CAP-UDF: Learning Unsigned Distance Functions Progressively from Raw Point Clouds with Consistency-Aware Field Optimization

Junsheng Zhou*, Baorui Ma*, Shujuan Li, Yu-Shen Liu, *Member, IEEE,* Yi Fang, Zhizhong Han

Abstract—Surface reconstruction for point clouds is an important task in 3D computer vision. Most of the latest methods resolve this problem by learning signed distance functions from point clouds, which are limited to reconstructing closed surfaces. Some other methods tried to represent open surfaces using unsigned distance functions (UDF) which are learned from ground truth distances. However, the learned UDF is hard to provide smooth distance fields due to the discontinuous character of point clouds. In this paper, we propose CAP-UDF, a novel method to learn consistency-aware UDF from raw point clouds. We achieve this by learning to move queries onto the surface with a field consistency constraint, where we also enable to progressively estimate a more accurate surface. Specifically, we train a neural network to gradually infer the relationship between queries and the approximated surface by searching for the moving target of queries in a dynamic way. Meanwhile, we introduce a polygonization algorithm to extract surfaces using the gradients of the learned UDF. We conduct comprehensive experiments in surface reconstruction for point clouds, real scans or depth maps, and further explore our performance in unsupervised point normal estimation, which demonstrate non-trivial improvements of CAP-UDF over the state-of-the-art methods.

Index Terms—surface reconstruction, point clouds, unsigned distance functions, scene reconstruction, normal estimation.

1 Introduction

Reconstructing surfaces from 3D point clouds is vital in 3D vision, robotics and graphics. It bridges the gap between raw point clouds that can be captured by 3D sensors and the editable surfaces for various downstream applications. Recently, Neural Implicit Functions (NIFs) have achieved promising results by training deep networks to learn Signed Distance Functions (SDFs) [1], [2], [3], [4] or occupancies [5], [6], [7], [8]. With the learned NIFs, we can extract a polygon mesh as a continuous iso-surface of a discrete scalar field using the marching cubes algorithm [9]. However, the NIFs approaches based on learning internal and external relations can only reconstruct closed surfaces. The limitation prevents NIFs from representing most real-world objects such as cars with inner structures, clothes with unsealed ends or 3D scenes with open walls and holes.

As a remedy, state-of-the-art methods [10], [11], [12] learn Unsigned Distance Functions (UDFs) as a more general representation to reconstruct surfaces from point clouds. However, these methods can not learn UDFs with smooth distance fields near surfaces, due to the discontinuous character of point clouds, even using ground truth distance values or large scale meshes during training. Moreover, most UDF approaches failed to extract surfaces directly from unsigned distance fields. Particularly, they rely on post-processing such as generating dense point clouds from the learned UDFs for Ball-Pivoting-Algorithm (BPA) [13] to extract surfaces, which is very time-consuming and also leads to surfaces with discontinuity and low quality.

To solve these issues, we propose a novel method named CAP-UDF to learn **C**onsistency-**A**ware UDFs **P**rogressively from raw point clouds. We learn to move 3D queries to reach the approximated surface progressively with a field consistency constraint, and introduce a polygonization algorithm to extract surfaces from the learned UDFs from a new perspective. Our method can learn UDFs from a single point cloud without requiring ground truth unsigned distances, point normals, or a large scale training set. Specifically, given queries sampled in 3D space as input, we learn to move them to the approximated surface according to the predicted unsigned distances and the gradient at the query locations. More appealing solutions [14], [15], [16], [17] have been proposed to learn SDFs from raw point clouds by inferring the relative locations of a query and its closest point in the point cloud. However, since the raw point cloud is a highly discrete approximation of the surface, the closest point of the query is always inaccurate and ambiguous, which makes the network difficult to converge to an accurate UDF due to the inconsistent or even conflicting optimization directions in the distance field.

Therefore, in order to encourage the network to learn a consistency-aware and accurate unsigned distance field, we propose to dynamically search the optimization target with a

- *Junsheng Zhou, Baorui Ma and Shujuan Li are with the School of Software, Tsinghua University, Beijing, China. E-mail: zhoujs21, mbr18, lisj22@mails.tsinghua.edu.cn*
- *Yu-Shen Liu is with the School of Software, Tsinghua University, Beijing, China. E-mail: liuyushen@tsinghua.edu.cn*
- *Yi Fang is with Center for Artificial Intelligence and Robotics, New York University Abu Dhabi, Abu Dhabi, UAE. E-mail: yfang@nyu.edu*
- *Zhizhong Han is with the Department of Computer Science, Wayne State University, USA E-mail: h312h@wayne.edu*

Junsheng Zhou and Baorui Ma contribute equally to this work. The corresponding author is Yu-Shen Liu. *This work was supported by National Key R&D Program of China (2022YFC3800600), and the National Natural Science Foundation of China (62272263, 62072268). Project page: https://junshengzhou.github.io/CAP-UDF.*

specially designed loss function constraining the consistency in the field. We also progressively infer the mapping between queries and the approximated zero iso-surface by using well-moved queries as additional priors for promoting further convergence. To extract a surface in a direct way, we propose to use the gradient of the learned UDFs to determine whether two queries are on the same side of the approximated surface or not. In contrast to NDF [10] which also learns UDFs but outputs dense point clouds for BPA [13] to generate meshes, our method shows great advantages in efficiency and accuracy due to the straightforward surface extraction.

With the ability of learning a consistency-aware and accurate unsigned distance field, we make a step forward and extend our method [18] for unsupervised point normal estimation, where our superior performance is demonstrated by both quantitative and qualitative results with the state-of-the-art unsupervised and supervised normal estimation methods. Furthermore, we evaluate the performance of our method using point clouds from depth sensors and show great advantages over the state-of-the-art NeRF-based [19] or TSDF-based [20], [21] methods using RGB-D images, where we only take depth maps as input without requiring colored images.

Our main contributions can be summarized as follows.

- We propose a novel neural network named CAP-UDF that learns consistent-aware UDFs from raw point clouds without requiring ground truth distance values or point normals. Our method gradually infers the relationship between 3D query locations and the approximated surface with a field consistent loss.
- We introduce an algorithm for directly extracting high-fidelity iso-surfaces with arbitrary topology using the gradients of the learned UDFs.
- We conduct comprehensive experiments in surface reconstruction for synthetic point clouds, real scans or depth maps, and further explore our performance in the unsupervised point normal estimation task. The experimental results demonstrate our significant improvements over the state-of-the-art methods under the widely used benchmarks.

2 Related Works

Surface reconstruction from 3D point clouds has been studied for decades. Classic optimization-based methods [13], [22], [23], [24] tried to resolve this problem by inferring continuous surfaces from the geometry of point clouds. With the rapid development of deep learning [25], [26], [27], [28], [29], [30], [31], [32], [33], the neural networks have shown great potential in reconstructing 3D surfaces [18], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45]. In the following, we will briefly review the studies of deep learning based methods.

2.1 Neural Implicit Surface Reconstruction

In the past few years, a lot of advances have been made in 3D surface reconstruction with Neural Implicit Functions (NIFs). The NIFs approaches [1], [2], [5], [6], [7], [46], [47], [48], [49] use either binary occupancies [5], [6], [7], [8] or signed distance functions (SDFs) [1], [2], [3], [4] to represent 3D shapes or scenes, and then use the marching cubes [9] algorithm to reconstruct surfaces from the learned implicit functions. Earlier studies [1], [5], [8] use an encoder [5], [8] or an optimization based method [1] to embed the shape into a global latent code, and then use a decoder to reconstruct the shape. To obtain more detailed geometry, some methods [2], [7], [46], [50], [51], [52], [53], [54], [55] proposed to leverage more latent codes to capture local shape priors. To achieve this, the point cloud is first split into different uniform grids [2], [46], [53] or local patches [50], [51], [52], and a neural network is then used to extract a latent code for each grid/patch. Some recent methods propose to learn NIFs from a new perspective, such as implicit moving least-squares surfaces [47], differentiable poisson solver [56], iso-points [57] , point convolution [58] or predictive context learning [54]. However, the NIFs approaches can only represent closed shapes due to the characters of occupancies and SDFs.

2.2 Learning Unsigned Distance Functions

To model general shapes with open and multi-layer surfaces, NDF [10] learns unsigned distance functions to represent shapes by predicting an unsigned distance from a query location to the continuous surface. However, NDF merely predicts dense point clouds as the output, which requires time-consuming post-processing for mesh generation and also struggles to retain high-quality details of shapes. In contrast, our method is able to extract surfaces directly from the gradients of the learned UDFs. Following studies use image features [11] or query side relations [59] as additional constraints to improve reconstruction accuracy, some other works advance UDFs for normal estimation [12] or semantic segmentation [60]. However, these methods require ground truth unsigned distances or even a large scale meshes during training, which makes it hard to provide smooth distance fields near the surface due to the discontinuous character of point clouds. While our method does not require any additional supervision but raw point clouds during training, which allows us to reconstruct surfaces for real point cloud scans. In a differential manner, a concurrent work named MeshUDF [61] meshes UDFs from the dynamic gradients during training using a voting schema. On the contrary, we learn a consistency-aware UDF first and extract the surface from stable gradients during testing. Moreover, our surface extraction algorithm is simpler to use, which is implemented based on the marching cube algorithm.

2.3 Surface Reconstruction from Raw Point Clouds

Learning implicit functions directly from raw point clouds without ground truth signed/unsigned distances or occupancy labels is more challenge. Current studies introduce sign agnostic learning with a specially designed network initialization [14], constraints on gradients [16] or geometric regularization [15] for learning SDFs from raw data. Neural-Pull [17] uses a new way of learning SDFs by pulling nearby space onto the surface. However, they aim to learn signed distances and hence can not reconstruct complex shapes with open or multi-layer surfaces. In contrast, our method is able to learn a continuous unsigned distance function from point clouds, which allows us to reconstruct surfaces for shapes and scenes with arbitrary typology.

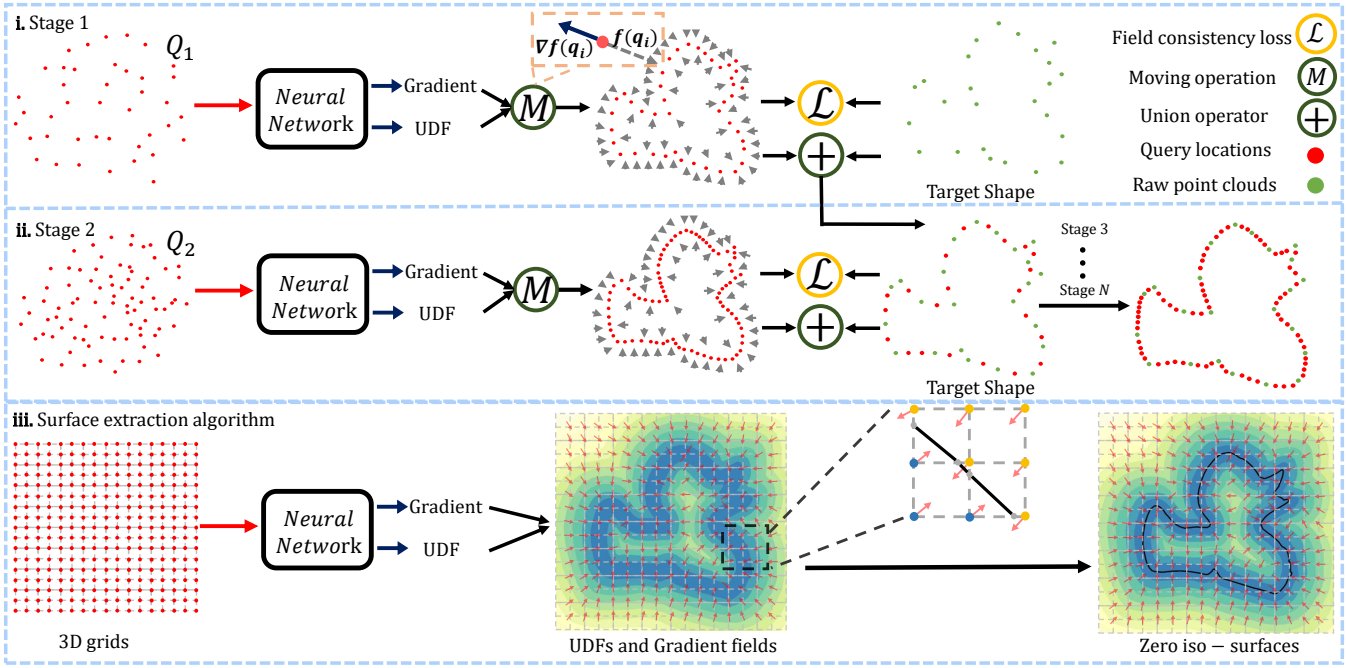

Fig. 1. Overview of CAP-UDF. Given a 3D query $q_i \in Q_1$ as input, the neural network predicts the unsigned distance $f(q_i)$ of q_i and moves q_i against the direction of gradient at q_i with a stride of $f(q_i)$. The field consistency loss is then computed between the moved queries q_i' and the target point cloud P as the optimization target. After the network converges in the current stage, we update P with a subset of q_i' as additional priors to learn more local details in the next stage. Finally, we use the gradients of the learned UDFs in the field to model the relationship between different 3D grids and extract iso-surfaces.

3 METHOD

In this section, we present CAP-UDF, a novel framework to learn consistency-aware UDFs progressively from raw point clouds. We introduce the schema of learning UDFs from raw point clouds in Sec. 3.1 with the consistency-aware field optimization shown in Sec. 3.2. We propose the progressive surface approximation strategy in Sec. 3.3. The gradient-based surface extraction algorithm for UDFs is described in Sec. 3.4. We further extend CAP-UDF to unsupervised point normal estimation in Sec. 3.5. The overview of CAP-UDF is shown in Fig. 1.

Method overview. We design a neural network to learn UDFs that represent 3D shapes and scenes. Given a 3D query location $q = [x, y, z]$, a learned UDF f predicts the unsigned distance value $s = f(q) \in \mathbb{R}$. Current methods rely on ground truth distance values generated from continuous surfaces and employ a neural network to learn f as a regression problem. Different from these methods, we aim to learn f from a raw point cloud $P = \{p_i, i \in [1, N]\}$ without using ground truth unsigned distances. Furthermore, these methods require post-processing [10] or additional supervision [59] to generate meshes. On the contrary, we introduce an algorithm to extract surfaces directly from f using the gradient field ∇f.

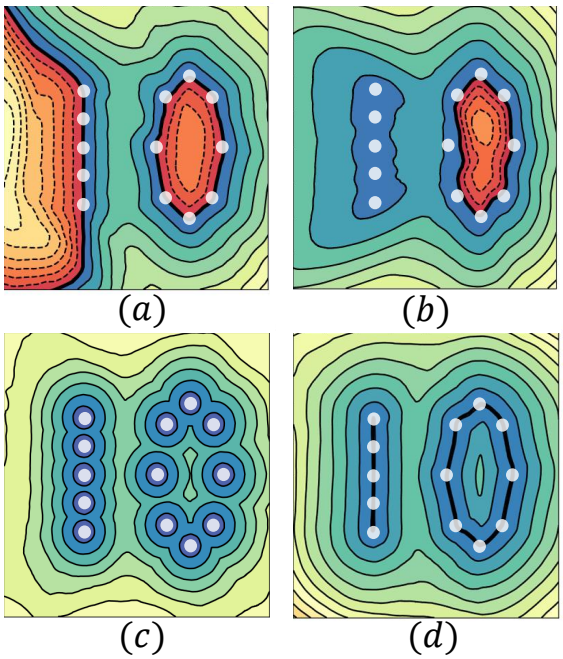

Fig. 2. The level-sets show the distance fields learned by (a) Neural-Pull, (b) SAL, (c) NDF, and (d) Ours. The color of blue and red represent positive and negative distance, respectively. The darker the color, the closer it is to the approximated surface.

3.1 Learn UDFs from Raw Point Clouds

We introduce a novel neural network to learn a continuous UDF f from a raw point cloud. We demonstrate our idea using a 2D point cloud S in Fig. 1(a), where S indicates some discrete points of a continuous surface. Specifically, given a set of query locations $Q = \{q_i, i \in [1, M]\}$ which is randomly sampled around S, the network moves q_i against the direction of the gradient g_i at q_i with a stride of predicted unsigned distance value $f(q_i)$. The gradient g_i is a vector that presents the partial derivative of f at $q_i = [x_i, y_i, z_i]$, which can be formulated as $g_i = \nabla f(q_i) = [\partial f/\partial x, \partial f/\partial y, \partial f/\partial z]$. g_i indicates the direction of the greatest unsigned distance

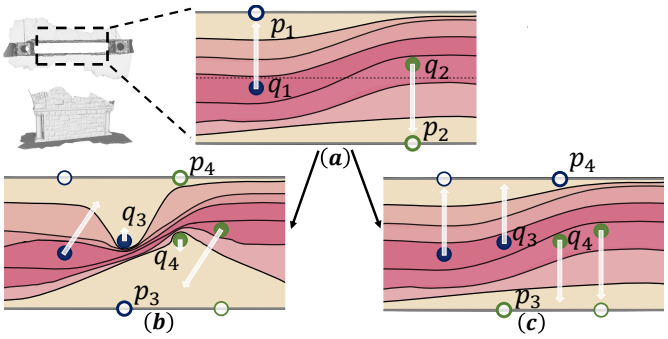

Fig. 3. Illustration of optimizing with different losses. (a) The initial distance field. (b) The distorted field with local minimum caused by inconsistent optimization with naive loss in Eq. (2). (c) Optimizing with our consistency-aware loss in Eq. (3) leads to correct and continuous distance field.

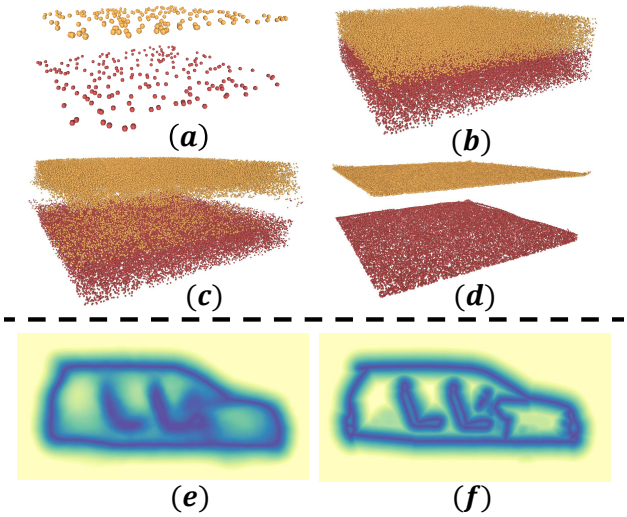

Fig. 4. Demonstration experiments on the effectiveness of consistency-aware loss. (a) The input point cloud of a double-deck wall. (b) The randomly sampled query points between two decks. (c, d) The moved queries optimized by naive loss in Eq. (2) and our loss in Eq. (3). (e,f) The learned distance field of a car with inner structure by the loss in Eq. (2) and our loss in Eq. (3).

change in 3D space, which points the direction away from the surface, therefore moving q_i against the direction of g_i will find a path to the surface of S. The moving operation can be formulated as:

$$z_i = q_i - f(q_i) \times \nabla f(q_i)/||\nabla f(q_i)||_2, \tag{1}$$

where z_i is the location of the moved query q_i, and $\nabla f(q_i)/||\nabla f(q_i)||_2$ is the normalized gradient g_i, which indicates the direction of g_i. The moving operation is differentiable in both the unsigned distance value and the gradient, which allows us to optimize them simultaneously during training.

The four examples in Fig. 2 show the distance fields learned by Neural-Pull [17], SAL [14], NDF [10] and our method for a sparse 2D point cloud P which only contains 13 points. One main branch to learn signed or unsigned distance functions for point clouds is to directly minimize the mean squared error between the predicted distance value $f(q_i)$ and the Euclidean distance between q_i and its nearest neighbour in P, as proposed in NDF and SAL. However, as shown in Fig. 2(c), NDF leads to an extremely discrete distance field. To learn a continuous distance field, NDF introduces ground truth distance values extracted from the continuous surface as extra supervision, which prevents it from learning from raw point clouds. SAL shows a great capacity in learning SDFs for watertight shapes using a carefully designed initialization. However, as shown in Fig. 2(b), SAL fails to converge to a multi-part structure since the network is initialized as a single layer shape prior. Neural-Pull uses a similar way as ours to pull queries onto the surface, thus also learns a continuous signed distance field as shown in Fig. 2(a). However, the nature of SDF prevents Neural-Pull from reconstructing open surfaces like the "1" on the left of Fig. 2(a). As shown in Fig. 2(d), our method can learn a continuous level set of distance field and can also represent open surfaces.

One way to extend Neural-Pull directly to learn UDFs is to predict a positive distance value for each query and pull it to the nearest neighbour in P. However, for shapes with complex topology, this optimization is often ambiguous due to the discontinuous character of raw point clouds. Hence, we resolve this problem by introducing consistency-aware field learning.

3.2 Consistency-Aware Field Learning

Neural-Pull leverages a mean squared error to minimize the distance between the moved query z_i and the nearest neighbour n_i of q_i in P:

$$\mathcal{L} = \frac{1}{M} \sum_{i \in [1,M]} ||z_i - n_i||_2^2. \tag{2}$$

However, the direct optimization of the loss in Eq. (2) will form a distorted field and lead some queries to get stuck in the local areas due to the conflict optimization which makes the network difficult to converge. We show a 2D demonstration of learning UDFs for a double-deck wall using the loss Eq. (2) in Fig. 3(b). Given p_1 and p_2 as two discrete points in two different decks of the wall, q_1 and q_2 are two queries whose closest neighbours are p_1 and p_2, respectively. Optimizing the network using q_1 and q_2 by minimizing Eq. (2) or our proposed loss in Eq. (3) will lead to an unsigned distance field as in Fig. 3(a). Assuming in the next training batch, q_3 and q_4 are two queries whose closest neighbours are p_3 and p_4. If we use the loss in Eq. (2), the optimization target of q_3 is to minimize $\mathcal{L} = ||z_3 - p_3||_2^2$. Notice that the target point p_3 is located on the lower surface, however the opposite direction of gradient around q_3 is upward at this moment. Therefore, the partial derivative $\partial\mathcal{L}/\partial z_3$ leads to a decrease in the unsigned distance value $f(q_3)$ predicted by the network. The case of q_4 is optimized similarly. An immediate consequence is that the inconsistent optimization directions will form a distorted fields that has local minima of unsigned distance values at q_3 and q_4 as in Fig. 3(b). However, this situation causes other query points around point q_3 or q_4 to get stuck in the distorted fields and unable to move to the correct location, thus making the network hard to converge.

To address this issue, we propose a loss function which can keep the consistency of unsigned distance fields to avoid the conflicting optimization directions. Specifically, instead

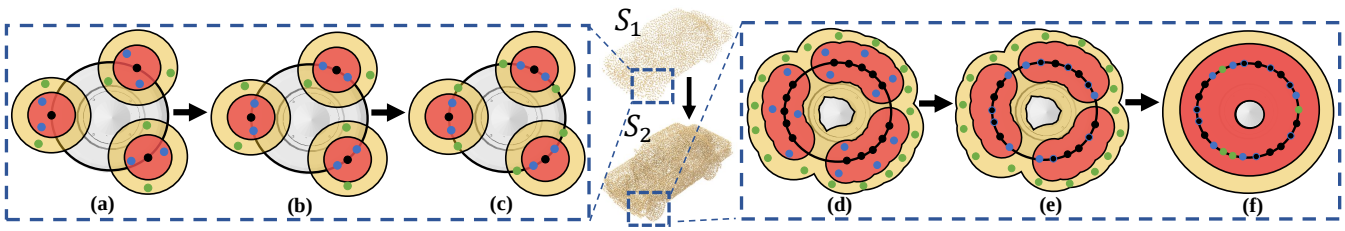

Fig. 5. Illustration of progressively approximating the surface. (a) We sample queries on the high confidence (red) and low confidence (yellow) regions. (b, c) The optimization process in the first stage for learning UDFs by moving queries as Eq. (1). (d) We update the raw points with the moved queries in both regions as additional priors for learning more local details in the next stage. (e,f) More continuous and accurate field is achieved with the progressive learning schema.

of strictly constraining the convergence target before forward propagation as Eq. (2), we first predict the moving path of a query location q_i and move it using Eq. (1) to z_i, then look for the surface point p_i in P which is the closest to z_i and minimize the distance between z_i and p_i. As shown in Fig. 3(c), after moving q_3 against the gradient direction with a stride of $f(q_3)$ to z_3, the closest surface point of z_3 lies on the upper deck, so the distance fields remain continuous and are optimized correctly. In practice, we can achieve this by using the Chamfer distance as a loss function, formulated as:

$$\mathcal{L}_{\text{CD}} = \frac{1}{M}\sum_{i\in[1,M]} \underset{j\in[1,N]}{min} ||z_i - p_j||_2 + \frac{1}{N}\sum_{j\in[1,N]} \underset{i\in[1,M]}{min} ||p_j - z_i||_2, \tag{3}$$

where $z_i, i \in [1, M]$ are the moved queries and $p_j, j \in [1, N]$ indicate the raw point cloud. We also use toy examples as shown in Fig. 4 to show the advantage of our proposed field consistency loss. We learn UDFs for a raw point cloud of a double-deck wall as shown in Fig. 4(a). Fig. 4(b) shows the randomly sampled query locations between the two decks of the wall where the different colors distinguish the queries that are closer to the upper or lower deck of the wall. Fig. 4(c) and Fig. 4(d) indicate the moved queries by loss in Eq. (2) and Eq. (3). It can be seen that our proposed loss can move most of the queries to the correct surface position, while the Neural-Pull loss can not move queries in many places or moves queries to the wrong places due to the field inconsistency in optimization. Fig. 4(e) and Fig. 4(f) show the learned distance field of a car with inner structure by the loss in Eq. (2) and our loss, respectively.

3.3 Progressive Surface Approximation

Moreover, in order to predict unsigned distance values more accurately and learn more local details, we propose a progressive learning strategy by taking the intermediate results of moved queries as additional priors. Given a raw point cloud which is a discrete representation of the surface, we have made a reasonable assumption: the closer the query location is to the given point cloud, the smaller the error of searching the target point on the given point cloud. We provide the proof of this assumption in Sec. 1 of the supplementary. Based on this assumption, we set up two regions: the high confidence region with small errors and the low confidence region with large errors. We sample query points in the high confidence region to help train the network and sample auxiliary points in the low confidence region to move onto the estimated surface by network gradient after network convergence at current stage, where the moved auxiliary points are regarded as the surface prior for the next stage. Notably, the auxiliary points do not participate in network training since these points with low confidence will lead to a large error and affect network training. Since the low confidence regions which are not optimized explicitly during training are distributed interspersed between the high confidence regions, according to the integral Monotone Convergence Theorem [62], the UDFs and gradients predicted by the low confidence region are a smooth expression of the trained high confidence region. We use the moved queries and auxiliary points to update S. According to the updated point cloud, we re-divide the regions with high confidence and low confidence and re-sample the query points and auxiliary points for the next stage.

We demonstrate our idea using a 2D case in Fig. 5.(a) We divided the regions with high confidence (red region) and low confidence (yellow region) based on the given raw point cloud S_1 (black dots) and then sample query points $Q_1 = \{q_i, i \in [1, M]\}$ (blue dots) and auxiliary points (green dots) $A_1 = \{a_i, i \in [1, M]\}$. (b) We train the network to learn UDFs by moving the query locations Q_1 using Eq. (1), and optimize the network by minimizing Eq. (3). (c) After the network convergence at current stage, we move query points Q and auxiliary points A to the estimated surface position by the gradient of network, $S_1^{'} = p - f(p) \times \nabla f(p)/||\nabla f(p)||_2, p \subset Q_1 \cup A_1$. (d) We use the moved points $S_1^{'}$ to update S, $S_2 = S_1 \cup S_1^{'}$. According to the updated point cloud S_2, we re-divided the regions with high confidence and low confidence and re-sampled the query points Q_2 and auxiliary points A_2. (e) We continue to train the network by moving query points Q_2 to the updated S_2, and then update S by combining the moved Q_2 and A_2 with S_2. (f) Because of the more continuous surface, the network can leverage a prior information to learn more accurate UDFs with more local details of shapes.

3.4 Surface Extraction Algorithm

Unlike SDFs, UDFs fail to extract surfaces using the marching cubes since UDFs cannot perform inside/outside tests on 3D grids. To address this issue, we propose to use the gradient field ∇f to determine whether two 3D grid locations are on the same side or the opposite side of the surface approximated by the point clouds P. We make an assumption that the space can be divided into two sides on a micro-scale of the surface, where the 3D query locations

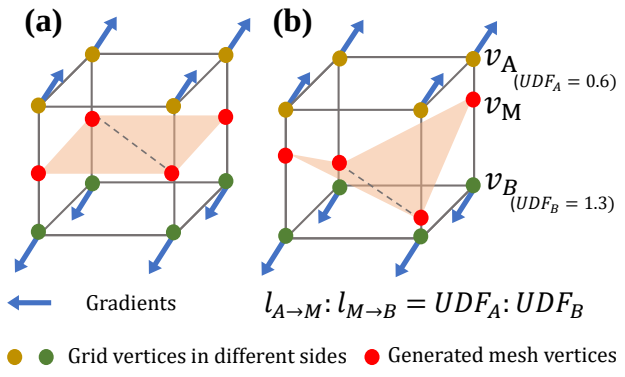

Fig. 6. Surface extraction algorithm.

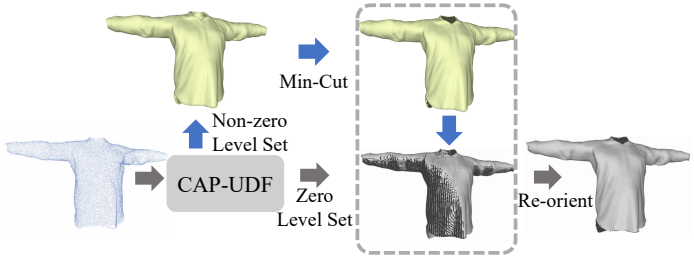

Fig. 7. Surface re-orientation algorithm for the extracted open surfaces. We extract a double-layer closed mesh from the non-zero level set of learned CAP-UDF with marching cubes [9], and cut the mesh with the Min-Cut algorithm [63] to only keep the outer layer surfaces, shown as the yellow meshes. We then leverage the outer layer surfaces as a guidance to re-orient the open surfaces extracted from the zero-level set.

of different sides denoted as $Q_{in} = \{{q_{in}}^i, i \in [1, L]\}$ and $Q_{out} = \{{q_{out}}^i, i \in [1, I]\}$. For two queries ${q_{in}}^i$ and ${q_{out}}^j$ in different sides of the surface, the included angle between the directions of the gradients $\nabla f({q_{in}}^i)$ and $\nabla f({q_{out}}^j)$ are more than 90 degrees, which can be formulated as $\nabla f({q_{in}}^i) \cdot \nabla f({q_{out}}^j) < 0$. On the contrary, for two queries ${q_{in}}^i$ and ${q_{in}}^j$ in the same side, the formula $\nabla f({q_{in}}^i) \cdot \nabla f({q_{in}}^j) > 0$ holds true. So, we can classify whether two points are in the same or the opposite side using dot product of gradients, $cls(q_i, q_j) = \nabla f(q_i) \cdot \nabla f(q_j)$. Based on that, we divide the space into 3D grids (e.g. 256^3), and perform gradient discrimination on the 8 vertices v_i in each cell grid according to $cls(q_i, q_j)$. As shown in Fig. 6(a), the gradient field separates the vertices into two sets, where we can further adapt the marching cubes algorithm [9] to create triangles for the grid using the lookup table. The complete surface is generated by grouping triangles of each grid together. To accelerate the surface extraction process and avoid extracting unexpected triangles in the multi-layers structures, we set a threshold θ to stop surface extraction on grids where $f(g_i) > \theta, i \in [0, 7]$.

Mesh refinement. The initial surface extracted by the gradient-based surface extraction algorithm is only a discrete approximation of the zero iso-surface. To achieve a more detailed mesh, we propose to refine it using the UDF values. As shown in Fig. 6(b), given the predicted UDF values UDF_A and UDF_B of grid vertices v_A and v_b, the mesh vertex v_M can be moved to a finer position where $l_{A\to M} : l_{M\to B} = UDF_A : UDF_B$. The $l_{A\to M}$ and $l_{M\to B}$ indicate the distance from v_A to v_M and the distance from v_M to v_B. In simpler terms, the mesh refinement strategy is to move the mesh vertices on to the zero-level set based on a linear interpolation using the UDF values.

Surface re-orientation. Another issue is that the global consistency of extracted meshes cannot be guaranteed since the surfaces are not closed. This is a common issue for most existing approaches in reconstructing open surfaces from point clouds, such as NDF [10], MeshUDF [61] and GIFS [59]. To solve this issue, we further propose a novel approach to re-direct the surface orientations with the help of non-zero level sets from the learned CAP-UDF. Our insight comes from the observation that although the open surfaces extracted from the zero-level set of learned CAP-UDF are orientation ambiguous, the surfaces extracted from the non-zero level sets with marching cubes algorithm [9] are usually closed and with correct normal orientations. For example, given a non-zero level set where UDF = 0.01, we can use marching cubes algorithm to obtain a double-layer closed mesh with a thickness of 0.02. Driven by this observation, we propose to leverage the closed surfaces from the non-zero level set of CAP-UDF as the guidance to re-direct the normal orientations on the ones extracted from the zero level set. We show the overview of surface re-orientation process in Fig. 7.

Given an input point cloud p, we learn an unsigned distance field from it with the proposed CAP-UDF. The open mesh surfaces S_0 are extracted from the zero level set with our proposed surface extraction algorithm, as described in Fig. 6 and Sec. 3.4. The mesh can be leveraged in some downstream tasks for AR, VR, physical simulation, etc. However, the surface orientation may not be correct due to the open structure, leading to negative affect on the downstream rendering and lighting applications. To solve this issue, we simultaneously extract another mesh S_σ from the non-zero level set of the learned distance field by using the marching cubes algorithm with a non-zero threshold σ (e.g. 0.01). The mesh S_σ is usually a closed surface which has consistent normal orientations. We then cut the double-layer closed mesh S_σ with the Min-Cut algorithm used in DCUDF [63], and keep the outer layer mesh surfaces S_σ^{out} as the guidance to re-direct the normals on the surface from the zero level set. Specifically, for each vertex $\{v_0^i\}_{i=1}^N$ on S_0, we re-direct its normal $\{n_0^i\}_{i=1}^N$ by first searching for its closest vertex v'_σ on S_σ, and then re-direct n_0^i as:

$$n_0^i = \begin{cases} n_0^i, & \text{if } n_0^i \times n'_\sigma > 0 \\ -n_0^i, & \text{if } n_0^i \times n'_\sigma < 0 \end{cases} \tag{4}$$

Note that we do not directly adopt the outer layer surfaces as the final result since they are from a non-zero level set of the learned UDF, which do not represent the actual surfaces.

3.5 Extension to Unsupervised Point Normal Estimation

Normal estimation for unstructured point clouds is to predict a normal for each point in a point cloud. Previous methods [64], [65], [66] achieve encouraging results by using large-scale annotated ground truth normal labels for supervised training, which are hard to collect. We show that our method can also be extended to estimate the normals for point clouds in an unsupervised way. Following most of the previous

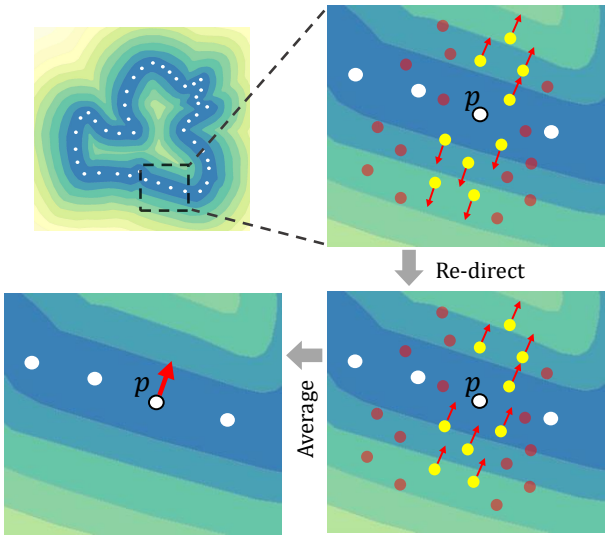

Fig. 8. Illustration of unsupervised point normal estimation. The white points indicates the raw point cloud P. To estimate the normal for point p, we first sample a set of queries near P and search for the subset of queries Q' whose nearest point on P is p (indicated as the yellow points). We then compute gradients at Q' and re-direct them. The normal highlighted as the red arrow at p is extracted by averaging the re-directed gradients.

works [64], [65], [67] on point normal estimation, we focus on estimating unoriented point cloud normals with neural networks. This means that we only care about if the predicted normals are collinear with the GT point normals, rather than the same direction. The globally consistent normal orientation can be further achieve by applying some off-the-shelf normal orientation methods, e.g. ODP [68] as a post-processing procedure on the estimated normals.

A simple implementation is to take the gradient $\nabla f(p)$ at p of the raw point cloud P as the predicted normal. However, it is not differentiable at the zero-level set of UDF, which leads to an unreliable gradient at the surface. We resolve this problem by fusing the gradients of nearby queries. Specifically, given a well-optimized unsigned distance field which is learned from the raw point cloud, we can estimate the normal for each point with a set of query points around it. To extract the normal for a point p in P, we sample a set of queries $Q' = q'_i, i \in [1, K]$, whose nearest point on P is p. Since we learn a continuous unsigned distance field around p, the normal n of p can be approximated as the fusion of the gradients $g'_i = \nabla f(q'_i), i \in [1, K]$ in Q'. A direct implementation is to calculate the average of g'_i, formulated as:

$$n = \frac{1}{K} \sum_{q' \in Q'} \nabla f(q'). \tag{5}$$

However, the simple averaging described above will lead to a large error since $q'_i, i \in [1, K]$ may lie in different side of the surface, which leads to the uncertain sign of g'_i. To solve this issue, we propose to normalize the signs of $g'_i, i \in [1, K]$ before averaging them together. Specifically, we randomly choose one query q'_v in Q' and take the gradient g'_v as the reference to re-direct the other gradients according to the dot product between q'_v and $q'_i, i \in [1, K]$ as below:

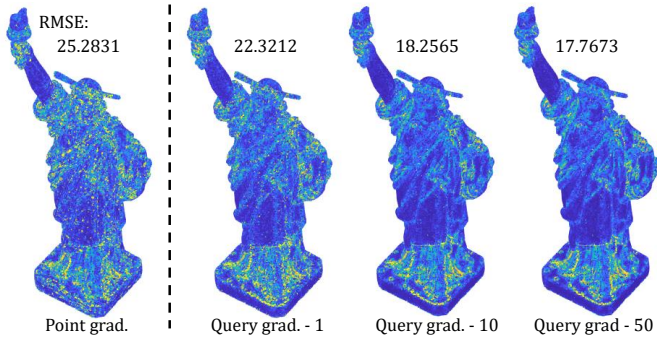

Fig. 9. Errors of normal estimation with different implementations. The "Point grad." means taking the gradient at the raw point cloud as normals and "Query grad. - K" means taking the fusion of gradients at K selected queries as normals. The bluer the color, the smaller the error.

$$\hat{g'_i} = \phi({q'_v}^T \cdot q'_i) g'_i, \tag{6}$$

where ϕ is the sign function with an output in $\{-1, 1\}$.

We illustrate the normal estimation algorithm in Fig. 8. To demonstrate the effectiveness of our proposed query gradient fusion strategy, we further provide the visual comparison with the baseline as shown in Fig. 9. The query sampling strategy is described in the implementation details of Sec. 4. For more experimental comparisons and ablation studies, please refer to Sec. 4.5 and Sec. 4.8.6.

4 Experiments

We evaluate our performance in surface reconstruction from raw point clouds or depth maps, and extend our method to unsupervised point normal estimation. We first demonstrate the ability of our method to reconstruct general shapes with open and multi-layer surfaces in Sec. 4.1. Next, we apply our method to reconstruct surfaces for real scanned raw data including 3D objects in Sec. 4.2 and complex scenes in Sec. 4.3. We then extend our method to reconstruct surfaces from the depth maps in Sec. 4.4 and the point normal estimation task in Sec. 4.5. Ablation studies are shown in Sec. 4.8. Finally, we provide the efficiency analysis and more visualizations in Sec. 4.9 and Sec. 4.10.

Implementation details. To learn UDFs for raw point clouds P, we adopt a neural network similar to OccNet [5] to predict the unsigned distance given 3D queries as input. Our network contains 8 layers of MLP where each layer has 256 nodes. We adopt a skip connection in the fourth layer as employed in DeepSDF [1] and ReLU activation functions in the last two layers of MLP. To make sure the network learns an unsigned distance, we further adopt a non-linear projection $g(x) = |x|$ before the final output.

Similar to Neural-Pull and SAL, given the single point cloud P as input, we do not leverage any condition and overfit the network to approximate the surface of P by minimizing the loss of Eq. (3). Therefore, we do not need to train our network on large scale training dataset in contrast to previous methods [2], [10], [59]. In addition, we use the same strategy as Neural-Pull to sample 60 queries around each point p_i on P as training data. A Gaussian function $\mathcal{N}(\mu, \sigma^2)$ is adopt to calculate the sampling probability where $\mu = p_i$ and σ is the distance between p_i and its 50-th nearest points

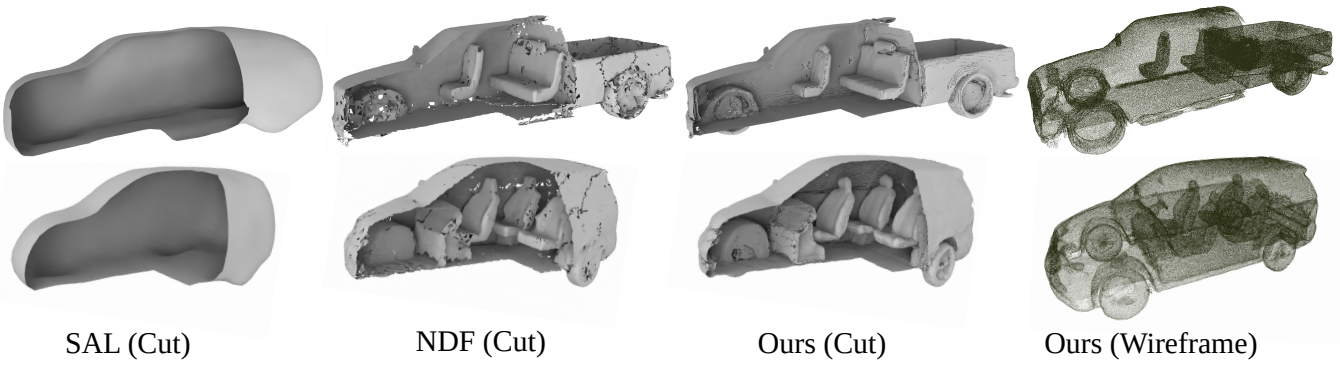

Fig. 10. Visual comparisons of surface reconstruction on ShapeNet cars.

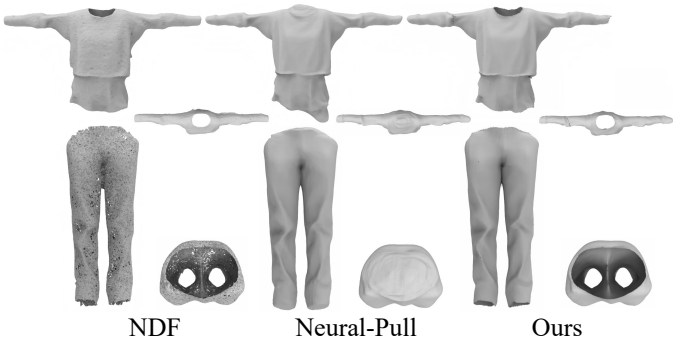

Fig. 11. Visual comparisons of surface reconstruction on MGN dataset.

on P. The queries used to estimate the normals are sampled in the same way as above. For sampling auxiliary points in the low confidence region, the standard deviation is set to 1.1σ. And the size of K for estimating normals in Eq. 5 is set to 50.

During training, we employ the Adam optimizer with an initial learning rate of 0.001 and a cosine learning rate schedule with 1k warn-up iterations. We start the training of next stage after the previous one converges. In experiment, we found the iteration of 40k, 60k and 70k suitable for the change of stages. Since the 3rd and 4th stages bring little advance as shown in Tab. 13 of the ablation studies, we specify the number of stages as two in practical, so the network has been trained in total 60k iterations. For the surface reconstruction of real scanned complex scenes, we increase the iterations to 300k for a better convergence.

4.1 Surface Reconstruction for Synthetic Shapes

Dataset and metrics. For the experiments on synthetic shapes, we follow NDF [10] to choose the "Car" category of the ShapeNet dataset which contains the greatest amount of multi-layer shapes and non-closed shapes. And 10k points is sampled from the surface of each shape as the input. Besides, we employ the MGN dataset [69] to show the advantage of our method in open surfaces. To measure the reconstruction quality, we follow GIFS [59] to sample 100k points from the reconstructed surfaces and adopt the Chamfer distance ($\times 10^4$), Normal Consistency (NC) [5] and F-Score with a threshold of 0.005/0.01 as evaluation metrics.

TABLE 1
Surface reconstruction for point cloud on ShapeNet cars dataset (Chamfer-L2$\times 10^4$).

Method	Chamfer-L2		F-Score	
	Mean	Median	$F1^{0.005}$	$F1^{0.01}$
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$_{BPA}$ [10]	0.202	0.193	77.40	97.97
NDF$_{gradRA}$	0.160	0.152	82.87	99.35
NDF$_{PC}$	0.126	0.120	88.09	99.54
GIFS [59]	0.128	0.123	88.05	99.31
Ours$_{BPA}$	0.141	0.138	84.84	99.33
Ours$_{gradRA}$	**0.119**	**0.114**	**88.55**	**99.82**
Ours$_{PC}$	**0.110**	**0.106**	**90.06**	**99.87**

Comparison. We compare our method with the state-of-the-art works NDF [10] and GIFS [59]. We quantitatively evaluate our method with NDF and GIFS in Tab. 1. We also report the results of points sampled from the watertight ground truth (watertight GT in table) as the upper bound of the traditional SDF-based or Occupancy-based implicit functions. To show the superior limit of this dataset, we sample two different sets of points from the ground truth mesh and report their results (GT in table). For a comprehensive comparison with NDF, we transfer our gradient-based reconstruction algorithm to extract surfaces from the learned distance field of NDF, and report three metrics of NDF and our method including generated point cloud ($*_{PC}$), mesh generated using BPA ($*_{BPA}$) and mesh generated using our gradient-based reconstruction algorithm ($*_{gradRA}$). As shown in Tab. 1, we achieve the best results in terms of all the metrics. Moreover, our gradient-based reconstruction algorithm shows great generality in transferring to the learned gradient field of other method (e.g. NDF) by achieving significant improvement over the traditional method (BPA). We also provide the results of surface reconstruction on MGN [69] dataset as shown in Tab. 2, where we significantly outperform other methods.

We further present a visual comparison with SAL and NDF in Fig. 10. Previous methods (e.g. SAL) take SDF as output and are therefore limited to single-layer shapes where the inner-structure is lost. NDF learns UDFs and is able to

TABLE 2
Surface reconstruction for point cloud on MGN dataset.

Method	Chamfer-L2	F-Score$^{0.01}$	NC
Neural-Pull [17]	4.447	94.49	91.83
NDF [10]	0.658	76.11	92.84
Ours	**0.117**	**99.68**	**97.80**

represent general shapes, but it outputs a dense point cloud and requires BPA to generate meshes, which leads to an uneven surface. On the contrary, we can extract surfaces directly from the learned UDFs, which are continuous surfaces with high fidelity. We also provide a visual comparison with Neural-Pull in MGN dataset as shown in Fig. 11, where we accurately reconstruct the open surfaces but Neural-Pull fails to reveal the original geometry.

4.2 Surface Reconstruction for Real Scans

Dataset and metrics. For surface reconstruction of real point cloud scans, we follow SAP to evaluate our methods under the Surface Reconstruction Benchmarks (SRB) [70]. We use Chamfer distance and F-Score with a threshold of 1% for evaluation. Note that the ground truth is dense point clouds.
Comparison. We compare our method with state-of-the-art classic and data-driven surface reconstruction methods in the real scanned SRB dataset, including IGR [15], Point2Mesh [71], Screened Poisson Surface Reconstruction (SPSR) [24], Shape As Points (SAP) [56], Neural-Pull [17] and NDF [10]. The numerical comparison is shown in Tab. 3, where we achieve the best accuracy. The visual comparisons in Fig. 12 demonstrate that our method is able to reconstruct a continuous surface with local geometry consistence while other methods struggle to reveal the geometry details. For example, IGR, Neural-Pull and SAP mistakenly mended or failed to reconstruct the hole of the anchor while our method is able to keep the correct geometry.

TABLE 3
Surface reconstruction for point cloud on SRB dataset.

Method	Chamfer-L1	F-Score
IGR [15]	0.178	75.5
Point2Mesh [71]	0.116	64.8
SPSR [24]	0.232	73.5
SAP [56]	0.076	83.0
Neural-Pull [17]	0.106	79.7
NDF$_{PC}$ [10]	0.185	72.2
NDF$_{mesh}$	0.238	68.6
Ours$_{PC}$	**0.068**	**90.4**
Ours$_{mesh}$	**0.073**	**84.5**

4.3 Surface Reconstruction for Scenes

Dataset and metrics. To further demonstrate the advantage of our method in surface reconstruction of real scene scans, we follow OnSurf [55] to conduct experiments under the 3D Scene dataset [72]. Note that the 3D Scene dataset is a challenging real-world dataset with complex topology and noisy open surfaces. We uniformly sample 100, 500 and 1000 points per m^2 at the original scale of scenes as the input and follow OnSurf to sample 1M points on both the reconstructed and the ground truth surfaces for evaluation. We leverage L1 and L2 Chamfer distance to evaluate the reconstruction quality.
Comparison. We compare our method with the state-of-the-arts scene reconstruction methods ConvONet [6], LIG [2], DeepLS [46], OnSurf [55] and NDF [10]. The numerical comparisons in Tab. 4 show that our method significantly outperform the other methods under different point densities. The visual comparisons in Fig. 13 further shows that our reconstructions present more geometry details in complex real scene scans. Note that all the other methods have been trained in a large scale dataset, from which they gain additional prior information. On the contrary, our method does not leverage any additional priors or large scale training datasets, and learns to reconstruct surfaces directly from the raw point cloud, but still yields a non-trivial performance.

TABLE 4
Surface Reconstruction under 3D Scene, L2CD$\times$1000.

Method	$100/m^2$		$500/m^2$		$1000/m^2$	
	L2CD	L1CD	L2CD	L1CD	L2CD	L1CD
ConvONet [6]	7.859	0.043	13.192	0.052	14.097	0.052
LIG [15]	6.265	0.049	5.633	0.048	6.190	0.048
DeepLS [46]	3.029	0.044	6.794	0.050	1.607	0.025
NDF$_{PC}$ [10]	0.409	0.012	0.377	0.014	0.561	0.017
NDF$_{mesh}$	0.452	0.014	0.475	0.016	0.872	0.022
OnSurf [55]	1.154	0.021	0.862	0.020	0.706	0.020
Ours$_{PC}$	**0.144**	**0.010**	**0.078**	**0.009**	**0.072**	**0.010**
Ours$_{mesh}$	**0.187**	**0.011**	**0.122**	**0.010**	**0.121**	**0.009**

4.4 Surface Reconstruction from Depth Maps

Dataset and metrics. For surface reconstruction from depth maps, we follow NeuralRGB-D [20] to evaluate our method under the 10 synthetic scenes dataset. To measure the reconstruction quality, we follow NeuralRGB-D and Go-Surf [21] to sample point clouds at a density of 1 point per cm^2 and adopt the Chamfer distance, Normal Consistency(NC) and F-Score with a threshold of 5cm as evaluation metrics.
Comparison. We compare our method with several state-of-the-art traditional or learning-based methods: BundleFusion [73], RoutedFusion [74], COLMAP [75] with Possion Reconstruction (PR) [24], NeRF [19] with additional depth supervision, Convolutional Occupancy Networks (COcc) [6], SIREN [76], Neural-Pull [17], NeuralRGB-D [20] and Go-Surf [21]. For a comprehensive comparison, we conduct the scene fusion experiments under the "Clean" setting and the "Noise" setting. For the "Clean" setting where the ground truth camera poses and clean depth maps are known, we simply back-project the depth maps into world space with the camera poses and fuse them together to achieve a global point cloud, and leverage our model to reconstruct surfaces. To further evaluate the ability of our method to handle noises in the real world situation, we follow NeuralRGB-D to apply noise and artifacts to the depth maps to simulate a real depth sensor, which is the "Noise" setting. And the ground truth camera poses are also unavailable in the "Noise" setting, where we adopt the estimated poses from Go-Surf as the initialization.

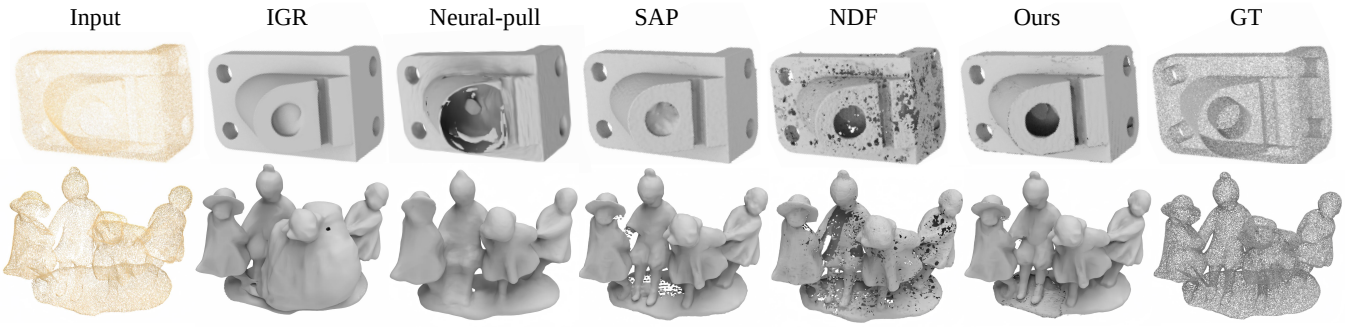

Fig. 12. Visual comparisons of surface reconstruction on the SRB dataset.

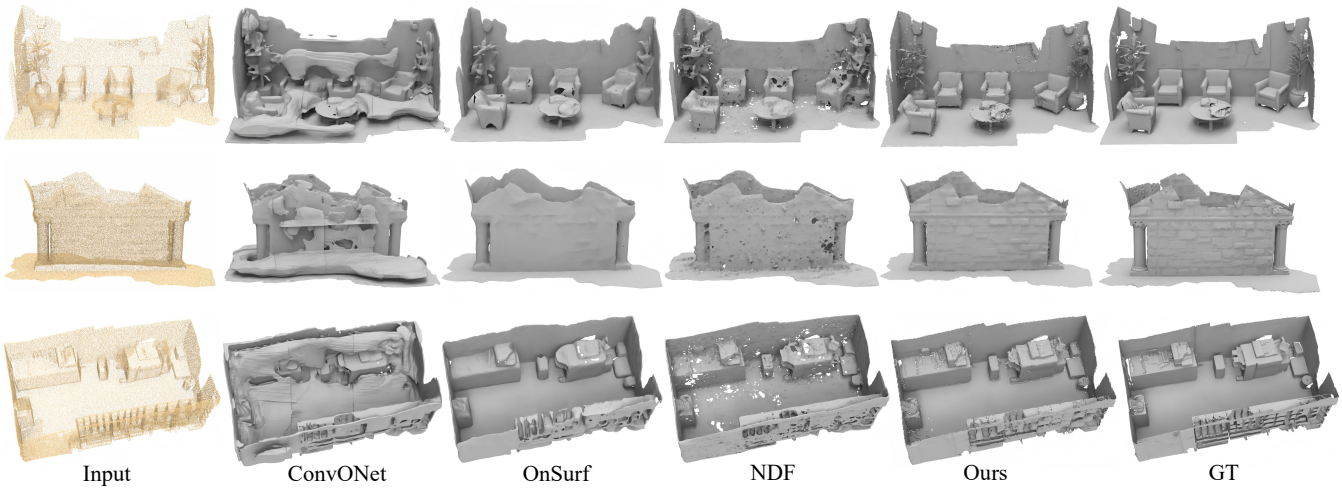

Fig. 13. Visual comparison with different methods under 3D Scene. Inputs contains 1k points/m^2.

The numerical comparison in Tab. 5 shows that our method significantly outperform the other depth-only methods. We also achieve comparable performance with the state-of-the-art RGBD-based methods NeuralRGB-D and Go-Surf, where both depth maps and colored images are required as inputs. We further present a visual comparison under the "Clean" setting with COcc, Neural-Pull and NeuralRGB-D in Fig. 14. Previous methods takes occupancy (COcc) or SDF (Neural-Pull and NeuralRGB-D) as the scene representation and is limited to the closed geometries, while our method can reconstruct surfaces with arbitrary architecture (e.g. open windows and thin table legs) and also reveal the geometry details.

TABLE 5
Surface reconstruction for depth maps on 10 synthetic scenes dataset.

	Method	Input	L1CD	NC	F-Score
Noise	BundleFusion [73]	RGB-D	0.062	0.892	0.805
	COLMAP [75]+PR [24]	RGB-D	0.057	0.901	0.839
	NeRF+Depth [19]	RGB-D	0.065	0.768	0.782
	NeuralRGB-D [20]	RGB-D	0.044	0.918	0.924
	RoutedFusion [74]	Depth	0.057	0.864	0.838
	COcc [6]	Depth	0.077	0.849	0.643
	SIREN [76]	Depth	0.060	**0.893**	0.816
	Ours	Depth	**0.052**	0.880	**0.854**
Clean	NeuralRGB-D [20]	RGB-D	0.0134	0.943	0.985
	Go-Surf [21]	RGB-D	0.0102	0.942	0.983
	COcc [6]	Depth	0.0943	0.860	0.693
	Neural-Pull [17]	Depth	0.0821	0.884	0.761
	Ours	Depth	**0.0081**	**0.934**	**0.987**

4.5 Point Normal Estimation

Dataset and metrics. For the point cloud normal estimation task, we adopt the widely-used benchmark PCPNet [77] dataset to evaluate our method. PCPNet samples 100k points on the mesh of each shape to obtain a point cloud. In addition to the "Clean" data which is sampled uniformly, two additional data with varying density settings (Stripes and Gradients) are added to evaluate the ability of different methods in handling irregular data. We aim at learning unsigned distance fields for unsupervised point cloud normal estimation, thus only the test set of PCPNet dataset is used. We adopt the angle root mean square error (RMSE) between the predicted normals and the ground truth normals as the metric to evaluate the performance of our method. Following previous works [64], [65], we compute the final result with a 5000 points subset for each shape.

Comparison. We conduct a comparison with traditional normal estimation methods including PCA [78], Jets [79] and the state-of-the-art learning-based methods PCPNet [77], HoughCNN [80], Nesti-Net [67], Iter-Net [81] and DeepFit [64]. Note that all the previous learning-based methods are

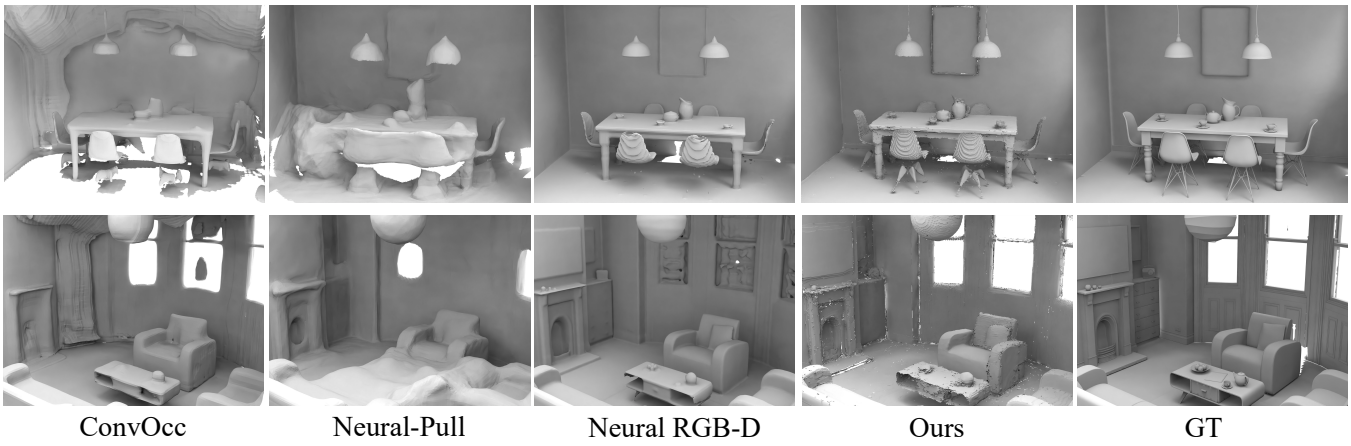

Fig. 14. Visual comparison of surface reconstruction from depth maps on the 10 synthetic scenes dataset.

TABLE 6
RMSE of point normal estimation on the PCPNet dataset (lower is better).

	Method	Clean	Strip	Gradient	average
Supervised	PCPNet [77]	9.66	11.47	13.42	11.61
	Hough [80]	10.23	12.47	11.02	11.24
	Nesti-Net [67]	6.99	8.47	9.00	8.15
	IterNet [81]	6.72	7.73	7.51	7.32
	DeepFit [64]	6.51	7.92	7.31	7.25
Unsupervised	Jet [79]	12.23	13.39	13.13	12.92
	PCA [78]	12.29	13.66	12.81	12.92
	Ours	**6.33**	**7.26**	**7.11**	**6.90**

TABLE 7
RMSE of oriented point normal estimation on the PCPNet dataset (lower is better).

	Method	Clean	Strip	Gradient	average
Supervised	PCPNet * [77]	33.34	37.95	35.44	35.58
	Nesti-Net [67] + ODP	28.87	32.07	22.27	27.73
	DeepFit [64] + ODP	26.60	26.74	**20.96**	24.77
Unsupervised	Jet [79] + ODP	28.08	23.86	23.96	25.30
	PCA [78] + ODP	28.96	28.70	23.00	28.89
	Ours + ODP	**26.47**	**23.64**	21.55	**23.89**

designed in a supervised manner where a large scale dataset is required for training. On the contrary, we resolve the problem from a new perspective, where the normals can be extracted directly from the learned unsigned distance fields in an unsupervised manner. The quantitative comparison in Tab. 6 shows that our method significantly outperform the traditional methods which also do not require ground truth normals for supervision. Meanwhile, our method also achieves a comparable or even better performance than the state-of-the-art supervised normal estimation methods which demonstrates that the expensive annotated normal labels are not necessary for point normal estimation task.

We further provide a visual comparison with the unsupervised methods PCA, Jets and the supervised methods PCPNet, Nesti-Net and DeepFit in Fig. 15. The color of the shape indicates errors, where the closer to yellow the larger the error and the closer to blue the smaller the error. As shown, our estimation results are more accurate and detailed compared to other methods, especially on the complex geometries such as sharp edges and corners.

Oriented Normal Comparison. We further conduct experiments on evaluating the oriented normals, which are achieved by applying the off-the-shelf normal orientation method ODP [68] as a post-processing procedure on the unoriented normals estimated by different methods. The numerical comparison is shown in Tab. 7, where CAP-UDF still achieves better oriented normal estimation results compared to the unsupervised and even some supervised methods. Note that PCPNet [77] can be trained to estimate oriented normals directly, where the result is shown as PCPNet*. All other methods estimated unoriented normals from point clouds, followed by ODP to estimate the orientation for each point normal.

The supervised method PCPNet* which directly estimates oriented normals from point clouds produces much worse results compared to all other methods which estimate unoriented normals with a post-processing procedure for the orientation consistency. The results demonstrate the necessity of decomposing oriented normal estimation task into the two subtasks to estimate the unoriented normals and then produce global consistent orientations. While we focus on the former one for fair comparisons with most of the previous methods.

4.6 Surface Reconstruction form Corrupted Data

The real world point clouds are often sparse or corrupted with noises and occlusions. To further apply CAP-UDF to reconstruct the corrupted point clouds, we propose to introduce priors from SoTA point cloud processing approaches. Our idea is to first improve the quality of point clouds, based on which we learn distance fields for reconstruction. To demonstrate the effectiveness of introducing off-the-shelf point cloud processing approaches as priors, we conduct comprehensive experiments on different datasets to evaluate the performance of CAP-UDF in reconstructing corrupted point clouds.

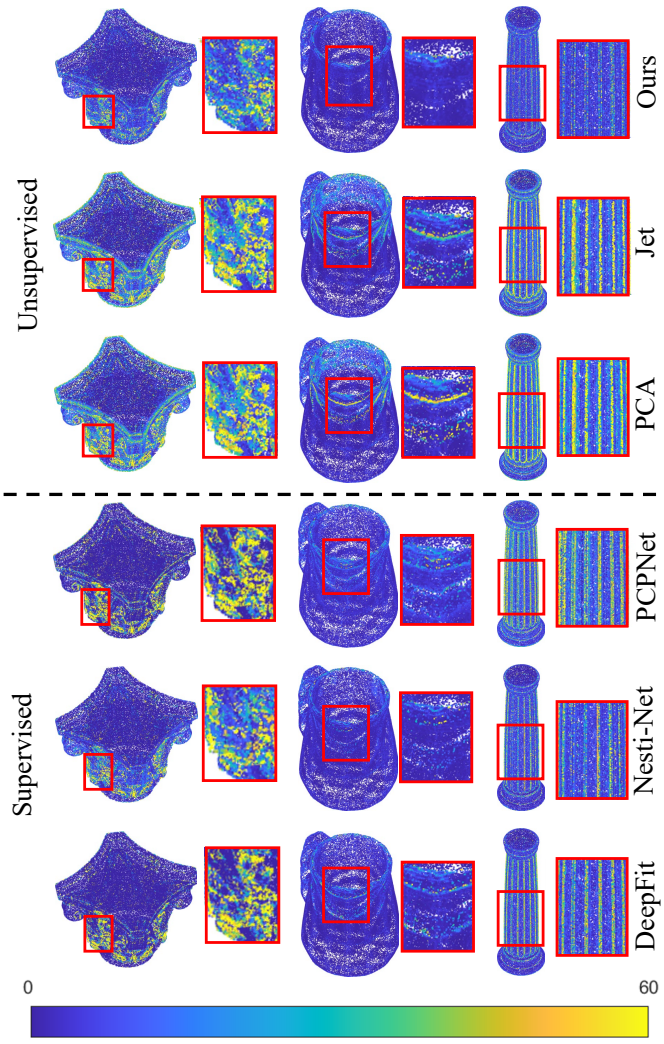

Fig. 15. Visual comparison on the error maps of point normal estimation under the PCPNet dataset.

TABLE 8
Surface reconstruction for occluded point clouds on PCN dataset (Chamfer-L1 $\times 10^3$).

Method	Chamfer-L1 Mean	Chamfer-L1 Median	F-Score $F1^{0.005}$	F-Score $F1^{0.01}$
ConvOcc [6]	47.51	37.75	15.74	38.66
POCO [58]	50.78	44.93	12.03	20.52
ConvOcc [6] + Prior	11.46	10.53	27.12	66.12
POCO [58] + Prior	15.52	13.09	29.24	61.89
Ours	**10.42**	**8.36**	**35.63**	**73.27**

4.6.1 Occluded point cloud surface reconstruction

We conduct experiments under the widely-used PCN [82] dataset for evaluating the reconstruction performances from occluded point clouds. We introduce the pretrained AdaPoinTr [83] as the point cloud completion prior and reconstruct surfaces from the completed point clouds.

The quantitative comparison on reconstructing occluded point clouds is shown in Tab. 8. As shown, the SoTA generalized methods (e.g. ConvOcc [6], POCO [58]) fail to produce complete reconstruction from the corrupted point clouds. The reason is that although they are trained under large and diverse datasets, they do not learn the completion knowledge, leading to a failure in generalizing to occluded shapes. While we believe that introducing expert point cloud completion methods as the prior is a more effective way to handle corrupted point clouds. As shown in the lower part of Tab. 8, the convincing results are achieved by introducing AdaPoinTr as the prior to complete the corrupted point clouds before feeding into the reconstruction frameworks. With the same completion prior, CAP-UDF achieves the best performance compared to the generalized methods ConvOcc and POCO, highlighting the superiority of CAP-UDF in faithfully reconstructing the completed point clouds. Please refer to Sec. 2.1 of the supplementary for visual comparisons on reconstructing occluded point clouds.

TABLE 9
Surface reconstruction for noisy point clouds on PUNet dataset (Chamfer-L1 $\times 10^3$).

Method	Chamfer-L1 Mean	Chamfer-L1 Median	F-Score $F1^{0.005}$	F-Score $F1^{0.01}$
ConvOcc [6]	22.02	20.09	5.74	32.72
POCO [58]	19.56	19.16	7.07	38.36
ConvOcc [6] + Prior	16.01	13.89	13.78	58.85
POCO [58] + Prior	12.38	11.04	23.11	74.79
Ours	**11.36**	**10.64**	**25.49**	**77.87**

4.6.2 Noisy point cloud surface reconstruction

We conduct experiments under the widely-used PUNet [84] dataset with 2% gaussian noises for evaluating the reconstruction performances from noisy point clouds. We introduce the pretrained IterativePFN [85] as the point cloud denoising prior and reconstruct surfaces from the denoised point clouds.

The quantitative comparison on reconstructing noisy point clouds is shown in Tab. 9, where the SoTA generalized results also fails to produce robust reconstruction from the noisy point clouds, demonstrating that training under large datasets in a generalized way still does not generalize well to unseen point noises. By introducing expert point cloud denoising methods as a prior to clean the corrupted point clouds, more accurate reconstructions can be achieved. CAP-UDF demonstrates superior performance in faithfully reconstructing cleaned point clouds when compared to the generalized methods ConvOcc [6] and POCO [58], all using the same denoising prior. Please refer to Sec. 2.1 of the supplementary for visual comparisons on reconstructing noisy point clouds.

4.6.3 Sparse point cloud surface reconstruction

For evaluating the reconstruction performances from sparse point clouds, we conduct experiments under the widely-used PU-GAN [86] dataset with extremely sparse point clouds containing only 256 or 512 points per shape. We introduce the pretrained APU-LDI [87] as the point cloud upsampling prior and reconstruct surfaces from the upsampled point clouds.

In Tab. 10 and Tab. 11, we show the quantitative comparison on reconstructing sparse point clouds with settings

TABLE 10
Surface reconstruction for sparse point clouds from PUGAN dataset where each shape only contains 256 points (Chamfer-L1 $\times 10^3$).

Method	Chamfer-L1		F-Score	
	Mean	Median	$F1^{0.005}$	$F1^{0.01}$
ConvOcc [6]	56.52	39.91	11.37	37.96
POCO [58]	61.91	57.35	1.83	5.45
ConvOcc [6] + Prior	13.66	13.60	18.93	66.25
POCO [58] + Prior	10.36	10.11	33.12	82.03
Ours	**9.44**	**9.24**	**39.95**	**86.18**

TABLE 11
Surface reconstruction for sparse point clouds from PUGAN dataset where each shape only contains 512 points (Chamfer-L1 $\times 10^3$).

Method	Chamfer-L1		F-Score	
	Mean	Median	$F1^{0.005}$	$F1^{0.01}$
ConvOcc [6]	16.30	16.81	16.29	57.05
POCO [58]	78.77	72.35	1.72	5.18
ConvOcc [6] + Prior	13.73	14.02	18.60	66.03
POCO [58] + Prior	10.29	10.02	34.60	83.36
Ours	**9.33**	**9.14**	**40.81**	**86.66**

of 256 points and 512 points, respectively. The results demonstrate that the SoTA generalized methods ConvOcc and POCO struggle to produce robust reconstruction from the extremely sparse point clouds. By introducing expert point cloud upsampling methods as a prior to increase point densities, more accurate reconstructions can be achieved. CAP-UDF demonstrates superior performance compared to the generalized methods ConvOcc and POCO when utilizing the same upsampling prior, emphasizing its proficiency in faithfully reconstructing the point clouds enhanced by the prior. Please refer to Sec. 2.1 of the supplementary for visual comparisons on reconstructing sparse point clouds.

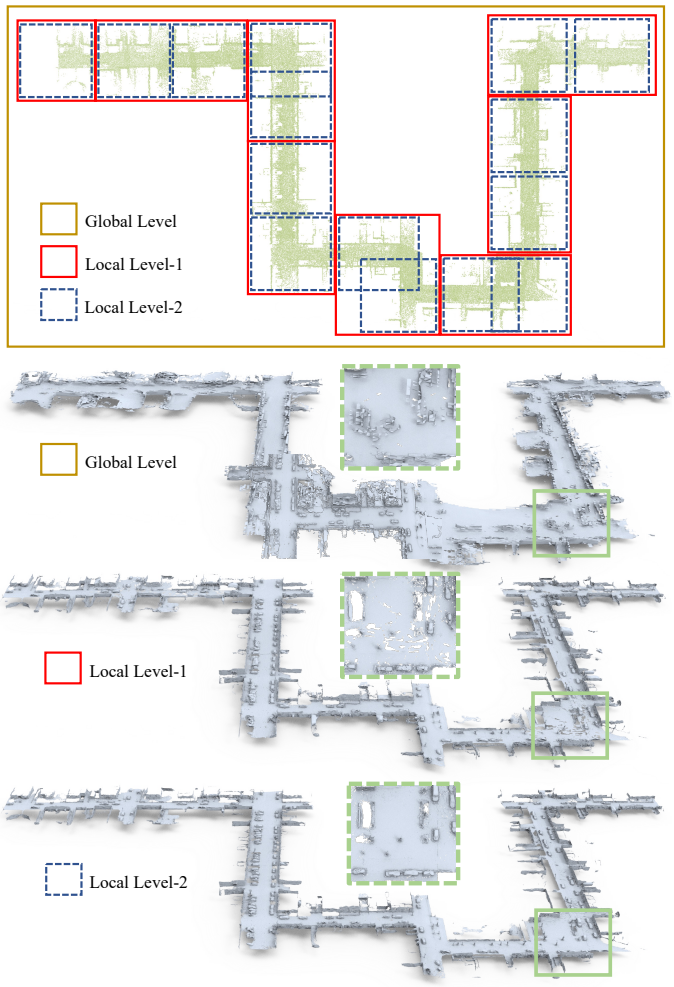

Fig. 16. Surface reconstruction results on the KITTI odometry dataset under different scale levels.

4.7 Large-Scale Scene Reconstruction

Representing the surfaces for large point clouds using only one single neural network may struggle to produce high-fidelity reconstructions, due to the catastrophic forgetting of neural networks. To overcome this challenge, we introduce a slide-window strategy to reconstruct extremely large-scale scenes by splitting scenes into local chunks where each local chunk is represented with a specific neural network. The final reconstruction is achieved by fusing the local scenes.

We conduct experiments under the large-scale driving scenes from the KITTI [88] dataset. We use the LiDAR scans of frame 3000 to 4000 in Squeece00 subset of KITTI dataset pre-processed by NGS [89] as the full input, which are transformed into world coordinates using the provided camera trajectories. The results of directly reconstructing the full scene are shown in the "Global Level" of Fig. 16, where only the overall outline of the scene is reconstructed and no geometric details are preserved, due to the catastrophic forgetting of neural networks, e.g., optimizing one local region results in degradation of other regions.

To overcome the limits of large-scale scene point cloud reconstruction with single neural network, we introduce a slide-window strategy to split the extremely large scene into local chunks at different scale levels and obtain the final scene reconstruction by fusing the reconstructed local geometries together. We reconstruct the scene under two local levels to split the full scene into 8 local trunks (Local Level-1) and 15 local trunks (Local Level-2). The size for local trunks in Local Level-2 is 51.2 m^3, and the trunks in Local Level-1 are achieved by merging two adjacent trunks in Local-Level-2. The results are shown in "Local-Level 1 / Local-Level 2" of Fig. 16, where more subdivided local chunks lead to more detailed local geometries with higher qualities. We find that the Local Level-2 already produces convincing reconstructions, which can be considered as the "maximum" size before seeing performance degrade. And splitting scenes into more local trunks may cause space and time complexity for representing the full scene.

4.8 Ablation Study

We conduct ablation studies to justify the effectiveness of each design in our method and the effect of some important parameters. We report the performance in terms of L2-CD under a subset of the ShapeNet Car dataset. By default, all the experimental settings are kept the same as in Sec. 4.1, except for modified part described in each ablation experiment below.

4.8.1 Framework design

We first justify the effectiveness of each design of our framework in Tab. 12. We first directly use the loss proposed in Neural-Pull and find that the performance degenerates dramatically as shown by "NP loss". We also use $g(x) = 1 - e^{(-x)}$ to replace $g(x) = |x|$ on the last layer of the network for f before output, but found no improvement as shown by "Exponent". We train the second stage from scratch as shown by "Scratch" and prove that an end-to-end training strategy is more effective.

TABLE 12
Effect of framework design.

$\times 10^4$	NP loss	Exponent	Scratch	Ours
L2CD	0.2381	0.1218	0.1497	**0.1112**

4.8.2 The effect of stage numbers

The number of stages during progressive surface approximation is also a crucial factor in the network training. We report the performance of training our network in different number of stages $St = [1, 2, 3, 4]$ in Tab. 13. We start the training of next stage after the previous one converges. We found that two stages training brings great improvement than training a single stage, and the improvements with 3rd and 4th stages are subtle. Therefore, we train CAP-UDF with two stages in practice.

TABLE 13
Effect of stage numbers.

$\times 10^4$	1	2	3	4
L2CD	0.1218	0.1112	0.1107	0.1107

4.8.3 The effect of low confidence range

We further explore the range of confidence region sample. Assume σ as the range of high confidence region, we use 0.9σ, 1.0σ, 1.1σ and 1.2σ as the range of low confidence region. The results in Tab. 14 show that a too small or too large range will degenerate the performance.

TABLE 14
Effect of low confidence range.

$\times 10^4$	0.9	1.0	1.1	1.2
L2CD	0.1133	0.1130	**0.1112**	0.1131

4.8.4 Surface extraction

We evaluate the effect of mesh refinement and the performance of different 3D grid resolutions. Tab. 16 shows the accuracy and efficiency of different resolutions. We observe that the mesh refinement highly improves the accuracy and higher resolutions leads to better reconstructions at a cost of speed.

We provide the visualizations of the extracted surfaces with different resolutions as shown in Fig. 4.8.4. It shows that a higher resolution leads to a more detailed reconstruction. Moreover, our designed mesh refinement operation brings great improvement in surface smoothness and local details due to the accurate unsigned distance values predicted by the neural network.

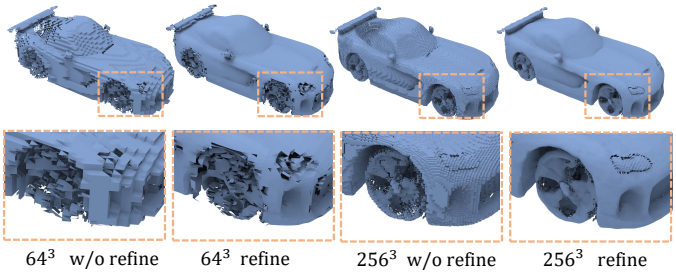

TABLE 15
Visualizations of extracted mesh with different settings.

$\times 10^4$	64^3	128^3	256^3	320^3
w/o refine	0.4169	0.1738	0.1294	0.1238
refine	0.1606	0.1174	**0.1112**	0.1105
Time	3.0 s	21.9 s	162.2 s	307.6 s

TABLE 16
Ablations on surface extraction.

4.8.5 The effect of training iterations

We further test the effect of training iterations of the first and second stage. Since we use an end-to-end training strategy, we set a relatively small number of training iterations for the second stage. In experiments, we set the numbers of iteration in the first stage to 30k, 40k and 50k, and the second stage to 15k, 20k and 25k. For testing the effect of the iteration numbers in the first stage, we set the iteration numbers of the second stage to the default 20k, and the iteration numbers of the first stage is set to the default 40k while testing the second stage. The results in Tab. 17 show that too few training iterations will lead to an under fitting problem and too many training iterations will result in network degeneration.

TABLE 17
Effect on training iterations of different stages.

Stage1	30k	40k	50k
Chamfer-L2	0.1158	**0.1112**	0.1137
Stage2	15k	20k	25k
Chamfer-L2	0.1125	**0.1112**	0.1133

4.8.6 The settings of normal estimation

We evaluate the effect of our designs for unsupervised point normal estimation in Tab. 18. We first directly use the gradients at the location of raw point cloud as the estimated normal and find that the performance degenerates dramatically as shown by "Point grad.". The reason is that the normals are ambiguous at the zero-level set of UDFs, where we solve this problem by estimating the normals as the fusion of gradients at the queries sampled nearby. We use $K = [1, 5, 10, 20, 50, 100]$ as the number of queries for estimating normals and find that a too small or too large K will degenerate the performance.

TABLE 18
Ablations on point normal estimation.

	K	Clean	Strip	Gradient	average
Point grad.	-	7.30	7.74	7.78	7.61
Query grad.	1	8.51	9.48	9.40	9.13
	5	6.80	7.80	7.55	7.38
	10	6.59	7.53	7.31	7.14
	20	6.42	7.39	7.19	7.00
	50	6.33	**7.26**	**7.11**	**6.90**
	100	**6.32**	7.27	7.14	6.91

4.9 Efficiency Analysis

We analyse the efficiency of our proposed method by comparing the computational cost of field learning and mesh extraction with the state-of-the-art methods.

4.9.1 Efficiency comparison on field learning

We make a comparison with Neural-Pull [17], IGR [15], Point2mesh [71] on the computational cost of optimizing for a single point cloud in Table 19. The results show that our proposed method converges faster than all the baselines with fewer memory requirements.

TABLE 19
Efficiency comparison on field learning.

Methods	Neural-Pull	IGR	Point2mesh	Ours
Time	1150 s	1212 s	4028 s	**667 s**
Memory	2.2 GB	6.1 GB	5.2 GB	**2.0 GB**

4.9.2 Efficiency comparison on mesh extraction

We further evaluate the efficiency of our method. The comparison is shown in Tab. 20. For reproducing the mesh generation process of NDF [10], we use the default 1×10^6 points and the parameters provided by NDF to generate a mesh with ball-pivoting-algorithm (BPA) [13]. It's clear that our method achieves a tremendous advantage over NDF even with a relatively high resolution (e.g.256^3). The reason is that our method allows a straightforward surface extracting from the learned UDFs while the ball-pivoting-algorithm used by NDF requires a lot of calculations for neighbour searching and normal estimation.

TABLE 20
Efficiency comparison of surface generation.

Method	NDF	Ours 64^3	Ours 128^3	Ours 256^3	Ours 320^3
Time	2080.7 s	3.0 s	21.9 s	**162.2 s**	307.6 s

4.10 More Visualizations and Analysis

4.10.1 Visualizations of the unsigned distance field

To further evaluate the effectiveness of our consistency-aware field learning, we visualize the learned unsigned distance fields with Neural-Pull [17] loss in Eq. (2) and our field consistency loss in Eq. (3). Fig. 17 shows the visual comparison under two different shapes with complex inner structures. For the storey bus, training with the loss proposed by Neural-Pull [17] fails to handle the rich structures, thus leads to a chaotic distance field where no details were kept. On the contrary, our proposed consistency-aware learning can build a consistent distance field where the detailed structures are well preserved. It's clear that our method learns a highly continuous unsigned distance field and has the ability to keep the field around complex structures correct (e.g. the seats, tires and stairs), which also helps to extract a high fidelity surface.

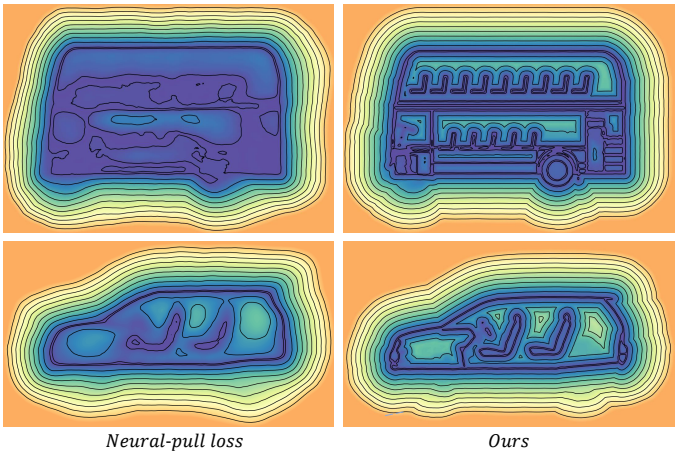

Fig. 17. Visualizations of the unsigned distance field. The darker the color, the closer it is to the approximated surface. For a clear contrast, we set the color of the space far away from the approximated surface to orange.

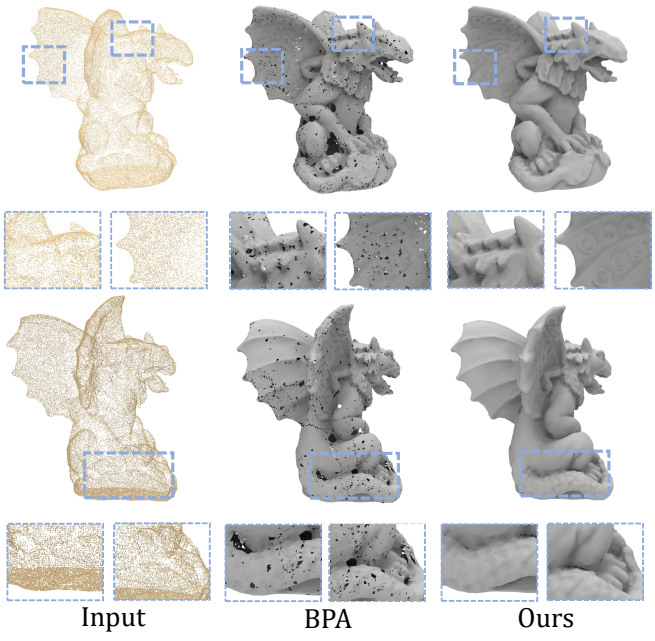

Fig. 18. Visual comparison with ball-pivoting-algorithm.

4.10.2 Visual comparison with ball-pivoting-algorithm

To further explore the advantage of our straightforward surface extraction algorithm, we use the same setting as NDF [10] to adopt ball-pivoting-algorithm (BPA) [13] to extract mesh from our generated point cloud and make a comparison with the mesh extracted using our method as shown in Fig. 18. Even with a carefully selected threshold, the mesh generated by BPA is still far from smooth and has a number of holes, and also fails to retain the detailed geometric information. On the contrary, our method allows to extract surfaces directly from the learned UDFs, thus is able to reconstruct a continuous and high-fidelity mesh where the geometry details is well preserved.

5 Conclusion, Limitation and Future Work

We propose a novel method to learn continuous UDFs directly from raw point clouds by learning to progressively move 3D queries onto the approximated surface. Our introduced reconstruction algorithm can extract surfaces directly from the gradient fields of the learned UDFs. Our method does not require ground truth distance values or point normals, and can reconstruct surfaces with arbitrary topology. We further extend our method to reconstruct surfaces from depth maps and estimate point normals without supervision, where we demonstrate our superior performance over the state-of-the-art in quantitative and qualitative results.

Finally, we acknowledge that there are some potential limitations to our method. First, directly leveraging CAP-UDF for reconstructing corrupted point clouds (e.g. occluded, noisy and sparse point clouds) may struggle to produce robust performances since we do not leverage any conditions or ground truth supervisions for training CAP-UDF. Although we provide an effective solution to improve the reconstruction quality by introducing priors from the state-of-the-art point cloud processing approaches in Sec. 4.6, we believe there is room for further improvements, e.g., training CAP-UDF under diverse datasets in a data-driven way. Second, we use uniformly divided grids to extract surface, which can be improved with a coarse-to-fine paradigm.

References

[1] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[2] C. Jiang, A. Sud, A. Makadia, J. Huang, M. Nießner, T. Funkhouser *et al.*, "Local implicit grid representations for 3D scenes," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6001–6010.

[3] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson, "Deep level sets: Implicit surface representations for 3D shape inference," *arXiv preprint arXiv:1901.06802*, 2019.

[4] Y. Duan, H. Zhu, H. Wang, L. Yi, R. Nevatia, and L. J. Guibas, "Curriculum deepsdf," in *European Conference on Computer Vision*. Springer, 2020, pp. 51–67.

[5] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger, "Occupancy networks: Learning 3D reconstruction in function space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4460–4470.

[6] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, and A. Geiger, "Convolutional occupancy networks," in *European Conference on Computer Vision*. Springer, 2020, pp. 523–540.

[7] Z. Mi, Y. Luo, and W. Tao, "SSRNet: Scalable 3D surface reconstruction network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 970–979.

[8] Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5939–5948.

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

[10] J. Chibane, G. Pons-Moll *et al.*, "Neural unsigned distance fields for implicit function learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 638–21 652, 2020.

[11] F. Zhao, W. Wang, S. Liao, and L. Shao, "Learning anchored unsigned distance functions with gradient direction alignment for single-view garment reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 674–12 683.

[12] R. Venkatesh, T. Karmali, S. Sharma, A. Ghosh, R. V. Babu, L. A. Jeni, and M. Singh, "Deep implicit surface point prediction networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 653–12 662.

[13] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin, "The ball-pivoting algorithm for surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 5, no. 4, pp. 349–359, 1999.

[14] M. Atzmon and Y. Lipman, "SAL: Sign agnostic learning of shapes from raw data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2565–2574.

[15] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3789–3799.

[16] M. Atzmon and Y. Lipman, "SALD: Sign agnostic learning with derivatives," in *International Conference on Learning Representations*, 2020.

[17] 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 *International Conference on Machine Learning*. PMLR, 2021, pp. 7246–7257.

[18] J. Zhou, B. Ma, L. Yu-Shen, F. Yi, and H. Zhizhong, "Learning consistency-aware unsigned distance functions progressively from raw point clouds," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[19] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision*, 2020.

[20] D. Azinović, R. Martin-Brualla, D. B. Goldman, M. Nießner, and J. Thies, "Neural rgb-d surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6290–6301.

[21] J. Wang, T. Bleja, and L. Agapito, "GO-Surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction," in *International Conference on 3D Vision (3DV)*, 2022.

[22] H. Edelsbrunner and E. P. Mücke, "Three-dimensional alpha shapes," *ACM Transactions on Graphics (TOG)*, vol. 13, no. 1, pp. 43–72, 1994.

[23] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics Symposium on Geometry Processing*, vol. 7, 2006.

[24] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, pp. 1–13, 2013.

[25] J. Zhou, J. Wang, B. Ma, Y.-S. Liu, T. Huang, and X. Wang, "Uni3D: Exploring Unified 3D Representation at Scale," in *International Conference on Learning Representations (ICLR)*, 2024.

[26] 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 Transactions on Image Processing*, vol. 31, pp. 4213–4226, 2022.

[27] X. Wen, J. Zhou, Y.-S. Liu, H. Su, Z. Dong, and Z. Han, "3D shape reconstruction from 2D images with disentangled attribute flow," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3803–3813.

[28] Y. Jiang, D. Ji, Z. Han, and M. Zwicker, "SDFDiff: Differentiable rendering of signed distance fields for 3D shape optimization," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[29] T. Hu, Z. Han, and M. Zwicker, "3D shape completion with multi-view consistent inference," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 10 997–11 004.

[30] J. Zhou, X. Wen, B. Ma, Y.-S. Liu, Y. Gao, Y. Fang, and Z. Han, "Self-supervised point cloud representation learning with occlusion auto-encoder," *arXiv preprint arXiv:2203.14084*, 2022.

[31] X. Wen, P. Xiang, Z. Han, Y.-P. Cao, P. Wan, W. Zheng, and Y.-S. Liu, "PMP-Net++: Point cloud completion by transformer-enhanced multi-step point moving paths," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 852–867, 2023.

[32] P. Xiang, X. Wen, Y.-S. Liu, Y.-P. Cao, P. Wan, W. Zheng, and Z. Han, "Snowflake point deconvolution for point cloud completion and generation with skip-transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 6320–6338, 2023.

[33] J. Zhou, B. Ma, W. Zhang, Y. Fang, Y.-S. Liu, and Z. Han, "Differentiable registration of images and lidar point clouds with voxelpoint-to-pixel matching," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[34] T. Li, X. Wen, Y.-S. Liu, H. Su, and Z. Han, "Learning deep implicit functions for 3D shapes with dynamic code clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 840–12 850.

[35] C. Chen, Y.-S. Liu, and Z. Han, "Latent partition implicit with surface codes for 3D representation," in *European Conference on Computer Vision (ECCV)*, 2022.

[36] C. Chen, Z. Han, Y.-S. Liu, and M. Zwicker, "Unsupervised learning of fine structure generation for 3D point clouds by 2D projections matching," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 466–12 477.

[37] Z. Han, B. Ma, Y.-S. Liu, and M. Zwicker, "Reconstructing 3D shapes from multiple sketches using direct shape optimization," *IEEE Transactions on Image Processing*, vol. 29, pp. 8721–8734, 2020.

[38] 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," *Advances in Neural Information Processing Systems*, vol. 34, pp. 27 171–27 183, 2021.

[39] M. Oechsle, S. Peng, and A. Geiger, "UNISURF: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5589–5599.

[40] D. B. Lindell, D. Van Veen, J. J. Park, and G. Wetzstein, "BACON: Band-limited coordinate networks for multiscale scene representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 16 252–16 262.

[41] Y. Wang, I. Skorokhodov, and P. Wonka, "HF-NeuS: Improved surface reconstruction using high-frequency details," *Advances in Neural Information Processing Systems*, 2022.

[42] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

[44] C. Jin, T. Wu, and J. Zhou, "Multi-grid representation with field regularization for self-supervised surface reconstruction from point clouds," *Computers & Graphics*, 2023.

[45] B. Ma, H. Deng, J. Zhou, Y.-S. Liu, T. Huang, and X. Wang, "GeoDream: Disentangling 2D and geometric priors for high-fidelity and consistent 3D generation," *arXiv preprint arXiv:2311.17971*, 2023.

[46] R. Chabra, J. E. Lenssen, E. Ilg, T. Schmidt, J. Straub, S. Lovegrove, and R. Newcombe, "Deep local shapes: Learning local sdf priors for detailed 3D reconstruction," in *European Conference on Computer Vision*. Springer, 2020, pp. 608–625.

[47] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1788–1797.

[48] S. Lombardi, M. R. Oswald, and M. Pollefeys, "Scalable point cloud-based reconstruction with local implicit functions," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 997–1007.

[49] J. N. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, and G. Wetzstein, "ACORN: adaptive coordinate networks for neural scene representation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–13, 2021.

[50] K. Genova, F. Cole, A. Sud, A. Sarna, and T. Funkhouser, "Local deep implicit functions for 3D shape," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4857–4866.

[51] K. Genova, F. Cole, D. Vlasic, A. Sarna, W. T. Freeman, and T. Funkhouser, "Learning shape templates with structured implicit functions," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7154–7164.

[52] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Stoll, and C. Theobalt, "PatchNets: Patch-based generalizable deep implicit 3D shape representations," in *European Conference on Computer Vision*. Springer, 2020, pp. 293–309.

[53] J. Chibane, T. Alldieck, and G. Pons-Moll, "Implicit functions in feature space for 3D shape reconstruction and completion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6970–6981.

[54] B. Ma, Y.-S. Liu, M. Zwicker, and Z. Han, "Surface reconstruction from point clouds by learning predictive context priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[55] B. Ma, Y.-S. Liu, and Z. Han, "Reconstructing surfaces for sparse point clouds with on-surface priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[56] S. Peng, C. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, and A. Geiger, "Shape as points: A differentiable poisson solver," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[57] W. Yifan, S. Wu, C. Oztireli, and O. Sorkine-Hornung, "Iso-points: Optimizing neural implicit surfaces with hybrid representations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 374–383.

[58] A. Boulch and R. Marlet, "POCO: Point convolution for surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 6302–6314.

[59] J. Ye, Y. Chen, N. Wang, and X. Wang, "GIFS: Neural implicit function for general shape representation," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[60] B. Wang, Z. Yu, B. Yang, J. Qin, T. Breckon, L. Shao, N. Trigoni, and A. Markham, "RangeUDF: Semantic surface reconstruction from 3D point clouds," *arXiv preprint arXiv:2204.09138*, 2022.

[61] B. Guillard, F. Stella, and P. Fua, "MeshUDF: Fast and differentiable meshing of unsigned distance field networks," *European Conference on Computer Vision*, 2022.

[62] J. Bibby, "Axiomatisations of the average and a further generalisation of monotonic sequences," *Glasgow Mathematical Journal*, vol. 15, no. 1, pp. 63–65, 1974.

[63] F. Hou, X. Chen, W. Wang, H. Qin, and Y. He, "Robust zero level-set extraction from unsigned distance fields based on double covering," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 6, pp. 1–15, 2023.

[64] Y. Ben-Shabat and S. Gould, "DeepFit: 3D surface fitting via neural network weighted least squares," in *European Conference on Computer Vision*. Springer, 2020, pp. 20–34.

[65] S. Li, J. Zhou, B. Ma, Y.-S. Liu, and Z. Han, "NeAF: Learning neural angle fields for point normal estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[66] Q. Li, Y.-S. Liu, J.-S. Cheng, C. Wang, Y. Fang, and Z. Han, "HSurf-Net: Normal estimation for 3D point clouds by learning hyper surfaces," *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[67] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "Nesti-Net: Normal estimation for unstructured 3D point clouds using convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 112–10 120.

[68] G. Metzer, R. Hanocka, D. Zorin, R. Giryes, D. Panozzo, and D. Cohen-Or, "Orienting point clouds with dipole propagation," *ACM Transactions on Graphics (TOG)*, vol. 40, no. 4, pp. 1–14, 2021.

[69] B. L. Bhatnagar, G. Tiwari, C. Theobalt, and G. Pons-Moll, "Multi-garment net: Learning to dress 3D people from images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5420–5430.

[70] F. Williams, T. Schneider, C. Silva, D. Zorin, J. Bruna, and D. Panozzo, "Deep geometric prior for surface reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 130–10 139.

[71] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or, "Point2Mesh: a self-prior for deformable meshes," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 4, pp. 126–1, 2020.

[72] Q.-Y. Zhou and V. Koltun, "Dense scene reconstruction with points of interest," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–8, 2013.

[73] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt, "BundleFusion: Real-time globally consistent 3D reconstruction using on-the-fly surface reintegration," *ACM Transactions on Graphics (TOG)*, vol. 36, no. 4, p. 1, 2017.
[74] S. Weder, J. Schonberger, M. Pollefeys, and M. R. Oswald, "RoutedFusion: Learning real-time depth map fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4887–4897.
[75] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4104–4113.
[76] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7462–7473, 2020.
[77] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "PCPNet: learning local shape properties from raw point clouds," in *Computer Graphics Forum*, vol. 37, no. 2. Wiley Online Library, 2018, pp. 75–85.
[78] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010.
[79] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Computer Aided Geometric Design*, vol. 22, no. 2, pp. 121–146, 2005.
[80] A. