∇-SDF: Learning Euclidean Signed Distance Functions Online with Gradient-Augmented Octree Interpolation and Neural Residual

Anonymous For Review

***Abstract*— Estimation of signed distance functions (SDFs) from point cloud data has been shown to benefit many robot autonomy capabilities, including localization, mapping, motion planning, and control. Methods that support online and large-scale SDF reconstruction tend to rely on discrete volumetric data structures, which affect the continuity and differentiability of the SDF estimates. Recently, using implicit features, neural network methods have demonstrated high-fidelity and differentiable SDF reconstruction but they tend to be less efficient, can experience catastrophic forgetting and memory limitations in large environments, and are often restricted to truncated SDFs. This work proposes ∇-SDF, a hybrid method that combines an explicit prior obtained from gradient-augmented octree interpolation with an implicit neural residual. Our method achieves non-truncated (Euclidean) SDF reconstruction with computational and memory efficiency comparable to volumetric methods and differentiability and accuracy comparable to neural network methods. Extensive experiments demonstrate that ∇-SDF outperforms the state of the art in terms of accuracy and efficiency, providing a scalable solution for downstream tasks in robotics and computer vision.**

I. Introduction

Accurate and differentiable geometric environment representations are critical for many functions in robot autonomy and computer vision, including simultaneous localization and mapping [1]–[3], rendering and AR/VR [4]–[6], autonomous navigation [7], [8] and manipulation [9]–[11]. In robotics, fast updates of the environment model from sensor observations and access to gradient information are important to enable robots to navigate reactively and safely and to interact with the environment precisely, while a small memory footprint is important for the scalability of the representation.

In this work, we focus on signed distance function (SDF) reconstruction. Given a query point, an SDF returns the signed distance to the nearest surface in the environment with sign indicating whether the query is in free (positive) or occupied (negative) space. SDFs have received increasing attention due to their constant-time complexity for distance and collision queries and their ability to capture complex obstacle surfaces implicitly as a zero-level set.

SDF reconstruction methods can roughly be organized into three categories: volumetric methods (e.g., [7], [12]), Gaussian Process (GP) methods (e.g., [13], [14]), and neural network methods (e.g., [15], [16]). We review representative papers from these categories in Sec. II. Volumetric methods utilize advanced data structures, like octrees and hashmaps, and are known for their real-time performance and scalability to large scenes. However, they provide non-differentiable SDF estimates and require significant storage to achieve higher accuracy. GP methods learn continuous SDF models with uncertainty quantification but often suffer from high computational complexity and poor scalability. Recently, neural network methods have shown great potential to learn compact and accurate SDF representations, but they tend to be restricted to truncated SDF and struggle with catastrophic forgetting in large scenes or in online settings.

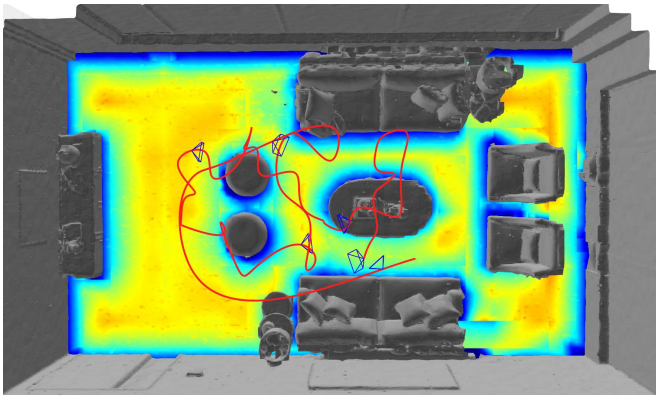


Fig. 1: ∇-SDF reconstructs an accurate Euclidean signed distance function online from streaming point cloud data.

We propose ∇-SDF, a hybrid method that combines the strengths of volumetric methods, in the form of an explicit SDF prior obtained from octree interpolation, and neural network methods, in the form of implicit features decoded into a residual correction of the explicit prior. To construct the explicit prior, we use a semi-sparse octree with SDF and gradient estimates stored at the octant vertices and design a new gradient-augmented interpolation approach to obtain smooth and accurate SDF priors at arbitrary query positions. We augment the prior prediction with a neural network residual, which recovers fine geometric details of the observed surface from implicit features. We train our hybrid explicit-implicit model using three loss functions that supervise both near-surface and distant SDF values and accelerate the convergence to achieve real-time highly accurate SDF reconstruction.

The closest works to ours are H_2-Mapping [17] and HIO-SDF [18]. Similar to our method, H_2-Mapping applies trilinear interpolation in a sparse octree to obtain an SDF prior and trains a neural network residual. In contrast with our method, H_2-Mapping reconstructs only truncated (near-surface) SDF. HIO-SDF achieves non-truncated SDF reconstruction using Voxfield [19] to generate an SDF prior and a neural network trained on a dataset of global SDF priors and local SDF approximations from the latest point cloud. The global priors accelerate the convergence and prevent forgetting, but the accuracy of the learned SDF is limited by the Voxfield prior.

Manuscript 4449 submitted to 2026 IEEE International Conference on Robotics and Automation (ICRA). Received September 16, 2025.

Instead, our method directly optimizes the parameters of the octree to learn a more accurate prior and uses the residual network to further improve the accuracy.

In summary, our work makes the following contributions.

- We introduce a new gradient-augmented interpolation in a semi-sparse octree to obtain an SDF prior, improving the accuracy, memory, and training speed for subsequent SDF residual learning.
- We formulate a hybrid model that combines the explicit priors with an implicit neural residual, enabling accurate SDF learning both near to and far from the observed surfaces. We also design loss functions to encourage globally accurate SDF learning and accelerate the training process to achieve real-time performance.

II. Related Work

Various methods have been proposed to learn SDF, which can be roughly categorized into three groups: volumetric methods [7], [12], [19]–[23], GP-based methods [13], [14], [24], and neural network based methods [1], [15]. We first review these three kinds of methods and then discuss the recent trend of using hybrid models for SDF reconstruction.

A. Volumetric SDF Reconstruction

Volumetric methods like Voxblox [7] achieve real-time SDF reconstruction. A regular grid with voxel hashing is used to efficiently look up voxels for updates and queries. Voxblox [7] builds a TSDF layer by projective distance, which is the distance between the voxel center and the observed surface point, then updates an SDF layer by breadth-first search (BFS) and the BFS path length. Both the projective distance and the BFS path length introduce inaccuracies in the SDF reconstruction. Subsequent works [19], [22] manage to narrow the errors. However, all of these methods rely on a discrete SDF representation, which is non-differentiable and limits the accuracy to the grid resolution. It is also difficult to scale up the dense grid to large scenes with high fidelity. In contrast, our work uses an octree to reduce the memory cost and employs a multi-resolution neural hash grid with an MLP decoder to learn a differentiable implicit compact representation, capturing the geometric details.

B. Gaussian Process SDF Reconstruction

GP-based methods [13], [14], [24] learn continuous SDF representations, which support SDF gradient computation and uncertainty quantification. Although GPIS [13] achieves accurate results in near-surface SDF prediction, it fails to extrapolate to positions away from the surface. Log-GPIS [24] and VDB-GPDF [14] learn unsigned distance functions in log space, which generalize well globally but omit the sign and struggle to scale to large environments due to the cubic complexity of the matrix inverse during training. In comparison, our method uses interpolation to compute the SDF prior, which has $O(1)$ complexity, and matrix multiplication during the computation of SDF residual, which has roughly quadratic complexity of the matrix multiplication. Besides, the octree structure and the neural hash grid have a smaller memory footprint than the gram matrix used by GP.

C. Neural Network SDF Reconstruction

DeepSDF [15] was among the first methods to demonstrate that neural networks can learn compact and continuous implicit SDF representations. This inspired many subsequent neural network methods for SDF reconstruction. iSDF [1] formulates an incremental learning approach for SDF reconstruction by iteratively updating the MLP with training samples generated from a key frame set and proposes Eikonal regularization to encourage the model to satisfy the Eikonal property of SDF. NeuS [16] learns SDF and neural radiance fields (NeRF) [25] simultaneously, allowing the two fields to improve each other by using an SDF-based unbiased volume density formulation. Besides, many other works like IGR [26], NGLoD [27] propose various neural network designs, loss functions, training procedures, and so on to achieve better SDF reconstruction. These works show that neural networks are able to learn SDF accurately near the surface, which is sufficient for high-fidelity surface reconstruction. However, they rarely pay attention to learning accurate SDF at locations away from the surface. Existing neural network methods [28] that learn non-truncated SDF, which are mostly object-level, require extensive training data and enough training time to achieve satisfactory accuracy.

D. Hybrid Methods for SDF Reconstruction

Recently, hybrid models that combine explicit geometric structures with implicit neural features show promising results. PIN-SLAM [2] stores neural features in near-surface voxels. Given a query point, its SDF prediction is a weighted sum of k SDF predictions, obtained by feeding k nearest neural features with local positions into a global decoder. H_2-Mapping [17] presents another model that combines an octree-based SDF prior with a neural residual. However, both PIN-SLAM and H_2-Mapping learn truncated SDF. HIO-SDF [18] removes truncation by running Voxfield [19] first to generate global SDF priors, which are combined with local SDF approximations to train a neural network. However, the accuracy and speed are limited by the volumetric method. As the observed area grows, the neural network, whose number of parameters is fixed, tends to learn an over-smooth SDF.

In contrast, our method, ∇-SDF, builds a semi-sparse octree to store the prior of SDF values and gradients, which is extendable as the scene grows and efficiently represents the SDF of the whole space. With gradient-augmented interpolation in the octree, our method can produce more accurate SDF priors, leaving more capacity for the subsequent network to recover surface geometric details.

III. Problem Statement

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

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

The SDF satisfies two key properties: 1) the obstacle surface is encoded as the zero-level set, $d(\mathbf{x}) = 0$, $\forall \mathbf{x} \in \partial\mathcal{O}$, and 2)

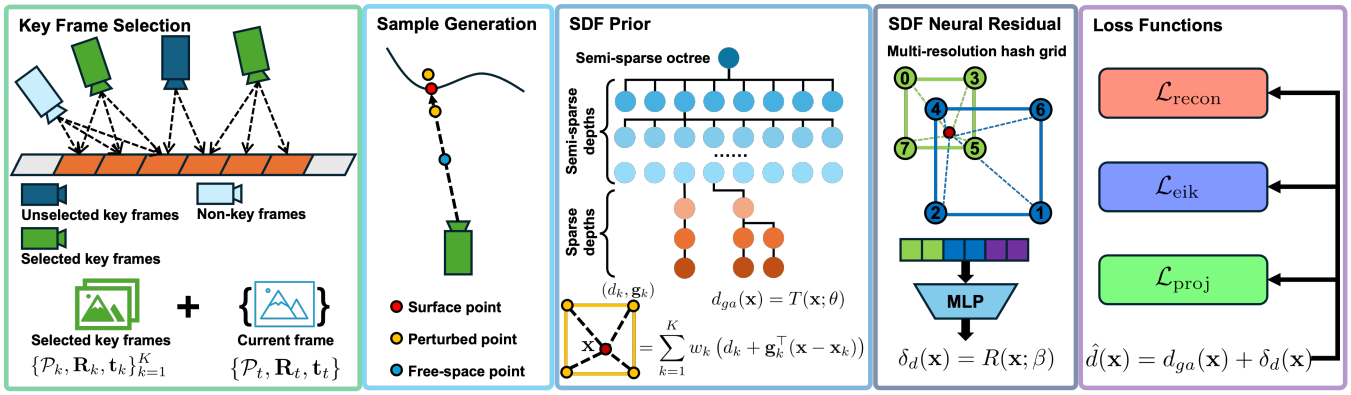


Fig. 2: Method Overview: a) We keep key frames with small overlap and those that maximize the surface coverage for training; b) with the selected key frames and the current frame, we generate three types of samples: surface points, perturbed points around the surface, and free-space points; c) to predict SDF, we first obtain an SDF prior $d_{ga}(\mathbf{x})$ with gradient-augmented interpolation in a semi-sparse octree, where each octant vertex has estimated SDF value and gradient; d) a multi-resolution hash grid with an MLP decoder is used to obtain an SDF residual correction $\delta_d(\mathbf{x})$; e) the SDF prior $d_{ga}(\mathbf{x})$ and the SDF residual $\delta_d(\mathbf{x})$ are combined as the final SDF prediction $\hat{d}(\mathbf{x}) = d_{ga}(\mathbf{x}) + \delta_d(\mathbf{x})$, and the parameters are trained with three loss functions: reconstruction loss, Eikonal loss and projection loss.

the gradient of $d(\mathbf{x})$ is the unit vector pointing away from the nearest surface point and satisfies an Eikonal equation [1]:

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

where $\mathbf{x}_* \in \arg\min_{\mathbf{y} \in \partial\mathcal{O}} \|\mathbf{x} - \mathbf{y}\|_2$.

Given a stream of point clouds obtained from range sensor measurements (e.g., from LiDAR or depth camera), $\{\mathbf{o}_t, \mathcal{P}_t\}_{t=1}$, where $\mathbf{o}_t$ is the sensor position at time step t and $\mathcal{P}_t$ is the set of observed surface points in the global frame, our objective is to approximate the SDF $d(\mathbf{x})$ of $\mathcal{O}$ as a scalar field, $\hat{d} : \mathbb{R}^3 \to \mathbb{R}$. We also aim to have $\hat{d}$ capture the SDF gradient accurately.

IV. ∇-SDF: Efficient Network for Learning Globally Accurate SDF

Our method employs a hybrid model to reconstruct SDF. We use a semi-sparse octree, where certain octants with no surface contained are created, to store explicit SDF and SDF gradient estimates in order to compute a coarse SDF prior, described in Sec. IV-A. To recover the geometric details, we use a multi-resolution hash grid of implicit neural features, which allows capturing the residuals at different scales, and an MLP decoder, which produces residual corrections to the coarse SDF from the octree. The neural feature hash grid and MLP decoder are described in Sec. IV-B. To train our hybrid model efficiently, we maintain a set of key frames that cover the observed surface with a small overlap between adjacent frames, as described in Sec. IV-C. Then, from the key frames and the latest frame, we generate a dataset containing different types of samples, discussed in Sec. IV-D, and use it to train our model with loss functions proposed in Sec. IV-E to speed up the network convergence. Our method is overviewed in Fig 2 with details presented in the subsections.

A. SDF Prior via Gradient-Augmented Octree Interpolation

1) Semi-Sparse Octree: The SDF prior for position $\mathbf{x}$ is obtained by interpolation in a semi-sparse octree data structure with N layers. We refer to an octree layer as *dense* when all octants are created as a regular grid; as *semi-sparse* when child octants containing surface points and all of their siblings (regardless of occupancy) are created simultaneously; and as *sparse* when only child octants containing surface points are created.

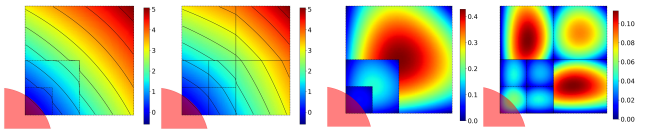


Fig. 3: 2D visualization of SDF interpolation without gradient augmentation using (a) a sparse octree and (b) a semi-sparse with corresponding interpolation error shown in (c) and (d) respectively. The bottom-left red region is an obstacle containing one vertex.

We use a semi-sparse octree of resolution r, where the first M layers are semi-sparse and the remaining $N - M$ layers are sparse. This is illustrated in Fig. 2c. In each vertex $\mathbf{x}_k$ of an octant with $k \in \{1, \ldots, 8\}$, we store estimates $d_k \in \mathbb{R}$ and $\mathbf{g}_k \in \mathbb{R}^3$ of the SDF $d(\mathbf{x}_k)$ and its gradient $\nabla d(\mathbf{x}_k)$, respectively, which are learnable during training. To maintain memory efficiency, a vertex is shared across neighboring octants from different tree depths. For example, the eight vertices of an octant are also included in the vertices of its eight child octants. This semi-sparse structure is essential to obtain a good SDF prior, especially for query positions away from the surface. The on-demand initialization of all child octants in the first M layers costs extra memory but improves the accuracy significantly.

Fig. 3 shows a 2D example of an SDF prior using trilinear interpolation in a sparse and a semi-sparse octree. In a sparse octree, the SDF interpolation has more discontinuities on the octant boundaries compared to the result in a semi-sparse octree. Given a query point $\mathbf{x}$, the semi-sparse octree provides the smallest octant containing $\mathbf{x}$ that is not larger than the smallest octant found in the sparse octree. This guarantees that we can find vertices closer to $\mathbf{x}$ for computing

the SDF prior because of the creation of sibling octants, which leads to a significantly smaller SDF interpolation error as indicated by Fig. 3c and Fig. 3d.

For any query position where the surface exists in the neighborhood, we can locate an octant no larger than $r \times 2^{N-M}$. For queries distant to the surface, an empty large octant is sufficient for computing an accurate SDF prior using gradient-augmented interpolation, which is discussed next.

2) Gradient-Augmented Interpolation: To achieve smaller SDF errors of the prior so that the subsequent neural network can focus on restoring the geometric details, we propose a new gradient-augmented trilinear interpolation method. Given the smallest octant that contains a query position $\mathbf{x}$, we first obtain an extrapolation result from each vertex $\mathbf{x}_k$:

$$d_k(\mathbf{x}) = d_k + \mathbf{g}_k^\top(\mathbf{x} - \mathbf{x}_k),\ k \in \{1, \dots, 8\}. \tag{3}$$

Given the extrapolation results, we compute the gradient-augmented (ga) interpolation:

$$d_{ga}(\mathbf{x}) = \frac{1}{\gamma}\sum_{k=1}^{8} w_k d_k(\mathbf{x}),\ \gamma = \sum_{k=1}^{8} w_k, \tag{4}$$

where $w_k = 1/|\operatorname{diag}(\mathbf{x}_i - \mathbf{x}_k)|$ is the interpolation weight. In contrast, the regular trilinear (tl) interpolation is:

$$d_{tl}(\mathbf{x}) = \frac{1}{\gamma}\sum_{k=1}^{8} w_k d_k,\ \gamma = \sum_{k=1}^{8} w_k. \tag{5}$$

Empirically, gradient-augmented interpolation generates more accurate SDF priors. Fig. 4 shows two 2D examples. Each row shows the ground truth SDF, interpolation results of using w/ and w/o gradient augmentation, the corresponding errors, and the Hessian spectral norm of SDF. As shown in Fig. 4d and 4e, gradient-augmented interpolation has smaller errors, especially when more obstacles are in the scene. The gradient-augmented interpolation requires extra memory and computation for the gradient $\mathbf{g}_k$ and the extrapolation, respectively. However, theoretically, interpolation with gradient augmentation causes a smaller error upper bound.

Proposition 1. *Consider an octant $\mathcal{V} \subset \mathbb{R}^3$ of size L. Assume that the SDF $d(\mathbf{x})$ is twice differentiable and the spectral norm of its Hessian is bounded:*

$$M := \sup_{\mathbf{x}\in\mathcal{V}} \|\nabla^2 d(\mathbf{x})\|_2 < \infty. \tag{6}$$

Assume that each vertex $\mathbf{x}_k$ has ground-truth SDF value $d_k = d(\mathbf{x}_k)$ and gradient $\mathbf{g}_k = \nabla d(\mathbf{x}_k)$. Then, given arbitrary $\mathbf{x} \in \mathcal{V}$, the errors of gradient-augmented interpolation in (4) and trilinear interpolation in (5) satisfy:

$$\begin{aligned} e_{ga}(\mathbf{x}) &= |d_{ga}(\mathbf{x}) - d(\mathbf{x})| \le \bar{e}_{ga} = \frac{3ML^2}{8}, \\ e_{tl}(\mathbf{x}) &= |d_{tl}(\mathbf{x}) - d(\mathbf{x})| \le \bar{e}_{tl} = \frac{\sqrt{3}L}{2}. \end{aligned} \tag{7}$$

