survey from the control perspective," *Journal of Intelligent & Robotic Systems*, vol. 72, no. 2, pp. 147–165, 2013.

[2] A. Mavrommati, E. Tzorakoleftherakis, I. Abraham, and T. D. Murphey, "Real-time area coverage and target localization using receding-horizon ergodic exploration," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 62–80, 2017.

[3] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1134–1154, 2012.

[4] W. Chen and L. Liu, "Pareto monte carlo tree search for multi-objective informative planning," in *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[5] M. Santos, Y. Diaz-Mercado, and M. Egerstedt, "Coverage control for multirobot teams with heterogeneous sensing capabilities," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 919–925, 2018.

[6] G. Mathew and I. Mezić, "Metrics for ergodicity and design of ergodic dynamics for multi-agent systems," *Physica D: Nonlinear Phenomena*, vol. 240, no. 4, pp. 432–442, 2011.

[7] L. M. Miller and T. D. Murphey, "Trajectory optimization for continuous ergodic exploration," in *2013 American Control Conference*. IEEE, 2013, pp. 4196–4201.

[8] G. De La Torre, K. Flaßkamp, A. Prabhakar, and T. D. Murphey, "Ergodic exploration with stochastic sensor dynamics," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 2971–2976.

[9] I. Abraham and T. D. Murphey, "Decentralized ergodic control: distribution-driven sensing and exploration for multiagent systems," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2987–2994, 2018.

[10] E. Ayvali, H. Salman, and H. Choset, "Ergodic coverage in constrained environments using stochastic trajectory optimization," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5204–5210.

[11] S. Schneyer, K. Nottensteiner, A. Albu-Schäffer, F. Stulp, and J. Silvério, "An ergodic approach to robotic surface finishing with learned motion preferences," *IEEE Transactions on Robotics*, vol. 42, pp. 342–361, 2026.

[12] Bilaloglu, Cem and Löw, Tobias and Calinon, Sylvain, "Whole-body ergodic exploration with a manipulator using diffusion," *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 8581–8587, 2023.

[13] C. Bilaloglu, T. Löw, and S. Calinon, "Tactile ergodic coverage on curved surfaces," *IEEE Transactions on Robotics*, vol. 41, pp. 1421–1435, 2025.

[14] S. Ivić, B. Crnković, L. Grbčić, and L. Matleković, "Multi-uav trajectory planning for 3d visual inspection of complex structures," *Automation in Construction*, vol. 147, p. 104709, 2023.

[15] J. Kwon, M. M. Sun, and T. Murphey, "Volumetric ergodic control," 2026. [Online]. Available: https://arxiv.org/abs/2511.11533

[16] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives." *Pacific Journal of Mathematics*, vol. 16, pp. 1–3, 1966.

[17] D. Dong, A. Xu, G. Gutow, H. Choset, and I. Abraham, "Ergodic exploration over meshable surfaces," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 13 088–13 094.

[18] I. Abraham, A. Prabhakar, and T. D. Murphey, "An ergodic measure for active learning from equilibrium," *IEEE Transactions on Automation Science and Engineering*, 2021.

[19] M. M. Sun, A. Gaggar, P. Trautman, and T. Murphey, "Fast ergodic search with kernel functions," *IEEE Transactions on Robotics*, vol. 41, pp. 1841–1860, 2025.

[20] C. Hughes, H. Warren, D. Lee, F. Ramos, and I. Abraham, "Ergodic trajectory optimization on generalized domains using maximum mean discrepancy," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 01–07.

[21] D. E. Dong, H. Berger, and I. Abraham, "Time-optimal ergodic search: Multiscale coverage in minimum time," *The International Journal of Robotics Research*, p. 02783649241273597, 2024.

[22] Z. Ren, A. K. Srinivasan, B. Vundurthy, I. Abraham, and H. Choset, "A

pareto-optimal local optimization framework for multiobjective ergodic search," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3452–3463, 2023.

[23] Y. Liu and Z. Ren, "Multi-robot ergodic trajectory optimization with relaxed periodic connectivity," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 8652–8659.

[24] B. Davis, I. Karamouzas, and S. J. Guy, "C-opt: Coverage-aware trajectory optimization under uncertainty," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 1020–1027, 2016.

[25] C. Mostegel, J. Ye, Y. Luo, and Y. Liu, "Overlap displacement error: Are your slam poses map-consistent?" in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5336–5343.

