

From Distances to Trajectories: Real-Time Signed Distance Function Mapping and Distance-Accelerated Motion Planning for UAVs

Jason Stanley^{1*}, Zhirui Dai^{1*}, Qihao Qian¹, Tzu-Chin Ho¹,
Tianxing Fan¹, Siddharth Saha², Christopher Barngrover²,
Ki Myung Brian Lee¹, Nikolay Atanasov¹

¹Electrical and Computer Engineering Department, UC San Diego, 9500 Gilman Dr, La Jolla, 92093, CA, USA.

²Shield AI, 600 W Broadway Suite #600, San Diego, 92101, CA, USA.

*Corresponding author(s). E-mail(s): jtstanle@ucsd.edu; zh dai@ucsd.edu;
Contributing authors: q2qian@ucsd.edu; tzh005@ucsd.edu; t2fan@ucsd.edu;
kmblee@ucsd.edu; natanasov@ucsd.edu;

[†]These authors contributed equally to this work.

Abstract

Autonomous flight in cluttered environments requires a robot to build a geometric map of its surroundings and plan safe, dynamically feasible trajectories, all onboard and in real time. Conventional approaches treat mapping and planning as separate stages and often rely on binary occupancy for collision checking. We argue that these two stages should be co-designed around a single representation: a signed distance function (SDF). By encoding distance to the nearest obstacle, an SDF provides richer information for planning and trajectory optimization than occupancy alone. We develop an Octree Residual Network (OREN) that pairs an explicit octree prior with an implicit neural residual to reconstruct SDFs online from point cloud observations with the efficiency of volumetric methods and the accuracy and differentiability of neural methods. In tandem, we develop Bubble*, a search-based planner that exploits the distance information to grow maximal collision-free balls, which we call bubbles, with formal guarantees of termination, completeness, and failure detection. Planning over a graph of bubbles significantly reduces collision checks compared to a grid-based A* search and returns a bubble sequence that forms a safe corridor for trajectory optimization. We demonstrate the integrated OREN-Bubble* approach onboard a quadrotor, navigating unseen indoor environments in real time under tight compute constraints. OREN improves SDF estimation by **22%** compared to baselines, while Bubble* finds trajectories spanning $\approx$ **90** m through a cluttered environment in **1–3** sec., whereas baselines take up to **10** sec. in the same environment.

Keywords: Signed Distance Function, Implicit Representation, Motion Planning, Unmanned Aerial Vehicle, Autonomous Flight

1 Introduction

Autonomous flight in unknown environments requires an unmanned aerial vehicle (UAV) to build a map of its surroundings and to plan safe, dynamically feasible motion through it, all onboard and in real time from streaming sensor data. This capability underpins many applications of aerial robotics, including inspection, delivery, and search and rescue, where UAVs must navigate safely and efficiently. Conventionally, mapping and planning are treated as two separate stages connected by a binary occupancy map. The planner queries the map to test for collisions at each robot configuration, disregarding distance and gradient information that a richer distance-based representation could provide. We argue that the two stages can instead be co-designed around a signed distance function (SDF) representation, exploiting the distance information to reduce the collision checking operations, accelerate the planning, and provide constraints for trajectory optimization.

Accurate environment representations are essential across the robot autonomy stack, including in simultaneous localization and mapping (Ortiz et al., 2022; Pan et al., 2024; Tian et al., 2025), navigation (Long et al., 2026; Oleynikova et al., 2017), and manipulation (Brunner et al., 2022; Li, Chi, et al., 2024; Li, Zhang, et al., 2024; P. Liu et al., 2022). SDF representations are particularly well suited to serve both mapping and planning. Given a query point, an SDF returns the signed distance to the nearest surface, with the sign indicating whether the query lies in free (positive) or occupied (negative) space. It captures obstacle surfaces implicitly as its zero-level set (Park et al., 2019), while simultaneously providing clearance to the nearest obstacle. To be useful in aerial autonomy, an SDF must be built and queried quickly from streaming observations, keeping a small memory footprint and remaining accurate under a tight onboard computational budget. However, existing SDF methods rarely meet all these requirements at once.

In this paper, we present a unified approach for mapping and planning for aerial robots, built around non-truncated SDF (Fig. 1). Our method reconstructs SDF from point cloud measurements

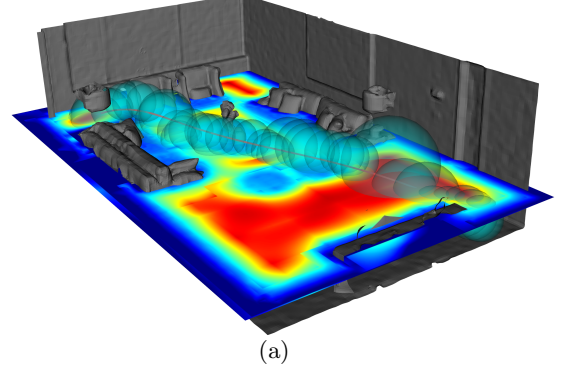


Fig. 1: We consider integrated mapping and planning for autonomous UAV flight. We develop OREN to reconstruct accurate Euclidean SDF online from streaming point clouds and Bubble* to plan safe flight corridors exploiting the distance information. (a) The color map shows distance to the nearest surface (large to small), and the planned safe corridor (composed of cyan balls). (b) OREN and Bubble* running onboard a quadrotor equipped with a Jetson Orin NX.

onboard a UAV and enables efficient distance-aware motion planning and trajectory optimization with safety and dynamic-feasibility guarantees. We extend our prior conference paper, Dai, Qian, et al. (2026), which introduced an Octree RESidual Network (OREN) for SDF mapping, combining an explicit prior from octree interpolation with implicit neural features decoded into a correction of the prior. To construct the prior, we use an octree with distance and gradient estimates stored at the octant vertices and gradient-augmented interpolation to compute SDF at arbitrary query positions. We correct the prior with a neural network residual, which recovers fine details of the observed surface from implicit features.

Given an SDF model of the environment, the second challenge is turning it into safe, dynamically feasible UAV trajectories. Commonly, this is done by first planning a collision-free path, e.g.,

via A* or RRT (Hart et al., 1968; Karaman & Frazzoli, 2011), then constructing a safe corridor of overlapping convex regions around the path (S. Liu, Watterson, et al., 2017), and finally optimizing a dynamically feasible trajectory inside the corridor (Z. Wang et al., 2022). Bubble Planner (Ren et al., 2022), a representative work, finds a path via search, grows a receding corridor of spheres around it, and optimizes a trajectory through the corridor. We observe that with SDF, the path search and the corridor construction need not be separate stages because the distance at any point already defines a maximal collision-free ball around it, which we call a *bubble*. We develop Bubble*, a motion planning algorithm that constructs a graph of bubbles from SDF queries to plan a safe bubble corridor. We establish formal guarantees of termination, completeness, and failure detection under mild clearance assumptions, then optimize a dynamically feasible trajectory within the corridor using MINCO (Z. Wang et al., 2022), a minimum-control-effort polynomial parameterization. Because Bubble* reads signed distances directly from OREN, mapping and planning share a single representation, requiring far fewer collision checks than grid-based A*. In our experiments, OREN improves SDF estimation by 22% over baseline methods and runs efficiently enough for real-time deployment onboard a UAV, while Bubble* finds trajectories ≈ 90 m long through a cluttered environment in 1–3 sec., versus up to 10 sec. for existing methods.

In summary, our work makes the following contributions.

- We develop OREN, a mapping approach that uses an explicit octree prior and implicit neural correction to reconstruct accurate, differentiable, non-truncated SDF in real time.
- We propose Bubble*, a search-based planner that exploits an SDF map to construct a graph of collision-free spheres (bubbles), unifying path search and safe-corridor construction into a single algorithm. We provide formal guarantees of termination, completeness, and failure detection, and perform trajectory optimization within the bubble corridor to produce dynamically feasible UAV trajectories.
- We compare our methods to mapping and planning baselines and demonstrate the complete

approach in real-world autonomous quadrotor flight in unknown environments.

2 Related Work

This section reviews existing methods for SDF reconstruction and integrated mapping and planning for aerial robots.

2.1 SDF Reconstruction

Methods for learning SDFs fall into three broad groups: volumetric, Gaussian Process (GP), and neural network.

Volumetric methods (Curless & Levoy, 1996; Han et al., 2019; Kähler et al., 2015; Millane et al., 2024; Newcombe et al., 2011; Oleynikova et al., 2017; Pan et al., 2022) fuse observations into a regular grid using voxel hashing for efficient updates and queries. Voxblox (Oleynikova et al., 2017) builds a TSDF layer from projective distance, then propagates it to an SDF layer via breadth-first search (BFS). Both steps introduce errors that subsequent works (Han et al., 2019; Pan et al., 2022) reduce by using non-projective distance and replacing BFS path length with distance to the nearest oriented surface point. Nonetheless, these representations are discrete, non-differentiable, and hard to scale to large scenes. In comparison, OREN learns a continuous, differentiable SDF in real time, with low memory usage that scales to large scenes. OREN first estimates an SDF prior from explicit discrete priors stored in an octree, then uses implicit neural features to predict residuals that correct the prior, forming a compact, differentiable representation of continuous SDF.

Gaussian Process (GP) methods (Dai, Fan, et al., 2026; B. Lee et al., 2019; Wu et al., 2021, 2025) learn continuous fields supporting gradient computation and uncertainty quantification. GP implicit surface (GPIS) (B. Lee et al., 2019) iteratively estimates oriented surface points and regresses SDF online, learning the surface implicitly as the SDF’s zero-level set. GPIS is accurate near the surface but fails to extrapolate away from it, since training data comes only from near-surface points and distant queries fall back to the zero-mean prior. Based on the connection between the heat equation and the unsigned distance function (UDF) (Varadhan, 1967), Log-GPIS (Wu

et al., 2021) learns globally-generalized unsigned distances in log space. VDB-GPDF (Wu et al., 2025) extends this log-GP technique to jointly learn surface estimation and UDF, using OpenVDB (Museth et al., 2025) for memory efficiency. However, both methods omit the sign and struggle to scale to large scenes on resource-constrained platforms due to the cubic complexity of matrix inversion during training. In comparison, OREN computes the prior via $O(1)$ trilinear interpolation in the octree and the residual via matrix multiplication, roughly $O(n^2)$ for n hidden dimensions, while using less memory than GP.

Neural network methods are attractive for their native speed on GPUs. DeepSDF (Park et al., 2019) showed that neural networks can learn continuous implicit SDFs, inspiring many follow-up works. iSDF (Ortiz et al., 2022) learns SDF incrementally with Eikonal regularization. NeuS (P. Wang et al., 2021) jointly learns SDF and radiance fields (Mildenhall et al., 2020). Gropp et al. (2020); Takikawa et al. (2021) refine the network architecture and loss designs. These methods learn accurate SDF near the surface, which suffices for surface reconstruction, but rarely far from it. HotSpot (Z. Wang et al., 2025) learns non-truncated SDF but is only verified at the object level and requires extensive data and training.

Recent work develops promising hybrid models that combine an explicit geometric structure with implicit neural features. PIN-SLAM (Pan et al., 2024) stores neural features in near-surface voxels and decodes the SDF from nearby features, while H₂-Mapping (Jiang et al., 2023) combines an octree-based SDF prior with a neural residual. However, both methods learn only truncated SDF. HIO-SDF (Vasilopoulos et al., 2024) removes truncation by training on global priors from Voxfield (Pan et al., 2022), but inherits the volumetric method’s limited accuracy, and over-smoothes as the scene grows due to fixed network capacity.

In contrast, our method, OREN, builds an extendable semi-sparse octree that stores SDF values and gradients to efficiently capture the SDF of the whole space as the environment grows. Through gradient-augmented interpolation, OREN produces more accurate SDF priors, leaving more capacity for the subsequent neural network to recover surface detail from implicit features. Furthermore, the loss function is designed to encourage the network to learn accurate SDF far

from the surface, which is critical for planning, and enables faster convergence than existing methods, allowing OREN to run in real time on a UAV.

2.2 Integrated Mapping and Planning for Aerial Robots

Motion planning for quadrotors involves local trajectory optimization and, most often, global path planning, both relying on *differential flatness* (Faessler et al., 2018; Mellinger & Kumar, 2011): the property that a dynamical system’s full states and controls can be computed analytically from a reduced state, the *flat output*, and its derivatives up to the relative degree.

Quadrotor dynamics are differentially flat, with 3D position and yaw as flat outputs of relative degree four: all quadrotor states (e.g., orientation) and controls (e.g., thrust) can be computed analytically from up to the fourth derivatives of position and yaw (Mellinger & Kumar, 2011), even with drag (Faessler et al., 2018). Thus, planning methods need only produce a position trajectory that is at least four times differentiable, and optionally yaw, without considering the full dynamics.

Mapless local trajectory optimization methods plan directly from observations to avoid obstacles, trading global optimality for fast computation. Song et al. (2021) achieve high-speed drone racing by using reinforcement learning to generate trajectories from a single image, treating planning as a black box. Jacquet et al. (2025) take a grey-box approach, learning a network that converts a depth image to a local SDF for local nonlinear model-predictive control (NMPC). All these methods share a limited field of view, risking dead-end failures. Ji et al. (2021) partially mitigate this by efficiently storing sensor history. Since they do not rely on a map, these methods often lack formal guarantees of optimality, leading to local optima.

