I-Filtering: Implicit Filtering for Learning Neural Distance Functions From 3D Point Clouds

Shengtao Li, Yudong Liu, Ge Gao, Ming Gu, and Yu-Shen Liu, *Member, IEEE*

***Abstract*—Neural implicit functions including signed distance functions (SDFs) and unsigned distance functions (UDFs) have shown powerful ability in fitting the shape geometry. However, inferring continuous distance fields from discrete unoriented point clouds still remains a challenge. The neural network typically fits the shape with a rough surface and omits fine-grained geometric details such as shape edges and corners. In this paper, we propose a novel non-linear implicit filter to smooth the implicit field while preserving high-frequency geometry details. Our novelty lies in that we can filter the surface (zero level set) by the neighbor input points with gradients of the signed distance field. By moving the input raw point clouds along the gradient, our proposed implicit filtering can be extended to non-zero level sets to keep the promise consistency between different level sets, which consequently results in a better regularization of the zero level set. Since the unsigned distance function is non-differentiable at the zero level set and lacks a stable gradient field, we further propose a gradient immutable training schema to migrate the filter to the unsigned distance function learned from point clouds. By leveraging the UDF training schema, we also improve sparse-view reconstruction results. We conduct comprehensive experiments in surface reconstruction from objects, complex scene point clouds, and multi-view images, and we further extend to the point normal estimation and point cloud upsampling tasks. The numerical and visual comparisons demonstrate our improvements over the state-of-the-art methods under the widely used benchmarks.**

***Index Terms*—Implicit filtering, signed distance functions, unsigned distance functions, sparse-view reconstruction, normal estimation, point cloud upsampling.**

I. Introduction

RECONSTRUCTING surfaces from 3D point clouds is an important task in 3D computer vision. Recently signed distance functions (SDFs) learned by neural networks have been a widely used strategy for representing high-fidelity 3D geometry. These methods train the neural networks to predict the signed distance for every position in the space by signed distances from ground truth or inferred from the raw 3D point cloud. With the learned signed distance field, we can obtain the surface by running the marching cubes algorithm [1] to extract the zero level set.

Without signed distance ground truth, inferring the correct gradient and distance for each query point could be hard. Since the gradient of the neural network also indicates the direction in which the signed distance field changes, recent works [2], [3], [4], [5], [6], [7], [8] typically add constraints on the network gradient to learn a stable field. In terms of the rate at which the field is changing, the eikonal term [2], [3], [5], [9] is widely used to ensure the norm of the gradient to be one everywhere. For the gradient direction constraint, some methods [7], [10] use the direction from the query point to the nearest point on the surface as guidance. Leveraging the continuity of the neural network and the gradient constraint, all these methods could reconstruct discrete points. However, the continuity cannot guarantee that the prediction is correct everywhere. Therefore, reconstructed surfaces of previous methods usually contain noise and ignore geometry details when there are not enough points to guide the reconstruction, as shown in Fig. 1.

The above-mentioned issue arises from the fact that these methods overlook the geometric information within the neighborhood but only focus on adding constraints on individual points to optimize the network. To resolve this issue, we introduce the bilateral filter for implicit fields that reduces surface noise while preserving the high-frequency geometric characteristics of the shape. Our designed implicit filter takes into account both the position of point clouds and the gradient of learned implicit fields. Based on the assumption of all input points lying on the surface, we can filter noise points on the zero level set by minimizing the weighted projection distance to gradients of the neighbor input points. Moreover, by moving the input points along the gradient of the field to other level sets, we can easily extend the filter to the whole field. This helps constrain the signed distance field near the surface and achieve better consistency through different level sets.

Unsigned distance functions differ from signed distance functions in that they measure the absolute distance from a query point to the surface. Because unsigned distance functions are non-differentiable at the zero level set, they do not have a well-defined zero level set like signed distance functions do. As a result, filtering each level set based on the gradient estimated by

Received 11 February 2025; revised 16 August 2025; accepted 21 August 2025. Date of publication 26 August 2025; date of current version 3 December 2025. This work was supported by The Innovative Research Service for BIM Model Inspection in Xiong'an under Grant 20232001544, directed by Jisheng Ma, Xiongan Xiongchuang Digital Technology CO., LTE. Recommended for acceptance by K. Yoon. *(Corresponding author: Ge Gao.)*

Shengtao Li, Yudong Liu, Ge Gao, and Ming Gu are with the Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100084, China, and also with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: list21@mails.tsinghua.edu.cn; liuyd23@mails.tsinghua.edu.cn; gaoge@tsinghua.edu.cn; guming@tsinghua.edu.cn).

Yu-Shen Liu is with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: liuyushen@tsinghua.edu.cn).

Project page: https://list17.github.io/ImplicitFilter.

This article has supplementary downloadable material available at https://doi.org/10.1109/TPAMI.2025.3602830, provided by the authors.

Digital Object Identifier 10.1109/TPAMI.2025.3602830

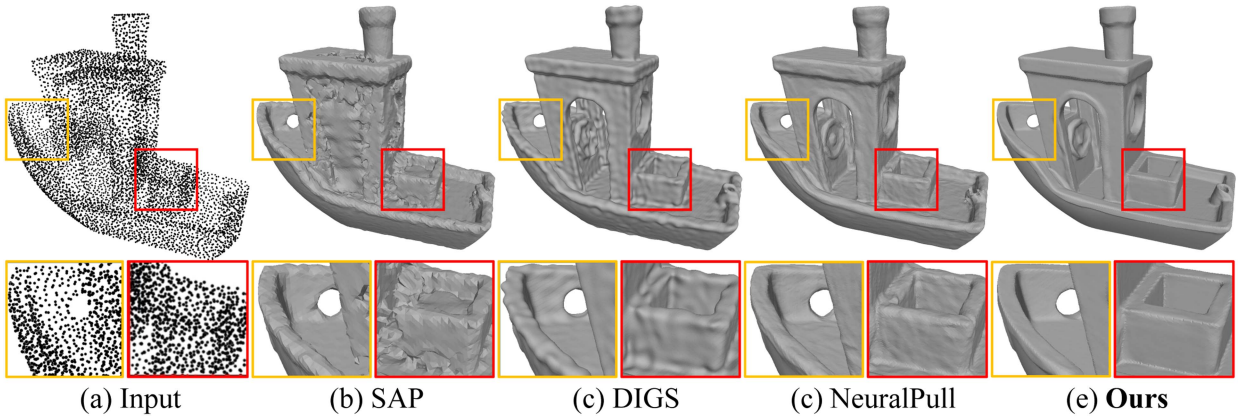


Fig. 1. Visualization of the comparisons on FAMOUS dataset [11]. Our implicit filter can improve the reconstruction by removing the noise and keeping the geometric details compared with other methods, including SAP [12], DIGS [9], and NeuralPull [7].

the neural network becomes challenging. To address this issue, we propose a gradient immutable training schema for unsigned distance functions. A new level set alignment loss function is used to constrain the projected surface points on the same level set as the query point.

Furthermore, we explore the implicit filter in the multi-view reconstruction. By leveraging our UDF filter, we learn a better UDF prior from the points estimated from Structure from Motion (SfM) methods for the sparse-view reconstruction.

We originally present our implicit filter on the signed distance function at ECCV 2024 [13], and then extend our method to the more general unsigned distance function and multi-view reconstruction. We also explore the performance of the filtering in point normal estimation and point cloud upsampling applications. To evaluate the effectiveness of our proposed implicit filtering, we validate it under several widely used benchmarks, including object and scene reconstructions from points, sparse-view reconstruction, point normal estimation, and point cloud upsampling. Our contributions are listed below.

- We introduce the implicit filter on SDFs to smooth the surface while preserving geometry details for learning better neural networks to represent shapes or scenes.
- We improve the implicit filter by extending it to non-zero level sets of signed distance fields. This regularization of the field aligns different level sets and provides better consistency within the whole SDF field.
- We propose a new gradient immutable training schema for extending the filter to the UDFs which allows for the reconstruction of arbitrary shapes.
- We extend the filter to the implicit field learned from multi-view images by learning a better UDF prior.
- We conduct experiments on surface reconstruction from points and sparse-view images, point normal estimation, and point cloud upsampling. These experiments validate our implicit filter, demonstrating its effectiveness and ability to produce high-fidelity reconstruction results.

II. Related Work

With the rapid development of deep learning, neural networks have shown great potential in surface reconstruction from 3D point clouds. In the following, we briefly review methods related to implicit learning for 3D shapes and reconstructions from point clouds.

A. Implicit Learning From 3D Supervision

The most commonly used strategy to train the neural network is to learn priors in a data-driven manner. These methods require signed distances or occupancy labels as 3D supervision to learn global priors [11], [14], [15], [16], [17] or local priors [18], [19], [20], [21], [22], [23], [24], [25], [26]. With large-scale training datasets, the neural network can perform well with similar shapes, but may not generalize well to unseen cases with large geometric variations. These models often have limited inputs that can be difficult to scale for varying sizes of point clouds.

B. Learning Signed Distance Functions

Different from the supervised methods, we can learn signed distance functions by overfitting neural networks on single point clouds or multi-view images to learn SDFs [2], [3], [7], [10], [12], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36]. These unsupervised methods rely on neural networks to infer implicit functions without learning any priors. Therefore, apart from the guidance of original input point clouds, we also need constraints on the direction [7], [10], [27], [28] or the norm [2], [3], [30] of the gradients, specially designed priors [28], [29], or differentiable poisson solver [12] to infer SDFs. This unsupervised approach heavily depends on the fitting capability and continuity of neural networks. However, these SDFs lack accuracy because there is no reliable guidance available for each query point across the entire space when working with discrete point clouds. Therefore, deducing the correct geometry for free space becomes particularly crucial. Our implicit filtering enhances SDFs by inferring the geometric details through the implicit field information of neighbor points.

C. Learning Unsigned Distance Functions

The unsigned distance field is a representation capable of modeling arbitrary shapes, including open and multi-layer

surfaces. This capability is first demonstrated by NDF [37], which learns unsigned distance functions to accurately model complex shapes by predicting dense point clouds and subsequently reconstructing meshes using other reconstruction techniques. Subsequent methods have employed specialized representations to enhance the learning effect of UDFs, such as defining null areas [38], exploring the relationship between two points [39], or some other representations [40], [41]. Recent advancements have focused on directly learning UDFs by overfitting neural networks on singular point clouds [42], [43], [44], [45], [46]. Despite these advances, the inherent properties of UDFs make them more challenging to learn accurately compared to signed distance fields (SDFs). The unstable gradient and fuzzy zero level set of UDFs make it hard to apply the implicit filtering. Therefore, we propose a novel gradient immutable training schema aimed at extending implicit filtering to the learning of unsigned distance functions, enhancing the robustness and precision of UDF modeling.

D. Feature Preserving Point Cloud Reconstruction

Early works [47], [48], [49] reconstruct point clouds with sharp features usually by point cloud consolidation. The key idea of these methods is to enhance the quality of point clouds with sharp features. One popular category is the local projection operation (LOP) [50] and its variants [47], [48], [51], [52]. The projection operator provides a stable and easily generalizable method for point cloud filtering, which is also the foundation of our implicit filter. The difference lies in that we do not need any normal or other priors and our filtering can be directly applied to implicit fields to extract high-fidelity meshes. Some other learning-based methods [53], [54] try to consolidate point clouds with edge points in a data-driven manner. Although capable of generating high-quality point clouds, these methods still require a proper reconstruction method [55], [56] to inherit the details in meshes.

With the advancement of deep learning in point cloud reconstruction, some approaches [5], [9], [57], [58] also explored employing neural networks to reconstruct high-precision models. FFN [59], SIREN [5], and IDF [60] introduce high-frequency features into the neural network in different ways to preserve the geometric details of the reconstructed shape. DIGS [9] and EPI [57] smooth the surface by using the divergence as guidance to alleviate the implicit surface roughness. Compared with these methods, we first introduce local geometric features through filtering to optimize the implicit field, so that we can achieve higher accuracy.

III. Overview of Implicit Filtering.

Motivation: Learning a continuous implicit field representation from a point cloud consisting of finite points does not have a unique solution. The only guidance available comes from the positions of input points. As a result, the distance field inferred from the point cloud usually contains noise. To address this issue, we draw inspiration from edge-preserving bilateral filtering, a technique used in image processing. We propose leveraging neighboring input points and their gradients to reduce the field noise while preserving geometric features at the edges. The main distinction lies in using projected distance to measure the difference, rather than the absolute coordinates typically employed as pixel values in image processing.

Overview: Implicit representations applicable to our implicit filtering involve two types of distance fields: SDFs and UDFs. We begin by designing the vanilla implicit filter on SDFs in Section IV, which is composed of the zero level set filter and the field filter. In this section, we clarify the definition of SDFs and the principle behind implicit filtering. Additionally, we will provide a detailed description of how to address the gradient degradation problem and how to sample on the zero level set during the learning process. In Section V, we explain the difference between UDFs and SDFs and outline the process of transitioning the implicit filter to UDFs. This enhances our method's generality, allowing the reconstruction of arbitrary shapes. Subsequently, we extend the UDF filter to sparse-view reconstruction in Section VI. Furthermore, experiments of surface reconstruction from both points and multi-view images are carried out in Section VII which also covers the extension to point normal estimation and point cloud upsampling.

IV. Implicit Filter on SDFs

A. Neural SDFs Overview

This section will briefly describe the concepts we used in our implicit filtering. We focus on the SDF $f : \mathbb{R}^3 \to \mathbb{R}$ inferred from the point cloud $\boldsymbol{P} = \{\boldsymbol{p}_i | \boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^N$ without ground truth signed distances and normals. f predicts a signed distance $s \in \mathbb{R}$ for an arbitrary query point $\boldsymbol{q}$, as formulated by $s = f_\theta(\boldsymbol{q})$, where θ denotes the parameters of the neural network.

The level set $\mathcal{S}_d$ of SDF is defined as a set of continuous query points with the same signed distance d, formulated as $\mathcal{S}_d = \{\boldsymbol{q} | f_\theta(\boldsymbol{q}) = d\}$. The goal of our implicit filtering is to smooth each level set with geometry details. Then we can extract the zero level set as a mesh by running the marching cubes algorithm [1].