[26] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-mcts: Decentralized planning for multi-robot active perception," *The International Journal of Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[27] T. Patten, M. Zillich, R. Fitch, M. Vincze, and S. Sukkarieh, "Viewpoint evaluation for online 3-d active object classification," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 73–81, 2015.

[28] Z. Tang and U. Ozguner, "Motion planning for multitarget surveillance with mobile sensor agents," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 898–908, 2005.

[29] H. Coffin, I. Abraham, G. Sartoretti, T. Dillstrom, and H. Choset, "Multi-agent dynamic ergodic search with low-information sensors," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 11 480–11 486.

[30] X. Wang, J. Xu, C. Gao, Y. Chen, J. Zhang, C. Wang, Y. Ding, and B. M. Chen, "Sensor-based multi-robot coverage control with spatial separation in unstructured environments," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 10 623–10 629.

[31] S. A. Sadat, J. Wawerla, and R. Vaughan, "Fractal trajectories for online non-uniform aerial coverage," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 2971–2976.

[32] E. Galceran and M. Carreras, "Efficient seabed coverage path planning for asvs and auvs," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 88–93.

[33] F. Stache, J. Westheider, F. Magistri, C. Stachniss, and M. Popović, "Adaptive path planning for uavs for multi-resolution semantic segmentation," *Robotics and Autonomous Systems*, vol. 159, p. 104288, 2023.

[34] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2015.

[35] Y. Liu and Z. Ren, "A Probabilistic Measure of Multi-Robot Connectivity and Ergodic Optimal Control," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, Jun. 2025.

[36] T. A. Howell, B. E. Jackson, and Z. Manchester, "Altro: A fast solver for constrained trajectory optimization," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7674–7679.

[37] N. Sharp and K. Crane, "A Laplacian for Nonmanifold Triangle Meshes," *Computer Graphics Forum (SGP)*, vol. 39, no. 5, 2020.

[38] B. Levy, "Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry," in *IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, 2006, pp. 13–13.

[39] C. Luis and J. L. Ny, "Design of a trajectory tracking controller for a nanoquadcopter," 2016.

[40] W. Giernacki, M. Skwierczyński, W. Witwicki, P. Wroński, and P. Kozierski, "Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering," in *2017 22nd International Conference on Methods and Models in Automation and Robotics (MMAR)*, 2017, pp. 37–42.

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

[42] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science robotics*, vol. 7, no. 66, p. eabm6074, 2022.

APPENDIX

This appendix details the proofs of lemmas in Sec. VI.

A. Proof of Lemma 1

For a fixed tangent plane with parameters $(\mathbf{p}_0, \mathbf{n})$ and a fixed ray with parameters (β, φ), recall that the footprint point associated with ray (β, φ) is $\mathbf{q}(\beta, \varphi) = \mathbf{p} + l^*(\beta, \varphi)\,\mathbf{d}(\beta, \varphi)$, where $l^*(\beta, \varphi) = h/D(\beta, \varphi)$, $h = \mathbf{n}^\top(\mathbf{p} - \mathbf{p}_0)$, and $D(\beta, \varphi) = -\mathbf{n}^\top \mathbf{d}(\beta, \varphi)$. Then,

$$\frac{\partial h}{\partial \mathbf{p}} = \frac{\partial \mathbf{n}^\top(\mathbf{p} - \mathbf{p}_0)}{\partial \mathbf{p}} = \mathbf{n}^\top, \tag{70}$$

$D(\beta, \varphi)$ is constant, and

$$\frac{\partial l^*}{\partial \mathbf{p}} = \frac{\partial}{\partial \mathbf{p}}\left(\frac{h}{D}\right) = \frac{1}{D}\frac{\partial h}{\partial \mathbf{p}} = \frac{1}{D}\mathbf{n}^\top. \tag{71}$$

Then, Eq. (56) to be proved is obtained by:

$$\frac{\partial \mathbf{q}}{\partial \mathbf{p}} = \frac{\partial}{\partial \mathbf{p}}\left(\mathbf{p} + l^*\mathbf{d}\right) = \mathbf{I}_3 + \mathbf{d}\frac{\partial l^*}{\partial \mathbf{p}} = \mathbf{I}_3 + \frac{1}{D}\,\mathbf{d}\,\mathbf{n}^\top. \tag{72}$$

B. Proof of Lemma 2

1) Cone Basis Derivatives: The cone basis vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ are defined in Definition 10:

$$\mathbf{e}_2 = -\frac{\mathbf{a} \times \mathbf{n}}{\|\mathbf{a} \times \mathbf{n}\|}, \qquad \mathbf{e}_1 = \mathbf{e}_2 \times \mathbf{a}. \tag{73}$$