Global planning methods store obstacle information in a map and plan safe paths against it. Global motion planning is dominated by search-based algorithms, e.g., A* (Hart et al., 1968), and sampling-based ones, e.g., RRT (LaValle & Kuffner, 2001) and its asymptotically optimal variant RRT* (Karaman & Frazzoli, 2011). Sampling-based methods are widely used, e.g., in OMPL (Sucan et al., 2012), with quadrotor examples by Funk et al. (2021); Gao et al. (2019).

Because they rely on random sampling, their completeness and optimality guarantees are only probabilistic or asymptotic. In contrast, search-based methods operate on regular grids or state lattices (Likhachev & Ferguson, 2009), guaranteeing deterministic (resolution-)completeness or optimality, albeit confined by the chosen resolution. Dharmadhikari et al. (2020); S. Liu, Watterson, et al. (2017) use search-based methods for quadrotor navigation. Our approach mitigates this cost-versus-speed trade-off in resolution choice by building safe bubbles from clearance information, obviating collision checking within them.

Both search-based and sampling-based planners can be either *geometric* or *kinodynamic*. Geometric planners produce a sequence of waypoints, assuming any pair of waypoints is achievable, whereas kinodynamic planners account for kinematic or dynamic constraints by requiring a valid motion between waypoints. To plan dynamically feasible quadrotor trajectories, one can either 1) use a geometric planner followed by trajectory optimization to track the resulting waypoints, or 2) use a single kinodynamic planner directly.

Kinodynamic planning may appear simpler, as it uses a single planner. LaValle and Kuffner (2001) originally designed RRT for kinodynamic planning. However, Richter et al. (2016) show that for real quadrotors, full kinodynamic planning performs worse than combining geometric planning with trajectory optimization, due to its additional computation. Allen and Pavone (2019) speed up real-time sampling-based kinodynamic planning by introducing a learned reachability classifier between states. Meanwhile, Mueller et al. (2015) present efficient methods for computing *motion primitives*: a library of trajectories used to connect states during planning. S. Liu, Atanasov, et al. (2017) use such motion primitives in a search-based kinodynamic planner, showing they discretize the state space into a lattice suitable for efficient search. Zhou et al. (2019) improve search-based planning with continuous optimization, and Ryll et al. (2019) present a receding-horizon variant for unknown environments. To handle state uncertainty alongside dynamic feasibility, Agha-Mohammadi et al. (2011); Majumdar and Tedrake (2017) plan between distributions rather than states, composing feedback controllers to reach one distribution from another. Our approach is

most similar to Richter et al. (2016), combining a geometric planner with subsequent trajectory optimization. However, our geometric planner composes collision-free regions, leaving room for dynamic feasibility while ensuring collision avoidance.

Safe corridor methods, such as Deits and Tedrake (2015); K.M.B. Lee et al. (2024); S. Liu, Watterson, et al. (2017); Ren et al. (2022); Tordeillas and How (2022); Z. Wang et al. (2022), similarly find a sequence of collision-free regions (safe corridors). These methods are popular for quadrotor navigation because they greatly accelerate trajectory optimization, making them suitable for onboard and even GPS-denied navigation (Mohta et al., 2018). Most safe corridor methods first plan a geometric path, then expand convex regions around the waypoints. S. Liu, Watterson, et al. (2017) first introduce safe corridors: overlapping collision-free polyhedra expanded around a path found via jump point search (Harabor & Grastien, 2011) on an OctoMap (Hornung et al., 2013), with an optimal continuous trajectory found via quadratic programming (QP). Z. Wang et al. (2022) accelerate optimization by introducing a closed-form trajectory parameterization enabling unconstrained optimization within safe corridors. Subsequent approaches (Ren et al., 2022, 2025) speed computation using lighter local maps instead of an OctoMap. In particular, Ren et al. (2022) build an efficient KD-Tree for proximity queries and use clearance information to build spherical bubbles along a geometric path. Our approach also builds spherical bubbles from clearance but directly searches for a sequence of them. In contrast to inflating corridors around a pre-computed path, few recent methods directly search for a sequence of corridors. Graph-of-convex-sets formulations (Marcucci et al., 2024, 2023) jointly optimize corridor choice and the trajectory within them via mixed-integer optimization. However, these methods require a known corridor set, and mixed-integer optimization can be expensive. In aerial robotics, Gao et al. (2019) search for a sequence of spherical corridors using a modified RRT*, enabling high-speed LiDAR-based navigation. Similarly, Funk et al. (2021) find convex regions via RRT search with collision-checking against a multi-resolution SDF grid built from visual sensors (Laina et al., 2025).

Our prior work, [K.M.B. Lee et al. \(2024\)](#), used a sampling-based formulation that first covers free space with a graph of bubbles and searches it via Dijkstra’s shortest-path algorithm. Here, we use a search-based formulation that improves efficiency by interleaving the graph construction and the search and by biasing the search towards the goal via a heuristic function, similarly to A^* .

3 Problem Statement

Consider a 3D environment $\mathcal{R} \subset \mathbb{R}^3$ with obstacles represented as a subset $\Omega \subset \mathcal{R}$. The SDF $d : \mathbb{R}^3 \rightarrow \mathbb{R}$ of Ω is defined as the shortest distance from any point $\mathbf{v} \in \mathbb{R}^3$ to the obstacle surface $\partial\Omega$, with a sign indicating whether $\mathbf{v}$ is inside or outside of Ω :

$$d(\mathbf{v}) = \begin{cases} \min_{\mathbf{y} \in \partial\Omega} \|\mathbf{v} - \mathbf{y}\|_2, & \mathbf{v} \notin \Omega, \\ -\min_{\mathbf{y} \in \partial\Omega} \|\mathbf{v} - \mathbf{y}\|_2, & \mathbf{v} \in \Omega. \end{cases} \quad (1)$$

The SDF satisfies two key properties: 1) the obstacle surface is encoded as the zero-level set, $d(\mathbf{v}) = 0$, $\forall \mathbf{v} \in \partial\Omega$; and 2) the gradient of $d(\mathbf{v})$ is the unit vector pointing away from the nearest surface point and satisfies an Eikonal equation ([Ortiz et al., 2022](#)):

$$\nabla d(\mathbf{v}) = \frac{\mathbf{v} - \mathbf{v}_*}{d(\mathbf{v})}, \quad \|\nabla d(\mathbf{v})\|_2 = 1, \text{ a.e.}, \quad (2)$$

where $\mathbf{v}_* \in \arg \min_{\mathbf{y} \in \partial\Omega} \|\mathbf{v} - \mathbf{y}\|_2$.

We consider a quadrotor robot, equipped with a range sensor (e.g., LiDAR or depth camera), operating in the environment. Given a stream of point clouds $\mathcal{P}_t$, our objective is to: 1) obtain an estimate $\hat{d} : \mathbb{R}^3 \rightarrow \mathbb{R}$ of the SDF of Ω and 2) plan a safe dynamically feasible trajectory for the robot. Let $\mathbf{x} = (\mathbf{p}, \boldsymbol{\nu}, R, \boldsymbol{\omega}) \in \mathcal{X} := \mathbb{R}^3 \times \mathbb{R}^3 \times SO(3) \times \mathbb{R}^3$ be the quadrotor state, consisting of its position $\mathbf{p}$ and velocity $\boldsymbol{\nu}$ in the inertial frame, orientation $R \in SO(3)$, and body angular velocity $\boldsymbol{\omega}$. Given control input $\mathbf{u} = (F, \boldsymbol{\tau}) \in \mathbb{R} \times \mathbb{R}^3$, including the collective motor thrust F and body torque $\boldsymbol{\tau}$, the quadrotor dynamics are:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{G}(\mathbf{x})\mathbf{u} = \begin{cases} \dot{\mathbf{p}} = \boldsymbol{\nu}, \\ \dot{\mathbf{p}} = -g\mathbf{e}_3 + \frac{1}{m}RF\mathbf{e}_3, \\ \dot{R} = R\hat{\boldsymbol{\omega}}, \\ \dot{\boldsymbol{\omega}} = J^{-1}(\boldsymbol{\tau} - \hat{\boldsymbol{\omega}}J\boldsymbol{\omega}), \end{cases} \quad (3)$$

where m is the mass, J is the inertia matrix, $\mathbf{e}_3 = (0, 0, 1)^\top$, g is the gravitational acceleration, and $\hat{\boldsymbol{\omega}}$ is a skew-symmetric matrix formed from $\boldsymbol{\omega}$.

The quadrotor dynamics (3) are differentially flat ([Mellinger & Kumar, 2011](#); [Z. Wang et al., 2022](#)); there exists a flat output $\mathbf{z}$ such that the state and control can be expressed as algebraic functions of $\mathbf{z}$ and k of its derivatives,

$$\mathbf{x} = \Psi_{\mathbf{x}}(\mathbf{z}, \dot{\mathbf{z}}, \dots, \mathbf{z}^{(k-1)}), \quad \mathbf{u} = \Psi_{\mathbf{u}}(\mathbf{z}, \dot{\mathbf{z}}, \dots, \mathbf{z}^{(k)}).$$

For a quadrotor, the flat output $\mathbf{z} = (\mathbf{p}, \psi)$ consists of its position $\mathbf{p}$ and yaw angle ψ . The state $\mathbf{x}$ and control $\mathbf{u}$ can be recovered from derivatives of the flat output up to order $k = 4$ with $\Psi_{\mathbf{x}}$ and $\Psi_{\mathbf{u}}$ known in closed form ([Mellinger & Kumar, 2011](#)). The yaw ψ can be determined separately, e.g., along the velocity direction, so motion planning reduces to designing a continuous position trajectory $\mathbf{p} : [0, T] \rightarrow \mathbb{R}^3$ from a start $\mathbf{p}_s$ to a goal $\mathbf{p}_g$ that remains collision-free and respects the vehicle dynamics. We express this as a trajectory optimization problem:

$$\begin{aligned} \min_{\mathbf{p}(\cdot), T} \quad & \int_0^T \|\mathbf{p}^{(4)}(t)\|_2^2 dt + \rho T \\ \text{s.t.} \quad & \mathbf{p}(0) = \mathbf{p}_s, \quad \mathbf{p}(T) = \mathbf{p}_g, \\ & \hat{d}(\mathbf{p}(t)) \geq r, \quad \forall t \in [0, T], \\ & \|\dot{\mathbf{p}}(t)\|_2 \leq v_{\max}, \quad \forall t \in [0, T], \\ & \|\ddot{\mathbf{p}}(t)\|_2 \leq a_{\max}, \quad \forall t \in [0, T], \end{aligned} \quad (4)$$

where the objective minimizes the trajectory snap $\|\mathbf{p}^{(4)}\|_2^2$ with $\rho > 0$ trading off smoothness against duration. In the constraints, $r > 0$ is a safety radius accounting for the robot size and errors in the SDF estimate $\hat{d}$, while $v_{\max}$ and $a_{\max}$ are velocity and acceleration bounds.

In summary, we consider an integrated mapping and planning problem. Given point cloud measurements, we estimate the SDF $\hat{d}$ and plan a dynamically feasible trajectory $\mathbf{p}(t)$ from the current robot position $\mathbf{p}_s$ to a desired goal $\mathbf{p}_g$ that maintains clearance $\hat{d}(\mathbf{p}(t)) \geq r$ for all $t \in [0, T]$.