Proof. For each octant vertex $\mathbf{x}_k$, by Taylor's theorem: $d(\mathbf{x}) = d_k + \mathbf{g}_k^\top(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_k)^\top \nabla^2 d(\boldsymbol{\xi}_k)(\mathbf{x} - \mathbf{x}_k)$, for some $\boldsymbol{\xi}_k$ on the line segment joining $\mathbf{x}$ and $\mathbf{x}_k$. Using (4) and (6), the gradient-augmented interpolation error satisfies:

$$\begin{aligned} e_{ga}(\mathbf{x}) &= \frac{1}{2\gamma}\left|\sum_{k=1}^{8} w_k(\mathbf{x} - \mathbf{x}_k)^\top \nabla^2 d(\boldsymbol{\xi}_k)(\mathbf{x} - \mathbf{x}_k)\right| \\ &\le \frac{M}{2\gamma}\sum_{k=1}^{8} w_k\|\mathbf{x} - \mathbf{x}_k\|_2^2 \le \frac{3ML^2}{8} = \bar{e}_{ga}, \end{aligned} \tag{8}$$

where equality holds when $\mathbf{x}$ is the octant center so that $\frac{1}{\gamma}\sum_{k=1}^{8} w_k\|\mathbf{x} - \mathbf{x}_k\|_2^2 = 3L^2/4$.

Similarly, for the error of the regular trilinear interpolation, we have by Taylor's theorem:

$$d(\mathbf{x}) = d_k + \nabla d(\boldsymbol{\zeta}_k)^\top(\mathbf{x} - \mathbf{x}_k) \tag{9}$$

for some $\boldsymbol{\zeta}_k$ on the line segment joining $\mathbf{x}$ and $\mathbf{x}_k$. Then, since $\|\nabla d(\boldsymbol{\zeta}_k)\| = 1$ and using (5), the trilinear interpolation error satisfies:

$$\begin{aligned} e_{tl}(\mathbf{x}) &= \left|\frac{1}{\gamma}\sum_{k=1}^{K} w_k \nabla d(\boldsymbol{\xi}_k)^\top(\mathbf{x} - \mathbf{x}_k)\right| \\ &\le \frac{1}{\gamma}\sum_{k=1}^{K} w_k\|\mathbf{x} - \mathbf{x}_k\|_2 \le \frac{\sqrt{3}L}{2} = \bar{e}_{tl}. \end{aligned} \tag{10}$$

□

When an octant does not contain positions where the gradient is not well defined (e.g., the medial axes where the closest point projection is not unique), (6) holds. For positions without well-defined gradients, although the Hessian norm blows up mathematically, gradient-augmented interpolation has $e_{ga}(\mathbf{x})$ significantly lower than $\bar{e}_{ga}$ based on empirical observation. The second row of Fig. 4d and 4f shows an example of such cases, that the Hessian spectral norm is large on the medial axes, but gradient-augmented interpolation has smaller errors. Since we are looking for an upper bound on the error, we ignore such cases.

As shown in Fig. 4, the gradient-augmented interpolation has smaller errors than without gradient augmentation. Empirically, as shown in Fig. 4f, $M \ll 1$ so that $\bar{e}_{ga}/\bar{e}_{tl} = \sqrt{3}ML/4 < 1$. Especially, when the octant is surrounded by multiple obstacles, the SDF prior values stored at the vertices are smaller than the ground truth SDF values inside the octant. This makes SDF priors obtained from interpolation without gradient augmentation no larger than the vertex SDF values, leading to significantly larger errors.

Hence, the prior network $T(\mathbf{x};\theta)$ of our method is a semi-sparse octree where each vertex has an estimate of SDF and gradient, i.e., $\theta = \{d_k, \mathbf{g}_k\}_{k=1}^K$, which are learnable parameters optimized together with the residual network. In the experiments, we maintain a semi-sparse octree for each scene with $N = 9$, $M = 5$ and $r = 10$ cm.

B. SDF Residual via Neural Feature Decoding

The accuracy of the SDF prior is limited to the octree resolution, causing lack of geometric details. To achieve high fidelity, we propose to learn a residual correction to the SDF values via a neural network $R(\mathbf{x};\beta)$, which consists of a multi-resolution hash grid encoder [29] and an MLP decoder.

As shown in Fig. 2d, for each query point $\mathbf{x}$, the multi-resolution neural hash grid [29] encodes the point $\mathbf{x}$ at l levels, where each level produces an F-dimensional feature by interpolation. The l features from all levels are concatenated $\mathbf{f} = E(\mathbf{x};\beta_E) \in \mathbb{R}^{lF}$ and used as the input to an MLP, which predicts the SDF residual $\delta_d(\mathbf{x}) = D(\mathbf{f};\beta_D)$. In the experiments, we have $l = 4$, $F = 2$ and the MLP has five 64-dim hidden layers with LeakyReLU activation.

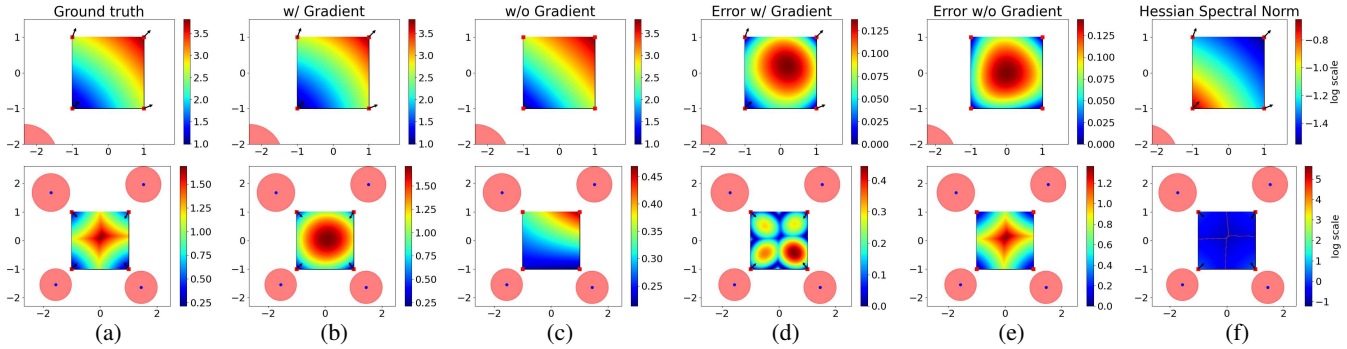


Fig. 4: 2D visualization of interpolation with and without gradient augmentation for one (red region, top row) and four obstacles (red regions, bottom row). Gradient-augmented interpolation produces a better SDF prior (b) with smaller error (d). Empirically, positions where the SDF gradient is not well defined (large Hessian spectral norm), as shown in (f), have small interpolation error with gradient augmentation as shown in (d).

C. Key Frame Selection

To learn SDF in real-time, it is important to keep a compact and representative set of sensor frames for training. We adopt the key-frame selection strategy of H_2-Mapping [17]. As shown in Fig. 2a, we insert a new sensor frame when the newly observed area compared with the last key frame is large enough, i.e., $\frac{|V_c \cap V_l|}{|V_c \cup V_l|} > c_{\min}$, where V_c is the set of surface octants observed by the current frame, V_l is the set of surface octants observed by the last inserted key frame, and $c_{\min} \in [0, 1]$ is a threshold. This strategy makes sure that the frames cover the observed surface with little overlap between adjacent key frames.

