OREN: Octree Residual Network for Real-Time Euclidean Signed Distance Mapping

Zhirui Dai[1*] Qihao Qian[1*] Tianxing Fan[1] Nikolay Atanasov[1]

***Abstract*— Reconstructing signed distance functions (SDFs) from point cloud data benefits many robot autonomy capabilities, including localization, mapping, motion planning, and control. Methods that support online and large-scale SDF reconstruction often rely on discrete volumetric data structures, which affects the continuity and differentiability of the SDF estimates. Neural network methods have demonstrated high-fidelity differentiable SDF reconstruction but they tend to be less efficient, experience catastrophic forgetting and memory limitations in large environments, and are often restricted to truncated SDF. This work proposes OREN, a hybrid method that combines an explicit prior from octree interpolation with an implicit residual from neural network regression. 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 OREN 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 3D geometric reconstruction is critical in many robot autonomy and computer vision settings, including localization and mapping [1], [2], rendering and AR/VR [3], [4], autonomous robot navigation [5], [6] and robot manipulation [7], [8]. In robotics, fast updates of the environment model from sensor observations and access to gradient information are important for safe navigation and precise environment interaction, while low memory footprint is important for scalability and onboard operation.

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 fall into roughly three categories: volumetric methods (e.g., [5], [9]), Gaussian Process (GP) methods (e.g., [10], [11]), and neural network methods (e.g., [12], [13]). We review representative works in Sec. II.

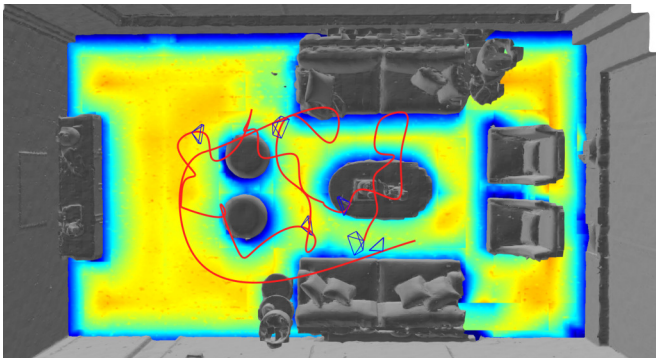


Fig. 1: OREN reconstructs an accurate Euclidean signed distance function online from streaming point cloud data.

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 growing storage for 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 are often restricted to truncated SDF and struggle with catastrophic forgetting in large scenes or in online settings.

We propose OREN, 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 prior. To construct the prior, we use a semi-sparse octree with SDF and gradient estimates stored at the octant vertices and gradient-augmented interpolation to obtain 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 accurate Euclidean SDF reconstruction.

The closest works to ours are H_2-Mapping [14] and Gradient-SDF [15]. Like ours, H_2-Mapping applies trilinear interpolation in a sparse octree to obtain an SDF prior and trains a neural network residual. In contrast, H_2-Mapping reconstructs only truncated (near-surface) SDF. Gradient-SDF is voxel-based and also uses SDF gradients but only for improving surface reconstruction and photometric bundle adjustment, ignoring free-space SDF estimation. In contrast, our method optimizes the octree parameters to obtain an

*Equal contribution

[1]The authors are with the Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA, e-mails: {zhdai,q2qian,t2fan,natanasov}@ucsd.edu.

We gratefully acknowledge support from ARL DCIST CRA W911NF-17-2-0181 and the Ministry of Trade, Industry and Energy (MOTIE), Korea, under the Strategic Technology Development Program, supervised by Korea Institute for Advancement of Technology (KIAT) [Grant No. P0026052].

Code available at https://github.com/ExistentialRobotics/oren

accurate Euclidean SDF prior, corrects the prior via neural residual predictions and achieves real-time operation.

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 from point cloud data. They can be categorized into three groups: volumetric methods [5], [9], [16]–[20], GP-based methods [10], [11], [21], and neural network based methods [1], [12]. We review these categories and then discuss the recent trend of using hybrid models for SDF reconstruction.

A. Volumetric SDF Reconstruction

Volumetric methods [5], [19] achieve real-time truncated SDF (TSDF) reconstruction using voxel hashing for efficient updates and queries. Voxblox [5] builds a TSDF layer by projective distance (from voxel center to observed surface point) and updates values via breadth-first search (BFS). Both the projective distance and the BFS path length introduce inaccuracies in SDF estimates. Subsequent works [18], [19] reduce these errors by using non-projective distance and replacing BFS path length with the distance to the nearest oriented surface point. Gradient-SDF [15] shows that estimating TSDF together with its gradient improves accuracy and can be used for SDF-based camera tracking and bundle adjustment. However, these methods rely on discrete SDF representations, which are non-differentiable and have accuracy limited to the grid resolution. In contrast, our method enables real-time learning of continuous and differentiable SDF. We first estimate SDF with explicit discrete priors stored in an octree, and then use implicit neural features to predict residuals and correct the prior, which forms a compact differentiable representation of continuous SDF.

B. Gaussian Process SDF Reconstruction

GP methods learn continuous SDF representations, and support gradient computation and uncertainty quantification [10], [11], [21]. GPIS [10] uses GP to estimate oriented surface points and regress SDF online, learning the surface implicitly as the zero-level set of SDF. However, GPIS cannot extrapolate SDF predictions away from the surface. Log-GPIS [21] and VDB-GPDF [11] learn unsigned distance function in log space, which generalizes well globally but does not provide sign information. GP methods scale poorly to large environments due to the cubic complexity of the matrix inverse during training. In comparison, our method uses octree interpolation, which has $O(1)$ complexity, and matrix multiplication, which has roughly quadratic complexity $O(n^2)$ for n hidden dimension, to compute the SDF prior and the SDF residual respectively. Our octree structure also has a smaller memory footprint than GP's gram matrix.

C. Neural Network SDF Reconstruction

DeepSDF [12] was among the first to demonstrate that neural networks can learn compact and continuous implicit SDF representations. This inspired many subsequent neural SDF methods. iSDF [1] incrementally learns SDF by iteratively updating an MLP with training samples from a key frame set. iSDF also uses Eikonal regularization to enforce the Eikonal property of SDF. NeuS [13] jointly learns SDF and a neural radiance field (NeRF) [22], letting the two improve each other via SDF-based volume density estimates. Other works, like IGR [23] and NGLoD [24], propose various network designs, loss functions, and training procedures to improve SDF reconstruction accuracy. These works show that neural networks can learn near-surface SDF accurately enough for high-fidelity surface reconstruction. However, accuracy far from the surface is rarely evaluated. Neural network methods [12], [23]–[25] that learn Euclidean SDF are often object-level and require extensive training data and time for satisfactory accuracy.

D. Hybrid Methods for SDF Reconstruction

Recently, hybrid models combining explicit geometric structures with implicit neural features show promising results. PIN-SLAM [2] stores neural features in near-surface voxels. Given a query, its SDF prediction is a weighted sum of k predictions, obtained by feeding the k nearest neural features and local positions into a global decoder. H_2-Mapping [14] combines an octree-based SDF prior with a neural residual. LCP-Fusion [26] extends H_2-Mapping to SLAM, combining the SDF prior and implicit neural feature for localization. However, these methods still learn truncated SDF. HIO-SDF [27] 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 fixed-size network tends to learn an over-smooth SDF.

In contrast, our method, OREN, 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 a sign indicating whether $\mathbf{x}$ is inside or outside of $\mathcal{O}$:

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

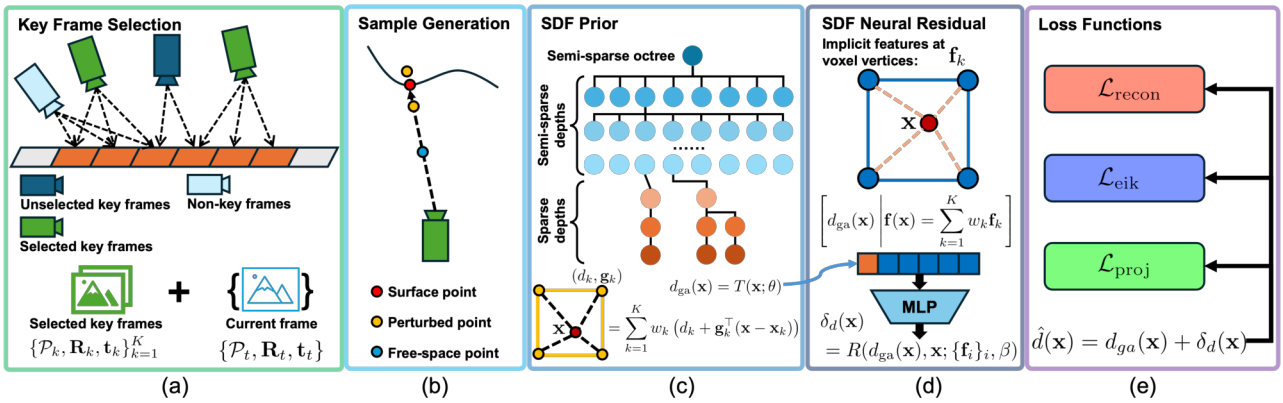


Fig. 2: Method overview: a) We keep key frames with small overlap that maximize surface coverage; b) From the selected key frames and the current frame, we generate three sample types: surface points, perturbed points around the surface, and free-space points; c) To predict SDF, we first obtain a prior $d_{ga}(\mathbf{x})$ via gradient-augmented interpolation in a semi-sparse octree, where each vertex stores an SDF value and gradient; d) We also interpolate an implicit neural feature for $\mathbf{x}$ from features stored at the octant vertices, and feed it with the prior $d_{\text{ga}}(\mathbf{x})$ into an MLP decoder to obtain an SDF residual $\delta_d(\mathbf{x})$; e) The prior and residual are combined into the final SDF prediction $\hat{d}(\mathbf{x}) = d_{ga}(\mathbf{x}) + \delta_d(\mathbf{x})$, and the model is trained with three loss functions: reconstruction loss, Eikonal loss and projection loss.

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) 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 from a range sensor (e.g., LiDAR or depth camera), $\{\mathbf{o}_t, \mathcal{P}_t\}_{t=1}$, where $\mathbf{o}_t$ is the sensor position at time 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 want $\hat{d}$ to capture the SDF gradient accurately.

IV. Octree Residual Network for Learning SDF

Our method employs a hybrid model to reconstruct SDF. We use a semi-sparse octree, where certain octants contain no surface, to store explicit SDF and gradient estimates and compute a coarse SDF prior, described in Sec. IV-A. To recover geometric details, we also store implicit neural features at the octant vertices and use an MLP decoder to correct the prior SDF estimates. The implicit neural residual prediction is described in Sec. IV-B. To train our model efficiently, we maintain key frames that cover the observed surface with small overlap between adjacent frames, as described in Sec. IV-C. Then, from the key frames and the latest frame, we generate a dataset of different sample types, discussed in Sec. IV-D, and use it to train the model with loss functions proposed in Sec. IV-E. Fig 2 shows an overview.

A. SDF Prior via Gradient-Augmented Octree Interpolation

1) Semi-Sparse Octree: The SDF prior at position $\mathbf{x}$ is obtained by interpolation in a semi-sparse octree with N layers. We call an octree layer *dense* when all octants form a regular grid; *semi-sparse* when child octants containing surface points and all their siblings are created; and *sparse* when only child octants containing surface points are created.

We use a semi-sparse octree of resolution r, where the first M layers are semi-sparse and the remaining $N - M$ layers

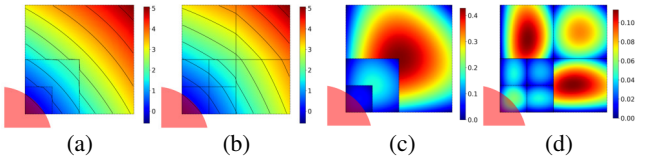


Fig. 3: 2D visualization of SDF interpolation without gradient augmentation using (a) a sparse octree and (b) a semi-sparse octree with corresponding interpolation error shown in (c) and (d) respectively. The bottom-left red region is an obstacle containing one vertex.

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. The choice of M trades memory for accuracy: more semi-sparse layers yield a more accurate prior but use more memory. Since the octant count grows roughly eight-fold per layer, making the finest layers semi-sparse adds substantial memory but little accuracy, as the prior is already accurate near the surface. We therefore keep only the first M layers semi-sparse and the rest sparse.

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. It is a standard formulation but expressed in vector form for brevity. 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:

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

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 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 $N = 8$, $M = 5$ and $r = 10$ cm.

B. SDF Residual via Neural Feature Decoding

The SDF prior's accuracy is limited by the octree resolution, lacking geometric details. To achieve high fidelity, OREN learns a residual correction to the SDF prior via a neural network $R(d_{\text{ga}}(\mathbf{x}), \mathbf{x}; \{\mathbf{f}_i\}_i, \beta)$, a composition of octree feature interpolation $\mathbf{f}(x) = \sum_k w_k \mathbf{f}_k$ with implicit neural features $\mathbf{f}_i \in \mathbb{R}^F$ stored at octree vertices and an MLP decoder $D(d, \mathbf{f}(x); \beta)$.

To enable continual learning as the sensor moves, we assign each octant vertex an implicit feature $\mathbf{f}_i$. Octree expansion automatically adds more features $\mathbf{f}_i$ to near-surface regions in smaller octants. The features $\mathbf{f}_i$ are initialized to zero and optimized together with the network weights β.

As shown in Fig. 2d, for each query point $\mathbf{x}$, we locate the leaf octant containing $\mathbf{x}$ and compute the interpolated neural feature $\mathbf{f}(\mathbf{x}) = \sum_{k=1}^{8} w_k \mathbf{f}_k$ with $\mathbf{f}_k$ at the octant vertices,

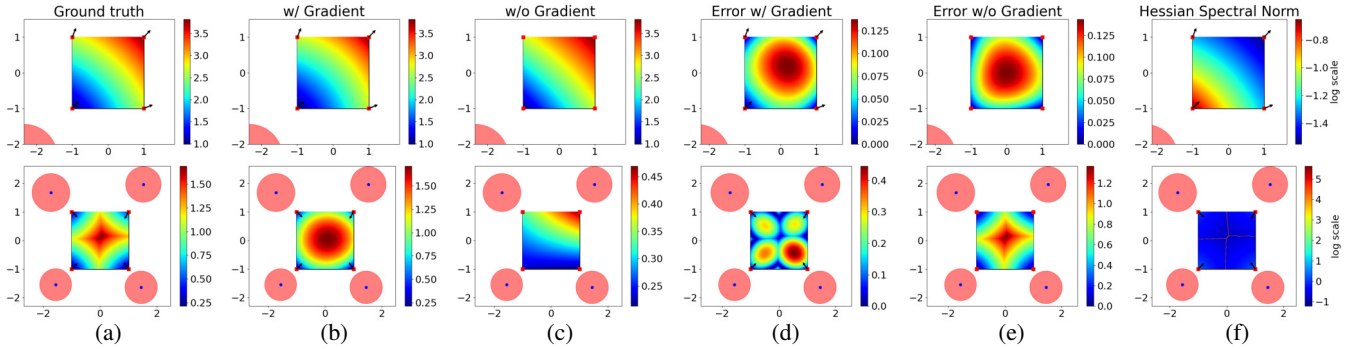


Fig. 4: 2D interpolation with and without gradient augmentation for one (red region, top row) and four obstacles (red regions, bottom row). Gradient-augmented interpolation produces a better SDF prior in (b) with smaller error in (d). Empirically, positions where the SDF gradient is not well defined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

where w_k is the same weight used in (4). The feature $\mathbf{f}(\mathbf{x})$ and the prior $d_{\text{ga}}(\mathbf{x})$ are fed into a decoder to predict the SDF residual $\delta_d(\mathbf{x}) = D(d_{\text{ga}}(\mathbf{x}), f(\mathbf{x}); \beta)$. In the evaluation, we have $F = 3$ and the MLP has two 32-dim hidden layers with LeakyReLU activation.

C. Key Frame Selection

Real-time SDF learning requires a compact, representative set of training frames. We adopt the key-frame selection of H_2-Mapping [14]. 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 and V_l are the sets of surface octants observed by the current frame and the last inserted key frame, respectively, and $c_{\min} \in [0, 1]$ is a threshold. This ensures the frames cover the observed surface with little overlap.

As key frames accumulate, we select only W of them to maintain real-time operation. We incrementally select the frame that observes the most octants, mask them out, and repeat until W frames are collected. If all octants are masked, we reset the mask except those from the last selected key frame, and continue. This ensures the selected key frames maximally cover the scene. In the evaluation, we set $W = 8$.

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, i.e., $\mathbf{x}_n = \mathbf{q}_j + \alpha(\mathbf{q}_j - \mathbf{o}_j)/\|\mathbf{q}_j - \mathbf{o}_j\|_2$, where α has uniform distribution on $(-3\sigma, \sigma) \cup (\sigma, 3\sigma)$.

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 $\tilde{d}(\mathbf{x}) = \text{sign}(\mathbf{x}) \min_{\mathbf{y} \in \mathcal{P}_S} \|\mathbf{x} - \mathbf{y}\|_2$.

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 $\delta_d(\mathbf{x}) = R(d_{\text{ga}}(\mathbf{x}), \mathbf{x}; \{\mathbf{f}_i\}_i, \beta)$ are combined together to obtain a final SDF prediction:

$$\hat{d}(\mathbf{x}) = d_{ga}(\mathbf{x}) + \delta_d(\mathbf{x}). \tag{11}$$

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:

$$\mathcal{L}_{\text{recon}} = \underbrace{\frac{w^S_{\text{recon}}}{|\mathcal{P}_S|} \sum_{\mathbf{x} \in \mathcal{P}_S} \left|\hat{d}(\mathbf{x})\right|}_{\mathcal{L}_{\text{surface}}} + \underbrace{\frac{w^P_{\text{recon}}}{|\mathcal{P}_P|} \sum_{\mathbf{x} \in \mathcal{P}_P} \left(c_j^{\text{lo}} + c_j^{\text{up}}\right)}_{\mathcal{L}_{\text{perturbation}}},$$
$$c_j^{\text{lo}} = \max\left(e^{\eta(\underline{b}_j - |\hat{d}(\mathbf{x})|)} - 1, 0\right), c_j^{\text{up}} = \max\left(|\hat{d}(\mathbf{x})| - \bar{b}_j, 0\right), \tag{12}$$

where w^S_{recon} and w^P_{recon} are the corresponding weights and $\underline{b}_j$ and $\bar{b}_j$ are lower and upper bounds, respectively. We set $\eta = 10$. For perturbed points, $\underline{b}_j = \sigma$, $\bar{b}_j = |\tilde{d}(\mathbf{x})|$.

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

$$\mathcal{L}_{\text{eik}} = \sum_{k \in \{S,P,F\}} \frac{w^k_{\text{eik}}}{|\mathcal{P}_k|} \sum_{\mathbf{x} \in \mathcal{P}_k} \left| \|\hat{\mathbf{g}}(\mathbf{x})\|_2 - 1 \right|, \tag{13}$$

over three different kinds of samples $\mathcal{P}_S$, $\mathcal{P}_P$ and $\mathcal{P}_F$, where w^k_{eik} is the weight for the corresponding sample set $\mathcal{P}_k$. 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. Although the gradient prior $\mathbf{g}_k$ may be constrained to a unit vector directly, we found that

applying the Eikonal loss to $\hat{\mathbf{g}}(\mathbf{x})$, which depends on $\mathbf{g}_k$, performs better.

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 $w^S_{\text{recon}} = 1000$, $w^P_{\text{recon}} = 200$, $w^S_{\text{eik}} = w^F_{\text{eik}} = 10$, $w^P_{\text{eik}} = 3$, and $w_{\text{proj}} = 100$.

V. Evaluation

In this section, we compare OREN with four baselines: Voxblox [5], H_2-Mapping [14], PIN-SLAM [2], and HIO-SDF [27]. We first examine mesh and SDF reconstructions 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 [28], which provides eight synthesized scenes with one trajectory per scene to compare OREN with the baselines. We also test all methods on real-data from the Newer College dataset [29]. For all methods and datasets, we use the ground truth poses. For PIN-SLAM, we disable its localization module.

A. Qualitative Results

1) Mesh Reconstruction: Fig. 5 shows mesh reconstruction results on the Replica room 0 scene. H_2-Mapping, PIN-SLAM, and OREN 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 [27], which also estimates continuous and differentiable SDFs, our reconstructions demonstrate substantially higher fidelity. We also present the reconstructed meshes of the Newer College dataset by OREN and PIN-SLAM in Fig. 6, which shows that our method also outperforms the baselines on large scale real dataset. We also show reconstructions on four diverse large-scale outdoor scenes in Fig. 7. OREN runs in real-time on these scenes.

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 a zero-level set, which is crucial for robotic tasks where accurate perception of obstacle boundaries is required. In contrast, OREN faithfully reconstructs the surface position and provides reliable SDF estimates in free space. The predictions of OREN 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

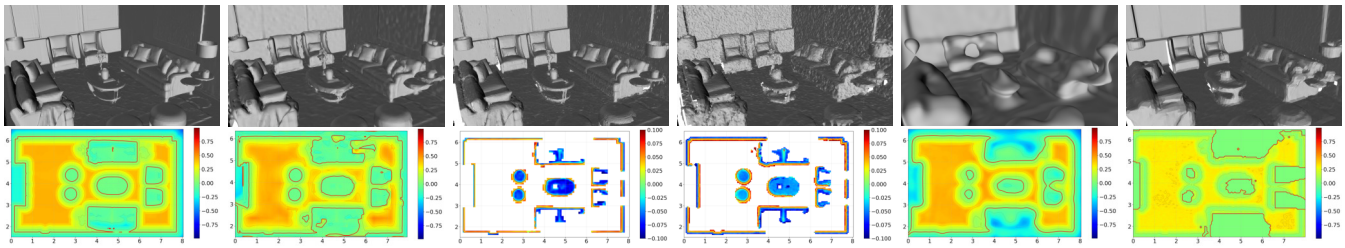
We compute two sets of metrics: mesh metrics to evaluate the surface reconstruction quality and SDF metrics to evaluate the overall SDF predictions.

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-L1 distance*, *F1 score*, *precision*, *recall*, *completion*, *completion ratio*, and *accuracy* [2], [14].

As shown in Table I, OREN 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 performs mostly worse than other methods. Since HIO-SDF relies on a volumetric method like Voxfield [19] to generate the SDF dataset, it also performs worse than OREN.

2) SDF Metrics: We evaluate the SDF predictions of all methods using a regular grid of resolution 1.25 cm (20 cm for Newer College). For each scene, the ground-truth SDF d of each grid center is computed from the ground-truth mesh (point cloud for Newer College). 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. OREN marginally performs better than the baselines in all SDF metrics. 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 ratios are extremely low, while HIO-SDF and OREN both cover the whole scene.

3) Runtime Metrics: We measure the timing and peak memory of each method on Intel 14900K CPU and NVIDIA RTX 3090 GPU. As shown in Table III, OREN is the fastest on both the small-scale Replica room 0 and the large-scale Newer College sequence, running at 13.96 and 14.28 fps, respectively. Voxblox is known for its fast TSDF estimation but its ESDF estimation is significantly slower due to the integration. OREN uses the least GPU memory among all methods (1.33 GB on Replica and 1.59 GB on Newer College). Its CPU memory is higher than the lightest baselines because the semi-sparse octree allocates additional vertices but stays far below HIO-SDF and modest even on the large Newer College scene. This efficiency is also confirmed by deployment on an NVIDIA Jetson Orin with 16 GB RAM, where OREN runs at 7 fps.



(a) Ground Truth (b) Ours (c) H_2-Mapping [14] (d) PIN-SLAM [2] (e) HIO-SDF [27] (f) Voxblox [5]

Fig. 5: Qualitative comparison of mesh reconstruction (top row) and z-plane slice of SDF reconstruction (bottom row) on Replica room 0 [28]. OREN 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 [28] and the Newer College dataset [29]. The best and second best results are bold and underlined, respectively. For the Replica dataset, $\delta = 5$cm. For the Newer College dataset, $\delta = 20$cm.

Metric	Method	room 0	room 1	room 2	office 0	office 1	office 2	office 3	office 4	Newer College
Completion [cm] ↓	OREN	**2.55**	**1.71**	**1.72**	**1.52**	**1.50**	**2.41**	**2.83**	**2.76**	**10.66**
	H_2-Mapping	3.05	2.51	2.14	1.64	1.98	3.06	3.09	3.35	21.94
	PIN-SLAM	3.46	2.90	2.89	1.96	2.58	3.54	3.34	3.77	16.71
	HIO-SDF	4.50	3.63	3.51	3.26	3.87	4.09	4.50	4.17	72.86
	Voxblox	3.69	2.16	2.31	1.71	2.07	3.52	4.37	4.38	21.30
Completion Ratio [$< \delta$]% ↑	OREN	**93.32**	**96.06**	**95.66**	**97.13**	**96.51**	**92.44**	**89.52**	**90.67**	**94.20**
	H_2-Mapping	90.79	91.87	92.87	94.75	92.11	89.19	88.45	87.88	61.58
	PIN-SLAM	88.84	90.30	89.71	92.96	89.34	86.72	86.64	86.36	72.83
	HIO-SDF	78.41	83.04	84.68	84.83	79.82	79.20	78.65	80.30	10.05
	Voxblox	87.09	91.26	90.41	91.92	89.85	84.33	80.21	83.06	60.31
Recall [$< \delta$]% ↑	OREN	**92.87**	**95.84**	**95.48**	**96.81**	**96.21**	**92.26**	88.49	**90.51**	**93.99**
	H_2-Mapping	91.53	92.46	93.32	95.00	92.65	90.21	**89.56**	89.14	57.96
	PIN-SLAM	89.83	91.03	90.55	93.29	90.22	88.31	87.87	87.82	70.40
	HIO-SDF	79.93	84.00	85.53	85.75	81.44	80.31	80.52	81.74	4.72
	Voxblox	88.17	91.81	90.79	92.18	90.62	85.67	82.21	84.94	56.64
Precision [$< \delta$]% ↑	OREN	87.04	90.77	91.70	86.90	88.69	90.14	80.59	89.00	**90.69**
	H_2-Mapping	**99.57**	**99.78**	**99.59**	**99.65**	**99.44**	**99.66**	**99.08**	**99.52**	52.97
	PIN-SLAM	98.57	98.39	98.59	97.95	98.40	98.06	96.83	98.39	64.63
	HIO-SDF	85.86	89.04	90.54	91.24	88.55	84.83	88.24	88.15	4.46
	Voxblox	96.18	97.94	94.56	95.31	98.13	93.69	91.44	95.55	51.84
F1 Score [$< \delta$]% ↑	OREN	89.86	93.24	93.55	91.59	92.29	91.19	84.36	89.75	**92.31**
	H_2-Mapping	**95.38**	**95.98**	**96.35**	**97.27**	**95.92**	**94.70**	**94.08**	**94.05**	55.35
	PIN-SLAM	93.00	94.57	94.40	95.57	94.13	92.93	92.13	92.81	67.39
	HIO-SDF	82.83	86.45	87.96	88.41	84.85	82.51	84.20	84.82	4.59
	Voxblox	92.00	94.77	92.64	93.72	94.23	89.50	86.58	89.94	54.14
Chamfer-L1 Distance [cm] ↓	OREN	2.58	1.96	1.85	2.07	1.85	2.35	3.26	2.60	**9.36**
	H_2-Mapping	**2.29**	1.86	**1.73**	1.83	1.49	**2.24**	**2.39**	**2.46**	28.4
	PIN-SLAM	2.56	2.13	2.19	1.70	1.99	2.57	2.60	2.73	21.64
	HIO-SDF	3.79	3.24	3.14	2.93	3.33	3.79	3.90	3.60	422.29
	Voxblox	2.72	**1.65**	1.90	**1.47**	**1.44**	2.60	3.26	3.02	23.37
Accuracy [cm] ↓	OREN	2.60	2.21	1.98	2.62	2.20	2.29	3.69	2.44	**8.07**
	H_2-Mapping	**1.54**	1.21	**1.33**	**1.22**	1.01	**1.43**	**1.69**	**1.58**	34.86
	PIN-SLAM	1.66	1.37	1.49	1.45	1.31	1.61	1.87	1.70	26.58
	HIO-SDF	3.09	2.85	2.78	2.62	2.79	1.49	3.31	3.03	771.72
	Voxblox	1.74	**1.14**	1.50	1.22	**0.81**	1.68	2.16	1.67	25.44

TABLE II: SDF reconstruction metrics on the Replica dataset [28] and the Newer College dataset [29].

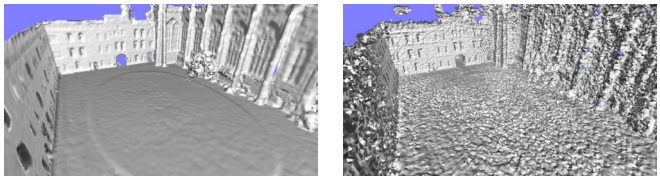
Metric	Region	Method	room 0	room 1	room 2	office 0	office 1	office 2	office 3	office 4	Newer College
SDF MAE [cm]↓	All	OREN	**2.15**	**2.13**	**2.26**	**1.93**	**2.16**	**2.32**	**2.65**	**2.36**	**56.00**
		HIO-SDF	3.27	2.79	39.98	2.60	2.72	3.34	3.40	3.39	301.56
		Voxblox	3.13	2.54	2.73	3.02	2.73	3.47	3.74	3.08	62.92
	Near	OREN	**1.67**	**1.26**	**1.82**	**1.30**	**1.43**	**1.84**	**2.15**	**2.11**	16.96
		HIO-SDF	3.45	2.90	28.32	2.57	3.00	3.36	3.52	3.62	58.34
		H_2-Mapping	6.13	6.01	5.54	5.88	5.75	5.81	5.99	6.19	**9.60**
		PIN-SLAM	4.65	8.10	8.06	8.06	8.15	8.11	8.13	8.07	22.64
		Voxblox	2.78	2.15	2.23	2.20	2.17	2.85	3.41	2.96	19.63
	Far	OREN	**2.39**	**2.66**	**2.49**	**2.33**	2.70	**2.58**	**2.98**	**2.49**	**59.62**
		HIO-SDF	3.12	2.69	50.22	2.62	**2.39**	3.33	3.30	3.20	324.12
		Voxblox	3.37	2.85	3.08	3.71	3.27	3.92	4.00	3.16	67.32
Grad. MAE [rad]↓	All	OREN	**0.153**	**0.157**	**0.160**	**0.160**	**0.174**	**0.162**	**0.158**	**0.154**	**0.184**
		HIO-SDF	0.924	1.315	1.534	1.272	1.101	1.340	1.326	1.256	1.529
		Voxblox	0.230	0.195	0.222	0.251	0.190	0.271	0.279	0.232	0.777
	Near	OREN	**0.120**	**0.116**	**0.117**	**0.128**	**0.129**	**0.117**	**0.130**	**0.117**	**0.156**
		H_2-Mapping	1.076	1.095	1.043	1.095	1.081	1.085	1.081	1.096	1.379
		PIN-SLAM	0.870	1.566	1.571	1.569	1.530	1.578	1.550	1.572	1.533
		HIO-SDF	0.975	1.301	1.560	1.281	1.115	1.324	1.331	1.257	1.565
		Voxblox	0.282	0.183	0.204	0.259	0.179	0.301	0.336	0.299	1.435
	Far	OREN	**0.165**	**0.177**	**0.176**	**0.175**	0.199	**0.180**	**0.171**	**0.167**	**0.198**
		HIO-SDF	0.884	1.328	1.512	1.262	1.084	1.356	1.325	1.255	1.526
		Voxblox	0.208	0.201	0.230	0.246	**0.198**	0.257	0.256	0.205	0.720
SDF Valid Ratio [%]↑		H_2-Mapping	16.05	**16.62**	**18.15**	**17.46**	**20.30**	**16.84**	**17.05**	**15.43**	28.22
		PIN-SLAM	**25.01**	6.47	5.82	7.86	8.63	7.09	7.19	5.67	**47.19**

TABLE III: Runtime and peak memory on Replica room 0 [28] and Newer College [29]. H_2-Mapping [14] is tested on Newer College by converting each LiDAR scan into 4 depth images since its implementation lacks LiDAR support.

		OREN	H_2-Mapping	PIN-SLAM	HIO-SDF	Voxblox
Replica room 0	FPS ↑	**13.96**	12.36	8.43	1.99	0.87
	CPU [GB] ↓	3.81	2.46	2.23	10.33	**1.19**
	GPU [GB] ↓	**1.33**	13.96	6.12	2.14	N/A
Newer College	FPS ↑	**14.28**	9.03*	8.95	3.80	0.20
	CPU [GB] ↓	3.10	2.37	5.98	7.08	**2.04**
	GPU [GB] ↓	**1.59**	9.90	7.56	2.63	N/A

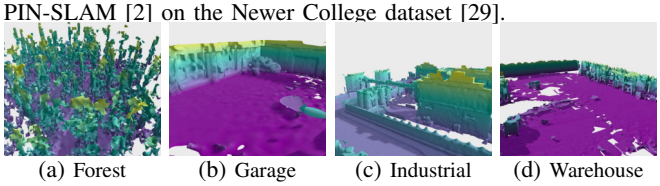
TABLE IV: Ablation study results on Replica room 1 [28].

Metric		OREN	Prior Only	Sparse Octree	w/o Grad. Aug.	w/o Proj. Loss
F1 Score %↑		**93.33**	93.07	93.01	92.09	89.22
Precision %↑		**90.98**	90.48	90.45	88.61	84.04
Recall %↑		95.79	95.82	95.71	**95.86**	95.07
Completion Ratio %↑		96.00	96.05	95.95	**96.17**	95.65
Completion [cm]↓		**1.705**	1.709	1.717	1.709	1.775
Accuracy [cm]↓		**2.201**	2.234	2.228	2.441	3.045
Chamfer Distance ↓		**1.953**	1.971	1.972	2.075	2.410
SDF	All	**1.862**	2.185	2.090	2.208	18.688
MAE	Near	**1.203**	1.320	1.278	1.612	2.303
[cm]↓	Far	**2.468**	2.981	2.837	2.757	33.756
Grad.	All	**0.1527**	0.1529	0.1626	0.1661	0.1713
MAE	Near	0.1166	**0.1107**	0.1229	0.1298	0.1328
[rad]↓	Far	**0.1770**	0.1823	0.1895	0.1890	0.2519



(a) OREN (b) PIN-SLAM

Fig. 6: Comparison of mesh reconstruction using OREN versus PIN-SLAM [2] on the Newer College dataset [29].



(a) Forest (b) Garage (c) Industrial (d) Warehouse

Fig. 7: Reconstructions by OREN on four large outdoor scenes.

C. Ablation Study

To assess each component, we train four variants: without the neural residual, with a regular sparse octree, without gradient augmentation, and without the projection loss. As shown in Table IV, the neural residual improves both mesh reconstruction quality and SDF prediction accuracy. With a sparse octree, our method performs worse due to discontinuities. Without gradient augmentation, the SDF priors become worse, leading to worse performance. Removing the projection loss significantly worsens metrics, as it provides important guidance on the SDF scale and gradient direction. Finally, the numbers of semi-sparse layers trades off memory for accuracy. On Replica room 0, making all layers semi-sparse nearly doubles the allocated octants from 21418 to 41073 (+92%) and the vertices from 37130 to 51174 (+38%), without significant accuracy gain, confirming that only the coarser layers need to be semi-sparse.