4 Octree Residual Network for SDF Mapping

We first focus on reconstructing the SDF in (1) from streaming point cloud measurements. We

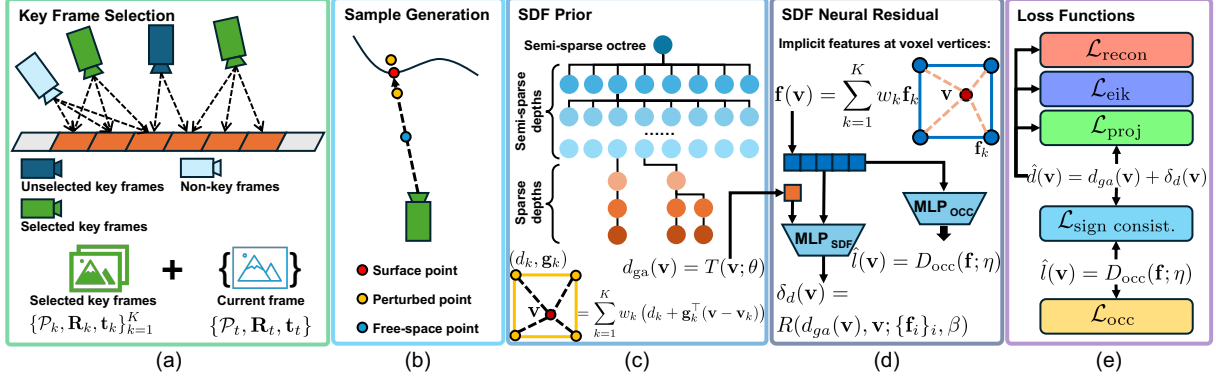


Fig. 2: Overview of OREN: a) We keep key frames with small overlap and those that maximize the surface coverage for training; b) with the selected key frames and the current frame, we generate three types of samples: **surface** points, **perturbed** points around the surface, and **free-space** points; c) to predict SDF, we first obtain an SDF prior $d_{ga}(\mathbf{v})$ with gradient-augmented interpolation in a semi-sparse octree, where each octant vertex has estimated SDF value and gradient; d) we also obtain an implicit neural feature for $\mathbf{v}$ by trilinear interpolation of implicit features stored at the voxel’s vertices, which is fed into an MLP decoder to obtain an SDF residual correction $\delta_d(\mathbf{v})$, and another MLP decoder to predict occupancy; e) the SDF prior $d_{ga}(\mathbf{v})$ and the SDF residual $\delta_d(\mathbf{v})$ are combined as the final SDF prediction $\hat{d}(\mathbf{v}) = d_{ga}(\mathbf{v}) + \delta_d(\mathbf{v})$, and the parameters are trained with five loss functions: **reconstruction** loss, **Eikonal** loss, **projection** loss, **occupancy** loss and **sign consistency** loss.

develop OREN, a hybrid model that combines an explicit octree prior with an implicit neural correction. We present an overview of OREN in Fig. 2. An octree data structure stores explicit SDF and gradient estimates, from which a coarse SDF prior is obtained by gradient-augmented interpolation (Sec. 4.1). To recover the geometric details that the octree resolution cannot capture, implicit neural features are stored at the octant vertices and are decoded by an MLP into a residual correction of the prior (Sec. 4.2). We also decode the implicit features with a second MLP to predict occupancy, which is used to supervise the SDF sign during online training and improve the robustness to sensor noise. The resulting non-truncated SDF estimate $\hat{d}$ can subsequently be used in the UAV trajectory optimization problem in (4).

4.1 SDF Prior From Octree Interpolation

OREN computes the SDF prior through interpolation of SDF values and gradients stored in an octree with *sparse* and *semi-sparse* layers. This allows efficient storage compared to a dense, regular grid. Of the N layers of our octree, the first M layers are designed to be *semi-sparse* in the sense that all siblings of an occupied child octant

are created regardless of occupancy. The remaining $N - M$ layers are *sparse*, where only child octants containing surface points are populated. This is illustrated in Fig. 2c. Each octant vertex $\mathbf{v}_k$, with $k \in \{1, \dots, 8\}$, stores learnable estimates $d_k \in \mathbb{R}$ and $\mathbf{g}_k \in \mathbb{R}^3$ of the SDF $d(\mathbf{v}_k)$ and its gradient $\nabla d(\mathbf{v}_k)$, and vertices are shared across neighboring octants at different tree depths to save memory. The semi-sparse layers cost extra memory but yield a more accurate prior, especially for query positions away from the surface, because creating sibling octants places vertices closer to an arbitrary query $\mathbf{v}$ and reduces the interpolation discontinuities at octant boundaries that arise in a purely sparse octree (Dai, Qian, et al., 2026). Using a semi-sparse octree of resolution ℓ , for a query near the surface we can locate an octant no larger than $\ell \times 2^{N-M}$, while for distant queries a large empty octant suffices for an accurate prior under the gradient-augmented interpolation described next.

To produce an accurate enough prior for the residual network to only capture fine details, we use gradient-augmented trilinear interpolation. At the smallest octant containing a query position $\mathbf{v}$, we first extrapolate from each vertex $\mathbf{v}_k$:

$$d_k(\mathbf{v}) = d_k + \mathbf{g}_k^\top (\mathbf{v} - \mathbf{v}_k), \quad k \in \{1, \dots, 8\}, \quad (5)$$

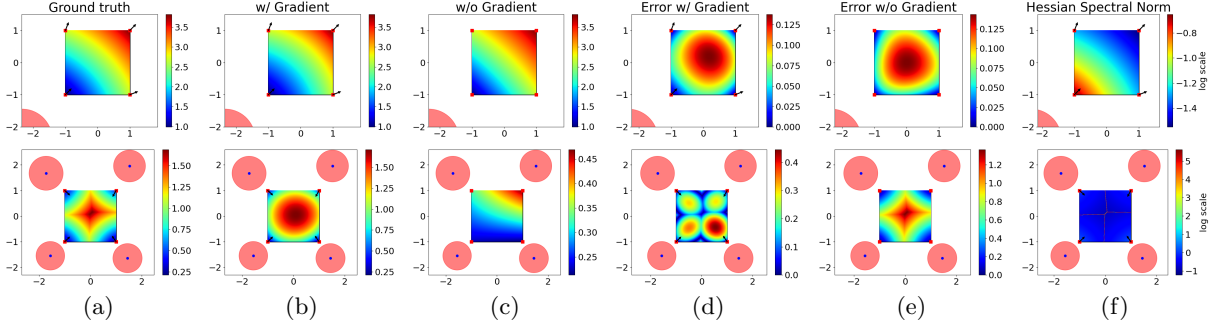


Fig. 3: 2D visualization of interpolation with and without gradient augmentation for one (red region, top row) and four obstacles (red regions, bottom row). Gradient-augmented interpolation produces a better SDF prior (b) with smaller error (d). Empirically, positions where the SDF gradient is not well defined (large Hessian spectral norm), as shown in (f), have small interpolation error with gradient augmentation as shown in (d).

and combine the extrapolations into the gradient-augmented (ga) interpolation

$$d_{ga}(\mathbf{v}) = \frac{1}{\gamma} \sum_{k=1}^8 w_k d_k(\mathbf{v}), \quad \gamma = \sum_{k=1}^8 w_k, \quad (6)$$

where $w_k = 1/|\text{diag}(\mathbf{v} - \mathbf{v}_k)|$ is the trilinear interpolation weight. Unlike regular trilinear interpolation, which ignores the stored gradients, the gradient-augmented form admits a tighter error upper bound; we present the derivation in Dai, Qian, et al. (2026).

Fig. 3 illustrates the benefit of gradient-augmented interpolation in 2D scenes with one and four obstacles. Each row shows the ground-truth SDF (a), the interpolation results with (b) and without (c) gradient augmentation, the corresponding errors (d, e), and the Hessian spectral norm of the SDF (f). Gradient-augmented interpolation produces smaller errors, as seen in Fig. 3d and 3e, and the improvement grows with the number of obstacles. Although the SDF gradient is not well defined on the medial axes, where the Hessian spectral norm is large (Fig. 3f), gradient-augmented interpolation still attains small error there in practice (Fig. 3d).

The prior $d_{ga}(\mathbf{v})$ is thus computed from a semi-sparse octree with learnable SDF and gradient estimates d_k and $\mathbf{g}_k$ at each vertex, which are optimized jointly with the residual network. In the experiments, we use the octree configuration of Dai, Qian, et al. (2026), with $N = 8$ total octree layers with $M = 5$ semi-sparse layers, and a resolution of $\ell = 10$ cm.

4.2 SDF Residual From Implicit Feature Decoding

The accuracy of the SDF prior is limited by the octree resolution, so it lacks geometric detail. To achieve high fidelity, we learn a residual correction with a neural network $R(d_{ga}(\mathbf{v}), \mathbf{v}; \{\mathbf{f}_k\}_k, \beta)$ that composes octree feature interpolation $\mathbf{f}(\mathbf{v}) = \sum_k w_k \mathbf{f}_k$, using implicit neural features $\mathbf{f}_k \in \mathbb{R}^F$ stored at the octree vertices, with an MLP decoder $D_{\text{SDF}}(d, \mathbf{f}(\mathbf{v}); \beta)$. Each octant vertex is assigned a feature $\mathbf{f}_k$, initialized to zero and optimized together with the decoder weights β . Octree expansion automatically allocates more features to the near-surface regions as smaller octants are created, enabling continual learning as the sensor moves. As shown in Fig. 2d, for a query point $\mathbf{v}$ we locate the leaf octant containing it, interpolate the feature $\mathbf{f}(\mathbf{v}) = \sum_{k=1}^8 w_k \mathbf{f}_k$ with the same weights w_k as in (6), and decode the SDF residual $\delta_d(\mathbf{v}) = D_{\text{SDF}}(d_{ga}(\mathbf{v}), \mathbf{f}(\mathbf{v}); \beta)$. The final SDF prediction combines the prior and the residual,

$$\hat{d}(\mathbf{v}) = d_{ga}(\mathbf{v}) + \delta_d(\mathbf{v}). \quad (7)$$

In our experiments $F = 3$ and the MLP has two 32-dimensional hidden layers with LeakyReLU activations.

Extending Dai, Qian, et al. (2026), we add a second decoder MLP $D_{\text{occ}}(\mathbf{f}(\mathbf{v}); \eta)$ to predict the occupancy log-odds $\hat{l}(\mathbf{v}) \in \mathbb{R}$, a continuous value that can be converted to an occupancy probability as $\sigma(\hat{l}(\mathbf{v}))$, where $\sigma(l) := (1 + e^{-l})^{-1}$ is the sigmoid function. Since $\sigma(l) > \frac{1}{2}$ if and only if $l > 0$, a positive prediction $\hat{l}(\mathbf{v}) > 0$ indicates

$\mathbf{v} \in \Omega$, and $\hat{l}(\mathbf{v}) < 0$ indicates $\mathbf{v} \notin \Omega$. Unlike the SDF branch, which decodes a residual correction to the octree prior, the occupancy decoder predicts $\hat{l}(\mathbf{v})$ directly from the interpolated features $\mathbf{f}(\mathbf{v})$, without a prior. The occupancy prediction $\hat{l}(\mathbf{v})$ is used to supervise the SDF sign as a regularization term, which significantly improves the robustness to noise. The occupancy labels are obtained by ray-casting the depth measurements. Samples along the ray between the sensor and the measured surface, including samples perturbed to lie in front of the surface, are labeled free ($o = 0$), while samples perturbed to lie behind the surface are labeled occupied ($o = 1$). The sample at the measured surface receives $o = 0.5$, which places the decision boundary $\sigma(\hat{l}) = \frac{1}{2}$ on the surface, aligned with the zero level-set of the SDF.

We jointly train the octree parameters $\theta = \{d_k, \mathbf{g}_k, \mathbf{f}_k\}_k$ and the decoder weights β and η online following Dai, Qian, et al. (2026), with the extension of adding loss terms $\mathcal{L}_{\text{occ}}$ and $\mathcal{L}_{\text{sign consistency}}$ for learning occupancy and enforcing its consistency with the predicted SDF sign:

$$\mathcal{L}_{\text{occ}}(\hat{l}, o) = \text{BCE}(\sigma(\hat{l}), o), \quad (8)$$

$$\mathcal{L}_{\text{sign consistency}}(\hat{l}, \hat{d}) = \mathbb{1}_{|\hat{l}| > \tau} \max(0, s\hat{d}), \quad (9)$$

where $\text{BCE}(\cdot, \cdot)$ is the binary cross-entropy loss, $o \in \{0, 1\}$ is the ground truth of occupancy (0: free, 1: occupied), $s = \text{sign}(\hat{l})$, and $\tau > 0$ is a confidence margin such that only occupancy predictions with enough confidence are used. In our experiments, $\tau = 3$, corresponding to $\sigma(\tau) \approx 0.95$, and both loss terms enter the total training loss with unit weight.

Fig. 2 summarizes OREN. A compact set of key frames is maintained so that it covers the observed surface with little overlap between adjacent frames (Fig. 2a). From these key frames and the current frame, we generate a dataset of points that are: 1) on the surface, 2) perturbed, and 3) in the free-space (Fig. 2b). With the generated dataset, the model is optimized with a combination of reconstruction, Eikonal, projection, sign consistency, and occupancy losses (Fig. 2e). We refer the reader to Dai, Qian, et al. (2026) for the key-frame criterion, the sampling scheme, and the definitions and hyperparameters of the remaining losses.

The result is a continuously updated implicit map that returns accurate, non-truncated distance and gradient at any query point in the explored workspace. Its continuous clearance and Eikonal regularity (2) are precisely the properties Bubble* exploits next, turning signed-distance queries directly into safe, dynamically feasible flight corridors.

5 Distance-Accelerated Motion Planning

In this section, we consider motion planning and trajectory optimization given an SDF representation of the environment. In order to solve (4), we split the problem into two parts: first finding a sequence of bubbles to the goal via global search to minimize path length, followed by local trajectory optimization within the bubble corridor to minimize the objective in (4). Since an SDF representation provides information not only about occupancy but also about the distance to the nearest occupied space, it can be used to accelerate motion planning by reducing the number of collision checks within large free-space regions. We refer to the ball of free space indicated by an SDF query as a *bubble* and develop a new search-based motion planning algorithm, Bubble*, to plan a sequence of bubbles from the start to the goal. Bubble* constructs a graph of nodes on a grid, each associated with a bubble, and edges that connect to the bubble boundaries (Sec. 5.1). Using a heuristic similar to A*, Bubble* expands the most promising node at each step by adding the boundary nodes as successors (Sec. 5.2). We prove completeness of Bubble* under a mild clearance assumption (Sec. 5.3). The overlapping bubbles along the recovered path form a safe corridor that we use for trajectory optimization (Sec. 5.4).