Over time, the number of key frames grows. It is essential to select only W key frames to maintain real-time operation. We incrementally select the frame that observes the most octants, mask out these octants, and repeat until W frames are collected. If all octants are masked out, we reset the mask except for the octants masked out by the last selected key frame and continue the selection. This strategy ensures that the selected key frames maximally cover the scene.

D. Dataset Generation

During online training, it is important to generate a high-quality dataset consisting of a small number of representative samples. At time step t, suppose the set of selected key frame time steps is $\mathcal{T} = \{k, 1 \le k \le t\}, |\mathcal{T}| \le W$. For each frame $\mathcal{P}_{i \in \mathcal{T} \cup \{t\}}$, we randomly choose $\lfloor N/|\mathcal{T} \cup \{t\}| \rfloor$ rays $\{\mathbf{o}_j = \mathbf{o}_i, \mathbf{q}_j \in \mathcal{P}_i\}_{j=1}^{N}$ to generate samples. As shown in Fig. 2b, we generate three types of points for training:

1) Free-space Points: To learn the SDF in free space, we sample free-space points $\mathcal{P}_F$ by drawing n_F points $\{\mathbf{x}_n\}_{n=1}^{n_F}$ along each ray j: $\mathbf{x}_n = \mathbf{o}_j + \lambda(\mathbf{q}_j - \mathbf{o}_j)$, where λ is drawn from the uniform distribution $\mathcal{U}(\delta, 1-\delta)$ with margin $\delta > 0$;

2) Surface Samples and Perturbed Points: To provide supervision for the surface reconstruction, we generate $\mathcal{P}_S$ by collecting the surface point $\mathbf{q}_j$ of each ray j and the perturbed points $\mathcal{P}_P$ by sampling n_P points $\{\mathbf{x}_n\}_{n=1}^{n_P}$ along each ray j: $\mathbf{x}_n = \mathbf{o}_j + \alpha'(\mathbf{q}_j - \mathbf{o}_j)$, where $\alpha \sim \mathcal{N}(1, \sigma^2), \alpha' = \min(\max(\alpha, 1-2\sigma), 1+2\sigma)$ with $\sigma > 0$;

3) Ground Truth SDF Computation: For surface points $\mathcal{P}_S$, we have ground truth SDF $d(\mathbf{x}) = 0, \mathbf{x} \in \mathcal{P}_S$. For perturbed points and free-space points, we approximate the ground truth as $\tilde{d}(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{P}_S} \|\mathbf{x} - \mathbf{y}\|_2$.

In our experiments, we have $W = 8$, $N = 20480$, $\delta = 0.05$, $\sigma = 0.06$, $n_F = 1$, and $n_P = 2$.

E. Loss Functions

As shown in Fig. 2e, the SDF prior $d_{ga}(\mathbf{x}) = T(\mathbf{x}; \theta)$ and the neural residual $\delta_d(\mathbf{x}) = R(\mathbf{x}; \beta)$ are combined together to obtain a final SDF prediction:

$$\hat{d}(\mathbf{x}) = d_{ga}(\mathbf{x}) + \delta_d(\mathbf{x}) = T(\mathbf{x}; \theta) + R(\mathbf{x}; \beta). \tag{11}$$

It is important to design appropriate loss functions to train the octree and neural network parameters θ, β.

1) Reconstruction Loss: We apply an L1 loss over the surface points $\mathcal{P}_S$ and the perturbation points $\mathcal{P}_P$ to capture the surface geometry, which is critical for accurate 3D reconstruction:

$$\mathcal{L}_{\text{recon}} = \frac{w_{\text{recon}}^S}{|\mathcal{P}_S|} \sum_{\mathbf{x} \in \mathcal{P}_S} \left|\hat{d}(\mathbf{x})\right| + \frac{w_{\text{recon}}^P}{|\mathcal{P}_P|} \sum_{\mathbf{x} \in \mathcal{P}_P} \left|\hat{d}(\mathbf{x}) - \tilde{d}(\mathbf{x})\right|, \tag{12}$$

where w_{recon}^S and w_{recon}^P are the corresponding weights.

2) Eikonal Loss: To enforce the Eikonal property in (2), we apply another L1 loss for the gradient norm:

$$\begin{aligned} \mathcal{L}_{\text{eik}} = &\frac{w_{\text{eik}}^S}{|\mathcal{P}_S| + |\mathcal{P}_P|} \sum_{\mathbf{x} \in \mathcal{P}_S \cup \mathcal{P}_P} |\|\hat{\mathbf{g}}(\mathbf{x})\|_2 - 1| \\ &+ \frac{w_{\text{eik}}^{\neg S}}{|\mathcal{P}_F|} \sum_{\mathbf{x} \in \mathcal{P}_F} |\|\hat{\mathbf{g}}(\mathbf{x})\|_2 - 1|, \end{aligned} \tag{13}$$

where w_{eik}^S is the weight for surface points $\mathcal{P}_S \cup \mathcal{P}_P$ and $w_{\text{eik}}^{\neg S}$ is the weight for non-surface points $\mathcal{P}_F$. Here, $\hat{\mathbf{g}}(\mathbf{x})$ is obtained from numerical differentiation instead of the auto gradient graph because the numerical differentiation involves more positions $\mathbf{x} \pm \epsilon \mathbf{e}_i$ for each dimension i, which helps the network converge and shows better training stability when the SDF gradient is not well defined at certain positions.

3) Projection Loss: Although the Eikonal loss $\mathcal{L}_{\text{eik}}$ enforces the gradient magnitude, the supervision for the gradient direction and SDF in the distant space is still missing. Hence, we propose a projection loss for free-space points $\mathcal{P}_F$ that are collected along rays:

$$\mathcal{L}_{\text{proj}} = \frac{w_{\text{proj}}}{|\mathcal{P}_F|} \sum_{\mathbf{x} \in \mathcal{P}_F} \left|\hat{d}(\mathbf{x}) - \tilde{d}(\mathbf{x})\right|. \tag{14}$$

Although the above loss has a form similar to (12), we call it *projection* loss because $\tilde{d}(\mathbf{x})$ is actually a loose upper bound for the ground truth SDF $d(\mathbf{x})$. The purpose of this loss is not to make the model predict $\tilde{d}(\mathbf{x})$ exactly at $\mathbf{x}$ but to provide the implicit supervision of the gradient direction so that it speeds up the convergence of other loss functions.

In our experiments, we set $w^S_{\text{recon}} = 1000$, $w^P_{\text{recon}} = 200$, $w^S_{\text{eik}} = 10$, $w^P_{\text{eik}} = 3$, and $w_{\text{proj}} = 100$.