B. Level Set Bilateral Filtering

Filtering for 2D images replaces the intensity of each pixel with the weighted intensity values from nearby pixels. Different from images, the resolution of implicit fields is infinite and we need to find the neighborhood on each level set for filtering. By minimizing the following loss function,

$$L_{dist} = \frac{1}{N} \sum_{i=1}^{N} |f_\theta(\boldsymbol{p}_i)|, \tag{1}$$

we can approximate that all points in $\boldsymbol{P}$ are located on level set $\mathcal{S}_0$, which makes it feasible to find neighbor points on $\mathcal{S}_0$. For a given point $\bar{\boldsymbol{p}}$ on $\mathcal{S}_0$, one simple strategy of filtering is to average positions of neighbor points $\mathcal{N}(\bar{\boldsymbol{p}}, \mathcal{S}_0) \subset \boldsymbol{P}$ on $\mathcal{S}_0$ by a Gaussian filter based on relative positions as follows:

$$\bar{\boldsymbol{p}}_{\text{average}} = \frac{\sum_{\boldsymbol{p}_j \in \mathcal{N}(\bar{\boldsymbol{p}}, \mathcal{S}_0)} \boldsymbol{p}_j \phi(||\bar{\boldsymbol{p}} - \boldsymbol{p}_j||)}{\sum_{\boldsymbol{p}_j \in \mathcal{N}(\bar{\boldsymbol{p}}, \mathcal{S}_0)} \phi(||\bar{\boldsymbol{p}} - \boldsymbol{p}_j||)}, \tag{2}$$

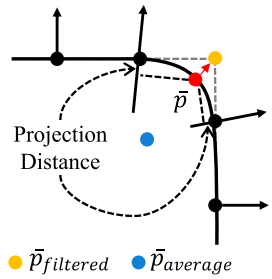


Fig. 2. By minimizing the weighted projection distance, our filter can preserve the sharp feature but the average method leads to a wrong result.

where the Gaussian function ϕ is defined as $\phi(||\bar{\boldsymbol{p}}-\boldsymbol{p}_j||) = \exp(-\frac{||\bar{\boldsymbol{p}}-\boldsymbol{p}_j||^2}{\sigma_p^2})$ and σ_p is a hyperparameter. However, as depicted in Fig. 2, it is evident that this weighted mean position yields excessively smooth surfaces, causing sharp features and details to be further obscured. To keep the geometric details, our filtering operator suggests measuring the projection distance to the gradient of neighbor points as shown in Figs. 2 and 3(b). When calculating weights, it is vital to account for both the impact of relative positions and the gradient similarity. Following the principles of bilateral filtering, to compute the filtered point for $\bar{\boldsymbol{p}}$, we simply need to minimize the following distance equation:

$$d(\bar{\boldsymbol{p}},\mathcal{S}) = \frac{\sum_{\boldsymbol{p}_j\in\mathcal{N}(\bar{\boldsymbol{p}},\mathcal{S})} |\boldsymbol{n}_{p_j}^T(\bar{\boldsymbol{p}}-\boldsymbol{p}_j)|\phi(||\bar{\boldsymbol{p}}-\boldsymbol{p}_j||)\psi(\boldsymbol{n}_{\bar{p}},\boldsymbol{n}_{p_j})}{\sum_{\boldsymbol{p}_j\in\mathcal{N}(\bar{\boldsymbol{p}},\mathcal{S})}\phi(||\bar{\boldsymbol{p}}-\boldsymbol{p}_j||)\psi(\boldsymbol{n}_{\bar{p}},\boldsymbol{n}_{p_j})}, \tag{3}$$

where the gradient $\boldsymbol{n}_{\bar{p}}$, $\boldsymbol{n}_{p_j}$ and the Gaussian function ψ are defined as $\boldsymbol{n}_{\bar{p}} = \frac{\nabla f_\theta(\bar{\boldsymbol{p}})}{||\nabla f_\theta(\bar{\boldsymbol{p}})||}, \boldsymbol{n}_{\boldsymbol{p}_j} = \frac{\nabla f_\theta(\boldsymbol{p}_j)}{||\nabla f_\theta(\boldsymbol{p}_j)||}, \psi(\boldsymbol{n}_{\bar{p}},\boldsymbol{n}_{p_j}) = \exp(-\frac{1-\boldsymbol{n}_{\bar{p}}^T\boldsymbol{n}_{p_j}}{1-\cos(\sigma_n)})$. The σ_n is the hyperparameter for ψ. The $\bar{\boldsymbol{p}}$ is situated on the level set $\mathcal{S}$, and for the mesh surface, it denotes the zero level set $\mathcal{S}_0$.

In addition to projection to the gradient $\boldsymbol{n}_{p_j}$, we observe that the projection distance to $\boldsymbol{n}_{\bar{p}}$ can assist in learning a more stable gradient for point $\bar{p}$ which is also adopted in EAR [47]. Taking into account the bidirectional projection, our final bilateral filtering operator can be formulated as follows:

$$d_{bi}(\bar{\boldsymbol{p}},\mathcal{S}) = \frac{\sum_{\boldsymbol{p}_j\in\mathcal{N}(\bar{\boldsymbol{p}},\mathcal{S})} (\boldsymbol{d_1}+\boldsymbol{d_2})\,\phi(||\bar{\boldsymbol{p}}-\boldsymbol{p}_j||)\psi(\boldsymbol{n}_{\bar{p}},\boldsymbol{n}_{p_j})}{\sum_{\boldsymbol{p}_j\in\mathcal{N}(\bar{\boldsymbol{p}},\mathcal{S})}\phi(||\bar{\boldsymbol{p}}-\boldsymbol{p}_j||)\psi(\boldsymbol{n}_{\bar{p}},\boldsymbol{n}_{p_j})}, \tag{4}$$

where $\boldsymbol{d_1} = |\boldsymbol{n}_{p_j}^T(\bar{\boldsymbol{p}}-\boldsymbol{p}_j)|, \boldsymbol{d_2} = |\boldsymbol{n}_{\bar{p}}^T(\bar{\boldsymbol{p}}-\boldsymbol{p}_j)|$. Although similar filtering methods have been widely studied in applications such as point cloud denoising and resampling [47], [54], there are two critical problems when applying these methods in implicit fields:

1) Filtering the zero level set needs to sample points on the level set $\mathcal{S}_0$, which necessitates the resolution of the equation $f_\theta = 0$, or the utilization of the marching cubes algorithm [1]. Both methods pose challenges in achieving fast and uniform point sampling. For the randomly sampled point $\boldsymbol{q}$ on non-zero level set $\mathcal{S}_{f_\theta(\boldsymbol{q})}$, we can also not filter this level set since there are no neighboring points on $\mathcal{S}_{f_\theta(\boldsymbol{q})}$.
2) The normals utilized in our filtering are derived from the gradients of the neural network f_θ. While the network typically offers reliable gradients, we may find that $\nabla f_\theta = 0$ is also the optimal solution to the minimum value of (3) and (4). This degenerate solution is unexpected, as it implies a scenario where there is no surface when the gradient is zero everywhere.

We will focus on addressing the two issues in the subsequent sections.

C. Sampling Points for Filtering

Inspired by NeuralPull [7], we can pull a query point to the zero level set by the gradient of the neural network f_θ. For a given query point $\boldsymbol{q}$ as input, the pulled location $\hat{\boldsymbol{q}}$ can be formulated as follows:

$$\hat{\boldsymbol{q}} = \boldsymbol{q} - f_\theta(\boldsymbol{q})\nabla f_\theta(\boldsymbol{q})/||\nabla f_\theta(\boldsymbol{q})||. \tag{5}$$

The points $\boldsymbol{q}$ and $\hat{\boldsymbol{q}}$ lie respectively on level sets $\mathcal{S}_{f_\theta(q)}$ and $\mathcal{S}_0$ as illustrated in Fig. 4(b). By adopting the sampling strategy in NeuralPull, we can generate samples $\boldsymbol{Q} = \{\boldsymbol{q}_i|\boldsymbol{q}_i\in\mathbb{R}^3\}_{i=1}^M$ on different level sets near the surface and pull them to $\mathcal{S}_0$ by (5), to obtain $\hat{\boldsymbol{Q}} = \{\hat{\boldsymbol{q}}_i|\hat{\boldsymbol{q}}_i = \boldsymbol{q}_i - f_\theta(\boldsymbol{q}_i)\nabla f_\theta(\boldsymbol{q}_i)/||\nabla f_\theta(\boldsymbol{q}_i)||, \boldsymbol{q}_i\in\boldsymbol{Q}\}_{i=1}^M$. Hence, we can filter the zero level set by minimizing (4) across all pulled query points $\hat{\boldsymbol{Q}}$, which is equivalent to optimizing the following loss:

$$L_{zero} = \frac{1}{M}\sum_{\hat{\boldsymbol{q}}\in\hat{\boldsymbol{Q}}} d_{bi}(\hat{\boldsymbol{q}},\mathcal{S}_0), \tag{6}$$

where for each $\hat{\boldsymbol{q}}\in\hat{\boldsymbol{Q}}$, $\mathcal{N}(\hat{\boldsymbol{q}},\mathcal{S}_0)$ denotes finding the neighbors of $\hat{\boldsymbol{q}}$ within the input points $\boldsymbol{P}$, since $\boldsymbol{P}$ is assumed to be located on $\mathcal{S}_0$.

This filtering mechanism can be easily extended to non-zero level sets in a similar inverse manner. To be more specific, as for level set $S_{f_\theta(\boldsymbol{q})}$, the neighbor points for query point $\boldsymbol{q}\in\boldsymbol{Q}$ are required. These points should lie on the level set $S_{f_\theta(\boldsymbol{q})}$, the same as $\boldsymbol{q}$, allowing us to filter the level set $S_{f_\theta(\boldsymbol{q})}$ using the same filter as described in (4).

However, obtaining $\mathcal{N}(\boldsymbol{q},S_{f_\theta(q)})$ in $\boldsymbol{P}$ is not feasible, since all input points $\boldsymbol{P}$ are situated on the zero level set instead of the $S_{f_\theta(\boldsymbol{q})}$ level set. To address this issue, we propose a technique for identifying neighbors of $\boldsymbol{q}$ on level set $S_{f_\theta(\boldsymbol{q})}$, by projecting the input points $\boldsymbol{P}$ inversely onto the specific level set $\mathcal{S}_{f_\theta(\boldsymbol{q})}$ based on the gradient, as depicted in Fig. 4(b). The projected neighbor points can be represented as in (7). Filtering across multiple level sets helps to enhance the performance of our method by optimizing the consistency between different level sets within the SDF field. We further showcase this evidence in the ablation study detailed in Section VII-I.

$$\mathcal{N}(\boldsymbol{q},\mathcal{S}_{f_\theta(q)}) = \left\{\boldsymbol{p} + f_\theta(\boldsymbol{q})\frac{\nabla f_\theta(\boldsymbol{p})}{||\nabla f_\theta(\boldsymbol{p})||}, \boldsymbol{p}\in\mathcal{N}(\hat{\boldsymbol{q}},\mathcal{S}_0))\right\}. \tag{7}$$

Based on the above analysis, we can filter the level sets $S_{f_\theta(\boldsymbol{q})}$ by minimizing (4) over all sample points $\boldsymbol{Q}$ through (7),

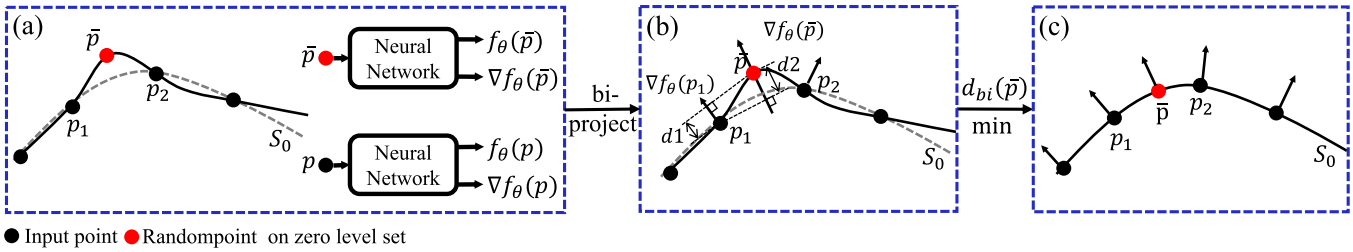


Fig. 3. Overview of filtering the zero level set. (a) We assume all input points lying on the surface and compute gradients as normals. (b) Calculating bidirectional projection distances $d1 = |\boldsymbol{n}_{p_j}^T(\bar{\boldsymbol{p}} - \boldsymbol{p}_j)|$, $d2 = |\boldsymbol{n}_{\bar{p}}^T(\bar{\boldsymbol{p}} - \boldsymbol{p}_j)|$ and the weights in (4). (c) By minimizing (4), we can remove the noise on the zero level set. The gradient ∇f_θ in this figure defaults to be regularized.

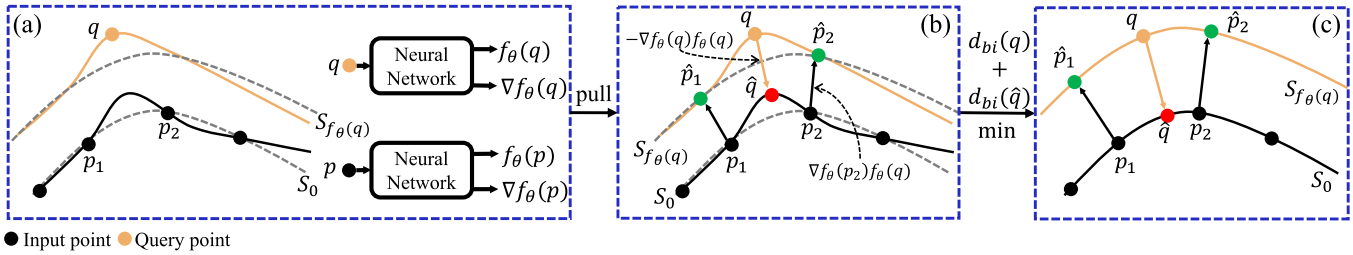


Fig. 4. Overview of sampling points. (a) Sampling query points near the surface. (b) Pulling the query point to the zero level set and input points to the level set where the query point is located. (c) Applying the filter on each level set. The gradient ∇f_θ in this figure defaults to being regularized.