5.1 Bubble Graph Construction

For planning, we discretize the 3D environment of Sec. 3 using grid resolution $\Delta > 0$ and define the set of grid nodes $\mathcal{V} := \Delta\mathbb{Z}^3 \cap \mathcal{R}$, where $\mathbb{Z}^3$ is the integer lattice in 3D. For each node $\mathbf{p} \in \mathcal{V}$, we define the associated grid cell $C_\Delta(\mathbf{p}) := \mathbf{p} + [-\frac{\Delta}{2}, \frac{\Delta}{2}]^3$, and the set of all grid cells $\mathcal{C}_\Delta := \{C_\Delta(\mathbf{p}) \mid \mathbf{p} \in \mathcal{V}\}$. We denote by $\mathcal{O} := \{C_\Delta(\mathbf{p}) \in \mathcal{C}_\Delta \mid C_\Delta(\mathbf{p}) \cap \Omega \neq \emptyset\}$ the occupied cells, i.e., those whose region intersects an obstacle, and by

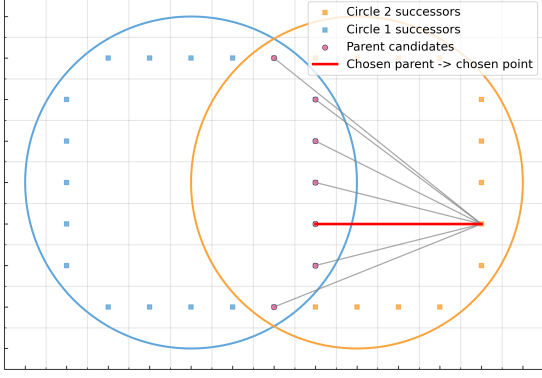


Fig. 4: CALCULATESUCCESSORS: Each successor selects a parent in the current bubble according to the minimum cost-to-come and updates its cost. The OPEN set then contains both existing (blue) and new (orange) nodes.

$\mathcal{F} := \mathcal{C}_\Delta \setminus \mathcal{O}$ the free cells. The corresponding set of free grid nodes is $\mathcal{V}^{\text{free}} := \{\mathbf{p} \in \mathcal{V} \mid \mathcal{C}_\Delta(\mathbf{p}) \in \mathcal{F}\}$. The free region can be represented explicitly using the SDF reconstructed using OREN in Sec. 4. By computing the SDF $d(\mathbf{c})$ at a node $\mathbf{c} \in \mathcal{V}^{\text{free}}$, we define an open ball of radius at least $\frac{\Delta}{2}$. We refer to this obstacle-free region around the node as a bubble and denote it as $\mathbb{B}_\mathbf{c} = \mathbb{B}(\mathbf{c}, d(\mathbf{c}))$, where:

$$\mathbb{B}(\mathbf{c}, r) := \{\mathbf{p} \in \mathbb{R}^3 \mid \|\mathbf{p} - \mathbf{c}\|_2 < r\}. \quad (10)$$

For the graph search over grid nodes, we use the free grid nodes that lie inside the bubble:

$$\mathcal{B}_\mathbf{c} = \mathcal{B}(\mathbf{c}, d(\mathbf{c})) := \mathbb{B}_\mathbf{c} \cap \mathcal{V}^{\text{free}}. \quad (11)$$

We define a graph over the free grid nodes $\mathcal{V}^{\text{free}}$, in which each bubble connects the nodes it contains to the nodes on its boundary. The boundary nodes are characterized using the axis-aligned neighbors of a node:

$$\mathcal{N}_6(\mathbf{p}) := \{\mathbf{p} \pm \Delta \mathbf{e}_i \in \mathcal{V} \mid i \in \{1, \dots, 3\}\}, \quad (12)$$

where $\{\mathbf{e}_i\}_{i=1}^3$ are the standard basis vectors of $\mathbb{R}^3$. The *successor set* of a bubble collects the boundary nodes, those with at least one axis-aligned neighbor (12) lying outside the bubble, as illustrated in Fig. 4:

$$S(\mathcal{B}_\mathbf{c}) = \{\mathbf{p} \in \mathcal{B}_\mathbf{c} \mid \exists \mathbf{q} \in \mathcal{N}_6(\mathbf{p}) \text{ s.t. } \mathbf{q} \notin \mathcal{B}_\mathbf{c}\}. \quad (13)$$

Two nodes $\mathbf{p}, \mathbf{q} \in \mathcal{V}^{\text{free}}$ are connected by an edge if some bubble (11) contains $\mathbf{p}$ and has $\mathbf{q}$ on its boundary. That is, $\mathbf{p} \in \mathcal{B}_\mathbf{c}$ and $\mathbf{q} \in S(\mathcal{B}_\mathbf{c})$ for some $\mathbf{c} \in \mathcal{V}^{\text{free}}$. Every edge is collision-free: the bubble $\mathbb{B}_\mathbf{c}$ (10) is obstacle-free and convex, so it contains the straight line segment between the two nodes.

5.2 Bubble* Search Algorithm

We present the Bubble* planner in Algorithm 1, which performs a heuristic graph search over bubbles to compute a collision-free path from the start node $\mathbf{p}_s$ to the goal node $\mathbf{p}_g$. The algorithm expands bubbles constructed from SDF clearance, allowing large collision-free regions to be explored efficiently. This greatly reduces the required number of collision checks compared to other search-based planners.

Bubble* maintains two sets of nodes: an OPEN set containing candidate successor nodes to be expanded, and a CLOSED set containing nodes whose associated bubbles have already been explored. Each node $\mathbf{p} \in \mathcal{V}^{\text{free}}$ is associated with a cost-to-come value $g(\mathbf{p})$, representing the minimum path length from the start to $\mathbf{p}$, a parent pointer $\text{Parent}(\mathbf{p})$ recording the predecessor node used to reach $\mathbf{p}$, and a bubble pointer, $\text{Bubble}(\mathbf{p})$, recording the bubble that contains the segment from $\mathbf{p}$ to $\text{Parent}(\mathbf{p})$. Analogous to an A* search, the OPEN set is prioritized according to the sum of the cost-to-come and an optional heuristic estimate of the distance to the goal.

At each iteration, Bubble* selects the node $\mathbf{c} \in \text{OPEN}$ with minimum priority and constructs the corresponding bubble $\mathcal{B}_\mathbf{c} = \mathcal{B}(\mathbf{c}, d(\mathbf{c}))$ defined in (11), along with its successor set $S_\mathbf{c} = S(\mathcal{B}_\mathbf{c})$ in (13). We refer to this as *expanding* $\mathbf{c}$. If $\mathbf{c} = \mathbf{p}_g$, the search terminates. Otherwise, if the goal lies within $\mathcal{B}_\mathbf{c}$, it is appended to the current successor set $S_\mathbf{c}$, so that the termination condition $\mathbf{c} = \mathbf{p}_g$ holds in a future iteration. The CALCULATESUCCESSORS procedure in Algorithm 1 processes each successor $\mathbf{j} \in S_\mathbf{c}$: it selects the predecessor $\mathbf{k}^*$ among nodes inside $\mathcal{B}_\mathbf{c}$ that minimizes $g(\mathbf{k}) + \|\mathbf{j} - \mathbf{k}\|_2$ and updates $g(\mathbf{j})$, $\text{Parent}(\mathbf{j})$, and $\text{Bubble}(\mathbf{j})$ whenever this improves the cost-to-come. Interior nodes of $\mathcal{B}_\mathbf{c}$ and its center $\mathbf{c}$ are then marked as CLOSED. This procedure assigns each successor node of the new bubble a parent node according to Line 21 of Algorithm 1.

Algorithm 1: Bubble* Planner

```
1 Algorithm Bubble* ( $\mathbf{p}_s, \mathbf{p}_g, d(\cdot), h(\cdot)$ )
2   OPEN  $\leftarrow \{\mathbf{p}_s\}$ ,   CLOSED  $\leftarrow \emptyset$ 
3    $g(\mathbf{p}_s) = 0$ ,    $g(\mathbf{p}) = \infty \quad \forall \mathbf{p} \neq \mathbf{p}_s$ 
4   while OPEN  $\neq \emptyset$  do
5      $\mathbf{c} \leftarrow \arg \min_{\mathbf{p} \in \text{OPEN}} (g(\mathbf{p}) + h(\mathbf{p}))$ 
6     OPEN  $\leftarrow \text{OPEN} \setminus \{\mathbf{c}\}$ 
7      $\mathcal{B}_c \leftarrow \mathcal{B}(\mathbf{c}, d(\mathbf{c}))$ 
8     if  $\mathbf{c} = \mathbf{p}_g$  then
9       | return PathTo( $\mathbf{p}_g$ )
10     $S_c \leftarrow S(\mathcal{B}_c)$ 
11    if  $\mathbf{p}_g \in \mathcal{B}_c$  then
12      |  $S_c \leftarrow S_c \cup \{\mathbf{p}_g\}$ 
13    CalculateSuccessors( $\mathcal{B}_c, S_c, \text{OPEN}, \text{CLOSED}$ )
14    foreach  $\mathbf{p} \in \mathcal{B}_c$  with  $\mathbf{p} \notin S_c$  do
15      | CLOSED  $\leftarrow \text{CLOSED} \cup \{\mathbf{p}\}$ 
16    CLOSED  $\leftarrow \text{CLOSED} \cup \{\mathbf{c}\}$ 
17  return failure

18 Procedure CalculateSuccessors( $\mathcal{B}_c, S_c, \text{OPEN}, \text{CLOSED}$ )
19  foreach  $\mathbf{j} \in S_c$  with  $\mathbf{j} \notin \text{CLOSED}$  do
20    |  $\mathbf{k}^* \leftarrow \arg \min_{\mathbf{k} \in \text{OPEN} \cap \mathcal{B}_c} (g(\mathbf{k}) + \|\mathbf{j} - \mathbf{k}\|_2)$ 
21    |  $g_{\min} \leftarrow g(\mathbf{k}^*) + \|\mathbf{j} - \mathbf{k}^*\|_2$ 
22    | if  $g(\mathbf{j}) > g_{\min}$  then
23      | |  $g(\mathbf{j}) \leftarrow g_{\min}$ 
24      | | Parent( $\mathbf{j}$ )  $\leftarrow \mathbf{k}^*$ ,   Bubble( $\mathbf{j}$ )  $\leftarrow \mathcal{B}_c$ 
25      | | if  $\mathbf{j} \in \text{OPEN}$  then
26      | | | update priority of  $\mathbf{j}$ 
27      | | else
28      | | | OPEN  $\leftarrow \text{OPEN} \cup \{\mathbf{j}\}$ 

29 Procedure PathTo( $\mathbf{p}$ )
30  Path  $\leftarrow \{\mathbf{p}\}$ 
31  BubblePath  $\leftarrow \{\mathcal{B}(\mathbf{p}, d(\mathbf{p}))\}$ 
32  current  $\leftarrow \mathbf{p}$ 
33  while Parent(current) exists do
34    | Path  $\leftarrow \text{Path} \cup \{\text{current}\}$ 
35    | BubblePath  $\leftarrow \text{BubblePath} \cup \{\text{Bubble}(\text{current})\}$ 
36    | current  $\leftarrow \text{Parent}(\text{current})$ 
37  return {Path.reverse(), BubblePath.reverse()}
```

The search terminates when the goal $\mathbf{p}_g$ is drawn from the open set (success), or if the open set is empty (failure). In the successful case, the parent pointers are followed to reconstruct a sequence of overlapping bubbles connecting the start and goal, which is done in the PATHTO procedure. If the OPEN set becomes empty before reaching the goal, the algorithm reports failure.

Because the parent candidate $\mathbf{k}^*$ may be any OPEN node inside the bubble, the recovered path can skip intermediate nodes unlike A*. Bubble* therefore produces any-angle paths in that the nodes on the path are connected through one straight segment if they belong to the same bubble. Therefore, the resulting paths are no longer than the shortest path restricted to the grid and are typically shorter.

5.3 Termination, Completeness, and Failure Detection

We now establish formal guarantees for Bubble*: it always terminates, it returns a path whenever one with sufficient clearance exists, and it reports failure otherwise. Throughout, we use the fact that the search space is bounded by $\mathcal{R}$, so there are finitely many nodes $\mathcal{V}$, and that the start and goal are free grid nodes, $\mathbf{p}_s, \mathbf{p}_g \in \mathcal{V}^{\text{free}}$. We assume the heuristic h is consistent, i.e., $h(\mathbf{p}_g) = 0$ and $h(\mathbf{p}) \leq \|\mathbf{p} - \mathbf{p}'\|_2 + h(\mathbf{p}')$ for adjacent $\mathbf{p}, \mathbf{p}'$, which holds for the Euclidean distance $h(\mathbf{p}) = \|\mathbf{p} - \mathbf{p}_g\|_2$ used in our experiments.

