Kernel-SDF: An Open-Source Library for Real-Time Signed Distance Function Estimation using Kernel Regression

Zhirui Dai[1,*] Tianxing Fan[1,*] Mani Amani[1] Jaemin Seo[2]
Ki Myung Brian Lee[1] Hyondong Oh[2] Nikolay Atanasov[1]

***Abstract*—Accurate and efficient scene representation is crucial for robotic tasks such as motion planning, manipulation, and navigation. Signed distance functions (SDFs) have emerged as a powerful representation for encoding distance to obstacle boundaries, enabling efficient collision-checking and trajectory optimization. However, existing methods are limited for large-scale uncertainty-aware SDF estimation from streaming sensor data: voxel-based approaches have fixed resolution and lack uncertainty quantification, neural network methods require significant training time, and Gaussian process (GP) methods struggle with scalability, sign estimation, and uncertainty calibration. In this letter, we develop an open-source library, Kernel-SDF, using kernel regression to learn SDF with calibrated uncertainty in real-time. It combines a front-end learning a continuous occupancy field via kernel regression with a back-end that estimates accurate SDF via GP regression using samples from the front-end surface boundaries. Kernel-SDF provides accurate SDF, gradient, uncertainty, and mesh construction in real-time. Evaluations show it achieves superior accuracy over existing methods while maintaining real-time performance, making it suitable for robotics tasks requiring reliable uncertainty-aware geometry.**

***Index Terms*—Mapping, Range Sensing, Signed Distance Function, Kernel Regression.**

I. Introduction

GEOMETRIC scene representations play a key role for many robot autonomy tasks, including mapping, localization, motion planning, manipulation, and navigation. An ideal representation should meet several requirements: 1) accuracy to reflect the scene structure, 2) efficiency to allow real-time processing of streaming sensor data, 3) scalability to large scenes, 4) robustness to noise, 5) uncertainty quantification to support risk-aware decision making, 6) differentiability to support gradient-based optimization, 7) adaptability to dynamic scenes, and 8) compatibility with different sensors.

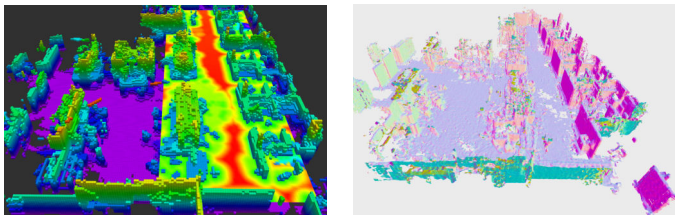
Manuscript received: March 30, 2026; Revised June 17, 2026; Accepted July 6, 2026. This paper was recommended for publication by Editor Ayoung Kim upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported by ARL DCIST CRA W911NF-17-2-0181, NSF FRR CAREER 2045945, the Ministry of Trade, Industry, and Energy (MOTIE), Korea, under the Strategic Technology Development Program supervised by the Korea Institute for Advancement of Technology (KIAT) [Grant No. P0026052], and Korea Institute of Planning and Evaluation for Technology in Food, Agriculture, Forestry(IPET) through (Smart Farm Innovation Technology Development Program), funded by Ministry of Agriculture, Food and Rural Affairs(MAFRA) (RS-2025-02219411).

[1]Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA.

[2]Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34051, Republic of Korea. Emails: {zhdai,t2fan,mamai5250,kmblee,natanasov}@ucsd.edu; {jam.seo,h.oh}@kaist.ac.kr.

[*]Equal contribution. Code and additional results: https://github.com/ExistentialRobotics/kernel_sdf.



(a) Signed distance function (b) Reconstructed mesh

Fig. 1: Example of SDF estimation, visualized as a heatmap in (a), and mesh reconstruction in (b) of a large-scale environment, obtained using Kernel-SDF.

Many effective representations have been proposed: occupancy maps [1], point clouds [2], meshes [3], and implicit functions such as SDFs [4]–[19] and neural radiance fields (NeRFs) [20]. Among these, SDFs have attracted attention for providing the distance to the nearest obstacles, useful for mapping [4], [19] and motion planning [21]. Moreover, differentiable SDF methods can define safety constraints in navigation [22] and manipulation [23].

Various methods have been proposed to learn SDFs from range sensors such as LiDAR or depth cameras. Voxel-based methods [4], [5], [7], [11] often focus only on truncated SDF (TSDF) estimation, though some [5], [7], [24], [25] reconstruct non-truncated (Euclidean) SDF on a discrete grid. These methods maintain a dense, fixed-resolution approximation of (T)SDF, which trades off accuracy against efficiency and prevents differentiability. In contrast, neural network methods [6], [13]–[16], [19] learn SDF as a continuous function, providing accurate and differentiable approximations without an explicit grid. However, the significant training time they require to converge makes them unsuitable for online learning.

Gaussian process (GP) methods [8], [9], [17], [18] are non-parametric and can estimate accurate Euclidean SDF, but scale poorly with environment size, lack robust sign estimation, and report variance inconsistent with actual SDF errors. For scalability, octree [9] and OpenVDB [18] data structures split the training data to train multiple GPs. For consistent sign and variance, GMMGP [17] uses an SDF prior from a hierarchical Gaussian mixture model, but this prior yields increasingly large errors away from the surface.

In this letter, we develop an open-source library, Kernel-SDF, for SDF estimation that meets the aforementioned requirements and addresses the limitations of existing GP methods. Kernel-SDF maintains an octree to split the estimation into smaller areas, estimates the occupancy/sign in each octant using a Bayesian Hilbert map (BHM) [26], and reconstructs SDF via GP regression from the BHM surface samples. The library is thus organized as a surface-estimation

front-end performing BHM occupancy estimation and an SDF-prediction back-end performing GP regression. The front-end learns a continuous occupancy field for robust sign estimation and provides surface point samples via the marching cubes algorithm [27]. The back-end uses these samples to train SDF GPs in each octant with integrated uncertainty quantification. We introduce several optimizations for real-time performance and map consistency: spatial partitioning and priority-queue-based updates for efficient computation, and BHM weight synchronization for consistency across neighboring local maps.

Kernel-SDF provides accurate SDF, gradient, uncertainty, and mesh construction in real time (see Fig. 1). In summary, our contributions include:

1) an open-source C++ implementation for 2D/3D SDF learning, with Python and ROS1/ROS2 interfaces;
2) a highly-optimized kernel regression method for real-time SDF learning via two components: a BHM front-end for continuous occupancy learning and uncertainty-aware surface extraction, and a GP back-end for SDF and gradient prediction with uncertainty;
3) five strategies for efficiency and robustness: octree-based partitioning for local region learning, optimized data processing for rapid training set generation, weight synchronization for map consistency, adaptive scaling for GP stability, and priority-queue-based updates to balance accuracy and responsiveness.

II. Related Work

SDF learning methods can be categorized as voxel-based, neural-net-based, and Gaussian-process-based.

A. Voxel Methods

Many methods learn SDF using a grid [28]. KinectFusion [4] pioneered real-time 3D reconstruction from depth sensors based on projective TSDF. Follow-up works such as Voxblox [5] extended this to Euclidean SDF via breadth-first-search (BFS) integration from the occupied voxels. However, projective TSDF inaccurately estimates the true SDF as distance along camera rays, and BFS integration adds further errors. FIESTA [7] integrates SDF estimation with an occupancy map, replacing the BFS-based distance with the distance to the nearest occupied voxel. VoxField [12] formulates non-projective TSDF, though its accuracy relies on estimated surface normals and SDF gradients. A common limitation is the use of fixed-resolution grids, which restricts scalability in larger environments. VDBblox [24] and nvblox [25] address this via OpenVDB [29] or GPU-based voxel hashing. However, voxel-based methods lack uncertainty quantification and are usually not differentiable.