We first derive the derivative of $\mathbf{e}_2$. Let $[\mathbf{x}]_\times \in \mathbb{R}^{3\times 3}$ denote the skew-symmetric matrix such that $[\mathbf{x}]_\times \mathbf{y} = \mathbf{x} \times \mathbf{y}$. Let $\mathbf{c} = \mathbf{a} \times \mathbf{n}$, so $\mathbf{e}_2 = -\mathbf{c}/\|\mathbf{c}\|$ with $\|\mathbf{c}\| = \sin\theta_{\text{inc}}$. With $\frac{\partial \mathbf{c}}{\partial \mathbf{a}} = -[\mathbf{n}]_\times$, we have:

$$\frac{\partial \mathbf{e}_2}{\partial \mathbf{a}} = \frac{1}{\sin\theta_{\text{inc}}}\left(\mathbf{I}_3 - \mathbf{e}_2\,\mathbf{e}_2^\top\right)[\mathbf{n}]_\times, \tag{74}$$

where $\mathbf{I}_3 - \mathbf{e}_2\,\mathbf{e}_2^\top$ is the projection onto the plane orthogonal to $\mathbf{e}_2$. An equivalent expansion of this expression is:

$$\frac{\partial \mathbf{e}_2}{\partial \mathbf{a}} = \frac{1}{\sin\theta_{\text{inc}}}[\mathbf{n}]_\times + \frac{1}{\sin^2\theta_{\text{inc}}}\mathbf{e}_2\mathbf{a}^\top + \frac{\cos\theta_{\text{inc}}}{\sin^2\theta_{\text{inc}}}\mathbf{e}_2\mathbf{n}^\top, \tag{75}$$

which can be verified by expanding $(\mathbf{I}_3 - \mathbf{e}_2\mathbf{e}_2^\top)[\mathbf{n}]_\times$. With $\mathbf{e}_1 = \mathbf{e}_2 \times \mathbf{a} = -[\mathbf{a}]_\times \mathbf{e}_2$, we have:

$$\frac{\partial \mathbf{e}_1}{\partial \mathbf{a}} = [\mathbf{e}_2]_\times - [\mathbf{a}]_\times \frac{\partial \mathbf{e}_2}{\partial \mathbf{a}}. \tag{76}$$

2) Derivative of Ray Direction: Combining cone basis derivatives (Eq. 73-Eq. 76) and Eq 52 yields

$$\frac{\partial \mathbf{d}}{\partial \mathbf{a}} = \cos\beta\,\mathbf{I}_3 + \sin\beta\cos\varphi\,\frac{\partial \mathbf{e}_1}{\partial \mathbf{a}} + \sin\beta\sin\varphi\,\frac{\partial \mathbf{e}_2}{\partial \mathbf{a}}. \tag{77}$$

3) Derivative of Denominator: We now derive $\frac{\partial D}{\partial a}$ with D defined in Eq. 53. The denominator D depends on $\mathbf{a}$ through θ_{inc}. Based on the geometry,

$$\cos\theta_{\text{inc}} = -\mathbf{n}^\top\mathbf{a}, \qquad \sin\theta_{\text{inc}} = \sqrt{1 - (\mathbf{n}^\top\mathbf{a})^2} \tag{78}$$

Taking derivatives with respect to $\mathbf{a}$ yields

$$\frac{\partial \cos\theta_{\text{inc}}}{\partial \mathbf{a}} = \frac{\partial(-\mathbf{n}^\top\mathbf{a})}{\partial \mathbf{a}} = -\mathbf{n} \tag{79}$$

For $\sin\theta_{\text{inc}}$, applying the chain rule yields

$$\frac{\partial \sin\theta_{\text{inc}}}{\partial \mathbf{a}} = \frac{\partial}{\partial \mathbf{a}}\sqrt{1 - (\mathbf{n}^\top\mathbf{a})^2} = \frac{\cos\theta_{\text{inc}}}{\sin\theta_{\text{inc}}}\mathbf{n}$$