Definition 1 (Clear grid path). *Let $\mathcal{V}^{\text{free}}$ be a set of free nodes with resolution Δ . A grid path from $\mathbf{p}_s$ to $\mathbf{p}_g$ is a finite sequence $(\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_N) \subset \mathcal{V}^{\text{free}}$ with $\mathbf{p}_0 = \mathbf{p}_s$, $\mathbf{p}_N = \mathbf{p}_g$, and $\mathbf{p}_{j+1} \in \mathcal{N}_6(\mathbf{p}_j)$ for all j . It is clear if the distance $d(\mathbf{p}_j) \geq 1.5\Delta$ for every j .*

The adjacency condition makes the sequence a connected walk on the grid, while the 1.5Δ clearance ensures each node's bubble contains all of its grid neighbors. Because the axis-aligned neighbors lie at distance Δ and bubbles are open balls, the radius must satisfy $d(\mathbf{p}_j) > \Delta$. Among the clearance values admissible under our grid-occupancy model, 1.5Δ is the smallest. The completeness proof below relies on this property.

Theorem 1 (Termination). *Bubble* (Algorithm 1) terminates after finitely many iterations.*

Proof The grid $\mathcal{V}$ is finite and each bubble is determined by its center, so there are finitely many bubbles. With a consistent heuristic, the cost-to-come $g(\mathbf{c})$ is optimal when $\mathbf{c}$ is expanded (Hart et al., 1968), so each node is expanded at most once and is never reinserted into OPEN after entering CLOSED. Each iteration removes one node from OPEN and inserts finitely many successors, so after at most $|\mathcal{V}|$ expansions OPEN is empty or the goal is returned. $\square$

Theorem 2 (Completeness). *If a clear grid path from $\mathbf{p}_s$ to $\mathbf{p}_g$ exists, then Bubble* (Algorithm 1) returns a valid path.*

Proof By Theorem 1, the search terminates, so it either returns a path or reports failure. Suppose, for

contradiction, that it reports failure. Then, OPEN is empty at termination. Let $(\mathbf{p}_0, \dots, \mathbf{p}_N)$ be the clear grid path from the hypothesis, with $\mathbf{p}_0 = \mathbf{p}_s$, $\mathbf{p}_N = \mathbf{p}_g$, $\mathbf{p}_{j+1} \in \mathcal{N}_6(\mathbf{p}_j)$, and $d(\mathbf{p}_j) \geq 1.5\Delta$ for all j . We prove by induction that every $\mathbf{p}_j$ lies in some bubble expanded during the search.

For the base case, $\mathbf{p}_0 = \mathbf{p}_s$ is placed in OPEN at initialization. Since OPEN is empty at termination, it was popped and its bubble expanded, so $\mathbf{p}_0$ lies in an expanded bubble. For the inductive step, suppose $\mathbf{p}_j$ lies in an expanded bubble $\mathcal{B}_{\mathbf{c}}$. If $\mathbf{p}_j$ is interior to $\mathcal{B}_{\mathbf{c}}$, that is $\mathbf{p}_j \notin S(\mathcal{B}_{\mathbf{c}})$, then by (13) all grid neighbors of $\mathbf{p}_j$, including $\mathbf{p}_{j+1}$, lie in $\mathcal{B}_{\mathbf{c}}$.

Otherwise $\mathbf{p}_j$ lies on the boundary, $\mathbf{p}_j \in S(\mathcal{B}_{\mathbf{c}})$, and was inserted into OPEN. Since OPEN is empty, it was also popped and its own bubble $\mathcal{B}_{\mathbf{p}_j}$ is expanded. Because $d(\mathbf{p}_j) \geq 1.5\Delta > \Delta$ and $\mathbf{p}_{j+1}$ is a grid neighbor at distance Δ , we have $\mathbf{p}_{j+1} \in \mathcal{B}_{\mathbf{p}_j}$. In either case $\mathbf{p}_{j+1}$ lies in an expanded bubble, completing the induction.

In particular, the goal $\mathbf{p}_N = \mathbf{p}_g$ lies in some expanded bubble $\mathcal{B}_{\mathbf{c}}$. The goal test $\mathbf{p}_g \in \mathcal{B}_{\mathbf{c}}$ in Algorithm 1 then inserts $\mathbf{p}_g$ into the successor set, so CALCULATESUCCESSORS inserts it into OPEN with finite cost. Since OPEN is empty at termination, $\mathbf{p}_g$ must have been popped. When a node equal to $\mathbf{p}_g$ is popped, the algorithm returns $\text{PATHTo}(\mathbf{p}_g)$. This contradicts the failure assumption, hence, Bubble* returns a valid path. $\square$

Corollary 1 (Failure detection). *Bubble* (Algorithm 1) reports failure in finite time and only when no clear grid path connects $\mathbf{p}_s$ and $\mathbf{p}_g$.*

Proof Since Theorem 1 shows termination in finite time and Theorem 2 shows completeness, it follows that if no path exists, the algorithm terminates with failure in finite time. $\square$

5.4 Trajectory Optimization Within Bubble Corridor

Given the sequence of bubbles returned by Bubble*, we compute a trajectory to solve (4) that lies within the convex bubble corridor. Pairs of consecutive bubbles define a convex overlap region through which the trajectory may pass. These overlap regions provide a sequence of safe sets that constrain the trajectory while preserving sufficient freedom for optimization.

Bubble Overlap Construction

Bubble* returns a sequence of bubbles (10) along the path, which we index as $\mathbb{B}_i := \mathbb{B}(\mathbf{c}_i, r_i)$ with centers $\mathbf{c}_i \in \mathbb{R}^3$ and radii $r_i := d(\mathbf{c}_i) > 0$. Consecutive bubbles $\mathbb{B}_i$ and $\mathbb{B}_{i+1}$ overlap by construction, since Bubble* generates each bubble center inside its predecessor. Their intersection is a lens whose widest cross-section is a disk in the plane perpendicular to the line between centers $\mathbf{a}_i := \mathbf{c}_{i+1} - \mathbf{c}_i$. We use this disk as a convex safe set linking the two bubbles. Its offset along $\mathbf{a}_i$, center, and radius are:

$$\begin{aligned}\lambda_i &= \frac{1}{2} + \frac{r_i^2 - r_{i+1}^2}{2 \|\mathbf{a}_i\|^2}, \\ \mathbf{o}_i &= \mathbf{c}_i + \lambda_i \mathbf{a}_i, \\ \rho_i &= \sqrt{r_i^2 - \lambda_i^2 \|\mathbf{a}_i\|^2}.\end{aligned}\quad (14)$$

Let $\boldsymbol{\nu}_i = \mathbf{a}_i / \|\mathbf{a}_i\|$ be the unit normal of the disk's plane and $B_i = [\mathbf{e}_1^i \ \mathbf{e}_2^i] \in \mathbb{R}^{3 \times 2}$ an orthonormal basis of that plane. With the center $\mathbf{o}_i$ and radius ρ_i from (14), the widest overlapping cross-section is the disk

$$\begin{aligned}\mathcal{S}_i &= \{ \mathbf{o}_i + B_i \mathbf{w} \mid \mathbf{w} \in \mathbb{R}^2, \|\mathbf{w}\|_2 \leq \rho_i \} \\ &\subset \mathbb{B}_i \cap \mathbb{B}_{i+1}.\end{aligned}\quad (15)$$

MINCO Trajectory Representation

We build upon the MINCO representation (Z. Wang et al., 2022), which parameterizes a piecewise polynomial trajectory using intermediate waypoints and segment durations. Let the trajectory consist of N polynomial segments. MINCO parameterizes the trajectory using spatial variables $\boldsymbol{\xi} = \{\boldsymbol{\xi}_i\}_{i=1}^{N-1}$ and temporal variables $\boldsymbol{\tau} = \{\tau_i\}_{i=1}^N$, which determine the intermediate waypoints and segment durations, respectively. These variables are mapped to physical waypoints $\mathbf{q} = \mathbf{q}(\boldsymbol{\xi})$ and segment durations $\mathbf{T} = \mathbf{T}(\boldsymbol{\tau})$, and the optimal polynomial coefficients $\mathbf{c}$ are uniquely determined from $(\mathbf{q}, \mathbf{T})$.

The trajectory optimization problem is formulated as

$$\min_{\boldsymbol{\xi}, \boldsymbol{\tau}} \mathcal{J}(\mathbf{q}(\boldsymbol{\xi}), \mathbf{T}(\boldsymbol{\tau})) \quad \text{s.t.} \quad \mathbf{q}_i(\boldsymbol{\xi}_i) \in \mathcal{S}_i, \forall i, \quad (16)$$

where $\mathcal{J}$ includes the minimum-snap and time penalties of (4) as well as corridor violation penalties. The dynamics constraints are penalized through the cost, following Z. Wang et al. (2022).

The clearance constraint in (4) is enforced by shrinking the radii of all bubbles in the corridor by the required clearance r , and the rest of the constraints can be written naturally in the MINCO problem formulation, so the optimization is (4) restricted to the MINCO polynomial class.

We design a smooth unconstrained parameterization that automatically satisfies the constraint of each waypoint lying within its corresponding overlap region of (15), i.e., $\mathbf{q}_i \in \mathcal{S}_i$. Specifically, we map unconstrained variables $\boldsymbol{\xi}_i \in \mathbb{R}^2$ to constrained waypoints $\mathbf{q}_i$ as:

$$\begin{aligned}\mathbf{u}_i &= \frac{B_i \boldsymbol{\xi}_i}{\sqrt{1 + \|B_i \boldsymbol{\xi}_i\|^2}}, \\ \mathbf{q}_i &= \mathbf{o}_i + \rho_i \mathbf{u}_i.\end{aligned}\quad (17)$$

Segment durations must satisfy the positivity constraint $T_i > 0$. Following the temporal constraint elimination scheme of MINCO (Z. Wang et al., 2022), we introduce unconstrained temporal variables $\tau_i \in \mathbb{R}$ and define the mapping:

$$T_i = \begin{cases} \frac{\tau_i^2}{2} + \tau_i + 1, & \tau_i > 0, \\ (\frac{1}{2}\tau_i^2 - \tau_i + 1)^{-1}, & \tau_i \leq 0, \end{cases} \quad (18)$$

which ensures positivity while preserving differentiability. Gradients with respect to τ_i are computed using the chain rule in conjunction with the gradients provided by the MINCO formulation.

Gradient-Based Optimization

MINCO provides gradients of the cost function with respect to waypoints and durations, $\partial \mathcal{J} / \partial \mathbf{q}$ and $\partial \mathcal{J} / \partial \mathbf{T}$. These gradients are propagated through the spatial parameterization using

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\xi}_i} = B_i^\top \rho_i (\alpha I_{3 \times 3} - \alpha \mathbf{u}_i \mathbf{u}_i^\top) \frac{\partial \mathcal{J}}{\partial \mathbf{q}_i}, \quad (19)$$

where $\alpha = (1 + \|B_i \boldsymbol{\xi}_i\|^2)^{-1/2}$. To improve convergence, we first prune the bubbles whose waypoints $\mathbf{q}_i$ make no progress toward the goal, and initialize the remaining waypoints at the centers of their overlap regions $\mathbf{o}_i$. The optimization then runs in two stages:

1. Optimize the spatial variables $\boldsymbol{\xi}$ with the durations $\boldsymbol{\tau}$ fixed to obtain a collision-free geometric path.

2. Jointly optimize ξ and τ to obtain a dynamically feasible trajectory.

This two-stage optimization produces smooth dynamically feasible trajectories through the sequence of bubbles from the Bubble* algorithm.

6 Evaluation

Our central claim is that combining OREN and Bubble* yields an efficient integrated mapping, planning, and trajectory optimization approach for UAVs that runs fully onboard. To support this claim, we first show that OREN provides an accurate continuous SDF representation with sufficient computational efficiency to permit real-time operation in simulated large-scale environments. Our prior work (Dai, Qian, et al., 2026) compares OREN extensively against SDF mapping baselines, and Sec. 6.1 summarizes those results. The key remaining comparison is against the mapping approaches commonly used in autonomous flight, and we evaluate OREN in four large simulated environments against OctoMap (Hornung et al., 2013), a widely used mapping library for quadrotors (Sec. 6.2). Then, building on the SDF reconstruction from OREN, another set of experiments shows that Bubble* is more efficient than grid-based A* and sampling-based (RRT, RRT*) planners (Sec. 6.3). Finally, we evaluate our approach on a real quadrotor with a Jetson Orin NX computer to show that it runs efficiently and fully onboard (Sec. 6.4).

6.1 Comparison with SDF Mapping Methods

In our conference paper (Dai, Qian, et al., 2026), OREN is evaluated on Replica (Straub et al., 2019) and Newer College (Zhang et al., 2021) datasets, in comparison with four baselines H₂-Mapping (Jiang et al., 2023), PIN-SLAM (Pan et al., 2024), HIO-SDF (Vasilopoulos et al., 2024) and Voxelblox (Oleynikova et al., 2017). OREN generates mesh results of quality similar to the baselines’ for small indoor Replica scenes and synthesized depth data. However, on the large outdoor Newer College scene with real LiDAR measurements, OREN reconstructs a much better mesh, as shown in Table 1. On the Newer College dataset, OREN improves over the best

baselines by 29–68% across all seven mesh metrics (Completion +36.2%, Completion Ratio +29.3%, Recall +33.5%, Precision +40.3%, F1 +37.0%, Chamfer-L1 +56.7%, Accuracy +68.3%), averaging $\approx 43.1\%$. As for SDF metrics, OREN outperforms the baselines by a larger margin on both Replica and Newer College datasets. As shown in Table 2, OREN provides at least 15% more accurate SDF than the baselines, which is an essential factor for the success of our Bubble* planner.