B. Neural Network Methods

Neural network methods are differentiable by design and can achieve highly accurate SDF reconstruction. DeepSDF [6] uses an MLP with latent features to learn the SDF of objects. NeuS [10] and NeuS2 [14] combine SDF with NeRFs [20] to jointly learn SDF and rendering. PIN-SLAM [16] and MISO [19] perform simultaneous localization and mapping (SLAM) on an SDF map encoded with neural features. H_2-Mapping [15] co-optimizes a voxel SDF prior and a neural TSDF residual with volumetric rendering. However, most neural network methods cannot meet real-time requirements in novel environments due to their significant training time. Some works, like H_2-Mapping [15] and PIN-SLAM [16], learn neural SDF online but sacrifice substantial accuracy.

C. Gaussian Process Methods

GPs are suitable for online SDF learning but face the high computation cost of matrix inversion as the environment scale increases. To mitigate this, GPIS [8] employs octree-based data partitioning and trains a GP per local partition. However, its SDF estimates quickly revert to the zero-mean prior away from the surface. Log-GPIS [9] exploits a connection between the heat kernel and the unsigned distance function (UDF) to formulate UDF GP regression in log space. However, the log transformation ignores sign information and distorts variance estimation, causing uncertainty to explode far from the surface. Its surface estimation is also sensitive to sensor noise and dynamic changes. VDB-GPDF [18] improves surface estimation through OpenVDB TSDF fusion but provides no sign estimates or reliable uncertainty quantification. To overcome these limitations, Kernel-SDF uses a continuous occupancy field front-end based on BHM [26], which provides robust sign prediction and accurate surface estimation even in dynamic or noisy environments. Our GP back-end then learns SDF from the surface samples and formulates an uncertainty estimate that respects the SDF errors, delivering precise SDF and gradient predictions with reliable uncertainty quantification.

III. Problem Statement

Consider a robot equipped with a depth camera or LiDAR sensor operating in an n-dimensional environment ($n = 2, 3$) with occupied space $\mathcal{O} \subset \mathbb{R}^n$. The sensor provides point cloud measurements $\mathcal{P}_t \subset \mathbb{R}^n$ of $\mathcal{O}$ from known (or estimated) sensor positions $\mathbf{p}_t \in \mathbb{R}^n$ and orientations $\mathbf{R}_t \in SO(n)$ at time steps t. Using these data, our objective is to learn an SDF representation of the occupied space $\mathcal{O}$:

$$d(\mathbf{x}) = \begin{cases} \min_{\mathbf{y}\in\partial\mathcal{O}} \|\mathbf{x}-\mathbf{y}\|_2, & \mathbf{x} \notin \mathcal{O}, \\ -\min_{\mathbf{y}\in\partial\mathcal{O}} \|\mathbf{x}-\mathbf{y}\|_2, & \mathbf{x} \in \mathcal{O}, \end{cases} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a query point and $\mathbf{y}$ is a surface point. The SDF $d(\mathbf{x})$ encodes the surface $\partial\mathcal{O}$ as its zero-level set. Further, the gradient $\nabla d(\mathbf{x})$ has unit norm whenever differentiable:

$$d(\mathbf{x}) = 0, \ \forall \mathbf{x} \in \partial\mathcal{O} \text{ and } \|\nabla d(\mathbf{x})\|_2 = 1, \text{ a.e. } \mathbf{x} \in \mathcal{O}. \tag{2}$$

We focus on estimating $d(\mathbf{x})$ using the streaming point cloud measurements $\mathcal{P}_t$ and the associated sensor poses $(\mathbf{p}_t, \mathbf{R}_t)$.

IV. Kernel-SDF

Kernel-SDF consists of a surface-estimation front-end using BHM [26] (Sec. IV-A) and an SDF-prediction back-end using GP regression [30] (Sec. IV-B). Both are kernel regression methods supporting continuous representations with uncertainty quantification. The BHM front-end fuses free and occupied measurements into a continuous occupancy field, which provides sign and per-surface-point uncertainty information to the SDF back-end. The SDF back-end provides a differentiable SDF with closed-form gradients and calibrated variance. However, applying kernel regression over the entire environment is

computationally expensive and impractical for real-time use. Kernel-SDF addresses this by partitioning the environment into local regions via an octree and applying kernel regression within each region with additional optimizations (Fig. 2a).

A. Surface Estimation Front-End

In this section, we review Bayesian Hilbert map (BHM) [26] and describe our extensions to achieve efficient updates, sign prediction, and surface point extraction.

1) **Review of Bayesian Hilbert Map**: BHM learns a continuous log-odds field $l(\mathbf{x})$ using kernel regression in Hilbert space. The log-odds $l(\mathbf{x})$ at position $\mathbf{x}$ is the ratio between the probabilities of being occupied and being free. BHM uses linear regression to learn the log-odds field, $l(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x})$, with weights $\mathbf{w}$ and features $\boldsymbol{\phi}(\mathbf{x})$. To obtain the features, a query $\mathbf{x}$ is mapped into Hilbert space using a radial basis function (RBF) kernel $k(\cdot,\cdot)$ evaluated with respect to a set of hinged points $\{\tilde{\mathbf{x}}_i\}_{i=1}^M$ organized in a grid structure:

$$\boldsymbol{\phi}(\mathbf{x}) = [k(\mathbf{x},\tilde{\mathbf{x}}_1),\cdots,k(\mathbf{x},\tilde{\mathbf{x}}_M),1]^\top. \tag{3}$$

Intuitively, the hinged points are fixed references tiling the region, and $\boldsymbol{\phi}(\mathbf{x})$ collects the similarity of the query $\mathbf{x}$ to each of them. The key step is to estimate the weights $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Given a point cloud $\mathcal{P}$, we generate a dataset $\mathcal{D}_{\text{BHM}} = \{\mathbf{x}_k, y_k\}_{k=1}^K \subset \mathbb{R}^3 \times \{-1,1\}$ by sampling free-space samples, $\mathbf{x}_k, y_k = -1$, along each segment between the sensor position $\mathbf{p}$ and the hit point $\mathbf{x} \in \mathcal{P}$ and adding the hit points as occupied samples, $\mathbf{x}_k = \mathbf{x}, y_k = 1$. Given $\mathcal{D}_{\text{BHM}}$, BHM uses expectation maximization (EM) [26] to update the weights mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ and predict the log-odds as $l(\mathbf{x}) = \boldsymbol{\mu}^\top \boldsymbol{\phi}(\mathbf{x})$.

2) **Octree of BHMs**: Since M scales with scene size, BHM updates become prohibitive. We address this by partitioning the scene with an octree, maintaining a local BHM only for octants with occupied voxels, which greatly reduces the necessary hinged points. To generate $\mathcal{D}_{\text{BHM}}$, we prune rays outside the local BHM's viewing frustum and sample only within its bounding box (Fig. 2b). We also assume independent hinged point weights, reducing $\boldsymbol{\Sigma}$ to a diagonal matrix and enabling element-wise EM updates that drastically lower complexity. Finally, we use a sparse representation of $\boldsymbol{\phi}(\mathbf{x})$, exploiting that the kernel vanishes when $\mathbf{x}$ is far from $\tilde{\mathbf{x}}$.

To maintain consistency between local BHMs, we overlap neighbors by a margin equal to the hinged point spacing and extend the sampling area when generating $\mathcal{D}_{\text{BHM}}$ to sufficiently update boundary weights (Fig. 2b). For global consistency, we sync weights (Fig. 2c): hinged point weights are categorized as core (unique to the local BHM), managed (owned locally but shared), and unmanaged (owned by a neighbor). Periodically syncing managed weights to their unmanaged counterparts ensures consistent occupancy predictions and accelerates convergence by broadcasting local geometry to neighbors.

3) **Sign Prediction**: The SDF sign of a near-surface position $\mathbf{x}_*$ can be predicted by the BHM as:

$$s(\mathbf{x}_*) = \begin{cases} +1, & l(\mathbf{x}_*) < \tau, \\ -1, & l(\mathbf{x}_*) \geq \tau, \end{cases} \tag{4}$$

where τ is a threshold (ideally 0) learned from data. Empirical evidence (Fig. 2d) shows τ varies with sensor viewing direction (e.g., ground vs. walls). Thus, we update τ online using a moving average of the log-odds from K hit samples:

$$\tau \leftarrow (1-\alpha)\tau + \frac{\alpha}{K}\sum_{k=1}^{K} l(\mathbf{x}_k), \quad \alpha \in (0,1], \tag{5}$$

where $\alpha \in (0,1]$ is the learning rate.