V. Evaluation

In this section, we compare ∇-SDF with four baselines: Voxblox [7], H_2-Mapping [17], PIN-SLAM [2], and HIO-SDF [18]. We first examine the mesh reconstruction and SDF visualizations as a qualitative comparison, then quantitatively evaluate the methods using different metrics. In addition, we perform an ablation study to evaluate the contribution of each component in our method.

We use the Replica dataset [30], which provides eight scenes with one trajectory per scene and 2000 frames per trajectory. To ensure full coverage of each scene, we augmented each trajectory with additional 40 camera views.

A. Qualitative Results

1) Mesh Reconstruction: We first compare the reconstructed meshes. Fig. 5 shows results on the Replica room 0 scene. H_2-Mapping, PIN-SLAM, and ∇-SDF generate complete and high-quality meshes. H_2-Mapping tends to produce smoother surfaces as it only allocates octree voxels near the surface. The completeness of our reconstructions is better than H_2-Mapping. Compared with PIN-SLAM [2], which only predicts SDF values in regions close to the surface, our meshes exhibit smoother geometry with less noise. Compared to HIO-SDF [18], which also estimates continuous and differentiable SDFs, our reconstructions demonstrate substantially higher fidelity.

2) SDF Reconstruction: We visualize z-plane slices of the SDF predictions in Fig. 5. H_2-Mapping and PIN-SLAM estimate truncated SDF only. Although HIO-SDF can predict SDF values over the entire space, it struggles to precisely encode the surface as the zero-level set, which is crucial for robotic tasks where accurate perception of obstacle boundaries is required. In contrast, ∇-SDF faithfully reconstructs the surface position and provides reliable SDF estimates in free space. The predictions of ∇-SDF outside the scene boundaries are less accurate due to the lack of sensor observations but this has little impact on applications where robots operate within the observed workspace.

B. Quantitative Results

We compute two sets of metrics: mesh metrics to evaluate the surface reconstruction quality and SDF metrics to evaluate the overall SDF predictions.

1) Mesh Metrics: We uniformly sample two point clouds, $\mathcal{P}_{\text{g.t.}}$ and $\mathcal{P}_{\text{recon}}$, of 200k points each from the ground-truth mesh and from the reconstructed mesh, and report *Chamfer distance*, *F1 score* ($< 5cm$), *precision*, *recall*, *completion*, *completion ratio* ($< 5cm$), and *accuracy* [2], [17].

As shown in Table I, ∇-SDF outperforms the baselines in recall, completion, and completion ratio. Our method is mostly the second best in the other mesh metrics. Since H_2-Mapping and PIN-SLAM are specially optimized for surface reconstruction, they perform slightly better in metrics like F1 score. However, their mesh has more holes, which can also improve metrics like precision. Voxblox has the best accuracy but performs the worst in other metrics. Since HIO-SDF relies on a volumetric method like Voxblox to generate the SDF dataset, it also performs worse than ∇-SDF.

2) SDF Metrics: We evaluate the SDF predictions of all methods using a regular grid of resolution 1.25 cm. For each scene, the grid boundary is the bounding box of the ground-truth mesh with 15 cm padding. The ground-truth SDF d of each grid center is computed from the ground-truth mesh. Only points with $d \geq -0.1$ m are kept for evaluation. In Table II, we report the *mean absolute error* (MAE) of SDF, the *angular MAE* of the SDF gradient, and the SDF *valid ratio*, defined as the proportion of test positions where a method can predict SDF. To examine the prediction quality in different regions, we categorize the grid points with $-0.1 \leq d \leq 0.2$ m as near surface, and the other points as far from the surface. ∇-SDF marginally performs better than the baselines in all SDF metrics except for near-surface angular MAE of the SDF gradient, where our method is the second best. HIO-SDF fails to train the network stably when its volumetric method does not provide good results. Since H_2-Mapping and PIN-SLAM are able to predict SDF only near the surface, their SDF valid ratio is extremely low, while HIO-SDF and ∇-SDF both cover the whole scene.

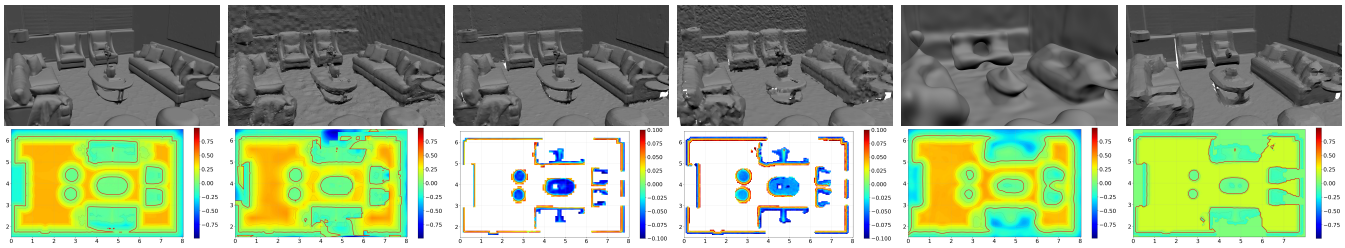
3) Runtime Metrics: We also measure the timing of each method. As shown in Table III, ∇-SDF runs at 8.51 fps, which is the second fastest.

C. Ablation Study

To investigate the contribution of the semi-sparse octree and gradient-augmented interpolation, we train three variants, without the neural residual network, using a regular sparse octree and interpolation without gradient augmentation. As shown in Fig. 6, the SDF prior of our method is smoother and the neural network residual helps recover the geometric details. The neural network residual does not improve the mesh metrics but improves the SDF metrics, as shown in Table IV. When trained with a sparse octree instead, our method performs worse due to more discontinuities caused by the sparsity. Without gradient augmentation, the SDF priors become worse, leading to worse performance. We also train our model without the projection loss, which shows significantly worse metrics because the projection loss provides important guidance about the SDF scale and gradient direction.

VI. Conclusion

This paper developed ∇-SDF, an online hybrid method that builds globally accurate SDFs from streaming point cloud data at scene scales. Our method combines an explicit SDF prior from gradient-augmented interpolation in



(a) Ground Truth (b) Ours (c) H_2-Mapping [17] (d) PIN-SLAM [2] (e) HIO-SDF [18] (f) Voxblox [7]

Fig. 5: Qualitative comparison of mesh reconstruction (top row) and z-plane slice of SDF reconstruction (bottom row) on Replica room 0 [30]. ∇-SDF reconstructs a mesh with the highest completion ratio and accurate SDF both near and far from the surface. H_2-Mapping and PIN-SLAM only learn truncated SDF. HIO-SDF learns an over smooth result. Voxblox significantly under-estimates the SDF.