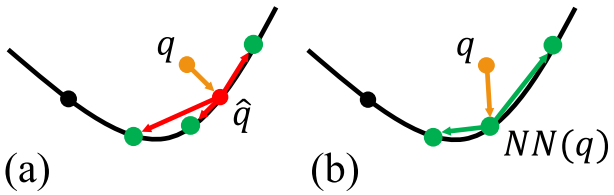


Fig. 5. (a) Searching neighbors directly for $\hat{\boldsymbol{q}}$. (b) Searching neighbors for $NN(\boldsymbol{q})$ instead of $\hat{\boldsymbol{q}}$.

equivalent to optimizing the following loss:

$$L_{field} = \frac{1}{M}\sum_{\boldsymbol{q}\in\boldsymbol{Q}} d_{bi}(\boldsymbol{q}, \mathcal{S}_{f_\theta(q)}). \tag{8}$$

It is worth noting that for a fixed query point $\boldsymbol{q}$, the pulled query point $\hat{\boldsymbol{q}}$ dynamically changes when training the neural network, which results in a time-consuming process to repeatedly searching for neighbors of $\hat{\boldsymbol{q}}$. To handle this matter, we substitute the $\mathcal{N}(\hat{\boldsymbol{q}}, \mathcal{S}_0)$ by $\mathcal{N}(NN(\boldsymbol{q}), \mathcal{S}_0)$, where $NN(\boldsymbol{q})$ denotes the nearest point of $\boldsymbol{q}$ within the point cloud $\boldsymbol{P}$ as shown in Fig. 5. While this substitution may introduce a slight bias during training, it also ensures the neighbor points are close to $\hat{\boldsymbol{q}}$, therefore this trade-off between efficiency and accuracy is reasonable.

D. Gradient Constraint

The other problem of implicit filtering is gradient degeneration. Overfitting the neural network requires the SDF to be geometrically initialized. We can consider the initialized implicit field as the noisy field and apply our filter directly to train the network from the beginning to fit the raw point cloud by removing the 'noise'. However, if the denoise target is too complex, gradient degeneration will occur during the training process. Therefore, we need to add a constraint to the gradient of the SDF.

There are two ways for training the neural network to pull query points onto the surface based on NeuralPull [7] and CAP-UDF [42]. One is minimizing the distance between the pulled point $\hat{\boldsymbol{q}}$ and the nearest point $NN(\boldsymbol{q})$ as formulated below:

$$L_{pull} = \frac{1}{M}\sum_{i\in[1,M]} ||\hat{\boldsymbol{q}}_i - NN(\boldsymbol{q}_i)||_2. \tag{9}$$

The other is minimizing the Chamfer distance between moved query points and the raw point cloud:

$$L_{CD} = \frac{1}{M}\sum_{i\in[1,M]} \min_{j\in[1,N]} ||\hat{\boldsymbol{q}}_i - \boldsymbol{p}_j||_2 + \frac{1}{N}\sum_{j\in[1,N]} \min_{i\in[1,M]} ||\boldsymbol{p}_j - \hat{\boldsymbol{q}}_i||_2. \tag{10}$$

A stable SDF can be trained by the losses above since they are trying to move the query points to be in the same distribution as the point cloud, which can provide the constraint for our implicit filter. Here we choose L_{CD} since the filtered points are likely not the nearest points and L_{CD} is a more relaxed constraint.

E. Loss Function

Finally, our loss function is formulated as:

$$L = L_{zero} + \alpha_1 L_{field} + \alpha_2 L_{dist} + \alpha_3 L_{CD}, \tag{11}$$

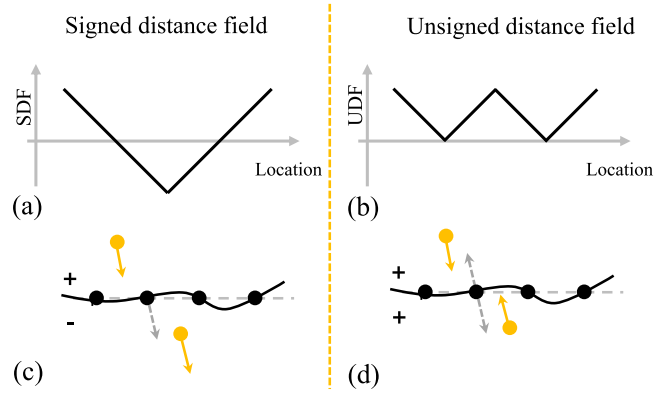


Fig. 6. The illustration of the difference between the signed distance field and the unsigned distance field is shown in (a) and (b). (c) and (d) show the difference between the learned zero level set and the gradients of the query and input points. The gradients of input points in (d) represent the possible directions.

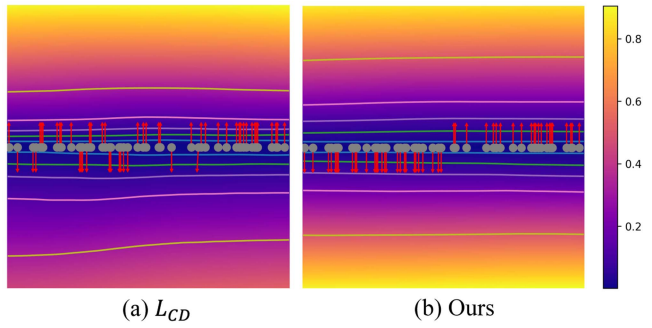


Fig. 7. The 2D level sets show the unsigned distance field and the normals of input points learned by L_{CD} and ours.

where α_1, α_2, and α_3 are the balance weights for our implicit filtering loss.

V. Implicit Filter on UDFs

A. Neural UDFs Overview

This section will briefly explain the basic concept of UDFs and the difference between SDFs and UDFs, which will serve as the foundation for migrating the filter to UDF learning. For a UDF $f^U : \mathbb{R}^3 \to \mathbb{R}$ inferred from the point cloud P, the unsigned distance $u \in \mathbb{R}$ is predicted for arbitrary query point $\boldsymbol{q}$, as formulated by $u = f_\theta^U(\boldsymbol{q})$, where θ denotes the parameters of the neural network. The level set $\mathcal{U}_d$ of a UDF can be formulated as $\mathcal{U}_d = \{\boldsymbol{q} | f_\theta^U(\boldsymbol{q}) = d, d >= 0\}$.

The difference between the signed distance field and the unsigned distance field is shown in Fig. 6(a) and (b). UDFs are not differentiable at zero level set compared to SDFs. Therefore, the gradients learned by UDFs are unoriented depicted in Fig. 6(d), which presents two potential scenarios for UDF learning. Moreover, the unsigned distance field's exclusive definition in the positive domain induces omnidirectional orientation of gradient directions at input points, particularly when such directional flexibility aligns with filtering objectives. This inherent instability may induce substantial perturbations in both the learned surface geometry and the gradients of input points.

Based on the above difference analysis, to migrate the filtering to UDF learning, two problems should be solved: unoriented gradients on the zero level set and unstable normals for the input point cloud when filtering the unsigned distance field. In the following sections, we will focus on these two problems.

B. Gradient Unoriented Bilateral Filtering

Due to the non-differentiable nature of UDF on the zero level set, the gradient direction of the input point extracted from the UDF is unoriented as shown in Fig. 7. The most intuitive effect of unoriented gradients is that it primarily impacts the weight calculation in the zero level set filtering, represented by the Gaussian function ψ in (4). Regarding the computation of the absolute value of the projection distance, the impact of unoriented gradients is negligible. In filtering of the other level sets, the unoriented gradients result in the opposite position of projected input points, as described in (7). While the projected points are, by definition, situated on the same level set as the query point, this may result in inaccuracies in determining the neighbors of the query point.

Confronted with these two challenges, we employ the same strategy by considering the gradient of the input points as bidirectional. For the function ψ, we choose the same direction as the query point $\boldsymbol{q}$. The modified function can be formulated as:

$$\psi^U(\boldsymbol{n}_{\bar{p}}, \boldsymbol{n}_{p_j}) = \exp\left(-\frac{1 - |\boldsymbol{n}_{\bar{p}}^T \boldsymbol{n}_{p_j}|}{1 - \cos(\sigma_n)}\right). \tag{12}$$

Given that the filtering point is proximal to the desired zero level set, the formula generally demonstrates efficacy across various scenarios, with exceptions noted in particular cases such as exceedingly thin plates.

To obtain the projected neighbor points, we project the neighbor points along the bidirectional gradients and then find the nearest points in these projected points. Since there is no issue with unoriented gradients on other level sets, although the distant neighbor points identified in this process may not lie on the same side of the zero level set as the query point, the gradients of these neighboring points are opposite to that of the query point and are located far away. As a result, the calculated weights for these points are also small and do not significantly impact the final outcome.

C. Gradient Immutable Training Schema

Another issue is the unstable normals in the input point cloud. To filter the zero level set, we need to minimize the projected distance $\boldsymbol{d}$, as shown in Fig. 8(a). However, in UDF learning, we can achieve this goal by either moving the position of $\hat{\boldsymbol{p}}$ or changing the gradient of $\boldsymbol{p_1}$. This flexibility arises because there are no negative values on one side of the surface to constrain the approximate direction of the gradient field, unlike in SDFs. We propose a gradient immutable training schema for the UDF implicit filtering, as illustrated in Fig. 8(b). The process begins by training the neural network using L_{CD} or other methods. Once training is complete, we copy the parameters from the trained

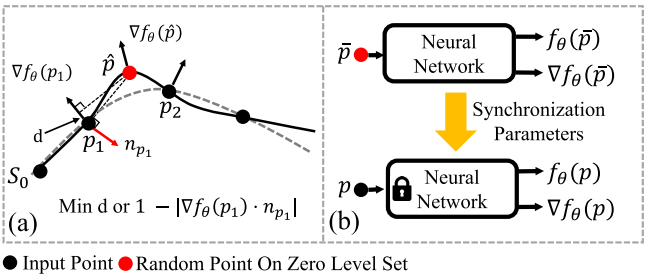


Fig. 8. Gradient immutable training schema.

neural network to a locked version of the network. Next, we extract the gradients of the input points from this locked, trained network. Additionally, we synchronize the parameters with the locked network after a predetermined number of training epochs. This approach not only stabilizes the extracted gradients but also allows for their subsequent updates.

D. Loss Function

In the SDF filtering loss function, we use L_{dist} to constrain all the points located on the zero level set. Although we can still leverage this function, the zero level set of the learned UDF is fuzzy. This will also impact the projection of the neighboring point to other level sets. Therefore, we propose a similar loss function:

$$L_{proj} = \frac{1}{M}\frac{1}{K}\sum_{\boldsymbol{q}\in\boldsymbol{Q}}\sum_{\hat{\boldsymbol{p}}\in\mathcal{N}(\boldsymbol{q},\mathcal{S}_{f_\theta(q)})}(f(\boldsymbol{q}) - f(\hat{\boldsymbol{p}})), \tag{13}$$

where $K = |\mathcal{N}(\boldsymbol{q}, \mathcal{S}_{f_\theta(q)})|$, to constrain the projected neighbor point located on the same level set of $\boldsymbol{q}$. The final loss function for the UDF implicit filtering can be expressed as:

$$L^U = L_{zero} + \alpha_1 L_{proj} + \alpha_2 L_{field} + \alpha_3 L_{dist} + \alpha_4 L_{CD}, \tag{14}$$

where $\alpha_1, \alpha_2, \alpha_3$ and α_4 are the balance weights for our UDF implicit filtering loss.

VI. Filtering UDF Prior for Sparse-View Reconstruction

A. Overview

Apart from filtering the SDF and UDF learned from point clouds, we further extend our method to multi-view reconstruction. Typically a signed distance function $f : \mathbb{R}^3 \to \mathbb{R}$ and a radiance function $c : \mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}^3$ that encodes the color associated with the query point $\boldsymbol{q} \in \mathbb{R}^3$ and a viewing direction $\boldsymbol{v} \in \mathbb{S}^2$ are learned from images $\mathbf{I} = \{I_i\}_{i=1}^K$ with poses $\mathbf{T} = \{T_i\}_{i=1}^K$ of a 3D object. The surface of the 3D object is also represented as the zero level set $\mathcal{S}_0$ of the implicit field. Volume rendering is used to render novel images from these two functions and the training target is minimizing the difference between the rendered and the input images. This makes it hard to filter the implicit field since our implicit filtering mechanism utilizes the input points as geometric guidance to achieve high-fidelity surface reconstruction. Although we can estimate points by SfM, heavily depending on the points to filter the surface is inconsistent with reconstructing surfaces from images since the points cannot represent the geometric details in images perfectly.

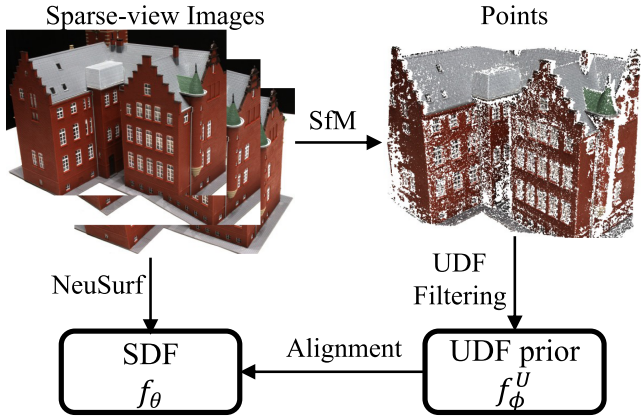


Fig. 9. UDF prior filtering for sparse-view reconstruction overview.

B. UDF Prior Filtering

While conventional multi-view reconstruction methods rely solely on image data, the NeuSurf introduced a UDF prior which is learned by the points estimated by SfM. The designed alignment loss function can be formulated as:

$$L_{align} = \frac{1}{|\boldsymbol{Q}|}\sum_{\boldsymbol{q}\in\boldsymbol{Q}}\big||f_\theta(\boldsymbol{q})| - f_\phi^U(\boldsymbol{q})\big|, \tag{15}$$

where $\boldsymbol{Q}$ represents the query points of the volume rendering, θ and ϕ represent the parameters of the SDF and the UDF prior network. It aligns $\mathcal{S}_0$ and $\mathcal{U}_0$ of these two fields. This learned prior enables better implicit field reconstruction under sparse-view conditions and also inspires us to optimize the multi-view reconstruction results by filtering the UDF prior.