VI. Conclusion

This paper developed OREN, an online hybrid method for globally accurate SDF estimation from streaming point cloud data at large scales. Our method combines an explicit octree prior with an implicit neural residual. Our evaluation shows that this combination achieves more accurate and efficient SDF reconstruction than state-of-the-art methods. Future work will focus on utilizing OREN in robot localization, navigation, and manipulation.

References

[1] J. Ortiz, A. Clegg *et al.*, "iSDF: Real-Time Neural Signed Distance Fields for Robot Perception," in *RSS*, 2022.

[2] Y. Pan, X. Zhong *et al.*, "PIN-SLAM: LiDAR SLAM Using a Point-Based Implicit Neural Representation for Achieving Global Map Consistency," *IEEE T-RO*, 2024.

[3] G. Chou, Y. Bahat, and F. Heide, "Diffusion-SDF: Conditional Generative Modeling of Signed Distance Functions," in *ICCV*, 2023.

[4] Y. Wang, Q. Han *et al.*, "NeuS2: Fast Learning of Neural Implicit Surfaces for Multi-view Reconstruction," in *ICCV*, 2023.

[5] H. Oleynikova, Z. Taylor *et al.*, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for On-board MAV planning," in *IROS*, 2017.

[6] K. Long, Y. Yi *et al.*, "Sensor-based Distributionally Robust Control for Safe Robot Navigation in Dynamic Environments," *IJRR*, 2025.

[7] P. Liu, K. Zhang *et al.*, "Regularized Deep Signed Distance Fields for Reactive Motion Generation," in *IROS*, 2022.

[8] Y. Li, X. Chi *et al.*, "Configuration Space Distance Fields for Manipulation Planning," 2024.

[9] R. A. Newcombe, S. Izadi *et al.*, "KinectFusion: Real-time Dense Surface Mapping and Tracking," in *ISMAR*, 2011.

[10] B. Lee, C. Zhang *et al.*, "Online Continuous Mapping using Gaussian Process Implicit Surfaces," in *ICRA*, 2019.

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

[12] J. J. Park, P. Florence *et al.*, "DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation," in *CVPR*, 2019.

[13] P. Wang, L. Liu *et al.*, "NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction," in *NeurIPS*, 2021.

[14] C. Jiang, H. Zhang *et al.*, "H2-Mapping: Real-time Dense Mapping Using Hierarchical Hybrid Representation," *IEEE RA-L*, 2023.

[15] C. Sommer, L. Sang *et al.*, "Gradient-SDF: A Semi-Implicit Surface Representation for 3D Reconstruction," in *CVPR*, 2022.

[16] B. Curless and M. Levoy, "A Volumetric Method for Building Complex Models from Range Images," in *SIGGRAPH*, 1996.

[17] O. Kähler, V. Adrian Prisacariu *et al.*, "Very High Frame Rate Volumetric Integration of Depth Images on Mobile Devices," *IEEE TVCG*, 2015.

[18] L. Han, F. Gao *et al.*, "FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots," in *IROS*, 2019.

[19] Y. Pan, Y. Kompis *et al.*, "Voxfield: Non-Projective Signed Distance Fields for Online Planning and 3D Reconstruction," in *IROS*, 2022.

[20] A. Millane, H. Oleynikova *et al.*, "nvblox: GPU-Accelerated Incremental Signed Distance Field Mapping," in *ICRA*, 2024.

[21] L. Wu, K. M. B. Lee *et al.*, "Faithful Euclidean Distance Field From Log-Gaussian Process Implicit Surfaces," *IEEE RA-L*, 2021.

[22] B. Mildenhall, P. P. Srinivasan *et al.*, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," in *ECCV*, 2020.

[23] A. Gropp, L. Yariv *et al.*, "Implicit Geometric Regularization for Learning Shapes," in *ICML*, 2020.

[24] T. Takikawa, J. Litalien *et al.*, "Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes," in *CVPR*, 2021.

[25] Z. Wang, C. Wang *et al.*, "HotSpot: Signed Distance Function Optimization with an Asymptotically Sufficient Condition," in *CVPR*, 2025.

[26] J. Wang, Y. Deng *et al.*, "LCP-Fusion: A Neural Implicit SLAM with Enhanced Local Constraints and Computable Prior," in *IROS*, 2024.

[27] V. Vasilopoulos, S. Garg *et al.*, "HIO-SDF: Hierarchical Incremental Online Signed Distance Fields," in *ICRA*, 2024.

[28] J. Straub, T. Whelan *et al.*, "The Replica Dataset: A Digital Replica of Indoor Spaces," *preprint arXiv:1906.05797*, 2019.

[29] L. Zhang, M. Camurri *et al.*, "Multi-Camera LiDAR Inertial Extension to the Newer College Dataset," *preprint arXiv:2112.08854*, 2021.