TABLE I: Mesh reconstruction metrics on the Replica dataset [30]. 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
Chamfer Distance [cm] ↓	∇-SDF	2.42	2.44	1.95	1.99	3.27	2.47	3.67	2.83
	H_2-Mapping	2.29	**1.86**	**1.73**	1.83	**1.49**	**2.24**	2.39	2.46
	PIN-SLAM	2.56	2.13	2.19	**1.70**	1.99	2.57	2.60	2.73
	HIO-SDF	3.79	3.24	3.14	2.93	3.33	3.79	3.90	3.60
	Voxblox	**2.29**	1.92	1.92	1.75	1.93	2.42	**2.39**	**2.42**
F1 Score [<5cm]% ↑	∇-SDF	92.26	91.99	93.34	93.99	87.96	90.86	83.03	89.36
	H_2-Mapping	**95.38**	**95.98**	**96.35**	**97.27**	**95.92**	**94.70**	**94.08**	**94.05**
	PIN-SLAM	93.00	94.57	94.40	95.57	94.13	92.93	92.13	92.81
	HIO-SDF	82.83	86.45	87.96	88.41	84.85	82.51	84.20	84.82
	Voxblox	93.01	93.79	92.6	93.21	92.03	90.71	90.6	91.11
Precision [<5cm]% ↑	∇-SDF	90.92	89.78	91.56	91.56	82.65	89.81	77.03	88.09
	H_2-Mapping	**99.57**	**99.78**	**99.59**	**99.65**	**99.44**	**99.66**	**99.08**	**99.52**
	PIN-SLAM	98.57	98.39	98.59	97.95	98.40	98.06	96.83	98.39
	HIO-SDF	85.86	89.04	90.54	91.24	88.55	84.83	88.24	88.15
	Voxblox	98.18	98.61	95.88	96.66	98.28	97.04	96.41	97.52
Recall [<5cm]% ↑	∇-SDF	**93.63**	**94.30**	**95.18**	**96.56**	**93.99**	**91.93**	**90.99**	**90.68**
	H_2-Mapping	91.53	92.46	93.32	95.00	92.65	90.21	89.56	89.14
	PIN-SLAM	89.83	91.03	90.55	93.29	90.22	88.31	87.87	87.82
	HIO-SDF	79.93	84.00	85.53	85.75	81.44	80.31	80.52	81.74
	Voxblox	88.36	89.42	89.54	89.99	86.53	85.16	85.46	85.5
Completion [cm] ↓	∇-SDF	**2.27**	**1.94**	**1.78**	**1.56**	**1.73**	**2.33**	**2.51**	**2.62**
	H_2-Mapping	3.05	2.51	2.14	1.64	1.98	3.06	3.09	3.35
	PIN-SLAM	3.46	2.90	2.89	1.96	2.58	3.54	3.34	3.77
	HIO-SDF	4.50	3.63	3.51	3.26	3.87	4.09	4.50	4.17
	Voxblox	4.02	3.63	3.27	3.08	3.67	4.24	3.94	4.22
Completion Ratio [<5cm]% ↑	∇-SDF	**93.82**	**94.57**	**95.36**	**96.73**	**94.71**	**92.12**	**91.49**	**90.95**
	H_2-Mapping	90.79	91.87	92.87	94.75	92.11	89.19	88.45	87.88
	PIN-SLAM	88.84	90.30	89.71	92.96	89.34	86.72	86.64	86.36
	HIO-SDF	78.41	83.04	84.68	84.83	79.82	79.20	78.65	80.30
	Voxblox	88.32	89.39	89.11	89.78	86.28	84.95	85.31	85.21
Accuracy [cm] ↓	∇-SDF	2.58	2.95	2.13	2.43	4.82	2.62	4.83	3.04
	H_2-Mapping	1.54	1.21	1.33	1.22	1.01	1.43	1.69	1.58
	PIN-SLAM	1.66	1.37	1.49	1.45	1.31	1.61	1.87	1.70
	HIO-SDF	3.09	2.85	2.78	2.62	2.79	1.49	3.31	3.03
	Voxblox	**0.97**	**0.91**	**1.19**	**1.16**	**0.87**	**1.09**	**1.22**	**1.01**

TABLE II: SDF reconstruction metrics on the Replica dataset [30].

Metric	Region	Method	room 0	room 1	room 2	office 0	office 1	office 2	office 3	office 4
SDF MAE [cm]↓	All	∇-SDF	**2.13**	**2.02**	**2.03**	**1.46**	**1.43**	**2.15**	**2.35**	**2.39**
		Lower by	-34%	-28%	-92%	-44%	-53%	-36%	-31%	-29%
		HIO-SDF	3.27	2.79	39.98	2.60	2.72	3.34	3.40	3.39
		Voxblox	21.03	17.93	24.63	18.23	14.71	19.36	19.15	21.35
	Near	∇-SDF	**1.97**	**1.64**	**1.79**	**1.33**	**1.68**	**2.14**	**2.23**	**2.49**
		HIO-SDF	3.45	2.90	28.32	2.57	3.00	3.36	3.52	3.62
		H_2-Mapping	6.13	6.01	5.54	5.88	5.75	5.81	5.99	6.19
		PIN-SLAM	4.65	8.10	8.06	8.06	8.15	8.11	8.13	8.07
		Voxblox	5.00	4.55	4.51	4.64	3.99	4.98	5.10	4.70
	Far	∇-SDF	**2.26**	**2.38**	**2.24**	**1.58**	**1.13**	**2.16**	**2.45**	**2.31**
		HIO-SDF	3.12	2.69	50.22	2.62	2.39	3.33	3.30	3.20
		Voxblox	32.78	29.41	26.79	29.76	26.55	30.97	29.90	33.56
Grad. MAE [rad]↓	All	∇-SDF	**0.325**	**0.328**	**0.395**	**0.348**	**0.349**	**0.362**	**0.397**	**0.340**
		HIO-SDF	0.924	1.315	1.534	1.272	1.101	1.340	1.326	1.256
		Voxblox	1.049	0.973	1.422	0.989	0.901	1.029	1.059	1.047
	Near	∇-SDF	0.323	0.306	0.396	0.346	0.363	0.364	0.419	0.379
		H_2-Mapping	1.076	1.095	1.043	1.095	1.081	1.085	1.081	1.096
		PIN-SLAM	0.870	1.566	1.571	1.569	1.530	1.578	1.550	1.572
		HIO-SDF	0.975	1.301	1.560	1.281	1.115	1.324	1.331	1.257
		Voxblox	**0.293**	**0.270**	**0.269**	**0.318**	**0.317**	**0.358**	**0.363**	**0.326**
	Far	∇-SDF	**0.326**	**0.348**	**0.394**	**0.350**	**0.333**	**0.360**	**0.343**	**0.311**
		HIO-SDF	0.884	1.328	1.512	1.262	1.084	1.356	1.325	1.255
		Voxblox	1.524	1.513	1.504	1.504	1.488	1.513	1.522	1.520
SDF Valid Ratio [%]↑		H_2-Mapping	16.05	16.62	18.15	17.46	20.30	16.84	17.05	15.43
		PIN-SLAM	25.01	6.47	5.82	7.86	8.63	7.09	7.19	5.67

(a) Ground Truth (b) ∇-SDF (c) ∇-SDF (prior only)

Fig. 6: Comparison of mesh reconstruction using ∇-SDF versus only the octree prior in ∇-SDF. The neural network residual helps recover geometric details.