Specifically, we propose to filter the UDF prior by leveraging SfM-estimated points for sparse-view reconstruction, as illustrated in Fig. 9. These sparse 3D points serve as geometric constraints during the filtering process, enabling the acquisition of an enhanced prior that better aligns with multi-view reconstruction objectives. Crucially, despite the sparsity of SfM points and inherent texture-image complexity surpassing point cloud resolution, our findings demonstrate that a regularized high-fidelity prior can still induce measurable improvements in surface reconstruction quality under specific conditions.

VII. Experiments

We conduct experiments to assess the performance of our implicit filter for surface reconstruction from raw point clouds. The results are presented for general watertight shapes in Section VII-A, real scanned raw data including 3D objects in Section VII-B, VII-D, and complex scenes in Section VII-E. We further explore the effect of our filter on UDF learning and sparse-view reconstruction. The results are presented in Section VII-C, VII-E, and VII-F. At the same time, since our implicit filter smooths the surface and keeps geometric details,

we find we can extend our method to unsupervised point normal estimation and point cloud upsampling. The normal estimation and the point cloud upsampling experiments are shown in Section VII-G and VII-H respectively. Additionally, ablation experiments are carried out to validate the theory and explore the impact of various parameters in Section VII-I.

Implementation details: We employ a neural network similar to OccNet [16] and the geometric network initialization proposed in SAL [2] with a smaller radius the same as GridPull [10] to learn the SDF. We use the strategy in NeuralPull [7] to sample queries around each point $\boldsymbol{p}$ in $\boldsymbol{P}$. We set the weight α_3 to 10 to constrain the learned SDF and α_1 and α_2 to 1. The parameters σ_n, σ_p are set to $15°, \max_{\boldsymbol{p}_j \in \mathcal{N}(\bar{\boldsymbol{p}}, \mathcal{S}_{f_\theta(\bar{\boldsymbol{p}})})}(||\bar{\boldsymbol{p}} - \boldsymbol{p}_j||)$ respectively. Instead of filtering the SDFs learned by other loss functions, we find our loss can directly learn the SDF from the raw point cloud. All the experiments are trained with our loss function from the beginning unless otherwise specified. In the experiment for scenes, we abort the L_{field} since there exist open surfaces that cause the gradient orientation of the input points to be inconsistent.

We utilize the Adam optimizer [61], setting an initial learning rate of 0.001 and employing a cosine learning rate schedule with 1 k warm-up iterations. For the shape reconstructions including ABC [62], FAMOUS [11], ShapeNet [63], and SRB [64] datasets, we train the neural network for 40 k iterations, and we use a resolution of 256^3 for the marching cubes algorithm [1]. For the 3D Scene dataset, we train the neural network for 100 k iterations to fully converge on each scene. Since there exist open surfaces, we use a resolution of 512^3 to extract the 0.001 level set as the reconstructed mesh. If not stated otherwise, all other methods of comparison will use consistent settings.

For the UDFs, we utilize the same network structure as the SDFs, but we incorporate a non-linear projection defined as $g(x) = |x|$ before the final output. This adjustment ensures that we can learn the unsigned distance field effectively. The parameter synchronization interval is set to 10 epochs. Unlike the SDFs, the UDFs do not utilize negative values to constrain the gradient. This limitation can lead to poor training results when starting from scratch. As a result, we opt to filter the existing neural network for our UDF experiments. We choose the LSUDF [43] as our baseline because it provides the more reliable normal estimation for the input points.

We set the learning rate for the UDF filtering to 0.0005 and employ a cosine learning rate schedule. For the shape reconstructions, we finetune the neural network for 20 k iterations, and we use the surface extraction algorithm proposed in CAPUDF [42]. The other settings are the same as SDF learning.

For the sparse-view reconstruction, we follow the setting the same as NeuSurf [65] and set the coefficient of the prior loss function to 0.5.

A. Surface Reconstruction for Shapes

Datasets and metrics: For surface reconstruction of general shapes from raw point clouds, we conduct evaluations on three widely used datasets including a subset of ShapeNet [63], ABC [62], and FAMOUS [11]. We use the same setting as NeuralPull [7] for the dataset ShapeNet. For datasets ABC and FAMOUS, we use the train/test splitting released by Points2Surf [11] and we sample points directly from the mesh in the ABC dataset without other mesh preprocessing to keep the sharp features.

TABLE I
COMPARISONS ON ABC AND FAMOUS DATASETS. THE THRESHOLD OF F-SCORE (F-S.) IS 0.01.

Methods	ABC			FAMOUS		
	CD_{L2}	CD_{L1}	F-S.	CD_{L2}	CD_{L1}	F-S.
P2S [11]	0.298	0.015	0.598	0.012	0.008	0.752
IGR [4]	2.675	0.063	0.448	1.474	0.044	0.573
NP [7]	0.095	0.011	0.673	0.100	0.012	0.746
PCP [28]	0.252	0.023	0.373	0.037	0.014	0.435
SIREN [5]	0.022	0.012	0.493	0.025	0.012	0.561
DIGS [9]	0.021	0.010	0.667	0.015	0.008	0.772
iPSR [68]	0.019	0.010	0.663	0.020	0.010	0.639
NCR [69]	0.024	0.010	0.643	0.030	0.011	0.667
NSH [70]	0.014	0.009	0.687	0.023	0.008	0.777
PGR [71]	0.011	0.009	0.676	0.012	0.008	0.767
Ours	**0.011**	**0.009**	**0.691**	**0.008**	**0.007**	**0.778**

TABLE II
COMPARISONS ON SHAPENET DATASET IN TERMS OF $CD_{L2} \times 100$

Class	SPSR [66]	NP [7]	LPI [67]	PCP [28]	GP [10]	Ours
Display	0.273	0.039	0.0080	0.0087	0.0082	**0.0009**
Lamp	0.227	0.080	0.0172	0.0380	0.0347	**0.0019**
Airplane	0.217	0.008	0.0060	0.0065	**0.0007**	0.0045
Cabinet	0.363	0.026	0.0179	0.0153	0.0112	**0.0055**
Vessel	0.254	0.022	0.0092	0.0079	0.0033	**0.0005**
Table	0.383	0.060	0.0436	0.0131	0.0052	**0.0025**
Chair	0.293	0.054	0.0187	0.0110	0.0043	**0.0070**
Sofa	0.276	0.012	0.0164	0.0086	**0.0015**	0.0027
Mean	0.286	0.038	0.0171	0.0136	0.0086	**0.0032**

For evaluating the performance, we follow NeuralPull to sample 1×10^5 points from the reconstructed surfaces and the ground truth meshes on the ShapeNet dataset and sample 1×10^4 on the ABC and FAMOUS datasets. For the evaluation metrics, we use L1 and L2 Chamfer distance (CD_{L1} and CD_{L2}) to measure the error. Moreover, we adopt normal consistency (NC) and F-score to evaluate the accuracy of the reconstructed surface, the threshold is the same as NeuralPull.

Comparisons: To evaluate the validity of our implicit filter, we compare our method with a variety of methods including SPSR [66], Points2Surf (P2S) [11], IGR [4], NeuralPull (NP) [7], LPI [67], PCP [28], GridPull (GP) [10], SIREN [5], DIGS [9], iPSR [68], NCR [69], NSH [70], PGR [71]. The quantitative results on ABC and FAMOUS datasets are shown in Table I, and selectively visualized in Fig. 10. Our model reaches state-of-the-art performance on both datasets, accomplishing the goal of eliminating noise on each level set while preserving the geometric details. To more intuitively validate the efficacy of our filtering, we visualize the level sets on a cross section in Fig. 11. We also report the results on ShapeNet which contains over 3000 objects in terms of CD_{L2} in Table II, NC in Table III, and F-Score with thresholds of 0.002 in Table IV and 0.004 in Table V. Our method outperforms previous methods over most classes. The visualization comparisons in Fig. 12 show that our method can reconstruct a smoother surface with fine details.

To validate the effect of our filter on sharp geometric features. We evaluate the edge points by the edge Chamfer distance metric

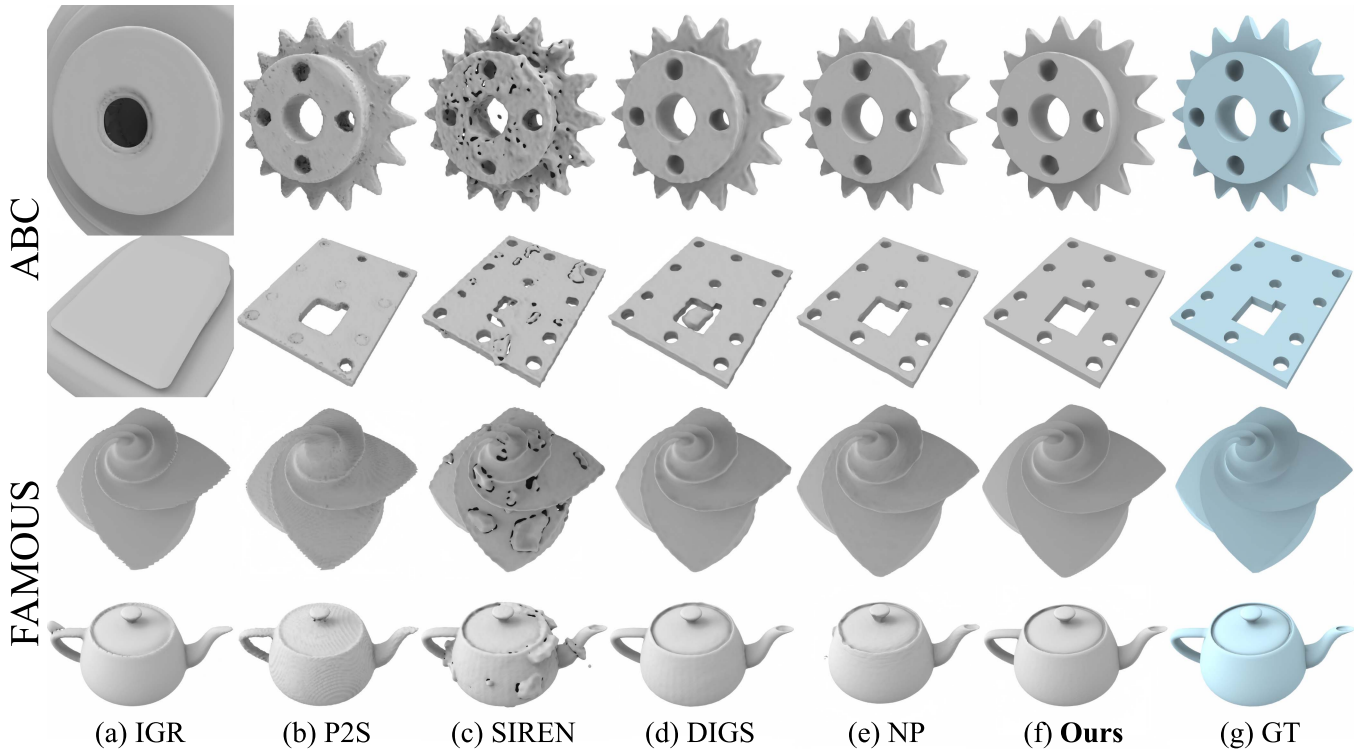


Fig. 10. Visual comparisons of surface reconstruction on ABC and FAMOUS datasets. Our method can reconstruct objects with sharp edges and less noise compared with other methods.

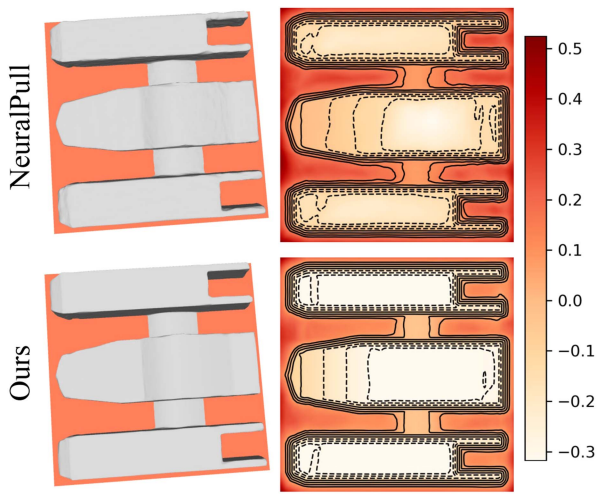


Fig. 11. Visualization of level sets on a cross section.