6.2 Mapping in Large Simulated Environments

The comparisons in Sec. 6.1 establish OREN’s accuracy against dedicated SDF mapping methods on standard benchmarks. This section evaluates the performance of OREN in an online setting with data streaming from a UAV in large simulated environments.

We build four simulation environments, a forest, an underground garage, an industrial site, and a warehouse, and simulate a quadrotor flying through each, carrying a single onboard depth camera and building the map online from the streaming point clouds. These scenes form our testbed for the mapping-and-planning approach. Here, we evaluate the map representation and Sec. 6.3 evaluates planning in the same environments. With our focus being on online deployment of autonomous robots, we compare against OctoMap (Hornung et al., 2013), an occupancy-grid mapping method widely used on compute-constrained platforms. OREN matches OctoMap’s occupancy prediction quality at a comparable update rate while providing the continuous distances that accelerate planning while recovering visually better surface meshes.

Table 3 reports occupancy prediction quality with the *Precision*, *Recall*, and *F1* metrics common in the mapping literature (Dai, Qian, et al., 2026; Jiang et al., 2023; Pan et al., 2024), evaluated against the ground-truth surface point cloud. OREN achieves better or comparable results across the four environments while additionally providing distance information and gradients. Fig. 5 visualizes the reconstructed surface meshes side by side in each scene, the marching-cubes mesh of the zero level-set of OREN’s SDF and the faces of OctoMap’s occupied cells.

Table 1: Mesh reconstruction metrics with $\delta = 20$ cm on the Newer College dataset (Zhang et al., 2021).

Metric	OREN	H ₂ -Mapping	PIN-SLAM	HIO-SDF	Voxblox
Completion [cm] ↓	10.66	21.94	<u>16.71</u>	72.86	21.30
Completion Ratio [$< \delta$] % ↑	94.20	61.58	<u>72.83</u>	10.05	60.31
Recall [$< \delta$] % ↑	93.99	57.96	<u>70.40</u>	4.72	56.64
Precision [$< \delta$] % ↑	90.69	52.97	<u>64.63</u>	4.46	51.84
F1 Score [$< \delta$] % ↑	92.31	55.35	<u>67.39</u>	4.59	54.14
Chamfer-L1 Distance [cm] ↓	9.36	28.40	<u>21.64</u>	422.29	23.37
Accuracy [cm] ↓	8.07	34.86	<u>26.58</u>	771.72	<u>25.44</u>

Table 2: Average improvement of OREN over other SDF estimation methods (the best baseline for each scene and metric). SDF MAE is the MAE of the predicted signed distance, and Gradient MAE is the MAE of the angle between the predicted and ground-truth SDF gradients.

Metric	Region	Avg. ↑
SDF MAE	All	22.0%
	Near	34.5%
	Far	15.0%
Gradient MAE	All	35.5%
	Near	54.1%
	Far	29.9%

Table 3: Comparison of OREN and OctoMap on 4 simulated environments. Both methods are compared against a ground-truth surface point cloud.

Env	Method	Precision	Recall	F1
Warehouse	OREN	0.672	0.884	0.764
	OctoMap	0.780	0.629	0.696
Garage	OREN	0.862	0.911	0.886
	OctoMap	0.888	0.819	0.852
Forest	OREN	0.652	0.948	0.773
	OctoMap	0.900	0.384	0.539
Industrial	OREN	0.556	0.901	0.688
	OctoMap	0.787	0.646	0.710

OctoMap builds a discrete occupancy grid of the free and occupied voxels at a fixed resolution. OREN instead represents the scene as a continuous SDF that provides the signed distance and gradient at every point (Sec. 4.2). Throughout the evaluation, both the occupancy predictions and the reconstructed meshes are derived from the SDF estimate $\hat{d}$ alone. The occupancy decoder could provide occupancy directly, but we choose to measure the representational power of the SDF itself, though the occupancy information is used during online training to improve the SDF training. Compared to OctoMap, OREN produces a

more accurate and complete reconstruction of the scene, with fewer holes and artifacts.

The continuous SDF also carries richer information for planning. It encodes the distance to the nearest obstacle that Bubble* uses for its efficiency and safety. This added information comes at only a modest expense in computation time. As Fig. 6 shows, OREN yields an update frequency comparable to OctoMap’s, only around 1 Hz slower in two of the four environments (warehouse and industrial), and 4–10 Hz slower in the others (garage and forest). The map update rate (19.1–28.4 Hz) remains amenable to real-time operation. Compared to the benefit of the continuous distance information, we regard the increase in computation as a small price to pay, since it enables Bubble*, which OctoMap cannot.

6.3 Planning Evaluation

We evaluate the efficiency of Bubble* by the number of collision checks the planner issues and by the total planning time, and report path length and tracking error to verify that efficiency does not come at the cost of trajectory quality. We first isolate search efficiency in a controlled 2D comparison against grid-based A*, and then evaluate the complete planner in the 3D industrial and forest environments of Sec. 6.2 against grid-based (A*) and sampling-based (RRT, RRT*) baselines.

Search Efficiency in 2D

We first evaluate Bubble* against A* in a 2D environment shown in Fig. 7. A* binarizes the same SDF to determine occupancy and expands one neighboring node at a time, whereas Bubble* uses the SDF magnitude to grow collision-free bubbles and expands only at their boundaries. Both planners use the Euclidean distance to the goal, $h(\mathbf{p}) = \|\mathbf{p} - \mathbf{p}_g\|_2$, as the heuristic. Over 1000

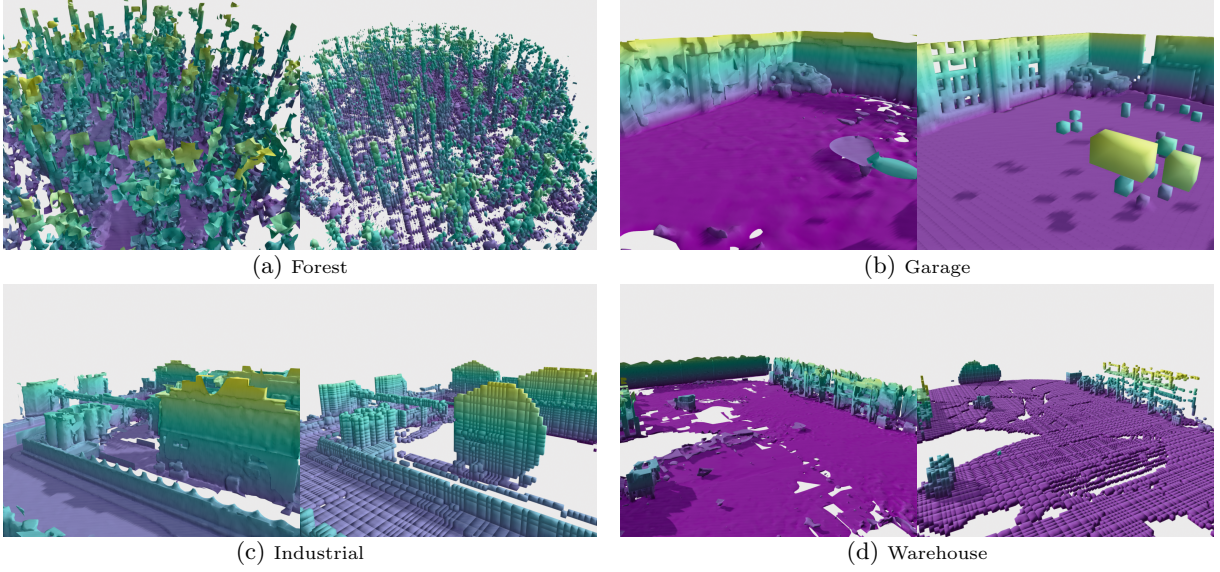


Fig. 5: Map representations compared across four large-scale simulated environments. In each pair, OREN (left) reconstructs a continuous SDF surface and OctoMap (Hornung et al., 2013) (right) a discretized occupancy grid of the same scene. Each map is built online from a single depth camera on a simulated quadrotor flown through a sequence of fixed waypoints.

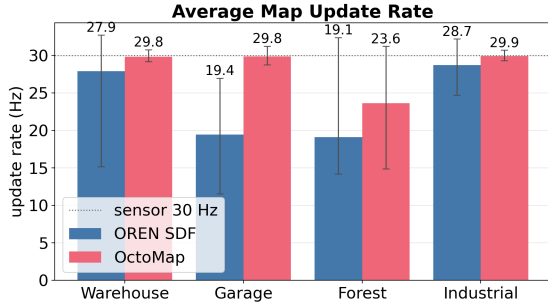


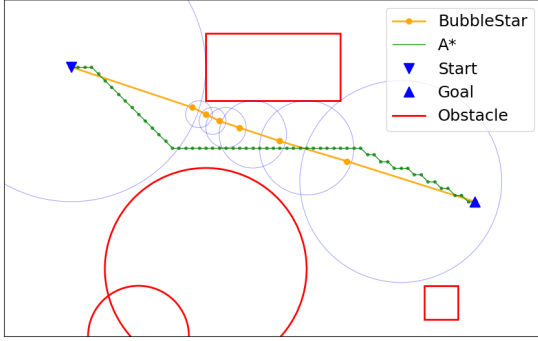
Fig. 6: Average mapping update rate (Hz) of OREN and OctoMap (Hornung et al., 2013) across the four simulated environments. The depth camera streams at 30 Hz (dotted line). Both run in real time. OctoMap is faster, as updating an occupancy grid is lighter than fitting an implicit field, whereas OREN provides the smooth, differentiable SDF that answers the distance queries Bubble* depends on.

trials at several resolutions, Bubble* performs 91–99% fewer collision checks than A* and finds a shorter or equal path in every case. Fig. 7b shows the runtime distribution, where Bubble* is about an order of magnitude faster, with a wider gap at finer resolutions.

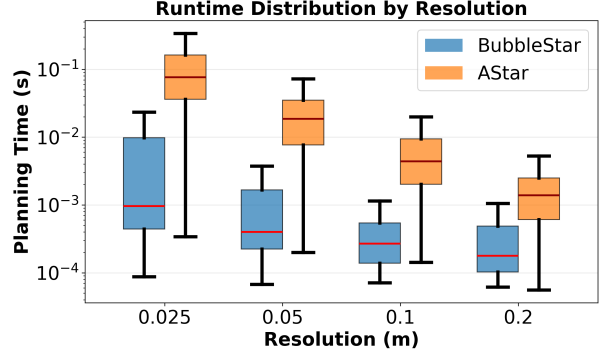
Planning in 3D Environments

We compare Bubble* in 3D environments against three baselines: grid-based A* and the sampling-based RRT and RRT* planners. To generate dynamically feasible trajectories for a quadrotor from the baselines, we place bubbles along the geometric paths found by the baseline planners, and use the same optimization routine as in Sec. 5.4. All planners use the same 3D SDF obtained using OREN. We evaluate on the industrial site and the forest scene environments from Sec. 6.2. A representative trajectory from each planner in the industrial scene is shown in Fig. 8a, and quantitative metrics for both scenes are reported in Table 4.

In the industrial environment, Bubble* achieves the lowest total planning time (1.0 s) while matching the shortest path within 1% (86.0 m vs 85.4 m by RRT*), well below A* (90.3 m) and RRT (102.5 m). The marginally shorter path by RRT* is expected because RRT* routes arbitrarily close to obstacles in continuous space, whereas Bubble* searches on a grid with fixed resolution and keeps its trajectory inside high-clearance bubbles, trading a fraction of length for larger clearance margin.

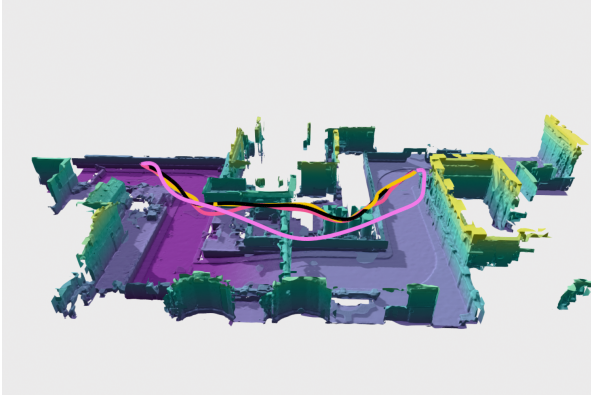


(a)

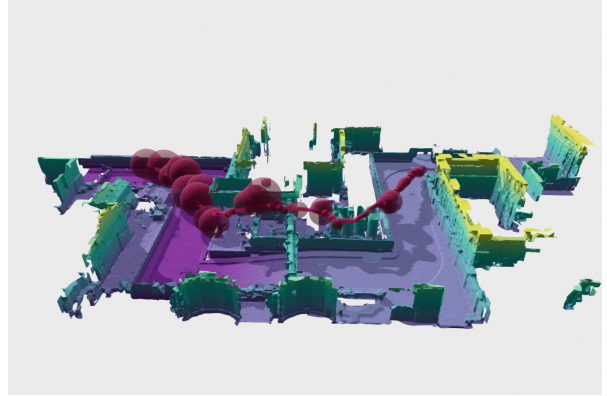


(b)

Fig. 7: 2D search-efficiency comparison between Bubble^{*} and A^{*}. (a) Representative paths in a 2D environment; SDF clearance lets Bubble^{*} take any-angle shortcuts that grid-restricted A^{*} cannot. (b) Runtime over 1000 randomly sampled start-goal pairs at several grid resolutions. Boxes span the interquartile range, the red line marks the median, and whiskers extend to the minimum and maximum.