TABLE III: Runtime comparison on Replica room 0 [30].

	∇-SDF	H_2-Mapping	PIN-SLAM	HIO-SDF	Voxblox
FPS	8.51	**12.36**	8.43	1.99	0.87

TABLE IV: Ablation study results on Replica office 0 [30].

Metric		∇-SDF	Prior Only	Sparse Octree	w/o Grad. Aug.	w/o Proj. Loss
Chamfer Distance ↓		1.98	**1.97**	2.09	2.23	7.38
F1 Score %↑		93.99	**94.14**	93.44	91.44	79.26
Precision %↑		91.56	**92.02**	90.54	86.61	68.19
Recall %↑		96.56	96.36	96.53	**96.84**	94.63
Completion [cm]↓		**1.56**	1.59	1.62	1.60	1.74
Completion Ratio %↑		96.73	96.52	96.75	**97.18**	96.13
Accuracy [cm]↓		2.43	**2.36**	2.57	2.85	13.01
SDF	All	**1.46**	1.52	2.65	3.69	15.76
MAE	Near	**1.33**	1.35	1.49	2.01	2.06
[cm]↓	Far	**1.58**	1.69	3.75	5.30	28.76
Grad.	All	0.348	**0.285**	0.565	0.730	0.931
MAE	Near	0.346	**0.290**	0.469	0.599	0.486
[rad]↓	Far	0.350	**0.280**	0.657	0.855	1.353

a semi-sparse octree with an implicit residual from a neural network feature decoder. Through extensive experiments, we demonstrate that ∇-SDF is more accurate and efficient than state-of-the-art methods. Future work will focus on utilizing ∇-SDF in robot localization, navigation, and manipulation.

REFERENCES

[1] J. Ortiz, A. Clegg, J. Dong, E. Sucar, D. Novotny, M. Zollhoefer, and M. Mukadam, "iSDF: Real-Time Neural Signed Distance Fields for Robot Perception," in *Robotics: Science and Systems (RSS)*, 2022.

[2] Y. Pan, X. Zhong, L. Wiesmann, T. Posewsky, J. Behley, and C. Stachniss, "PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency," *IEEE Transactions on Robotics*, 2024.

[3] Y. Tian, H. Cao, S. Kim, and N. Atanasov, "MISO: Multiresolution Submap Optimization for Efficient Globally Consistent Neural Implicit Reconstruction," in *Robotics: Science and Systems (RSS)*, 2025.

[4] G. Chou, Y. Bahat, and F. Heide, "Diffusion-SDF: Conditional Generative Modeling of Signed Distance Functions," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[5] D. Vicini, S. Speierer, and W. Jakob, "Differentiable Signed Distance Function Rendering," *ACM Trans. Graph.*, 2022.

[6] Y. Wang, Q. Han, M. Habermann, K. Daniilidis, C. Theobalt, and L. Liu, "NeuS2: Fast Learning of Neural Implicit Surfaces for Multi-view Reconstruction," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[7] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-board MAV planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[8] K. Long, Y. Yi, Z. Dai, S. Herbert, J. Cortés, and N. Atanasov, "Sensor-based Distributionally Robust Control for Safe Robot Navigation in Dynamic Environments," *The International Journal of Robotics Research*, 2025.

[9] P. Liu, K. Zhang, D. Tateo, S. Jauhri, J. Peters, and G. Chalvatzaki, "Regularized Deep Signed Distance Fields for Reactive Motion Generation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[10] Y. Li, X. Chi, A. Razmjoo, and S. Calinon, "Configuration Space Distance Fields for Manipulation Planning," 2024.

[11] Y. Li, Y. Zhang, A. Razmjoo, and S. Calinon, "Representing Robot Geometry as Distance Fields: Applications to Whole-body Manipulation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[12] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time Dense Surface Mapping and Tracking," in *IEEE International Symposium on Mixed and Augmented Reality*, 2011.

[13] B. Lee, C. Zhang, Z. Huang, and D. D. Lee, "Online Continuous Mapping using Gaussian Process Implicit Surfaces," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[14] L. Wu, C. Le Gentil, and T. Vidal-Calleja, "VDB-GPDF: Online Gaussian Process Distance Field with VDB Structure," *IEEE Robotics and Automation Letters*, 2025.

[15] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[16] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, and W. Wang, "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction," in *Conference on Neural Information Processing Systems*, 2021.

[17] C. Jiang, H. Zhang, P. Liu, Z. Yu, H. Cheng, B. Zhou, and S. Shen, "H2-Mapping: Real-time Dense Mapping Using Hierarchical Hybrid Representation," *IEEE Robotics and Automation Letters*, 2023.

[18] V. Vasilopoulos, S. Garg, J. Huh, B. Lee, and V. Isler, "HIO-SDF: Hierarchical Incremental Online Signed Distance Fields," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[19] Y. Pan, Y. Kompis, L. Bartolomei, R. Mascaro, C. Stachniss, and M. Chli, "Voxfield: Non-Projective Signed Distance Fields for Online Planning and 3D Reconstruction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[20] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1996.

[21] O. Kähler, V. Adrian Prisacariu, C. Yuheng Ren, X. Sun, P. Torr, and D. Murray, "Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices," *IEEE Transactions on Visualization and Computer Graphics*, 2015.

[22] L. Han, F. Gao, B. Zhou, and S. Shen, "FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[23] A. Millane, H. Oleynikova, E. Wirbel, R. Steiner, V. Ramasamy, D. Tingdahl, and R. Siegwart, "nvblox: GPU-Accelerated Incremental Signed Distance Field Mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[24] L. Wu, K. M. B. Lee, L. Liu, and T. Vidal-Calleja, "Faithful Euclidean Distance Field From Log-Gaussian Process Implicit Surfaces," *IEEE Robotics and Automation Letters*, 2021.

[25] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *European Conference on Computer Vision (ECCV)*, 2020.

[26] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit Geometric Regularization for Learning Shapes," in *International Conference on Machine Learning*, 2020.

[27] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[28] Z. Wang, C. Wang, T. Yoshino, S. Tao, Z. Fu, and T.-M. Li, "HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[29] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding," *ACM Trans. Graph.*, 2022.

[30] J. Straub, T. Whelan, L. Ma, Y. Chen, E. Wijmans, S. Green, J. J. Engel, R. Mur-Artal, C. Ren, S. Verma *et al.*, "The Replica Dataset: A Digital Replica of Indoor Spaces," *arXiv preprint arXiv:1906.05797*, 2019.