TABLE III
COMPARISONS ON SHAPENET DATASET IN TERMS OF NC

Class	SPSR [66]	NP [7]	LPI [67]	PCP [28]	GP [10]	Ours
Display	0.889	0.964	0.9780	0.9775	0.9847	**0.9880**
Lamp	0.876	0.930	0.9503	0.9450	**0.9693**	0.9692
Airplane	0.848	0.947	0.9560	0.9490	0.9614	**0.9800**
Cabinet	0.880	0.930	0.9576	0.9600	0.9689	**0.9711**
Vessel	0.861	0.941	0.9564	0.9546	0.9667	**0.9802**
Table	0.833	0.908	0.9527	0.9595	0.9755	**0.9780**
Chair	0.850	0.937	0.9545	0.9580	0.9733	**0.9740**
Sofa	0.892	0.951	0.9713	0.9680	0.9792	**0.9829**
Mean	0.866	0.939	0.9596	0.9590	0.9723	**0.9779**

TABLE IV
COMPARISONS ON SHAPENET DATASET IN TERMS OF F-SCORE WITH A THRESHOLD OF 0.002

Class	SPSR [66]	NP [7]	LPI [67]	PCP [28]	GP [10]	Ours
Display	0.468	0.989	0.9978	0.9939	0.9963	**0.9998**
Lamp	0.455	0.891	0.9889	0.9382	0.9455	**0.9986**
Airplane	0.415	0.996	**0.9989**	0.9942	0.9976	0.9959
Cabinet	0.392	0.980	0.9849	0.9888	0.9901	**0.9955**
Vessel	0.415	0.985	0.9955	0.9935	0.9956	**0.9998**
Table	0.233	0.922	0.9789	0.9969	0.9977	**0.9986**
Chair	0.382	0.954	0.98977	0.9970	**0.9979**	0.9952
Sofa	0.499	0.968	0.9946	0.9943	**0.9974**	0.9973
Mean	0.407	0.961	0.9912	0.9871	0.9896	**0.9976**

TABLE V
COMPARISONS ON SHAPENET DATASET IN TERMS OF F-SCORE WITH A THRESHOLD OF 0.004

Class	SPSR [66]	NP [7]	LPI [67]	PCP [28]	GP [10]	Ours
Display	0.666	0.991	0.9993	0.9958	0.9963	**0.9999**
Lamp	0.648	0.924	0.9954	0.9402	0.9538	**0.9994**
Airplane	0.619	0.997	**0.9998**	0.9972	0.9989	0.9971
Cabinet	0.598	0.989	0.9938	0.9939	0.9946	**0.9968**
Vessel	0.633	0.99	0.9985	0.9958	0.9972	**0.9999**
Table	0.442	0.973	0.9866	0.9985	0.9990	**0.9995**
Chair	0.617	0.969	0.994	0.9991	**0.9990**	0.9966
Sofa	0.725	0.974	0.9982	0.9987	0.9992	**0.9987**
Mean	0.618	0.976	0.9957	0.9899	0.9923	**0.9985**

used in [72]. We sample 100 k points uniformly on the surface of both the reconstructed mesh and the ground truth. The edge point $\boldsymbol{p}$ is calculated by finding whether there exists a point $\boldsymbol{q} \in \mathcal{N}_\epsilon(\boldsymbol{p})$ satisfied $|\boldsymbol{n}_q \boldsymbol{n}_p| < \sigma$, where $\mathcal{N}_\epsilon(\boldsymbol{p})$ represents the neighbor points within distance ϵ from $\boldsymbol{p}$. The results are shown in Table VI and visualized in Fig. 13. We set $\epsilon = 0.01$ and

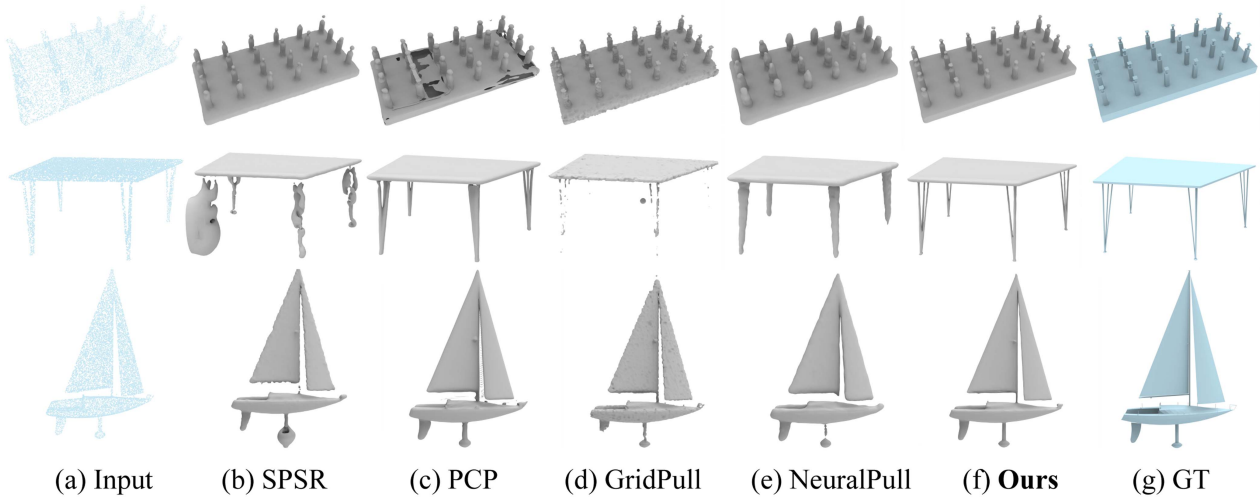


Fig. 12. Visual comparisons of surface reconstruction on ShapeNet dataset.

TABLE VI
EDGE CHAMFER DISTANCE COMPARISONS ON ABC DATASET, $ECD_{L2} \times 100$

Methods	P2S [11]	IGR [4]	NP [7]	PCP [28]	SIREN [5]	DIGS [9]	Ours
ECD_{L1}	0.0496	0.0835	0.0501	0.0628	0.0695	0.0786	**0.0256**
ECD_{L2}	1.055	2.365	1.255	1.265	1.407	2.493	**0.399**

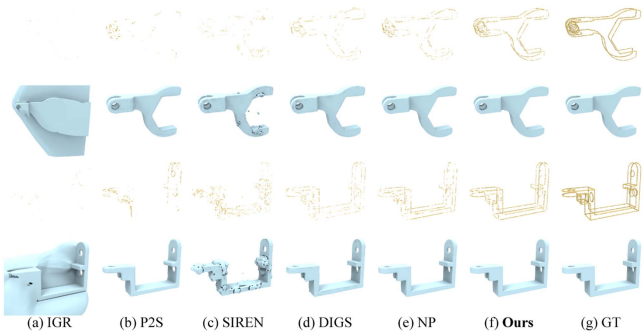


Fig. 13. Visual comparisons of edge points and reconstruction results.

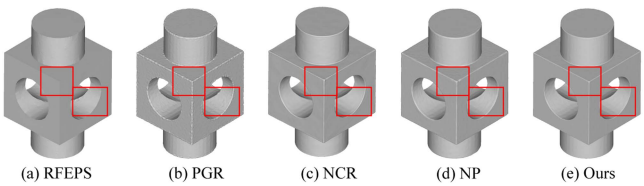


Fig. 14. Visual comparisons of shape feature reconstruction results.

$\sigma = 0.1$. We also compare the sharp reconstruction effect with feature preserving methods, including RFEPS [73], NCR [69], and PGR [71]. NP [7] is also used to show the effect without our filtering. The visualization comparisons in Fig. 14 show that our sharp reconstruction effect may not match the performance of RFEPS [73] due to the different reconstruction methods, but our method outperforms other implicit methods in maintaining the sharp features of the shape. Furthermore, our method excels in various other aspects, demonstrating its overall superiority.

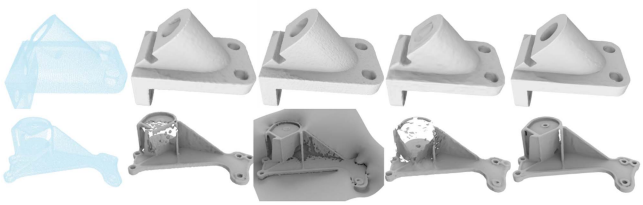
TABLE VII
COMPARISONS ON SRB DATASET

		SPSR [66]	IGR [4]	SIREN [5]	VisCo [74]	SAP [12]	NP [7]	GP [10]	DIGS [9]	Ours
Anchor	CD_{L1}	0.60	0.22	0.32	0.21	0.12	0.122	0.093	0.063	**0.052**
	HD	14.89	4.71	8.19	3.00	2.38	3.243	1.804	1.447	**1.232**
	$d_{\overrightarrow{C}}$	0.60	0.12	0.10	0.15	0.08	0.061	0.066	0.030	**0.025**
	$d_{\overrightarrow{H}}$	14.89	1.32	2.432	1.07	0.83	3.208	0.460	0.270	**0.265**
Daratech	CD_{L1}	0.44	0.25	0.21	0.21	0.26	0.375	0.062	0.049	**0.051**
	HD	7.24	4.01	4.30	4.06	0.87	3.127	**0.648**	0.858	0.751
	$d_{\overrightarrow{C}}$	0.44	0.08	0.09	0.14	0.04	0.746	0.039	0.025	**0.028**
	$d_{\overrightarrow{H}}$	7.24	1.59	1.77	1.76	0.41	3.267	**0.293**	0.441	0.423
DC	CD_{L1}	0.27	0.17	0.15	0.15	0.07	0.157	0.066	0.042	**0.041**
	HD	3.10	2.22	2.18	2.22	1.17	3.541	1.103	**0.667**	0.815
	$d_{\overrightarrow{C}}$	0.27	0.09	0.06	0.09	0.04	0.242	0.036	0.022	**0.019**
	$d_{\overrightarrow{H}}$	3.10	2.61	2.76	2.76	**0.53**	3.523	0.539	0.729	**0.724**
Gargoyle	CD_{L1}	0.26	0.16	0.17	0.17	0.07	0.080	0.063	0.047	**0.044**
	HD	6.80	3.52	4.64	4.40	1.49	1.376	1.129	**0.971**	1.089
	$d_{\overrightarrow{C}}$	0.26	0.06	0.08	0.11	0.05	0.063	0.045	0.028	**0.022**
	$d_{\overrightarrow{H}}$	6.80	0.81	0.91	0.96	0.78	0.475	0.700	0.271	**0.246**
Lord Quas	CD_{L1}	0.20	0.12	0.17	0.12	0.05	0.064	0.047	0.031	**0.030**
	HD	4.61	1.17	0.82	1.06	0.98	0.822	0.569	**0.496**	0.554
	$d_{\overrightarrow{C}}$	0.20	0.07	0.12	0.07	0.04	0.053	0.031	0.017	**0.014**
	$d_{\overrightarrow{H}}$	4.61	0.98	0.76	0.64	0.51	0.508	0.370	**0.181**	0.230

B. *Surface Reconstruction for Real Scans*

Dataset and metrics: For surface reconstruction of real point cloud scans, we follow VisCo [74] to evaluate our method under the Surface Reconstruction Benchmarks (SRB) [64]. We use Chamfer and Hausdorff distances (CD_{L1} and HD) between the reconstruction meshes and the ground truth. Furthermore, we report their corresponding one-sided distances ($d_{\overrightarrow{C}}$ and $d_{\overrightarrow{H}}$) between the reconstructed meshes and the input noisy point cloud.

Comparisons: We compare our method with state-of-the-art methods under the real scanned SRB dataset, including IGR [4], SPSR [66], Shape As Points (SAP) [12], NeuralPull (NP) [7], and GridPull (GP) [10]. The numerical comparisons are shown in Table VII, where we achieve the best accuracy in most cases. The visual comparisons in Fig. 15 demonstrate that our method can reconstruct a continuous and smooth surface with geometry details.



(a) Input (b) SAP (c) SPSR (d) NeuralPull (e) **Ours**

Fig. 15. Visual comparisons on SRB dataset.

TABLE VIII
COMPARISONS ON SHAPENET CARS DATASET (CHAMFER-L2$\times 10^4$)

Method	Chamfer-L2		F-Score		NC
	Mean	Median	$F_0.005$	$F_0.01$	
Input	0.363	0.355	48.50	88.34	-
Watertight GT	2.628	2.293	68.82	81.60	-
GT	0.076	0.074	95.70	99.99	-
NDF [37]	0.202	0.193	77.40	97.97	79.1
GIFS [39]	0.128	0.123	88.05	99.31	-
CAPUDF$_{BPA}$ [42]	0.141	0.138	84.84	99.33	81.8
CAPUDF [42]	0.119	0.114	88.55	99.82	82.5
LSUDF [43]	0.098	0.097	92.18	**99.90**	85.0
Ours	**0.095**	**0.091**	**92.73**	99.80	**86.0**

C. Surface Reconstruction for Non-Watertight Shapes

Dataset and metrics: To validate our improved implicit filter for learning UDFs, we conduct evaluations on the car subset of ShapeNet [63], without applying watertight processing. These models feature a high quantity of multi-layer and non-closed shapes. We sample 10 k points from the surface of each shape as input data. To evaluate the quality of the reconstruction, we follow the approach proposed in GIFS [39], sampling 100 k points from the reconstructed surfaces. We use Chamfer distance ($\times 10^4$), Normal Consistency (NC), and F-Score with thresholds of 0.005 and 0.01 as evaluation metrics.

Comparisons: We compare our implicit filtering method with the latest works NDF [37], GIFS [39], CAPUDF [42], and LSUDF [43]. The numerical comparisons are shown in Table VIII. Our method achieves the best performance, especially on the term of normal consistency. The visual comparisons are shown in Fig. 16, our method can reconstruct more smooth and continuous non-watertight surfaces.

D. Surface Reconstruction for Large Scans

Dataset and metrics: To evaluate the effect of our implicit filter on reconstruction of high-frequency details, we select three models from the ThreedScans dataset [75] following NSH [70]. We sample 50 k points from these meshes as input points. To evaluate the effect, 1 M points are sampled from the reconstructed and ground truth meshes, and we use the Chamfer distance (CD_{L1}) and F-score to measure the error.

Comparisons: To validate the effect on the meshes with high-frequency details. We compare our methods with the no-filter version and NSH [70], which aims to recover the details of the reconstructed meshes. The detailed comparisons are shown in Table IX and are visualized in Fig. 17. Our method performs filtering without excessively smoothing details. High-precision reconstruction typically requires a higher point density. In this case, since our neighborhood range is limited and L_{CD} constrains our gradient direction, details will be preserved throughout the reconstruction process.

TABLE IX
COMPARISONS ON THREEDSCANS DATASET

	Dragon		Hosmer		Eagle	
	CD_{L1}	F-S.	CD_{L1}	F-S.	CD_{L1}	F-S.
NSH [70]	0.068	0.876	0.066	0.866	0.085	0.687
w/o filter	0.075	0.888	0.074	0.809	0.086	0.667
Ours	**0.062**	**0.891**	**0.063**	**0.877**	**0.081**	**0.709**

E. Surface Reconstruction for Scenes

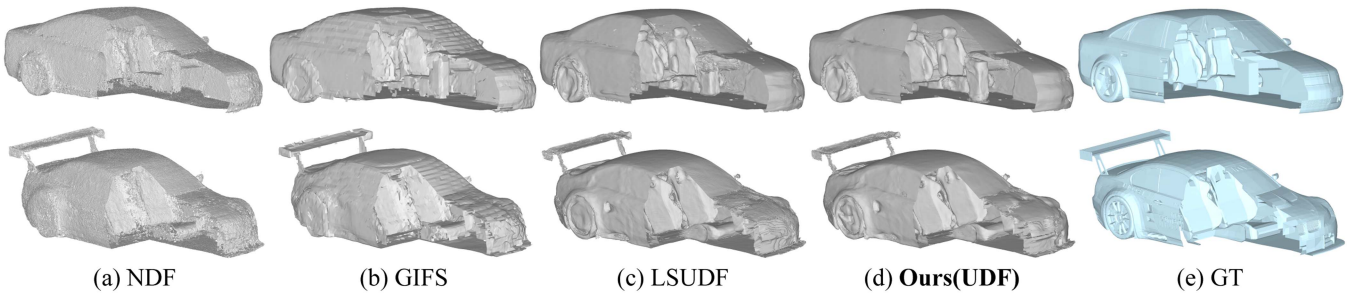
Dataset and metrics: To further demonstrate the advantage of our method in the surface reconstruction of real scene scans, we conduct experiments using the 3D Scene dataset. The 3D Scene dataset is a challenging real-world dataset with complex topology and noisy open surfaces. We uniformly sample 1000 points per m^2 of each scene as the input and follow PCP [28] to sample 1M points on both the reconstructed and the ground truth surfaces. We leverage L1 and L2 Chamfer distance (CD_{L1}, CD_{L2}) and normal consistency (NC) to evaluate the reconstruction quality.