4) **Online Mesh Extraction and Surface Point Uncertainty Estimation**: To recover surface points $\{\mathbf{x}_i\}_i$ at the τ-level set of $l(\mathbf{x})$, we use the marching cubes algorithm [27] on a grid of BHM log-odds values. We process only grid cells containing point cloud points and their neighbors. The uncertainty for each point $\mathbf{x}_i$ is computed as:

$$\sigma^2_{\mathbf{x}_i} = \beta(\tau - l(\mathbf{x}_i))^2 / \|\nabla l(\mathbf{x}_i)\|_2^2, \tag{6}$$

where $\beta > 0$ is a hyperparameter and τ is updated via (5).

We derive (6) via a first-order Taylor expansion of $l(\mathbf{x}_*)$ at the extracted point $\mathbf{x}_i$ relative to the true surface point $\mathbf{x}_*$:

$$\underbrace{l(\mathbf{x}_*)}_{\tau} \approx l(\mathbf{x}_i) + \underbrace{\nabla l(\mathbf{x}_i)}_{\nabla l}^\top \underbrace{(\mathbf{x}_* - \mathbf{x}_i)}_{\delta\mathbf{x}}, \tag{7}$$

and assuming the error $\delta\mathbf{x}$ aligns primarily with the gradient direction, i.e., $\delta\mathbf{x} \approx \nabla l / \|\nabla l\|_2 \delta_x$. Intuitively, uncertainty rises when $l(\mathbf{x}_i)$ deviates from τ (higher surface extraction error) or when the gradient is small (flat and noise-sensitive field). Hence, (6) reflects how confident the front-end is about each point, which allows the back-end to weight each point's contribution during SDF learning and uncertainty quantification.

B. SDF Prediction Back-End

In this section, we review the log-GP method for unsigned distance function (UDF) estimation, extend it to other kernel types to improve its numerical stability, and introduce a softmin SDF uncertainty quantification approach.

1) **Log-GP for Unsigned Distance Estimation:** Given the front-end sign prediction $s(\mathbf{x})$ in (4), we decompose the SDF as $d(\mathbf{x}) = s(\mathbf{x}) \cdot u(\mathbf{x})$, where $u(\mathbf{x}) \geq 0$ is a UDF. Thus, the back-end only needs to learn the UDF $u(\mathbf{x})$. Varadhan's formula [31] shows that the UDF is related to the short-time heat diffusion equation:

$$u(\mathbf{x}) = \lim_{t\to 0}\{-\sqrt{t}\log v(\mathbf{x},t)\}, \tag{8}$$

where $v(\mathbf{x},t)$ is the solution to the heat equation $\partial v/\partial t = \Delta v$ with boundary condition $v(\mathbf{x},t) = 1$ for $\mathbf{x} \in \partial\mathcal{O}$. Log-GPIS [9] shows that GP regression of $f(\mathbf{x}) = \exp(-\lambda u(\mathbf{x}))$ with Matérn 3/2 kernel can approximate (8) well when the kernel scale $\sqrt{3}/\lambda$ is small. The UDF can be recovered as $u(\mathbf{x}) \approx -\log \mathbb{E}[f(\mathbf{x})]/\lambda$. The GP $f(\mathbf{x})$ is trained on the dataset of surface points $\mathcal{D}_{\text{surf}} = \{\mathbf{x}_i, \sigma_i^2\}_{i=1}^N$ with labels $f(\mathbf{x}_i) = \exp(-\lambda \cdot 0) = 1$. The posterior mean and variance of $f(\mathbf{x}_*)$ at a query point $\mathbf{x}_*$ are given by:

$$\begin{aligned} \begin{bmatrix} \hat{f}(\mathbf{x}_*) \\ \nabla\hat{f}(\mathbf{x}_*) \end{bmatrix} &= \begin{bmatrix} \mathbf{k}_*^\top \\ \nabla_{\mathbf{x}_*}^\top \mathbf{k}_* \end{bmatrix} (\mathbf{K}+\boldsymbol{\Sigma})^{-1}\mathbf{1}, \\ \mathbb{V}[\hat{f}(\mathbf{x}_*)] &= k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^\top (\mathbf{K}+\boldsymbol{\Sigma})^{-1}\mathbf{k}_*, \end{aligned} \tag{9}$$

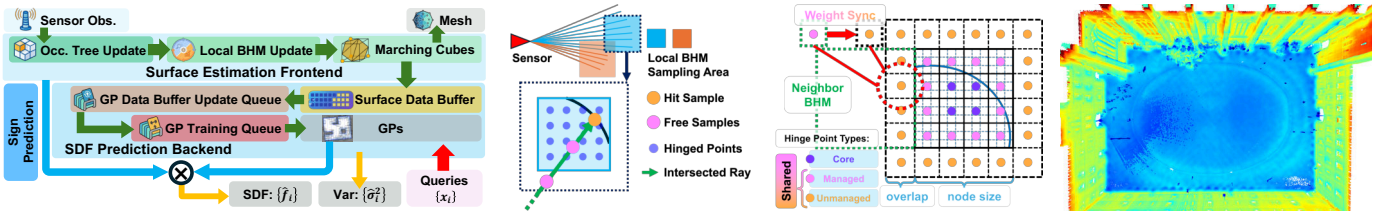
where $\mathbf{k}_* \in \mathbb{R}^N$ and $\mathbf{K} \in \mathbb{R}^{N\times N}$ are calculated by the kernel:

$$k_{*,i} = k(\mathbf{x}_i,\mathbf{x}_*) \quad 1 \leq i \leq N, \tag{10}$$

$$K_{ij} = k(\mathbf{x}_i,\mathbf{x}_j) \quad 1 \leq i \leq N, 1 \leq j \leq N, \tag{11}$$

and $\Sigma_{ii} = \sigma_i^2$. The UDF gradient can also be obtained as $\nabla u(\mathbf{x}) = -\nabla\hat{f}(\mathbf{x})/\|\nabla\hat{f}(\mathbf{x})\|_2$.

Each octant with a BHM also maintains a log-GP for SDF estimation. For inter-GP consistency, we collect surface points within boxes larger than the BHM sampling box and fuse



(a) Overview (b) Sample generation (c) Weight sync (d) Surface log-odds

Fig. 2: (a) Overview of Kernel-SDF: the front-end estimates surface points and normals from point clouds via BHM and marching cubes; these train the back-end, which learns SDF and its gradient using multiple GPs in an octree. (b) For each ray hitting or crossing the sampling region, we generate occupied and free samples to update the BHM. (c) Weight sync between neighboring BHMs: managed weights are synced to unmanaged weights. (d) Surface points colored by BHM log-odds; the ground has lower log-odds than the walls.

TABLE I: Kernel functions and their nonlinear transforms.

Kernel	$k(r)$	$r(\hat{f})$
RBF	$\exp\left(-r^2/2l^2\right)$	$\sqrt{-2l^2\log\hat{f}}$
Matérn 3/2	$\left(1+\sqrt{3}r/l\right)\exp\left(-\sqrt{3}r/l\right)$	$-l\log\hat{f}/\sqrt{3}$

the K nearest GPs via $\hat{u}(\mathbf{x}_*) = \min_k u_k(\mathbf{x}_*)$. Because the collection boxes overlap, neighboring GPs share samples near boundaries and yield almost identical predictions to a single global GP[1]. Therefore, the min-fusion of multiple local GPs yields continuous results across octant boundaries.