(a) Planner trajectories



(b) Bubble^{*} safe corridor

Fig. 8: Planning in the simulated industrial environment of Fig. 5c, all on the same OREN SDF. (a) Representative trajectories from Bubble^{*} (red), A^{*} (orange), RRT (magenta), and RRT^{*} (black). (b) The Bubble^{*} safe corridor: overlapping bubbles (red) grown along the optimized trajectory, whose large, mutually overlapping clearance gives the optimizer ample room and keeps the optimization fast (Table 4).

The two planning stages, search and optimization, trade off differently across the methods. A^{*} searches quickly (223 ms) but the high number of bubbles created around the path (164) leads to a dense cell corridor and slow trajectory optimization (1005 ms). RRT finds a discrete path almost instantly (2 ms). However, because the path is jagged, it takes a long time (6068 ms) to optimize a continuous trajectory within bubbles centered at the RRT path. RRT^{*} requires a stopping criterion for search, and we chose one that is based on finding a path shorter than Bubble^{*}, but this causes RRT^{*} to spend most of its budget on search (5.1 s). Bubble^{*} keeps both

stages low (485 ms search, 523 ms optimization). Tracking error is comparable across all planners, with Bubble^{*} achieving the lowest tracking error (0.291 m), confirming that its corridor does not compromise dynamic feasibility.

By construction, Bubble^{*} produces a corridor of overlapping safe regions, which keeps the subsequent trajectory optimization fast. The Bubbles and Mean r columns of Table 4 quantify this: Bubble^{*} covers its path with 82 overlapping bubbles of mean radius 1.08 m (Fig. 8b), whereas A^{*}'s dense cell path requires 164 bubbles of roughly half the radius (0.55 m). Fewer and larger bubbles

Table 4: Planning comparison in the simulated industrial and forest environments of Fig. 5. All planners operate on the same SDF map. Columns report the planning time (Search), the trajectory-optimization time (Opt.), their sum (Total), the path length (Path), the root-mean-square position error of the executed trajectory relative to the planned trajectory (RMSE), the number of overlapping bubbles forming the safe corridor (Bubbles), and their mean radius (Mean r). **Bold** marks the best value in each column and underline the second best, per environment. For the Bubbles and Mean r columns, which characterize the corridor handed to the trajectory optimizer, fewer bubbles and larger mean radius are better. Bubble* attains the lowest total planning time in both environments while producing a compact high-clearance corridor.

Env	Planner	Search (ms)	Opt. (ms)	Total (ms)	Path (m)	RMSE (m)	Bubbles	Mean r (m)
Industrial	Bubble*	485	523	1008	<u>86.0</u>	0.291	82	<u>1.08</u>
	A*	<u>223</u>	1005	<u>1228</u>	90.3	<u>0.297</u>	164	0.55
	RRT	2	6068	6069	102.5	0.327	82	1.17
	RRT*	5079	<u>927</u>	6006	85.4	0.314	<u>123</u>	0.69
Forest	Bubble*	298	<u>3478</u>	3776	87.2	<u>0.274</u>	311	0.30
	A*	<u>466</u>	3687	<u>4152</u>	<u>89.5</u>	0.265	341	0.27
	RRT	2931	3056	5987	90.1	0.281	348	0.27
	RRT*	5050	5318	10368	94.7	0.324	<u>328</u>	<u>0.28</u>

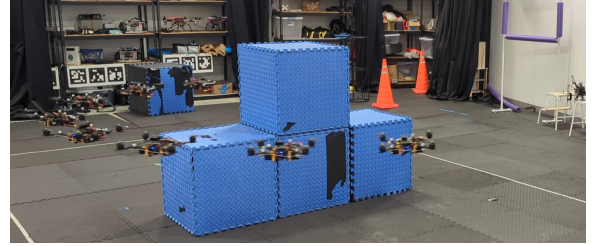
leave the optimizer more room around each way-point and fewer convex regions to stitch together, which is why Bubble* optimizes in 523 ms against the 1005 ms A* requires. Larger bubbles alone are not sufficient, however: RRT attains an even larger mean radius (1.17 m) with the same bubble count, yet its randomly sampled path threads them along an irregular route and is the slowest of all to optimize (6.1 s). Bubble* is the only method that pairs a compact, high-clearance corridor with a smooth underlying path, which keeps both its search and its optimization fast.

This pattern still holds in the more cluttered forest environment. Bubble* again attains the lowest total planning time (3.8 s) and produces the shortest path (87.2 m), with the most compact, highest-clearance corridor among the four planners (311 bubbles at a 0.30 m mean radius). The sampling-based planners remain the most expensive: RRT* spends over 10 s across search and optimization, and both produce longer paths than Bubble*.

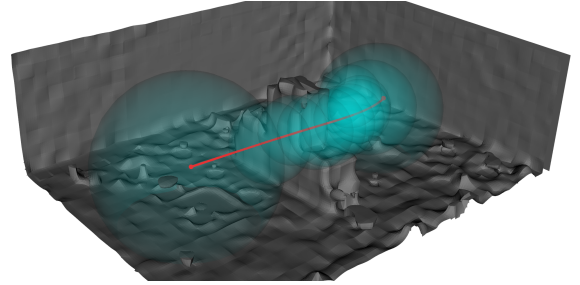
Across both experiments, the efficiency of Bubble* follows directly from the tight coupling between map and planner: by reading clearance straight from OREN’s non-truncated SDF map and expanding whole collision-free bubbles rather than testing occupancy cell by cell, it issues far fewer signed-distance/collision queries than A* while preserving the safety and completeness guarantees (Sec. 5.3).

6.4 Real-World Autonomous Flight

We have shown that OREN produces a non-truncated SDF efficiently (Sec. 6.2) and Bubble*



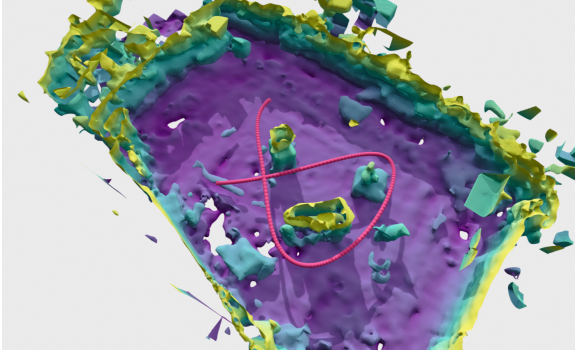
(a)



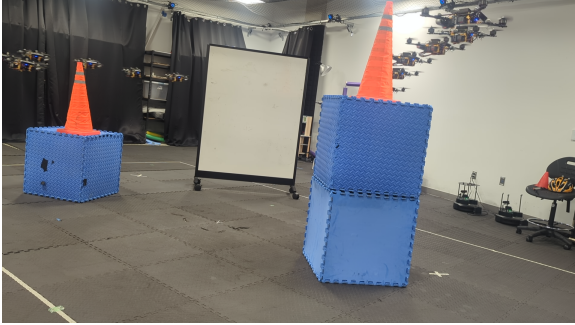
(b)

Fig. 9: Fully onboard flight in an indoor environment: (a) A quadrotor circles a set of obstacles with yaw fixed along the acceleration direction; (b) OREN reconstructs SDF online (shown as gray mesh extracted from the surface), Bubble* plans a bubble corridor (cyan) and optimizes a quadrotor trajectory (red), all computed in real time.

plans safe dynamically feasible trajectories using the non-truncated SDF values (Sec. 6.3). We now validate our integrated approach in real-world flight, with OREN mapping and Bubble* planning running onboard a quadrotor equipped with a Jetson Orin NX. The UAV navigates a previously unseen indoor environment, building the SDF online from streaming depth measurements and planning safe corridors in real time.



(a)



(b)

Fig. 10: (a) Mesh extracted from OREN SDF onboard the quadrotor in real time and a looping trajectory (red) obtained by concatenating several Bubble* plans into a single bubble sequence. (b) The robot executing the planned trajectory.

Fig. 9a shows a representative deployment of our method. OREN runs at 7 Hz to provide the mesh shown in Fig. 9b. Bubble* uses the SDF values from OREN to generate the trajectory shown in Fig. 9b. Beyond point-to-point planning, several Bubble* plans can be concatenated into a single sequence of overlapping bubbles, letting the quadrotor follow a longer, more complex route while preserving the per-segment clearance guarantees of Sec. 5.3. Fig. 10a shows a multi-segment trajectory executed onboard, threading a looping path through the SDF mesh reconstructed during the flight. This trajectory is created by selecting three waypoints and sequentially planning from one to the next. We then concatenate these sequences of bubbles and find a single trajectory through the entire route. Table 5 shows statistics on computation time, clearances, and trajectory lengths for this multi-segment trajectory. The planner maintains a safe distance from

Table 5: Runtime and tracking performance for the trajectory shown in Fig. 10b. Clearance values are shown after accounting for the robot footprint.

Planning time (Bubble*)	76.96 ms
Optimization time	142.33 ms
Total plan time	219.30 ms
Min. obstacle distance, planned	0.313 m
Min. obstacle distance, flown	0.169 m
Flown clearance (p5 / median)	0.297 / 0.804 m
Segments / pieces	3 / 39
Trajectory duration	17.38 s

obstacles as it navigates around them. Fig. 10b visualizes the robot flying this trajectory.

These results show that combining an SDF map with a planner that exploits SDF to construct safe corridors for trajectory optimization makes safe autonomous flight tractable under tight compute and memory budgets.

7 Conclusion

This paper developed an efficient unified approach for SDF reconstruction and SDF-accelerated motion planning for safe autonomous flight. Our mapping method, OREN, reconstructs differentiable non-truncated SDF online by combining an explicit octree prior with an implicit neural residual, attaining the accuracy of neural network methods at the memory and runtime efficiency of volumetric ones. Exploiting the SDF representation, Bubble* constructs a graph of collision-free bubbles, which forms a safe corridor with guarantees of termination, completeness, and failure detection. Such a bubble corridor defines safety constraints for dynamically feasible trajectory optimization. Compared with baselines, Bubble* is the only method that pairs a compact high-clearance corridor with a smooth underlying path, which keeps both its search and its trajectory optimization fast.

We demonstrate the integrated mapping and planning approach running fully onboard a quadrotor in real time. This shows that an accurate implicit map representation makes autonomous flight tractable under tight onboard resources. In our experiments, OREN maintains real-time speed while supplying the continuous differentiable SDF that occupancy-grid mappers cannot, and Bubble* exploits this representation

to issue 91–99% fewer collision checks than A* and to attain the lowest total planning time while matching the shortest path. Together, these results enable the complete mapping and planning loop to run efficiently onboard a Jetson Orin NX.

Funding

We gratefully acknowledge support from ARL DCIST CRA W911NF-17-2-0181 (N. Atanasov, Z. Dai), a research gift fund established by Shield AI (J. Stanley), and the Ministry of Trade, Industry and Energy (MOTIE), Korea, under the Strategic Technology Development Program, supervised by Korea Institute for Advancement of Technology (KIAT) [Grant No. P0026052] (K.M.B. Lee).

Statements and Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

Data Availability A public demonstration of OREN and Bubble*, including code to run the integrated mapping and planning approach, is available at https://github.com/ExistentialRobotics/erl_oren_bubble_star_demo.

Author Contributions OREN extension concepts: ZD, NA; Developing OREN: ZD, QQ, TF; Planner concept: JS, KMBL; Developing planning algorithm: JS, TH; Quadrotor experiments: JS, TH, QQ, SS, CB; Writing main manuscript: JS, ZD; Writing, review and editing: KMBL, NA; Supervision: NA.

References

- Agha-Mohammadi, A.-A., Chakravorty, S., Amato, N.M. (2011). FIRM: Feedback controller-based information-state roadmap-a framework for motion planning under uncertainty. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 4284–4291).
- Allen, R.E., & Pavone, M. (2019). A real-time framework for kinodynamic planning in dynamic environments with application to quadrotor obstacle avoidance. *Robotics and Autonomous Systems*, 115, 174–193, <https://doi.org/10.1016/j.robot.2018.11.017>
- Brunner, M., Rizzi, G., Studiger, M., Siegwart, R., Tognon, M. (2022). A planning-and-control framework for aerial manipulation of articulated objects. *IEEE Robotics and Automation Letters*, 7(4), 10689–10696, <https://doi.org/10.1109/LRA.2022.3191178>
- Curless, B., & Levoy, M. (1996). A volumetric method for building complex models from range images. *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (p. 303–312).
- Dai, Z., Fan, T., Amani, M., Seo, J., Lee, K.M.B., Oh, H., Atanasov, N. (2026). *Kernel-SDF: An open-source library for real-time signed distance function estimation using kernel regression*. (Preprint at <https://arxiv.org/abs/2603.29227>)
- Dai, Z., Qian, Q., Fan, T., Atanasov, N. (2026). OREN: Octree Residual Network for Real-Time Euclidean Signed Distance Mapping. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.
- Deits, R., & Tedrake, R. (2015). Computing large convex regions of obstacle-free space through semidefinite programming. In H.L. Akin, N.M. Amato, V. Isler, & A.F. van der Stappen (Eds.), *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics* (pp. 109–124). Cham: Springer International Publishing.
- Dharmadhikari, M., Dang, T., Solanka, L., Loje, J., Nguyen, H., Khedekar, N., Alexis, K. (2020). Motion primitives-based path planning for fast and agile exploration using aerial robots. *IEEE International Conference on Robotics and Automation (ICRA)* (p. 179–185).