Comparisons: We compare our method with the state-of-the-art SDF learning methods CONet [21], LIG [18], DeepLS [19], NeuralPull (NP) [7], PCP [28], GridPull (GP) [10] and UDF learning methods NDF [37], GIFS [39], CAPUDF [42], LSUDF [43]. The numerical comparisons in Table X demonstrate our superior performance in all scenes even compared with the local-based methods. We further present visual comparisons in Fig. 18. The visualization further shows that our method can achieve smoother with high-fidelity surfaces in complex scenes. It should be noted that the surface we extract here is not the zero level set but the 0.001 level set since the scene is not watertight. For NeuralPull we use the threshold of 0.005 instead of 0.001 to extract the complete surface therefore the mesh looks thicker. At the same time, we visualize the reconstruction results by our UDF learning filter in Fig. 18 and represent the quantitative comparisons in Table X. These comparisons of UDF learning can validate the priority of our method on open scene surface reconstruction.

F. Sparse-View Surface Reconstruction

Dataset and metric: To evaluate the effect of the filtered UDF prior, we conduct experiments under the DTU dataset [76]. Following the settings in previous methods [77], we report the results reconstructed from 3 images without mask supervision under 15 scenes. We measure the Chamfer Distances on the DTU dataset in the same way as VolRecon [77] to quantitatively evaluate the reconstruction quality.

Comparisons: We compare our method with the traditional method Colmap [78] and the state-of-the-art learning-based methods. The comparisons of Chamfer Distance on the DTU

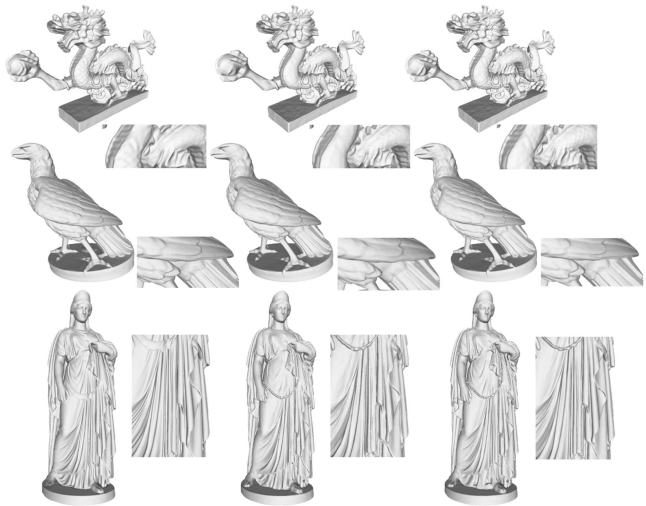


(a) NDF (b) GIFS (c) LSUDF (d) **Ours(UDF)** (e) GT

Fig. 16. Visual comparisons of multi-layer surface reconstruction on ShapeNet cars.

TABLE X
COMPARISONS ON 3D SCENE DATASET, $CD_{L2} \times 1000$

		Burghers			Lounge			Copyroom			Stonewall			Totempole		
		CD_{L2}	CD_{L1}	NC	CD_{L2}	CD_{L1}	NC	CD_{L2}	CD_{L1}	NC	CD_{L2}	CD_{L1}	NC	CD_{L2}	CD_{L1}	NC
SDF	CONet [21]	27.46	0.079	0.907	9.54	0.046	0.894	10.97	0.045	0.892	20.46	0.069	0.905	2.054	0.021	0.943
	LIG [18]	3.055	0.045	0.835	9.672	0.056	0.833	3.61	0.036	0.810	5.032	0.042	0.879	9.58	0.062	0.887
	DeepLS [19]	0.401	0.017	0.920	6.103	0.053	0.848	0.609	0.021	0.901	0.320	0.015	0.954	0.601	0.017	0.950
	GP [10]	1.367	0.028	0.873	4.684	0.053	0.827	2.327	0.030	0.857	2.234	0.024	0.913	2.278	0.034	0.878
	PCP [28]	1.339	0.031	0.929	0.432	0.014	**0.934**	0.405	0.014	**0.914**	0.266	0.014	**0.957**	1.089	0.029	**0.954**
	NP [7]	0.897	0.025	0.883	0.855	0.022	0.887	0.479	0.018	0.862	0.434	0.018	0.929	1.604	0.032	0.923
	Ours	**0.133**	**0.011**	**0.934**	**0.120**	**0.008**	0.926	**0.111**	**0.009**	0.913	**0.082**	**0.009**	**0.957**	**0.203**	**0.013**	0.944
UDF	NDF [37]	1.168	0.027	0.901	0.393	0.014	0.910	0.269	0.013	0.908	0.509	0.019	0.936	2.02	0.036	0.922
	CAPUDF [42]	0.191	0.010	0.910	0.092	0.008	0.927	0.113	0.009	0.911	0.066	0.007	0.962	0.139	0.009	0.955
	LSUDF [43]	0.146	0.009	0.921	**0.068**	**0.006**	**0.941**	0.093	0.007	0.925	0.050	**0.005**	0.970	**0.107**	**0.007**	0.960
	Ours(UDF)	**0.127**	**0.008**	**0.923**	0.079	**0.006**	0.937	**0.070**	**0.006**	**0.935**	**0.047**	**0.005**	**0.971**	0.115	**0.007**	**0.962**



(a) NSH (b) w/o filter (c) **Ours**

Fig. 17. Visual comparisons on ThreedScans dataset.

dataset are shown in Table XI where our method achieves better performance compared with other SDF-based methods. The visual comparisons in Fig. 19 show that our method can reconstruct high-fidelity meshes even with the guidance of complex images.

G. Unsupervised Point Normal Estimation

Dataset and metrics: We use the most widely-used dataset PCPNet [79] for the point normal estimation to evaluate our filtering. We conduct experiments on three kinds of point clouds with different sampling methods, including sampling uniformly on shapes("Clean"), and two additional settings with varying point densities ("Stripes" and "Gradients"). Following PCPNET [79], we sample 5000 points per shape and compute the angle root mean square error (RMSE) between the predicted normals and the ground truth normals as the evaluation metrics.

Comparisons: We compare our method with both supervised and unsupervised methods including PCPNet [79], HoughCNN [80], Nesti-Net [81], IterNet [82], DeepFit [83], Jet [84], PCA [85], CAPUDF [42], LSUDF [43]. The numerical comparisons presented in Table XII indicate that our method outperforms all other approaches, including those that rely on large-scale training datasets. The visual comparisons in Fig. 20 show that our method can learn better normals, especially on complex or sharp geometry surfaces. These results indicate that our filter can improve the normals of the input points by denoising the implicit field.

H. Unsupervised Point Cloud Upsampling

Dataset and Metric: We evaluate our implicit filtering on the dataset used in PU-GAN [86] for the task of point cloud upsampling. Each point cloud is upsampled to 8192 points from 2048 points. The evaluation metrics include Chamfer and Hausdorff distances (CD and HD), and Point-to-Surface distance (P2F).

Comparisons: We also conduct a comparison with supervised point cloud upsampling methods including PU-Net [87], MPU [88], PU-GAN [86], Dis-PU [89], Grad-PU [90] and

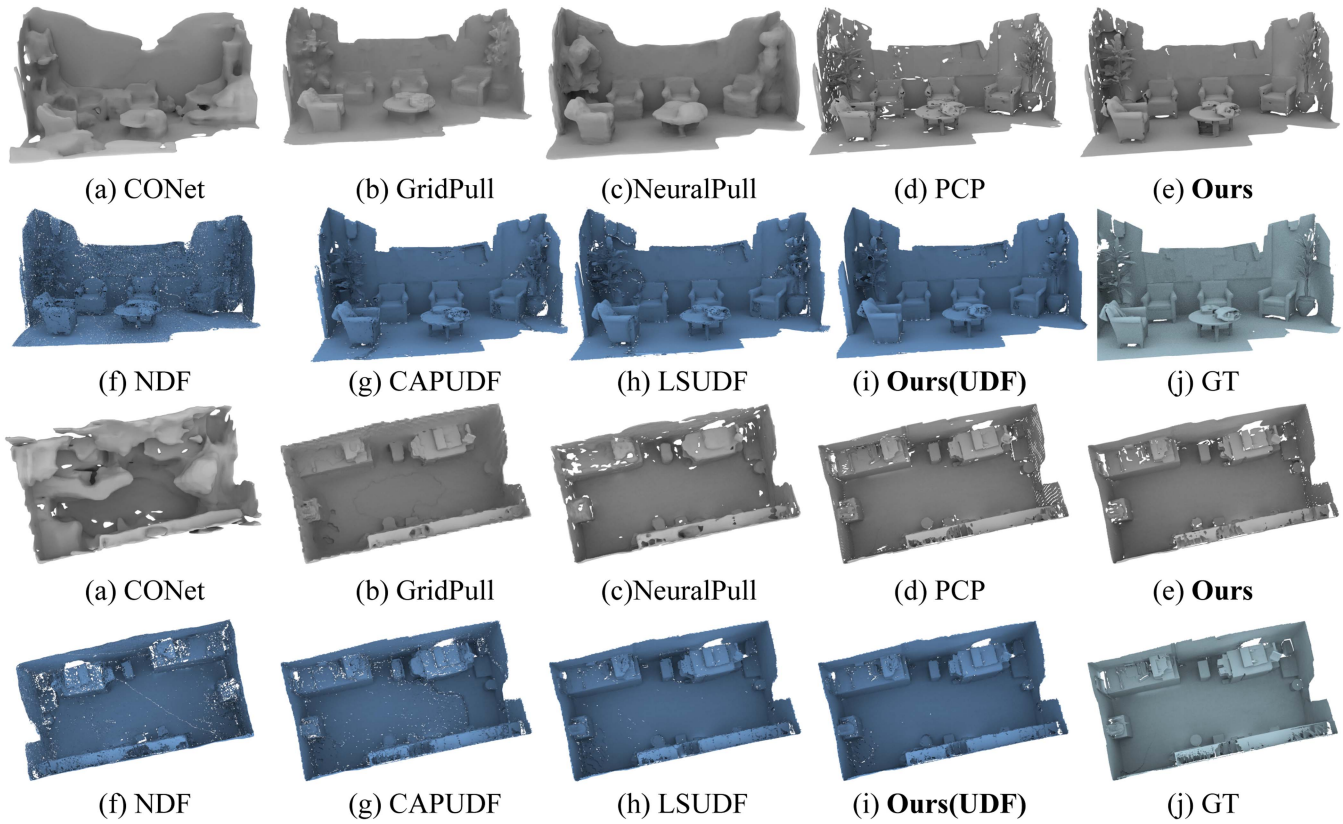


Fig. 18. Visual comparisons of surface reconstruction on 3D Scene dataset.

TABLE XI
COMPARISONS ON DTU DATASET (LARGE-OVERLAP SETTING). IN THIS TABLE, THE BEST RESULTS ARE IN BOLD, THE SECOND BEST ARE UNDERLINED.

Scan ID	24	37	40	55	63	65	69	83	97	105	106	110	114	118	122	Mean
COLMAP [78]	0.90	2.89	1.63	1.08	2.18	1.94	1.61	1.30	2.34	1.28	1.10	1.42	0.76	1.17	1.14	1.52
SparseNeuS$_{ft}$ [34]	1.29	2.27	1.57	0.88	1.61	1.86	1.06	1.27	1.42	1.07	0.99	0.87	0.54	1.15	1.18	1.27
VolRecon [77]	1.20	2.59	1.56	1.08	1.43	1.92	1.11	1.48	1.42	1.05	1.19	1.38	0.74	1.23	1.27	1.38
ReTR [94]	1.05	2.31	1.44	0.98	1.18	1.52	0.88	1.35	1.30	0.87	1.07	0.77	0.59	1.05	1.12	1.17
C2F2NeuS [95]	1.12	2.42	1.40	**0.75**	1.41	1.77	0.85	1.16	1.26	0.76	0.91	0.60	0.46	0.88	0.92	1.11
UFORecon [96]	0.76	**2.05**	1.31	0.82	1.12	**1.18**	0.74	1.17	1.11	0.71	0.88	0.58	0.54	0.86	0.99	0.99
NeuS [33]	4.57	4.49	3.97	4.32	4.63	1.95	4.68	3.83	4.15	2.50	1.52	6.47	1.26	5.57	6.11	4.00
VolSDF [97]	4.03	4.21	6.12	0.91	8.24	1.73	2.74	1.82	5.14	3.09	2.08	4.81	0.60	3.51	2.18	3.41
MonoSDF [35]	2.85	3.91	2.26	1.22	3.37	1.95	1.95	5.53	5.77	1.10	5.99	2.28	0.65	2.65	2.44	2.93
NeuSurf [65]	0.78	2.35	1.55	**0.75**	1.04	1.68	**0.60**	**1.14**	**0.98**	0.70	**0.74**	0.49	**0.39**	**0.75**	**0.86**	0.99
Ours	**0.69**	2.13	**1.30**	0.77	1.05	1.39	0.69	1.22	1.02	**0.69**	0.75	0.49	0.41	**0.75**	0.91	**0.95**

TABLE XII
COMPARISONS OF NORMAL ESTIMATION ON PCPNET DATASET

Method	Clean	Strip	Gradient	Average
PCPNet [79]	9.66	11.47	13.42	11.61
Hough [80]	10.23	12.47	11.02	11.24
Nesti-Net [81]	6.99	8.47	9.00	8.15
IterNet [82]	6.72	7.73	7.51	7.32
DeepFit [83]	6.51	7.92	7.31	7.25
Jet [84]	12.23	13.39	13.13	12.92
PCA [85]	12.29	13.66	12.81	12.92
CAPUDF [42]	7.30	7.74	7.78	7.61
LSUDF [43]	6.51	7.45	7.64	7.20
Ours	**5.76**	**6.94**	**6.79**	**6.50**

unsupervised methods including EAR [47], L2G-AR [91], SPU-Net [92], APU-LDI [93], CAPUDF [42] and LSUDF [43]. The quantitative comparisons are shown in Table XIII. We visualize the Point-to-Surface distance in Fig. 21. Our method effectively preserves the sharp geometric features of the original shape compared with the other methods.

I. Ablation Studies

We conduct ablation studies on the FAMOUS dataset to demonstrate the effectiveness of our proposed implicit filter and explore the effect of some important hyperparameters. We report the performance in terms of L1 and L2 Chamfer distance

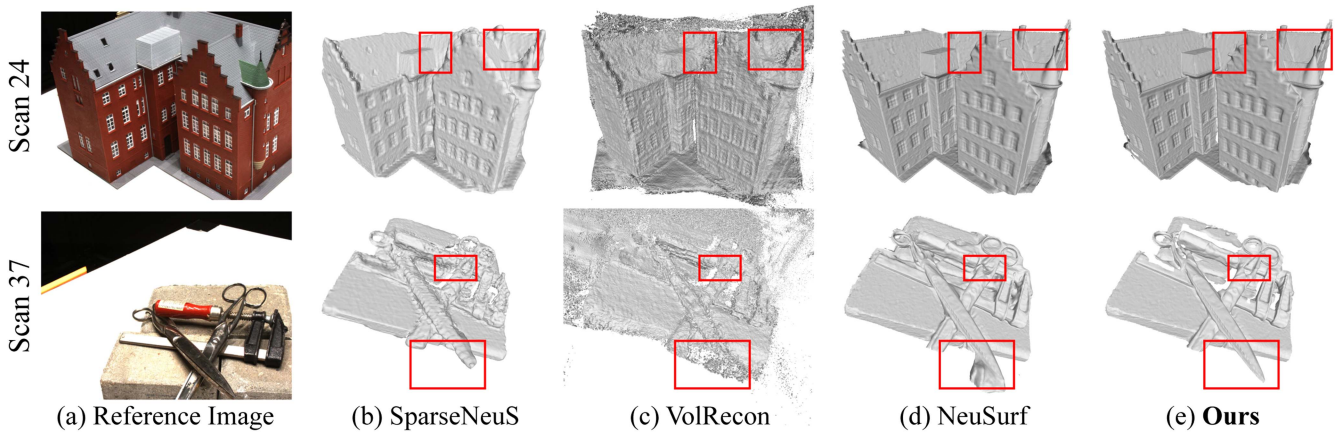


Fig. 19. Visual comparisons of surface reconstruction on DTU dataset.

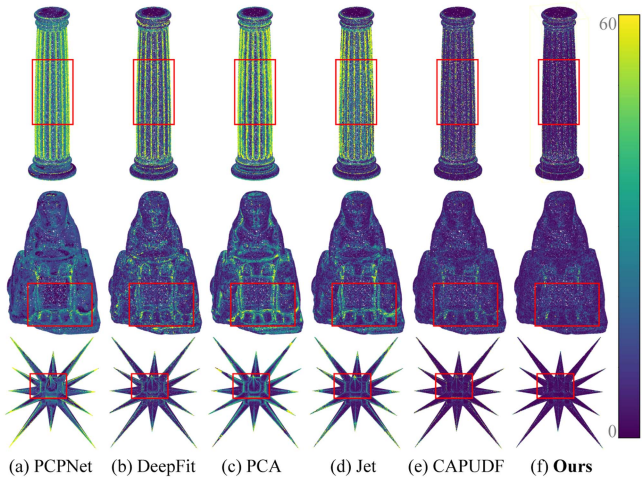


Fig. 20. Visual comparisons of normal estimations on PCPNet dataset.

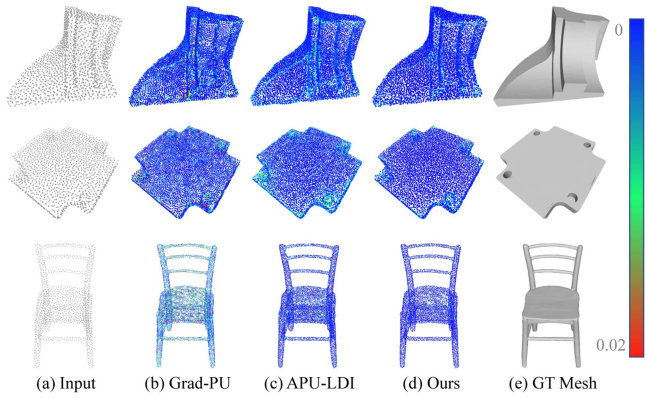


Fig. 21. Visual comparisons of point cloud upsampling on PU-GAN dataset.

TABLE XIII
COMPARISONS OF POINT CLOUD UPSAMPLING RESULTS ON PU-GAN DATASET

	Method	P2F	CD	HD
Supervised	PU-Net [87]	6.84	0.72	8.94
	MPU [88]	3.96	0.49	6.11
	PU-GAN [86]	2.33	0.28	4.64
	Dis-PU [89]	2.01	0.22	2.83
	Grad-PU [90]	1.95	0.26	2.46
Unspervised	EAR [47]	5.82	0.52	7.37
	L2G-AE	39.37	6.31	63.23
	SPU-Net (All2T) [92]	5.79	0.37	2.55
	CAPUDF [42]	1.43	0.24	2.16
	APU-LDI [93]	1.34	0.23	1.68
	LSUDF [43]	1.02	0.19	1.55
	Ours	0.77	0.12	1.40

$(CD_{L1}, CD_{L2} \times 10^3)$, normal consistency (NC), and F-Score (F-S.). We also conduct ablation experiments on the 3D Scene dataset to validate the effectiveness of our designed UDF implicit filtering.

Effect of Eikonal loss: We select the L_{CD} to prevent the degeneration of the gradient since it both constrains the value and the gradient of the SDF. It also guides how to pull the query point onto the surface. Therefore we omit the Eikonal term used in previous methods like the IGR [4], SIREN [5], and DIGS [9] which have no other direct supervision for the gradient. To verify this selection, we conduct the following experiments by trade-off these two functions. With the experimental results in Table XIV, we find that only applying the Eikonal term is not as effective as CD alone. At the same time combining the Eikonal term with CD does not further enhance the experiment results, but the difference is small.

TABLE XIV
EFFECT OF THE EIKONAL TERM

Loss	CD_{L1}	CD_{L2}	F-S.	NC
w/ Eikonal, w/o CD	0.009	0.021	0.738	0.899
w/ Eikonal, w/ CD	0.008	0.009	0.774	0.910
w/o Eikonal, w/ CD	**0.007**	**0.008**	**0.778**	**0.911**

Effect of level set filtering: To justify the effectiveness of each term in our loss function. We report the results trained by different combinations in Table XV. The L_{CD} is more applicable for training SDF from raw point clouds. The zero-level filter can help remove the noise and keep the geometric features. Filtering

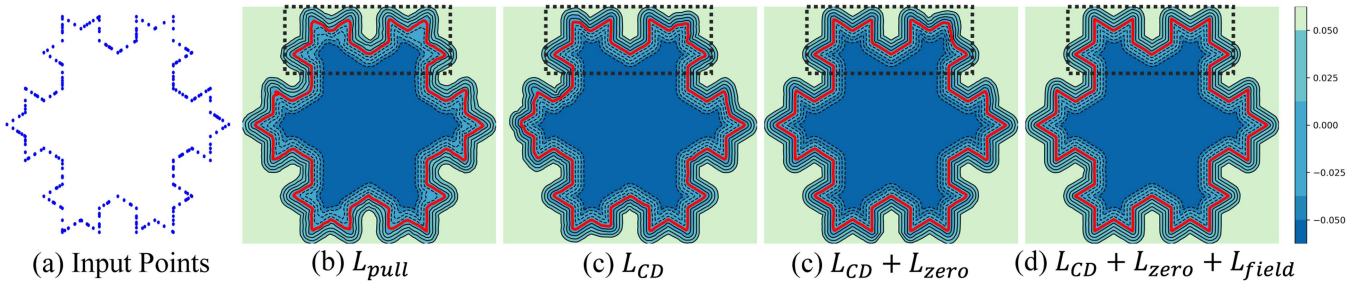


Fig. 22. The 2D level sets show the distance field learned by different losses. The red lines represent the learned zero level set.

TABLE XV
EFFECT OF DIFFERENT LOSSES

Loss	CD_{L1}	CD_{L2}	F-S.	NC
L_{pull}	0.012	0.083	0.742	0.884
L_{CD}	0.010	0.031	0.757	0.891
$L_{CD} + L_{zero}$	0.008	0.018	0.772	0.905
$L_{CD} + L_{zero} + L_{field}$	0.008	0.011	0.769	0.908
Ours	**0.007**	**0.008**	**0.778**	**0.911**

TABLE XVI
EFFECT OF BIDIRECTIONAL PROJECTION

	CD_{L1}	CD_{L2}	F-S.	NC
$d(\bar{\boldsymbol{p}})$	0.010	0.024	0.726	0.890
$d_{bi}(\bar{\boldsymbol{p}})$	**0.007**	**0.008**	**0.778**	**0.911**

TABLE XVII
EFFECT OF WEIGHT α_3

α_3	CD_{L1}	CD_{L2}	F-S.	NC
0	0.008	0.013	0.758	0.903
1	**0.007**	0.011	0.772	0.910
10	**0.007**	**0.008**	**0.778**	**0.911**
100	0.008	0.009	0.774	0.909

TABLE XVIII
EFFECT OF FILTER PARAMETERS σ_n AND σ_p

		CD_{L1}	CD_{L2}	F-S.	NC
σ_n	15°	**0.007**	**0.008**	**0.778**	**0.911**
	30°	**0.007**	0.011	0.771	0.907
	45°	0.008	0.012	0.764	0.903
	60°	0.008	0.010	0.767	0.901
σ_p	max	**0.007**	**0.008**	**0.778**	**0.911**
	diagonal	0.008	0.011	0.763	0.904

TABLE XIX
EFFECT OF UDF FILTER FRAMEWORK DESIGN

	w/o BiG	w/o L_{proj}	scratch	Ours
CD_{L2}	0.088	0.090	0.100	**0.087**
NC	94.40	94.47	94.45	**94.55**
	w/o GITS	Sync_5	Sync_{10}	Sync_{100}
CD_{L2}	0.091	0.089	**0.087**	0.088
NC	94.54	94.50	**94.55**	94.48

across non-zero level sets can improve the overall consistency of the entire signed distance field. Since we assume all input points lie on the surface, the function L_{dist} is also necessary. Fig. 22 shows a 2D comparison of these losses, showing that our filter loss functions can reconstruct a field that is aligned at all level sets and maintains geometric characteristics.

Effect of the bidirectional projection: To validate our bidirectional projection distance, we report the results in Table XVI. The numerical comparisons show that projecting the distance to both normals can improve the reconstruction quality. Note that only using $d(\bar{\boldsymbol{p}})$ can also improve the results.

Weight of level set projection loss: We explore the effect of the L_{CD} loss function by adjusting the weight α_3 in (11). We report our results with different candidates {0, 1, 10} in Table XVII, where 0 means we do not use the L_{CD} to constrain the gradient. The comparisons in Table XVII show that although our implicit filter can directly learn SDFs, it is better to adopt the L_{CD} for a more stable field. However, if the weight is too large, the filtering effect will decrease. Selecting the weight ranging from 1 to 10 is recommended, which is usually adequate. For the weights α_1 and α_2, setting them to 1 is always necessary.

Effect of filter parameters: We compare the effect of different parameters σ_n, σ_p in Table XVIII. The diagonal weight for σ_p means the length of the diagonal of the bounding box for the local patch mentioned in [54]. The results indicate that the method is relatively robust to parameter variation in a certain range.

Effect of UDF filter design: To validate the effectiveness of the modification we propose to migrate the implicit filter to the UDF learning, we report the results on the 3D Scene dataset in Table XIX. We report the performance without bidirectional gradient as "w/o BiG" and without project distance loss function as "w/o L_{proj}. We also report the results of training from scratch as "scratch". The second row shows the results without the gradient immutable training schema as "w/o GITS" and with the schema but under different synchronization epochs as "Sync_n". The results verify the effectiveness of our design for UDF learning.

VIII. LIMITATIONS

Given that our implicit filter smooths each level set via points in the neighborhood, the selection of a neighborhood with overly intricate geometry could yield inaccurate results when the points

are sparse. This limitation is not uncommon in the majority of analogous filtering processes. For example, it's hard to filter a low-resolution 2D image with complex content. Meanwhile, processing the neighbor points also increases our training time.

IX. Conclusion

We introduce implicit filtering on SDFs to reduce the noise of the signed distance field while preserving geometric features. We filter the distance field by minimizing the weighted bidirectional projection distance, where we can generate sampling points on the zero level set and neighbor points on non-zero level sets by the pulling procedure. By leveraging the Chamfer distance, we address the gradient degeneration problem. Additionally we extend the implicit filtering to the UDFs and apply it to the sparse-view reconstruction, point normal estimation, and point cloud upsampling tasks. The visual and numerical comparisons demonstrate our effectiveness and superiority over state-of-the-art methods.

References

[1] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.

[2] M. Atzmon and Y. Lipman, "SAL: Sign agnostic learning of shapes from raw data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2562–2571.

[3] M. Atzmon and Y. Lipman, "SALD: Sign agnostic learning with derivatives," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 4422–4438.

[4] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 3789–3799.

[5] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7462–7473.

[6] B. Ma, J. Zhou, Y.-S. Liu, and Z. Han, "Towards better gradient consistency for neural signed distance functions via level set alignment," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17724–17734.

[7] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, "Neural-pull: Learning signed distance function from point clouds by learning to pull space onto surface," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 7246–7257.

[8] Q. Li et al., "Learning signed hyper surfaces for oriented point cloud normal estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 9957–9974, Dec. 2024.

[9] Y. Ben-Shabat, C. H. Koneputugodage, and S. Gould, "DiGS: Divergence guided shape implicit neural representation for unoriented point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 19323–19332.

[10] C. Chen, Y.-S. Liu, and Z. Han, "GridPull: Towards scalability in learning implicit representations from 3D point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 18322–18334.

[11] P. Erler, P. Guerrero, S. Ohrhallinger, N. J. Mitra, and M. Wimmer, "Points2Surf: Learning implicit surfaces from point clouds," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 108–124.

[12] S. Peng, C. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, "Shape as points: A differentiable Poisson solver," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 13032–13044.

[13] S. Li, G. Gao, Y. Liu, M. Gu, and Y.-S. Liu, "Implicit filtering for learning neural signed distance functions from 3D point clouds," in *Proc. Eur. Conf. Comput. Vis.*, 2024, pp. 234–251.

[14] A. Boulch and R. Marlet, "POCO: Point convolution for surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6302–6314.

[15] Z. Mi, Y. Luo, and W. Tao, "SSRNet: Scalable 3D surface reconstruction network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 970–979.

[16] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4460–4470.

[17] S.-L. Liu, H.-X. Guo, H. Pan, P.-S. Wang, X. Tong, and Y. Liu, "Deep implicit moving least-squares functions for 3D reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 1788–1797.

[18] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, and T. Funkhouser et al., "Local implicit grid representations for 3D scenes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6001–6010.

[19] R. Chabra et al., "Deep local shapes: Learning local SDF priors for detailed 3D reconstruction," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 608–625.

[20] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt, "PatchNets: Patch-based generalizable deep implicit 3D shape representations," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 293–309.

[21] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 523–540.

[22] J. Tang, J. Lei, D. Xu, F. Ma, K. Jia, and L. Zhang, "SA-ConvONet: Sign-agnostic optimization of convolutional occupancy networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 6504–6513.

[23] Z. Wang et al., "ALTO: Alternating latent topologies for implicit 3D reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 259–270.

[24] S. Li, G. Gao, Y. Liu, Y.-S. Liu, and M. Gu, "GridFormer: Point-grid transformer for surface reconstruction," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 3163–3171.

[25] J. Huang, Z. Gojcic, M. Atzmon, O. Litany, S. Fidler, and F. Williams, "Neural kernel surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 4369–4379.

[26] F. Williams et al., "Neural fields as learnable kernels for 3D reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18500–18510.

[27] C. H. Koneputugodage, Y. Ben-Shabat, D. Campbell, and S. Gould, "Small steps and level sets: Fitting neural surface models with point guidance," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 21456–21465.

[28] B. Ma, Y.-S. Liu, M. Zwicker, and Z. Han, "Surface reconstruction from point clouds by learning predictive context priors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6326–6337.

[29] B. Ma, Y.-S. Liu, and Z. Han, "Reconstructing surfaces for sparse point clouds with on-surface priors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 6315–6325.

[30] Z. Marschner, S. Sellán, H.-T. D. Liu, and A. Jacobson, "Constructive solid geometry on neural signed distance fields," in *Proc. SIGGRAPH Asia Conf. Papers*, 2023, pp. 1–12.

[31] J. Zhou, B. Ma, Y.-S. Liu, and Z. Han, "Fast learning of signed distance functions from noisy point clouds via noise to noise mapping," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 8936–8953, Dec. 2024.

[32] Z. Wang et al., "Neural-IMLS: Self-supervised implicit moving least-squares network for surface reconstruction," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 8, pp. 5018–5033, Aug. 2024.

[33] 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 *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 27171–27183.

[34] X. Long, C. Lin, P. Wang, T. Komura, and W. Wang, "SparseNeuS: Fast generalizable neural surface reconstruction from sparse views," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 210–227.

[35] Z. Yu, S. Peng, M. Niemeyer, T. Sattler, and A. Geiger, "MonoSDF: Exploring monocular geometric cues for neural implicit surface reconstruction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 25018–25032.

[36] C. Chen, Y.-S. Liu, and Z. Han, "NeuralTPS: Learning signed distance functions without priors from single sparse point clouds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 1, pp. 565–582, Jan. 2025.

[37] J. Chibane et al., "Neural unsigned distance fields for implicit function learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 21638–21652.

[38] W. Chen, C. Lin, W. Li, and B. Yang, "3PSDF: Three-pole signed distance function for learning surfaces with arbitrary topologies," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 18522–18531.

[39] J. Ye, Y. Chen, N. Wang, and X. Wang, "GIFS: Neural implicit function for general shape representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 12819–12829.

[40] L. Wang et al., "HSDF: Hybrid sign and distance field for neural representation of surfaces with arbitrary topologies," *IEEE Trans. Vis. Comput. Graph.*, vol. 31, no. 9, pp. 5215–5228, Sep. 2025.

[41] H. Tian, C. Zhu, Y. Shi, and K. Xu, "SuperUDF: Self-supervised UDF estimation for surface reconstruction," *IEEE Trans. Vis. Comput. Graph.*, vol. 30, no. 9, pp. 5965–5975, Sep. 2024.

[42] J. Zhou, B. Ma, S. Li, Y.-S. Liu, Y. Fang, and Z. Han, "CAP-UDF: Learning unsigned distance functions progressively from raw point clouds with consistency-aware field optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, pp. 7475–7492, Dec. 2024.
[43] J. Zhou, B. Ma, S. Li, Y.-S. Liu, and Z. Han, "Learning a more continuous zero level set in unsigned distance fields through level set projection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 3181–3192.
[44] C. Xu, F. Hou, W. Wang, H. Qin, Z. Zhang, and Y. He, "Details enhancement in unsigned distance field learning for high-fidelity 3D surface reconstruction," in *Proc. AAAI Conf. Artif. Intell.*, 2025, pp. 8806–8814.
[45] M. Fainstein, V. Siless, and E. Iarussi, "DUDF: Differentiable unsigned distance fields with hyperbolic scaling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 4484–4493.
[46] Y. Lu et al., "Unsigned orthogonal distance fields: An accurate neural implicit representation for diverse 3D shapes," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 20551–20560.
[47] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 1–12, 2013.
[48] B. Liao, C. Xiao, L. Jin, and H. Fu, "Efficient feature-preserving local projection operator for geometry reconstruction," *Comput.-Aided Des.*, vol. 45, no. 5, pp. 861–874, 2013.
[49] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," *Comput. Graph. Forum*, vol. 28, no. 2, pp. 493–501, 2009.
[50] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 22–es, 2007.
[51] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–7, 2009.
[52] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, and M. Wimmer, "Continuous projection for fast L1 reconstruction," *ACM Trans. Graph.*, vol. 33, no. 4, 2014, Art. no. 47.
[53] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "EC-Net: An edge-aware point set consolidation network," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 386–402.
[54] D. Zhang, X. Lu, H. Qin, and Y. He, "Pointfilter: Point cloud filtering via encoder-decoder modeling," *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 3, pp. 2015–2027, Mar. 2021.
[55] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 544–552, 2005.
[56] C.-C. Kuo and H.-T. Yau, "A new combinatorial approach to surface reconstruction with sharp features," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 1, pp. 73–82, Jan./Feb. 2006.
[57] X. Wang, Y. Cheng, L. Wang, J. Lu, K. Xu, and G. Xiao, "Edge preserving implicit surface representation of point clouds," 2023, *arXiv:2301.04860*.
[58] D. B. Lindell, D. Van Veen, J. J. Park, and G. Wetzstein, "BACON: Band-limited coordinate networks for multiscale scene representation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 16252–16262.
[59] M. Tancik et al., "Fourier features let networks learn high frequency functions in low dimensional domains," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7537–7547.
[60] W. Yifan, L. Rahmann, and O. Sorkine-hornung, "Geometry-consistent neural shape representation with implicit displacement fields," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 23095–23114.
[61] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, San Diega, CA, USA, 2015, pp. 1–15.
[62] S. Koch et al., "ABC: A big CAD model dataset for geometric deep learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9601–9611.
[63] A. X. Chang et al., "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.
[64] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, and D. Panozzo, "Deep geometric prior for surface reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10130–10139.
[65] H. Huang, Y. Wu, J. Zhou, G. Gao, M. Gu, and Y.-S. Liu, "NeuSurf: On-surface priors for neural surface reconstruction from sparse input views," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 2312–2320.
[66] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 1–13, 2013.
[67] C. Chen, Y.-S. Liu, and Z. Han, "Latent partition implicit with surface codes for 3D representation," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 322–343.
[68] F. Hou, C. Wang, W. Wang, H. Qin, C. Qian, and Y. He, "Iterative poisson surface reconstruction (iPSR) for unoriented points," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 1–13, 2022.
[69] Q. Dong et al., "NeurCADRecon: Neural representation for reconstructing CAD surfaces by enforcing zero Gaussian curvature," *ACM Trans. Graph.*, vol. 43, no. 4, pp. 1–17, 2024.
[70] Z. Wang et al., "Neural-singular-hessian: Implicit neural representation of unoriented point clouds by enforcing singular Hessian," *ACM Trans. Graph.*, vol. 42, no. 6, pp. 1–14, 2023.
[71] S. Lin, D. Xiao, Z. Shi, and B. Wang, "Surface reconstruction from point clouds without normals by parametrizing the Gauss formula," *ACM Trans. Graph.*, vol. 42, no. 2, pp. 1–19, 2022.
[72] Z. Chen, A. Tagliasacchi, and H. Zhang, "BSP-Net: Generating compact meshes via binary space partitioning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 42–51.
[73] R. Xu et al., "RFEPS: Reconstructing feature-line equipped polygonal surface," *ACM Trans. Graph.*, vol. 41, no. 6, pp. 1–15, 2022.
[74] A. Pumarola, A. Sanakoyeu, L. Yariv, A. Thabet, and Y. Lipman, "VisCo Grids: Surface reconstruction with Viscosity and Coarea grids," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 18060–18071.
[75] O. Laric, "Three D scans," 2012. [Online]. Available: https://threedscans.com/
[76] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanæs, "Large scale multi-view stereopsis evaluation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 406–413.
[77] Y. Ren, F. Wang, T. Zhang, M. Pollefeys, and S. Süsstrunk, "VolRecon: Volume rendering of signed ray distance functions for generalizable multi-view reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16685–16695.
[78] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4104–4113.
[79] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "PCPNet learning local shape properties from raw point clouds," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 75–85, 2018.
[80] A. Boulch and R. Marlet, "Deep learning for robust normal estimation in unstructured point clouds," *Comput. Graph. Forum*, vol. 35, no. 5, pp. 281–290, 2016.
[81] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "Nesti-Net: Normal estimation for unstructured 3D point clouds using convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10112–10120.
[82] J. E. Lenssen, C. Osendorfer, and J. Masci, "Deep iterative surface normal estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11247–11256.
[83] Y. Ben-Shabat and S. Gould, "DeepFit: 3D surface fitting via neural network weighted least squares," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 20–34.
[84] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Comput. Aided Geometric Des.*, vol. 22, no. 2, pp. 121–146, 2005.
[85] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433–459, 2010.
[86] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: A point cloud upsampling adversarial network," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 7203–7212.
[87] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2790–2799.
[88] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, "Patch-based progressive 3D point set upsampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5958–5967.
[89] R. Li, X. Li, P.-A. Heng, and C.-W. Fu, "Point cloud upsampling via disentangled refinement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 344–353.
[90] Y. He, D. Tang, Y. Zhang, X. Xue, and Y. Fu, "Grad-PU: Arbitrary-scale point cloud upsampling via gradient descent with learned distance functions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 5354–5363.
[91] X. Liu, Z. Han, X. Wen, Y.-S. Liu, and M. Zwicker, "L2G auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 989–997.

[92] X. Liu, X. Liu, Y.-S. Liu, and Z. Han, "SPU-Net: Self-supervised point cloud upsampling by coarse-to-fine reconstruction with self-projection optimization," *IEEE Trans. Image Process.*, vol. 31, pp. 4213–4226, 2022.
[93] S. Li, J. Zhou, B. Ma, Y.-S. Liu, and Z. Han, "Learning continuous implicit field with local distance indicator for arbitrary-scale point cloud upsampling," in *Proc. AAAI Conf. Artif. Intell.*, vol. 38, no. 4,2024, pp. 3181–3189.
[94] Y. Liang, H. He, and Y. Chen, "ReTR: Modeling rendering via transformer for generalizable neural surface reconstruction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, pp. 62332–62351.
[95] L. Xu et al., "C2F2NeUS: Cascade cost frustum fusion for high fidelity and generalizable neural surface reconstruction," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 18291–18301.
[96] Y. Na, W. J. Kim, K. B. Han, S. Ha, and S.-E. Yoon, "UFORecon: Generalizable sparse-view surface reconstruction from arbitrary and unfavorable sets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 5094–5104.
[97] L. Yariv, J. Gu, Y. Kasten, and Y. Lipman, "Volume rendering of neural implicit surfaces," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 4805–4815.

Shengtao Li received the BS degree in software engineering in 2021 from Tsinghua University, Beijing, China, where he is currently working toward the PhD degree with the School of Software. His research interests include deep learning, 3D shape analysis, and 3D reconstruction.

Yudong Liu received the BS degree in computer science and technology from the Beijing Institute of Technology, Beijing, China, in 2023. He is currently working toward with the School of Software, Tsinghua University, Beijing. His research interests include deep learning and 3D computer vision.

Ge Gao received the BS degree from the College of Materials Science and Technology, Beijing Forestry University, Beijing, China, in 2006, the MS degree from the School of Software, Tsinghua University, Beijing, in 2010, and the PhD degree from the Department of Computer Science and Technology, Tsinghua University, in 2016. From 2016 to 2020, he was a postdoctoral researcher with Tsinghua University. He is currently a research assistant professor with the Beijing National Research Center for Information Science and Technology, Tsinghua University. His research interests include building information modeling, computer aided design, artificial intelligence.

Ming Gu received the BS degree from the National University of Defense Technology, China, in 1984, and the MS degree from Institute of Computer Science, Chinese Academy of Sciences, Shengyang, China, in 1987. She is currently a Professor with the School of Software, Tsinghua University. Her research interests include information model standards and architecture design, behavior modeling, and credibility analysis of software systems.

Yu-Shen Liu (Member, IEEE) received the BS degree in mathematics from Jilin University, China, in 2000, and the PhD degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2006. From 2006 to 2009, he was a postdoctoral researcher with Purdue University. He is currently an Associate Professor with the School of Software, Tsinghua University. His research interests include 3D computer vision, digital geometry processing, 3D reconstruction, and point cloud processing.