2) **Extension to Other Kernels:** While the Matérn 3/2 kernel is standard for log-GP, we show that more efficient kernels such as the RBF can be used by identifying the posterior mean $\hat{f}$ with a softmin-like function, i.e., the softmin approximation of the UDF over $\mathcal{D}_{\text{surf}}$: $u(\mathbf{x}_*) \approx h(\mathbf{x}_*, \{\mathbf{x}_i\}_{i=1}^N) = \mathbf{s}^\top\mathbf{z}$, where $z_i = \|\mathbf{x}_i - \mathbf{x}_*\|_2$ and $s_i = e^{-\alpha z_i}/\sum_i^N e^{-\alpha z_i}, \alpha > 0$. Since every label equals 1, the posterior mean (9) reduces to $\hat{f}_* = \mathbf{k}_*^\top(\mathbf{K}+\mathbf{\Sigma})^{-1}\mathbf{1}$ with $\mathbf{\Sigma} = \sigma^2\mathbf{I}$. In the *weakly-correlated* regime (small kernel scale relative to the sample spacing, $\mathbf{K} \to \mathbf{I}$), $\hat{f}_* \approx \frac{1}{1+\sigma^2}\sum_i k_{*i} = \frac{S}{1+\sigma^2}\max_i k_{*i}$ with $S = \sum_i k_{*i}/\max_i k_{*i} \geq 1$, so

$$\log\hat{f}_* \approx \log\max_i k_{*i} + \log S - \log(1+\sigma^2), \tag{12}$$

recovering the softmin as $S \to 1$ (a single dominant kernel bump). In the *strongly-correlated* regime (dense sampling, $\mathbf{K} \to \mathbf{1}\mathbf{1}^\top$), for a query at distance z from a tight surface cluster the Sherman–Morrison identity gives $\hat{f}_* \approx \frac{S}{\sigma^2+N}\max_i k_{*i} \approx \frac{N}{\sigma^2+N}k(z)$ with $S \to N$, so

$$\log\hat{f}_* \approx \log k(z) - \log\frac{\sigma^2+N}{N}, \tag{13}$$

where the bumps merge into one wide bump (uniform-weight, zero-temperature softmin). In both limits, the UDF is recovered by the kernel-specific transform $r(\hat{f})$ in Table I. In the weakly-correlated regime, the softmin inflation $\log S$ biases the recovered UDF slightly below the true distance, a deterministic under-estimate that shrinks with the kernel scale. In the strongly-correlated regime, it is absorbed by the σ^2+N normalization. The noise σ^2 down-weights uncertain samples and offsets the under-estimate. We pick the RBF kernel for its efficiency and better accuracy, shown in the ablation of Sec. V-E. While larger λ (smaller kernel scale) improves the approximation of (8), it triggers numerical underflow of $k(\mathbf{x}_*, \mathbf{x}_i)$. We address this by subtracting a constant d from the kernel exponent and adding it back in the inverse transforms: $r(\hat{f}') = \sqrt{-2l^2\log\hat{f}' + d^2}$ for RBF and $r(\hat{f}') = -\frac{l}{\sqrt{3}}\log\hat{f}' + d$ for Matérn 3/2.

3) **Consistent Uncertainty Quantification via Softmin-based Fusion:** Standard first-order variance propagation, $\mathbb{V}[\hat{d}] \approx (r')^2\mathbb{V}[\hat{f}]$, fails in log-GP: as $\mathbf{x}_*$ moves away from $\mathcal{D}_{\text{surf}}$, $\mathbb{V}[\hat{f}] \to 1$ and $|r'| \to \infty$, so $\mathbb{V}[\hat{d}] \to \infty$. Instead, we leverage the softmin property of the log-GP posterior to propagate surface point uncertainties σ_i^2 directly through the softmin.

The variances of $u(\mathbf{x}_*)$ and $\nabla u(\mathbf{x}_*)$ then follow from a first-order approximation of the softmin:

$$\mathbb{V}[u(\mathbf{x}_*)] \approx \sum_{i=1}^N \|\nabla_{\mathbf{x}_i} h\|_2^2\,\sigma_i^2, \tag{14}$$

$$\mathbb{V}[\nabla_k u(\mathbf{x}_*)] \approx \sum_{i=1}^N \left\|\nabla_{\mathbf{x}_i} g_k(\mathbf{x}_*, \{\mathbf{x}_i\}_{i=1}^N)\right\|_2^2\,\sigma_i^2, \tag{15}$$

where $\nabla_{\mathbf{x}_i} h = -s_i(\alpha\mathbf{s}^\top\mathbf{z} - \alpha z_i + 1)(\mathbf{x}_* - \mathbf{x}_i)/z_i$, $\mathbf{g}(\mathbf{x}_*, \{\mathbf{x}_i\}_{i=1}^N) = \nabla_{\mathbf{x}_*} h(\mathbf{x}_*, \{\mathbf{x}_i\}_{i=1}^N)$. This directly propagates the surface point location uncertainty to the SDF and gradient predictions. The deterministic-sign treatment is justified empirically: the occupancy field transitions sharply at the surface (on Replica `room0`, 97% of g.t. surface points have $q < 0.1$ or $q > 0.9$, with a median transition band ≈ 3.6 mm, far below the map resolution), so q is effectively binary off the surface. The sign mixture collapses, giving $\mathbb{V}[\hat{d}(\mathbf{x})] = \mathbb{V}[\hat{u}(\mathbf{x})]$. For near-surface points without a sharp transition, $q \in (0.1, 0.9)$, the deterministic-sign treatment theoretically underestimates the predictive variance.

C. Priority Queue Based Update

Beyond the core BHM and GP optimizations, we use priority-queue-based updates for efficiency. Running surface extraction, GP buffer updates, and GP training after every BHM update is wasteful given frequent sensor measurements, so we delay these processes until necessary using a three-queue priority system (Fig. 2a). To stay responsive, frequently queried GPs get higher priority for buffer updates and retraining, while the oldest BHM marching requests are prioritized to process likely stable regions. This strategy achieves a balance between responsiveness and map correctness.

V. Experiments

We compare Kernel-SDF to four baselines: Voxblox [5], FIESTA [7], iSDF [13], and VDB-GPDF [18] on the Replica [32], Cow&Lady [5] and Newer College [33]. For Replica, we simulate depth noise using an axial model with $\sigma = 0.0025z^2$ [34]. All baselines use the same input point clouds, poses, and frame rates as Kernel-SDF. Each grid-based method's

[1]The difference between the min-fusion distance and gradient and a global GP was around $\sim 10^{-8}$ m and $< 10^{-6}$ rad for a 2D benchmark.

TABLE II: Mesh reconstruction metrics on Replica, Cow&Lady (C.L. for the scene and Cow for the cow only), and Newer College (N.C.). When computing the ratio, $\delta = 5$ cm (20cm for Newer College). The best and second-best results are bold and underlined, respectively.