- Faessler, M., Franchi, A., Scaramuzza, D. (2018, April). Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters*, 3(2), 620–626, <https://doi.org/10.1109/LRA.2017.2776353>
- Funk, N., Tarrio, J., Papatheodorou, S., Popović, M., Alcantarilla, P.F., Leutenegger, S. (2021, April). Multi-resolution 3D mapping with explicit free space representation for fast and accurate mobile robot motion planning. *IEEE Robotics and Automation Letters*, 6(2), 3553–3560, <https://doi.org/10.1109/LRA.2021.3061989>
- Gao, F., Wu, W., Gao, W., Shen, S. (2019). Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments. *Journal of Field Robotics*, 36(4), 710–733, <https://doi.org/https://doi.org/10.1002/rob.21842>
- Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y. (2020). Implicit Geometric Regularization for Learning Shapes. *International Conference on Machine Learning*. JMLR.
- Han, L., Gao, F., Zhou, B., Shen, S. (2019). FIESTA: Fast incremental Euclidean distance fields for online motion planning of aerial robots. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (p. 4423–4430).
- Harabor, D., & Grastien, A. (2011). Online graph pruning for pathfinding on grid maps. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI)* (pp. 1114–1119). AAAI Press.
- Hart, P.E., Nilsson, N.J., Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100–107, <https://doi.org/10.1109/TSSC.1968.300136>
- Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3), 189–206, <https://doi.org/10.1007/s10514-012-9321-0> (Software available at <https://octomap.github.io>)
- Jacquet, M., Harms, M., Alexis, K. (2025). Neural NMPC through signed distance field encoding for collision avoidance. *The International Journal of Robotics Research*, , <https://doi.org/10.1177/02783649251401223> (OnlineFirst, December 13, 2025)
- Ji, J., Wang, Z., Wang, Y., Xu, C., Gao, F. (2021). Mapless-planner: A robust and fast planning framework for aggressive autonomous flight without map fusion. *IEEE International Conference on Robotics and Automation (ICRA)* (p. 6315–6321).
- Jiang, C., Zhang, H., Liu, P., Yu, Z., Cheng, H., Zhou, B., Shen, S. (2023). H2-Mapping: Real-time dense mapping using hierarchical hybrid representation. *IEEE Robotics and Automation Letters*, 8(10), 6787–6794, <https://doi.org/10.1109/LRA.2023.3313051>
- Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846–894, <https://doi.org/10.1177/0278364911406761>
- Kähler, O., Adrian Prisacariu, V., Yuheng Ren, C., Sun, X., Torr, P., Murray, D. (2015). Very high frame rate volumetric integration of depth images on mobile devices. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 21(11), 1241–1250, <https://doi.org/10.1109/TVCG.2015.2459891>

- Laina, S.B., Boche, S., Papatheodorou, S., Tzoumanikas, D., Schaefer, S., Chen, H., Leutenegger, S. (2025). Scalable outdoors autonomous drone flight with visual-inertial SLAM and dense submaps built without LiDAR. *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (p. 5440-5447).
- LaValle, S.M., & Kuffner, J.J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5), 378–400, <https://doi.org/10.1177/02783640122067453>
- Lee, B., Zhang, C., Huang, Z., Lee, D.D. (2019). Online continuous mapping using Gaussian process implicit surfaces. *IEEE International Conference on Robotics and Automation (ICRA)* (p. 6884–6890).
- Lee, K.M.B., Dai, Z., Le Gentil, C., Wu, L., Atanasov, N., Vidal-Calleja, T. (2024). *Safe bubble cover for motion planning on distance fields*. (Preprint at <https://arxiv.org/abs/2408.13377>)
- Li, Y., Chi, X., Razmjoo, A., Calinon, S. (2024). Configuration space distance fields for manipulation planning. *Robotics: Science and Systems (RSS)*.
- Li, Y., Zhang, Y., Razmjoo, A., Calinon, S. (2024). Representing robot geometry as distance fields: Applications to whole-body manipulation. *IEEE International Conference on Robotics and Automation (ICRA)* (p. 15351-15357).
- Likhachev, M., & Ferguson, D. (2009). Planning long dynamically feasible maneuvers for autonomous vehicles. *The International Journal of Robotics Research*, 28(8), 933–945, <https://doi.org/10.1177/0278364909340445>
- Liu, P., Zhang, K., Tateo, D., Jauhri, S., Peters, J., Chalvatzaki, G. (2022). Regularized deep signed distance fields for reactive motion generation. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (p. 6673-6680).
- Liu, S., Atanasov, N., Mohta, K., Kumar, V. (2017). Search-based motion planning for quadrotors using linear quadratic minimum time control. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (p. 2872-2879).
- Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C.J., Kumar, V. (2017). Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. *IEEE Robotics and Automation Letters*, 2(3), 1688-1695, <https://doi.org/10.1109/LRA.2017.2663526>
- Long, K., Yi, Y., Dai, Z., Herbert, S., Cortés, J., Atanasov, N. (2026). Sensor-based distributionally robust control for safe robot navigation in dynamic environments. *The International Journal of Robotics Research*, 45(2), 328-351, <https://doi.org/10.1177/02783649251352000>
- Majumdar, A., & Tedrake, R. (2017). Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8), 947–982, <https://doi.org/10.1177/0278364917712421>
- Marcucci, T., Nobel, P., Tedrake, R., Boyd, S. (2024). Fast path planning through large collections of safe boxes. *IEEE Transactions on Robotics*, 40, 3795-3811, <https://doi.org/10.1109/TRO.2024.3434168>
- Marcucci, T., Petersen, M., von Wrangel, D., Tedrake, R. (2023). Motion planning around obstacles with convex optimization. *Science Robotics*, 8(84), eadf7843, <https://doi.org/10.1126/scirobotics.adf7843>

- Mellinger, D., & Kumar, V. (2011). Minimum snap trajectory generation and control for quadrotors. *IEEE International Conference on Robotics and Automation (ICRA)* (p. 2520-2525).
- Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R. (2020). NeRF: Representing scenes as neural radiance fields for view synthesis. *European Conference on Computer Vision (ECCV)* (p. 405–421).
- Millane, A., Oleynikova, H., Wirbel, E., Steiner, R., Ramasamy, V., Tingdahl, D., Siegwart, R. (2024). nvblox: GPU-accelerated incremental signed distance field mapping. *IEEE International Conference on Robotics and Automation (ICRA)* (p. 2698-2705).
- Mohta, K., Watterson, M., Mulgaonkar, Y., Liu, S., Qu, C., Makineni, A., ... Kumar, V. (2018). Fast, autonomous flight in GPS-denied and cluttered environments. *Journal of Field Robotics*, 35(1), 101-120, <https://doi.org/https://doi.org/10.1002/rob.21774>
- Mueller, M.W., Hehn, M., D’Andrea, R. (2015). A computationally efficient motion primitive for quadcopter trajectory generation. *IEEE Transactions on Robotics*, 31(6), 1294-1310, <https://doi.org/10.1109/TRO.2015.2479878>
- Museth, K., Budsberg, J., Sirois-Vigneux, A., Hurst, G., Williams, F., Pradhana, A., ... Avramoussis, N. (2025). OpenVDB. *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Courses*.
- Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., ... Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (p. 127-136).
- Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., Nieto, J. (2017). Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (p. 1366–1373).
- Ortiz, J., Clegg, A., Dong, J., Sucar, E., Novotny, D., Zollhoefer, M., Mukadam, M. (2022). iSDF: Real-time neural signed distance fields for robot perception. *Robotics: Science and Systems (RSS)*.
- Pan, Y., Kompis, Y., Bartolomei, L., Mascaro, R., Stachniss, C., Chli, M. (2022). Voxfield: Non-projective signed distance fields for online planning and 3D reconstruction. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (p. 5331-5338).
- Pan, Y., Zhong, X., Wiesmann, L., Posewsky, T., Behley, J., Stachniss, C. (2024). PIN-SLAM: LiDAR SLAM using a point-based implicit neural representation for achieving global map consistency. *IEEE Transactions on Robotics*, 40, 4045-4064, <https://doi.org/10.1109/TRO.2024.3422055>
- Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S. (2019). DeepSDF: Learning continuous signed distance functions for shape representation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 165-174).
- Ren, Y., Zhu, F., Liu, W., Wang, Z., Lin, Y., Gao, F., Zhang, F. (2022). Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (p. 6332-6339).
- Ren, Y., Zhu, F., Lu, G., Cai, Y., Yin, L., Kong, F., ... Zhang, F. (2025). Safety-assured high-speed navigation for MAVs. *Science Robotics*, 10(98), eado6187, <https://doi.org/10.1126/scirobotics.ado6187>

- Richter, C., Bry, A., Roy, N. (2016). Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In M. Inaba & P. Corke (Eds.), *Robotics Research: The 16th International Symposium ISRR* (pp. 649–666). Cham: Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-319-28872-7_37
- Ryll, M., Ware, J., Carter, J., Roy, N. (2019). Efficient trajectory planning for high speed flight in unknown environments. *International Conference on Robotics and Automation (ICRA)* (p. 732-738).
- Song, Y., Steinweg, M., Kaufmann, E., Scaramuzza, D. (2021). Autonomous drone racing with deep reinforcement learning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (p. 1205-1212).
- Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., ... Newcombe, R. (2019). *The Replica dataset: A digital replica of indoor spaces*. (Preprint at <https://arxiv.org/abs/1906.05797>)
- Sucan, I.A., Moll, M., Kavraki, L.E. (2012). The Open Motion Planning Library. *IEEE Robotics and Automation Magazine*, 19(4), 72-82, <https://doi.org/10.1109/MRA.2012.2205651>
- Takikawa, T., Litalien, J., Yin, K., Kreis, K., Loop, C., Nowrouzezahrai, D., ... Fidler, S. (2021). Neural geometric level of detail: Real-time rendering with implicit 3D shapes. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 11353-11362).
- Tian, Y., Cao, H., Kim, S., Atanasov, N. (2025). MISO: Multiresolution submap optimization for efficient globally consistent neural implicit reconstruction. *Robotics: Science and Systems (RSS)*.
- Tordesillas, J., & How, J.P. (2022). FASTER: Fast and safe trajectory planner for navigation in unknown environments. *IEEE Transactions on Robotics*, 38(2), 922-938, <https://doi.org/10.1109/TRO.2021.3100142>
- Varadhan, S.R.S. (1967). On the behavior of the fundamental solution of the heat equation with variable coefficients. *Communications on Pure and Applied Mathematics*, 20(2), 431–455, <https://doi.org/10.1002/cpa.3160200210>
- Vasilopoulos, V., Garg, S., Huh, J., Lee, B., Isler, V. (2024). HIO-SDF: Hierarchical incremental online signed distance fields. *IEEE International Conference on Robotics and Automation (ICRA)* (p. 17537-17543).
- Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W. (2021). NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.
- Wang, Z., Wang, C., Yoshino, T., Tao, S., Fu, Z., Li, T.-M. (2025). HotSpot: Signed distance function optimization with an asymptotically sufficient condition. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (p. 1276-1286).
- Wang, Z., Zhou, X., Xu, C., Gao, F. (2022). Geometrically constrained trajectory optimization for multicopters. *IEEE Transactions on Robotics*, 38(5), 3259-3278, <https://doi.org/10.1109/TRO.2022.3160022>
- Wu, L., Lee, K.M.B., Liu, L., Vidal-Calleja, T. (2021). Faithful Euclidean distance field from log-Gaussian process implicit surfaces. *IEEE Robotics and Automation Letters*, 6(2), 2461-2468, <https://doi.org/10.1109/LRA.2021.3061356>
- Wu, L., Le Gentil, C., Vidal-Calleja, T. (2025). VDB-GPDF: Online Gaussian process distance field with VDB structure. *IEEE*

Robotics and Automation Letters, 10(1), 374-381, <https://doi.org/10.1109/LRA.2024.3505814>

Zhang, L., Camurri, M., Wisth, D., Fallon, M. (2021). *Multi-camera LiDAR inertial extension to the Newer College dataset*. (Preprint at <https://arxiv.org/abs/2112.08854>)

Zhou, B., Gao, F., Wang, L., Liu, C., Shen, S. (2019). Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4), 3529-3536, <https://doi.org/10.1109/LRA.2019.2927938>