Now we can compute $\frac{\partial D}{\partial \mathbf{a}}$:

$$\begin{aligned}\frac{\partial D}{\partial \mathbf{a}} &= \frac{\partial \cos\theta_{\text{inc}}}{\partial \mathbf{a}}\cos\beta + \frac{\partial \sin\theta_{\text{inc}}}{\partial \mathbf{a}}\sin\beta\cos\varphi \\ &= \left(-\cos\beta + \frac{\cos\theta_{\text{inc}}\sin\beta\cos\varphi}{\sin\theta_{\text{inc}}}\right)\mathbf{n} \\ &= \frac{-\sin\theta_{\text{inc}}\cos\beta + \cos\theta_{\text{inc}}\sin\beta\cos\varphi}{\sin\theta_{\text{inc}}}\mathbf{n} \end{aligned} \tag{80}$$

Let $N_u = -\sin\theta_{\text{inc}}\cos\beta + \cos\theta_{\text{inc}}\sin\beta\cos\varphi$, then the above expression can be written as

$$\frac{\partial D}{\partial \mathbf{a}} = \frac{N_u}{\sin\theta_{\text{inc}}}\mathbf{n} \tag{81}$$

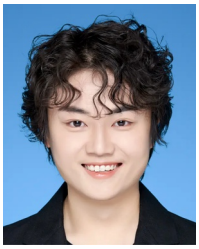
4) Complete Orientation Jacobian: Note that the footprint point is $\mathbf{q} = \mathbf{p} + l^*\mathbf{d}$, where $l^* = h/D$. Since $h = \mathbf{n}^\top(\mathbf{p} - \mathbf{p}_0)$ is independent of $\mathbf{a}$, we have

$$\frac{\partial l^*}{\partial \mathbf{a}} = -\frac{h}{D^2}\frac{\partial D}{\partial \mathbf{a}} \tag{82}$$

The complete orientation Jacobian is then:

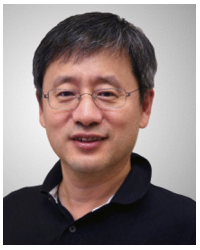
$$\mathbf{J}_a(\beta,\varphi) = \frac{\partial \mathbf{q}}{\partial \mathbf{a}} = l^*\frac{\partial \mathbf{d}}{\partial \mathbf{a}} + \mathbf{d}\frac{\partial l^*}{\partial \mathbf{a}} = \frac{h}{D}\frac{\partial \mathbf{d}}{\partial \mathbf{a}} - \frac{h}{D^2}\mathbf{d}\left(\frac{\partial D}{\partial \mathbf{a}}\right)^\top, \tag{83}$$

where $\frac{\partial D}{\partial \mathbf{a}}$ is given by (81), and $\frac{\partial \mathbf{d}}{\partial \mathbf{a}}$ is given by (77), which completes the proof.



Ziyue Zheng received the B.S. degree in optoelectronic information science and engineering from East China Normal University, Shanghai, China, in 2024. He is currently working toward the M.S. degree with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD, USA. His research interests include optimal control, active perception, and embodied artificial intelligence.

Yongce Liu received the B.Eng. degree in automation from Northeastern University, Shenyang, China, in 2024. He is currently working toward the M.S. degree in control science and engineering with Shanghai Jiao Tong University, Shanghai, China. His research interests include robotics, optimization, and physical artificial intelligence.



Hesheng Wang (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2002, and the M.Phil. and Ph.D. degrees in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, in 2004 and 2007, respectively. He is currently a Distinguished Professor with the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai, China. His current research interests include visual servoing, intelligent robotics, computer vision, and autonomous driving.

Dr. Wang is an Associate Editor of *Robotic Intelligence and Automation* and the *International Journal of Humanoid Robotics*, a Senior Editor of the *IEEE/ASME Transactions on Mechatronics*, and the Editor-in-Chief of *Robot Learning*. He served as an Associate Editor of the *IEEE Transactions on Robotics* from 2015 to 2019 and the *IEEE Transactions on Automation Science and Engineering* from 2021 to 2023. He was the General Chair of IEEE/RSJ IROS 2025, IEEE ROBIO 2022, and IEEE RCAR 2016.

Zhongqiang (Richard) Ren (Member, IEEE) received the dual B.E. degree in mechatronics from Tongji University, Shanghai, China, and FH Aachen University of Applied Sciences, Aachen, Germany, in 2015, and the M.S. degree in mechanical engineering and the Ph.D. degree in mechanical engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2017 and 2022, respectively. He is currently an Associate Professor with the Global College, Shanghai Jiao Tong University, Shanghai, China.

His research interests include robotics, path and motion planning, and multi-robot systems. He is an Associate Editor for IEEE Transactions on Robotics, Autonomous Robots, ICRA and IROS.