Metric	Method	room 0	room 1	room 2	office 0	office 1	office 2	office 3	office 4	C.L.	Cow	N.C.
F1 Score [$< \delta$]% ↑	Ours	**95.86**	**97.32**	**97.54**	**98.23**	**96.57**	**95.11**	**93.58**	**94.53**	**83.47**	**92.74**	**75.80**
	FIESTA	76.71	76.49	78.15	77.64	82.43	75.45	79.73	78.19	19.37	18.03	54.73
	Voxblox	92.00	94.77	92.64	93.72	94.23	89.50	86.58	89.94	70.30	79.38	54.14
	iSDF	87.85	85.11	90.64	89.80	91.37	87.53	87.52	91.21	78.33	91.84	70.15
	VDB-GPDF	60.06	57.20	58.45	60.99	56.47	58.00	54.78	56.92	78.37	91.08	61.94
Recall [$< \delta$]% ↑	Ours	**92.94**	**95.51**	**95.50**	**97.53**	**95.54**	**92.10**	**90.15**	90.06	**97.42**	**98.98**	75.10
	FIESTA	77.56	78.33	81.04	81.33	82.00	77.17	81.15	78.09	19.89	17.27	52.60
	Voxblox	88.17	91.81	90.79	92.18	90.62	85.67	82.21	84.94	72.71	78.03	56.64
	iSDF	82.88	77.54	88.64	85.51	86.30	83.30	85.46	**90.12**	82.70	93.47	**86.24**
	VDB-GPDF	58.03	55.50	57.29	60.30	55.36	56.55	51.47	55.12	80.73	91.61	62.44
Precision [$< \delta$]% ↑	Ours	**98.96**	**99.21**	**99.66**	**98.95**	97.62	**98.32**	**97.28**	**99.46**	73.01	87.23	**76.50**
	FIESTA	75.87	74.73	75.47	74.28	82.85	73.81	78.36	78.28	18.87	18.86	57.03
	Voxblox	96.18	97.94	94.56	95.31	**98.13**	93.69	91.44	95.55	68.05	80.77	51.84
	iSDF	93.45	94.33	92.74	94.56	97.08	92.21	89.68	92.33	74.40	90.27	59.12
	VDB-GPDF	62.24	59.01	59.66	61.69	57.63	59.52	58.53	58.85	**76.14**	**90.56**	61.45
Completion Ratio [$< \delta$]% ↑	Ours	**92.41**	**95.33**	**95.31**	**97.49**	**95.44**	**91.55**	**89.35**	88.99	**93.37**	**97.33**	74.76
	FIESTA	78.05	79.32	82.34	82.94	81.82	78.16	81.80	78.03	23.99	9.67	48.61
	Voxblox	87.09	91.26	90.41	91.92	89.85	84.33	80.21	83.06	74.46	77.26	60.31
	iSDF	80.69	72.68	88.12	83.98	84.59	81.51	84.74	**89.88**	84.43	93.69	**90.57**
	VDB-GPDF	54.98	52.69	55.53	59.39	53.53	54.26	44.82	52.08	81.83	91.70	63.03
Completion [cm] ↓	Ours	**2.85**	2.37	2.33	2.07	2.25	**3.02**	**3.01**	3.50	**2.60**	2.57	14.22
	FIESTA	3.98	3.16	2.88	2.73	4.71	4.40	3.69	3.98	18.68	25.77	71.87
	Voxblox	3.69	**2.16**	**2.31**	**1.71**	**2.07**	3.52	4.37	4.38	4.44	5.77	21.30
	iSDF	5.63	8.72	2.62	4.14	3.98	5.62	3.03	**2.23**	3.12	**2.24**	**9.75**
	VDB-GPDF	5.52	4.82	3.87	3.52	4.12	4.80	13.78	5.07	3.68	3.05	40.57
Accuracy [cm] ↓	Ours	1.93	1.82	1.94	1.90	1.86	1.98	**1.96**	2.00	4.69	3.25	**14.03**
	FIESTA	3.57	3.19	3.18	3.43	3.11	4.23	3.51	3.22	26.67	16.58	21.02
	Voxblox	**1.74**	**1.14**	**1.50**	**1.22**	**0.81**	**1.68**	2.16	**1.67**	4.98	3.64	25.44
	iSDF	2.04	1.60	1.52	1.64	1.11	1.89	2.30	1.83	**4.66**	**2.75**	31.66
	VDB-GPDF	3.42	3.34	3.17	3.30	3.55	3.32	3.55	3.37	4.94	3.28	18.56
Chamfer-L1 [cm] ↓	Ours	**2.39**	2.09	2.13	1.99	2.06	**2.50**	**2.48**	2.75	**3.65**	2.91	**14.13**
	FIESTA	3.77	3.17	3.03	3.08	3.91	4.31	3.60	3.60	22.68	21.17	46.45
	Voxblox	2.72	**1.65**	**1.90**	**1.47**	**1.44**	2.60	3.26	3.02	4.71	4.70	23.37
	iSDF	3.83	5.16	2.07	2.89	2.55	3.75	2.67	**2.03**	3.89	**2.49**	20.71
	VDB-GPDF	4.47	4.08	3.52	3.41	3.83	4.06	8.67	4.22	4.31	3.17	29.57

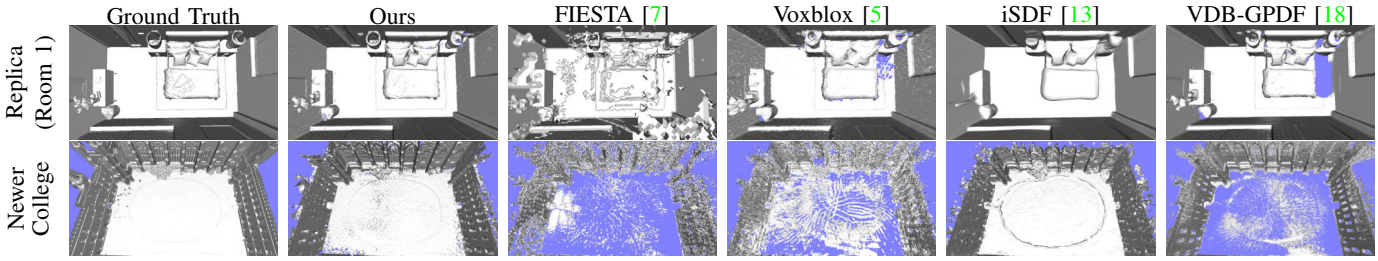


Fig. 3: Qualitative comparison of mesh reconstruction on Replica [32] (top) and Newer College [33] (bottom). Kernel-SDF achieves the best visual quality, accuracy, and completeness; e.g., the Replica bed-sheet textures are preserved where the baselines fail.

voxel resolution matches the evaluation resolution (0.05 m for Replica and Cow&Lady, 0.2 m for Newer College). Other parameters follow the authors' defaults. We analyze reconstruction quality (Sec.V-A), SDF accuracy (Sec. V-B), efficiency (Sec.V-C), scalability (Sec. V-D), kernel choice (Sec.V-E) and individual components (Sec.V-F), then verify the consistency between estimated SDF variance and actual error (Sec.V-G), and study the impact of sensor noise (Sec.V-H). Finally, we show a real-world robot navigation demonstration (Sec.V-I).

A. *Reconstruction Accuracy*

As shown in Figs. 3 and 4, Kernel-SDF produces high-quality meshes closely resembling the ground truth. On the low-noise Replica dataset, Voxblox, FIESTA and VDB-GPDF exhibit voxel-induced block artifacts, while iSDF yields over-smoothed surfaces. These baselines perform worse on the real-world datasets (Cow&Lady, Newer College), where sensor noise yields incomplete or distorted meshes: the discrete grids of FIESTA, Voxblox and VDB-GPDF struggle with fine details and completeness, while iSDF lacks geometric details.

Quantitative results in Table II confirm that Kernel-SDF achieves state-of-the-art or competitive performance across

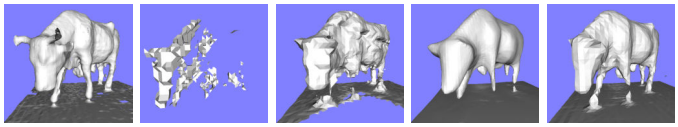


Fig. 4: Mesh reconstruction of the cow in Cow&Lady. Our method successfully recovers the detailed geometry (e.g., the horns, ears, and legs), which are missing in the baseline reconstructions.

nearly all metrics and datasets, consistently outperforming the baselines by mitigating sensor noise while preserving fine geometric details. Fine near-surface details come from extracting the mesh via marching cubes on the continuous BHM occupancy field, which recovers the surface at sub-voxel resolution and avoids the voxel-induced artifacts of the grid-based baselines. On the low-noise Replica scenes, Voxblox attains lower Accuracy because its TSDF mesh is extracted at the exact zero-crossing, whereas our marching-cubes surface sits at an online-estimated, view-dependent log-odds level τ (Sec. IV-A); the gap averages only 0.43 cm and reverses on the larger, noisier Newer College scene, where Kernel-SDF is the most accurate (Table II).

B. SDF Accuracy

We evaluate SDF and gradient accuracy using Mean Absolute Error (MAE) across three regions: *All* points, *Near* ($\leq$ 0.2m from surface), and *Far* ($>$ 0.2m). To isolate SDF accuracy from occupancy misclassification (e.g., by FIESTA), we compare the absolute values of predicted and g.t. SDFs. As summarized in Table III, Kernel-SDF consistently achieves the lowest MAE for both SDF values and gradients across nearly all datasets and regions. FIESTA is competitive on Replica but degrades significantly on Cow&Lady and Newer College due to sensor noise. Voxblox and iSDF exhibit higher errors due to voxelization and over-smoothing, respectively. Notably, while VDB-GPDF outperforms Voxblox in SDF accuracy, its gradient MAE is the highest among all methods.

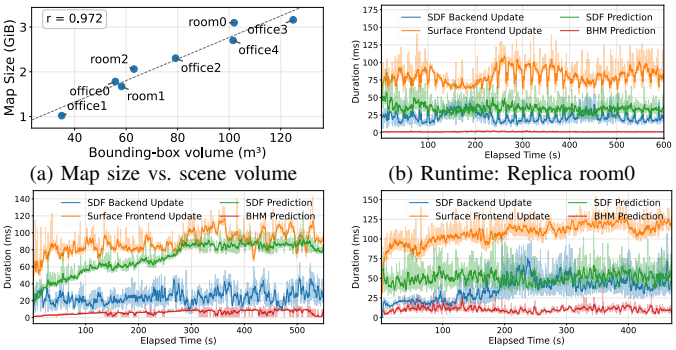
Kernel-SDF's superior performance in the *Near* region supports manipulation and navigation, while its accurate long-range predictions and gradients facilitate path planning and optimization-based control. In addition to SDF values, we evaluate the predicted sign (i.e. occupancy) against the ground truth on Replica and summarize the key findings here. Far from the surface, the sign is clear and every method exceeds 99%, so only the near-surface region is discriminating. Taking free space as the positive class, precision is important because it corresponds to low false-free (collision) rate. Kernel-SDF showed the highest near-surface precision, ranking first in 5 of 8 scenes and staying above 99% everywhere, whereas VDB-GPDF showed 69–82% precision, labeling nearly all near-surface points as free, yielding the highest false-free rate among all methods. The BHM sign prediction is therefore accurate and, importantly, conservative: it rarely mislabels occupied space as free, supporting collision avoidance and justifying the deterministic-sign treatment in the back-end.

C. Computation Efficiency

We evaluate efficiency by measuring the average per-frame SDF update time and the prediction latency for 1k query positions. As shown in Table IV, Kernel-SDF consistently achieves real-time performance with an average update time of approximately 150 ms across scene scales. While Voxblox builds the TSDF rapidly, its runtime is dominated by BFS-based SDF integration, slowed by the high volume of voxel updates from the large cutoff required for accurate SDF estimation. FIESTA achieves the fastest integration but, as shown in Sec. V-A and V-B, is inaccurate in noisy environments. iSDF remains slow due to its multi-iteration convergence and analytical gradient backpropagation overhead. Although VDB-GPDF also uses log-GP, its reliance on VDB structures for spatial partitioning yields significantly higher update latencies. Overall, Kernel-SDF balances computational speed and mapping precision, providing a robust solution for real-time robotic applications.

D. Scalability

The persistent map size scales linearly with the scene's spatial extent (Fig. 5a). Across the eight Replica scenes, the serialized map grows from about 1 GiB for the smallest ($\approx 35\,\mathrm{m}^3$) to about 3 GiB for the largest ($\approx 125\mathrm{m}^3$), strongly correlated with the bounding-box volume ($r = 0.972$). This footprint is set by the number of stored GP surface samples



(a) Map size vs. scene volume (b) Runtime: Replica room0

(c) Runtime: Cow&Lady (d) Runtime: Newer College

Fig. 5: Scalability of Kernel-SDF. (a) The serialized map size scales linearly with the environment volume across the eight Replica scenes ($r = 0.972$). (b)–(d) Per-component processing time over the full session (BHM front-end, SDF back-end, SDF prediction, and BHM prediction); all components stay stable with increasing map size and are independent of the global environment size.

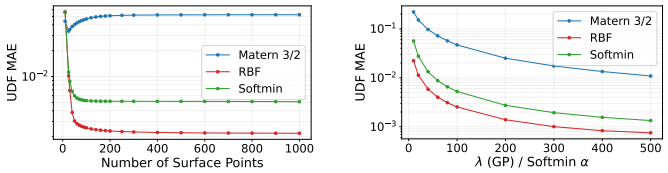


Fig. 6: Kernel ablation on a 2D unit-circle dataset, empirically validating the softmin view of the log-GP back-end. UDF MAE vs. number of surface samples (left) N and kernel scale λ (equivalently softmin α) (right). Except for the swept variable, defaults are $\lambda = 100$, $\alpha = 100$, and $N = 100$. The RBF kernel attains the lowest error and improves with larger scale and denser sampling.

(observed surface area times sampling density), so it grows with the observed extent rather than without bound. On Newer College, the serialized map reaches about 6 GiB. For memory-constrained onboard deployment in large environments, the memory use can be reduced by using a coarser map resolution or reducing the density of surface samples estimated by the front-end. The per-component update and prediction times (Fig. 5b–5d) do not grow with the accumulated map size, since the local octree-organized BHM/GP design ties each incremental update to the newly observed region rather than the global map. This bounded per-update cost lets Kernel-SDF scale to large environments in real time.

E. Kernel Ablation

We validate the softmin view (Sec. IV-B2) of the log-GP back-end by sweeping the kernel scale λ (equivalently softmin α) and the number of surface samples N on a 2D unit-circle dataset (Fig. 6). The UDF error shrinks as λ and N grow, matching the limit analysis in (12) and (13). The RBF kernel attains the lowest error and is therefore our default.

F. Component Analysis

To isolate the back-end's contribution, we replace the GP back-end with nearest-neighbor distance and with a pure softmin, keeping the same BHM front-end and surface samples (Table V). The three back-ends are essentially equivalent in the *All* region: GP and nearest-neighbor SDF MAE differ by less than 0.2 cm on every dataset. Near the surface, however, softmin is far less robust. Its SDF MAE on Newer College is over 6$\times$ the GP's (59.1 vs. 8.8 cm) and its Replica gradient

TABLE III: SDF reconstruction metrics on Replica, Cow&Lady (C.L. for the scene and Cow for cow only), and Newer College (N.C.).

Metric	Region	Method	room 0	room 1	room 2	office 0	office 1	office 2	office 3	office 4	C.L.	Cow	N.C.
SDF MAE [cm] ↓	All	Ours	**1.691**	**1.548**	**1.638**	**1.576**	**1.562**	**1.712**	**1.595**	**1.699**	**4.628**	**2.367**	22.989
		FIESTA	2.175	2.364	2.418	2.662	2.461	2.234	2.344	2.378	15.344	15.092	37.244
		Voxblox	3.128	2.539	2.733	3.017	2.725	3.470	3.740	3.077	6.297	4.000	32.030
		iSDF	3.745	4.230	4.259	3.890	4.226	5.094	4.245	4.147	7.327	4.258	64.304
		VDB-GPDF	2.925	2.900	2.778	2.902	2.968	2.861	3.176	2.929	4.683	3.056	**21.783**
	Near	Ours	**1.741**	**1.706**	**1.777**	**1.665**	**1.624**	**1.755**	**1.697**	**1.809**	**3.572**	**2.442**	8.813
		FIESTA	2.121	2.046	2.040	2.233	1.926	1.848	1.974	2.077	10.591	13.582	32.209
		Voxblox	2.780	2.153	2.233	2.196	2.172	2.852	3.409	2.959	4.287	3.404	20.186
		iSDF	3.649	3.632	3.295	3.206	3.646	3.821	3.397	3.301	4.258	3.595	**6.540**
		VDB-GPDF	2.847	2.924	2.802	2.827	2.936	2.770	3.136	2.870	3.938	3.025	20.693
	Far	Ours	**1.658**	**1.431**	**1.543**	**1.505**	**1.504**	**1.683**	**1.520**	**1.630**	**5.157**	**2.306**	23.899
		FIESTA	2.205	2.574	2.656	2.961	2.898	2.461	2.563	2.545	17.753	16.331	37.744
		Voxblox	3.368	2.845	3.081	3.711	3.270	3.918	4.002	3.158	7.338	4.471	33.267
		iSDF	3.811	4.705	4.932	4.466	4.797	6.016	4.918	4.718	8.864	4.802	69.652
		VDB-GPDF	3.486	2.620	2.606	3.778	3.585	3.472	3.511	3.320	7.066	3.272	**22.739**
Grad. MAE [rad] ↓	All	Ours	**0.151**	**0.134**	**0.157**	**0.146**	**0.140**	**0.165**	**0.159**	**0.149**	0.498	**0.297**	**0.299**
		FIESTA	0.220	0.207	0.184	0.216	0.250	0.198	0.224	0.180	1.141	1.093	0.498
		Voxblox	0.230	0.195	0.222	0.251	0.190	0.271	0.279	0.232	**0.477**	0.338	0.428
		iSDF	0.358	0.200	0.339	0.276	0.279	0.286	0.292	0.346	0.569	0.400	0.667
		VDB-GPDF	0.998	1.051	1.022	1.050	1.165	0.998	1.032	0.980	1.066	0.982	1.176
	Near	Ours	**0.183**	0.177	**0.174**	**0.201**	**0.179**	**0.195**	**0.199**	**0.178**	0.680	0.405	**0.948**
		FIESTA	0.304	0.276	0.230	0.313	0.323	0.253	0.300	0.232	1.329	1.224	1.398
		Voxblox	0.282	0.183	0.204	0.259	**0.179**	0.301	0.336	0.299	**0.679**	0.466	1.342
		iSDF	0.290	**0.176**	0.247	0.229	0.206	0.222	0.257	0.277	**0.679**	**0.395**	1.309
		VDB-GPDF	1.070	1.101	1.088	1.099	1.193	1.075	1.093	1.057	1.116	1.030	1.413
	Far	Ours	**0.138**	**0.111**	**0.148**	**0.117**	**0.116**	**0.151**	**0.140**	**0.138**	0.433	**0.229**	**0.275**
		FIESTA	0.182	0.171	0.162	0.166	0.203	0.171	0.189	0.158	1.050	0.985	0.412
		Voxblox	0.208	0.201	0.230	0.246	0.198	0.257	0.256	0.205	**0.406**	0.258	0.351
		iSDF	0.389	0.214	0.389	0.303	0.330	0.319	0.309	0.377	0.528	0.403	0.632
		VDB-GPDF	0.613	0.622	0.627	0.643	0.762	0.603	0.684	0.590	0.907	0.644	0.968

TABLE IV: Timing metrics: average per-frame processing time (FPT) and 1k-point query time (QT-1k) on Replica `room0` [32], Cow&Lady [5], and Newer College [33]. Best and second best are bold and underlined. Measured on an Intel i9-14900K and NVIDIA RTX 3090.

Method	Replica		Cow & Lady		Newer College	
	FPT (s)	QT-1k (ms)	FPT (s)	QT-1k (ms)	FPT (s)	QT-1k (ms)
Kernel-SDF	0.19	2.80	0.11	4.43	**0.17**	3.04
VDB-GPDF	0.68	10.66	0.10	8.29	0.26	10.16
Voxblox	21.23	103.81	4.48	99.42	94.01	85.20
iSDF	2.49	0.29	0.84	0.27	1.38	0.28
FIESTA	**0.05**	**0.05**	**0.02**	**0.03**	0.28	**0.04**

TABLE V: Back-end ablation: ours vs. nearest-neighbor (NN) and softmin fusion, sharing the same BHM front-end and surface samples. SDF MAE for *All* and *Near* query regions, and gradient angular MAE for the *Near* region (Replica is averaged over its 8 scenes).

Metric	Region	Method	Replica	Cow & Lady	Newer College
SDF MAE [cm] ↓	All	Ours (GP)	1.628	4.628	22.989
		NN	**1.622**	**4.605**	**22.811**
		Softmin	1.988	4.648	33.391
	Near (< 0.2m)	Ours (GP)	1.722	3.572	**8.813**
		NN	**1.683**	**3.524**	8.838
		Softmin	3.420	3.612	59.086
Grad. MAE [rad] ↓	Near	Ours (GP)	**0.1858**	**0.6800**	**0.9480**
		NN	0.2659	0.6982	0.9988
		Softmin	0.4221	0.7342	1.1575

error more than doubles (0.422 vs. 0.186 rad). The GP's advantage is therefore not distance accuracy but differentiability with closed-form gradients, calibrated variance (Sec. V-G), and graceful degradation under noise.

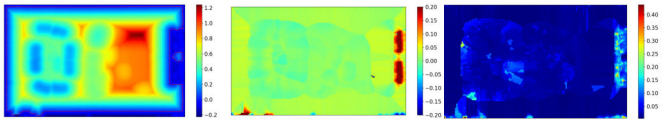
We also evaluate the BHM-derived uncertainty (6) by replacing the per-sample variance σ_i^2 with a constant, either 0 (pure softmin) or 0.1 (Table VI). The BHM-derived weighting improves both the SDF and gradient accuracy, most notably the gradient angle error, which drops from 0.172 to 0.151 rad (12%). This confirms the regularization effect analyzed in Sec. IV-B that down-weighting samples with high occupancy uncertainty sharpens both the SDF and its gradient.

G. Uncertainty Calibration

A key advantage of Kernel-SDF is its calibrated SDF uncertainty. Qualitatively, Fig. 7 shows that on a slice of Replica Office3 [32] regions of high SDF error coincide with regions of high estimated variance. Quantitatively, we examine the empirical first and second moments of normalized error $z = (u - \hat{u})/\hat{\sigma}$. Good calibration corresponds to $\mathbb{E}[z] = 0$ and $\mathbb{E}[z^2] = 1$. Across Replica room0, Cow&Lady, and Newer College, $\mathbb{E}[z^2] = 1.05, 1.03, 1.04$ (within 5.5% of 1), and the empirical coverage closely tracks the target with an expected calibration error (ECE) of $0.052, 0.031, 0.011$ (Fig. 8), showing the predicted variance closely matches the true error variance. The first moment is positive on all three scenes ($\mathbb{E}[z] = 0.49, 0.55, 0.02$), which is consistent with the softmin perspective (Sec. IV-B2) that the predicted distance is biased to be conservative. Such a bias is related to the scale

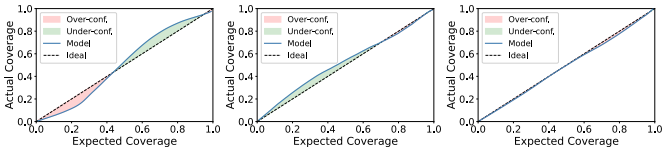
TABLE VI: Ablation on the BHM surface-uncertainty weighting (Replica room0, all query points). Ours uses per-sample, occupancy-derived variance σ_i^2; the alternatives replace it with a single constant variance, either σ^2=0 (pure softmin) or σ^2=0.1.

Weighting	SDF MAE [cm] ↓	Grad. MAE [rad] ↓
Ours (w/ BHM var.)	**1.691**	**0.1514**
w/o BHM var. (σ^2=0)	1.731	0.1722
w/o BHM var. (σ^2=0.1)	1.734	0.1749



(a) Predicted SDF (b) SDF Error (c) SDF Variance

Fig. 7: Consistency between SDF error and estimated SDF variance on a slice of Replica `Office3` [32]: (a) predicted SDF, (b) absolute SDF error against ground truth, and (c) estimated SDF variance. Regions of higher error coincide with higher estimated variance, showing that Kernel-SDF effectively captures uncertainty.



(a) Replica room0 (b) Cow&Lady (c) Newer College

Fig. 8: Reliability curves of the predicted SDF uncertainty: for each target confidence level (x-axis), the fraction of query points whose error falls within it (y-axis). Empirical coverage closely tracks the ideal diagonal (ECE 0.052, 0.031, 0.011), confirming calibration.

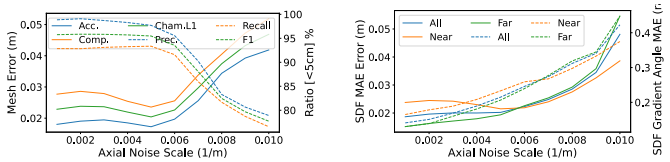
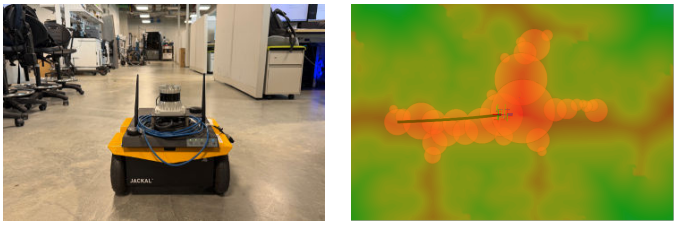


Fig. 9: Effect of sensor noise on (left) mesh reconstruction and (right) SDF metrics on Replica `room0` [32]. Kernel-SDF stays robust as noise increases, with only gradual degradation.



(a) Jackal robot (b) SDF and safe bubble cover

Fig. 10: Kernel-SDF enables safe bubble cover construction [21] and safe robot navigation. (a) A Clearpath Jackal robot with LiDAR sensor navigating in a lab environment. (b) Real-time Kernel-SDF estimate shown as a heatmap with safe bubble cover (union of orange circles) and planned trajectory (black line).

β in (6), the kernel scale λ and the surface sample density, which can be tuned to reduce it if desired.

H. Ablation Study on Sensor Noise

To assess noise robustness, we evaluated Kernel-SDF on Replica `room0` by varying the axial noise parameter k in $\sigma = kz^2$ with other parameters fixed. Fig. 9 shows the resulting mesh and SDF metrics. Kernel-SDF degrades only gradually as noise increases, confirming that the BHM front-end mitigates sensor noise to maintain reliable SDF mapping in adverse environments.

I. Autonomous Robot Navigation Demonstration

Real-time, uncertainty-quantified SDF mapping is essential for safe robot navigation. We demonstrate this by planning a safe corridor in the SDF map from a chain of overlapping bubbles, using the sampling-based approach of [21]. The planner builds a rapidly exploring random tree from free-space samples with associated safe bubbles, whose radii are the SDF values minus a safety margin, optionally scaled by the estimated SDF uncertainty. As shown in Fig. 10, Kernel-SDF allows the planner to generate a connected chain of bubbles spanning the start and goal, keeping the path strictly within the mapped free space. This demonstrates Kernel-SDF's utility in providing accurate distance and uncertainty information for risk-aware motion planning.

VI. Conclusion

We presented Kernel-SDF, a real-time kernel regression framework for differentiable SDF learning with uncertainty quantification. By coupling BHMs for robust surface estimation with GPs for distance prediction, our method achieves superior reconstruction accuracy and efficiency. The hierarchical tree structure ensures scalability to large environments, while our softmin-based uncertainty quantification enables risk-aware decision-making for downstream robotic tasks. Extensive evaluations on synthetic and real-world datasets show that Kernel-SDF outperforms state-of-the-art baselines in both quality and speed, while its BHM front-end mitigates sensor noise and adapts to dynamic environments, providing a robust, practical solution for real-time robotic applications.

References

[1] A. Hornung, *et al.*, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013.
[2] J. Behley and C. Stachniss, "Efficient surfel-based SLAM using 3D laser range data in urban environments," *RSS*, 2018.
[3] A. Rosinol, *et al.*, "Kimera: an open-source library for real-time metric-semantic localization and mapping," in *ICRA*, 2020.
[4] R. A. Newcombe, *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *ISMAR*, 2011.
[5] H. Oleynikova, *et al.*, "Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning," in *IROS*, 2017.
[6] J. J. Park, *et al.*, "DeepSDF: Learning continuous signed distance functions for shape representation," in *CVPR*, 2019.
[7] L. Han, *et al.*, "FIESTA: Fast incremental euclidean distance fields for online motion planning of aerial robots," in *IROS*, 2019.
[8] B. Lee, *et al.*, "Online continuous mapping using gaussian process implicit surfaces," in *ICRA*, 2019.
[9] L. Wu, *et al.*, "Faithful euclidean distance field from log-gaussian process implicit surfaces," *RA-L*, 2021.
[10] P. Wang, *et al.*, "NeuS: Learning neural implicit surfaces by volume rendering for multi-view reconstruction," in *NeurIPS*, 2021.
[11] I. Vizzo, *et al.*, "VDBFusion: Flexible and efficient tsdf integration of range sensor data," *Sensors*, 2022.
[12] Y. Pan, *et al.*, "Voxfield: Non-projective signed distance fields for online planning and 3D reconstruction," in *IROS*, 2022.
[13] J. Ortiz, *et al.*, "iSDF: Real-time neural signed distance fields for robot perception," in *RSS*, 2022.
[14] Y. Wang, *et al.*, "NeuS2: Fast learning of neural implicit surfaces for multi-view reconstruction," in *ICCV*, 2023.
[15] C. Jiang, *et al.*, "H2-Mapping: Real-time dense mapping using hierarchical hybrid representation," *RA-L*, 2023.
[16] Y. Pan, *et al.*, "PIN-SLAM: LiDAR SLAM using a point-based implicit neural representation for achieving global map consistency," *T-RO*, 2024.
[17] Q. Zou and M. Sester, "3D uncertain implicit surface mapping using GMM and GP," *RA-L*, 2024.
[18] L. Wu, *et al.*, "VDB-GPDF: Online gaussian process distance field with vdb structure," *RA-L*, 2025.
[19] Y. Tian, *et al.*, "MISO: Multiresolution submap optimization for efficient globally consistent neural implicit reconstruction," in *RSS*, 2025.
[20] B. Mildenhall, *et al.*, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *ECCV*, 2020.
[21] K. M. B. Lee, *et al.*, "Safe bubble cover for motion planning on distance fields," in *preprint arXiv:2408.13377*, 2024.
[22] K. Long, *et al.*, "Sensor-based distributionally robust control for safe robot navigation in dynamic environments," *IJRR*, 2025.
[23] Y. Li, *et al.*, "Representing robot geometry as distance fields: Applications to whole-body manipulation," in *ICRA*, 2024.
[24] Y. Bai, *et al.*, "VDBblox: Accurate and efficient distance fields for path planning and mesh reconstruction," in *IROS*, 2023.
[25] A. Millane, *et al.*, "nvblox: GPU-accelerated incremental signed distance field mapping," in *ICRA*, 2024.
[26] R. Senanayake and F. Ramos, "Bayesian hilbert maps for dynamic continuous occupancy mapping," in *CoRL*, 2017.
[27] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *ACM SIGGRAPH*, 1987.
[28] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *ACM SIGGRAPH*, 1996.
[29] K. Museth, "VDB: High-resolution sparse volumes with dynamic topology," *ACM Trans. Graph.*, 2013.
[30] O. Williams and A. Fitzgibbon, "Gaussian process implicit surfaces," in *Gaussian Processes in Practice*, 2006.
[31] S. R. S. Varadhan, "On the behavior of the fundamental solution of the heat equation with variable coefficients," *Communications on Pure and Applied Mathematics*, 1967.
[32] J. Straub, *et al.*, "The Replica dataset: A digital Replica of indoor spaces," *preprint arXiv:1906.05797*, 2019.
[33] L. Zhang, *et al.*, "Multi-camera LiDAR inertial extension to the newer college dataset," in *preprint arXiv:2112.08854*, 2021.
[34] M. S. Ahn, *et al.*, "Analysis and noise modeling of the Intel RealSense D435 for mobile robots," in *UR*, 2019.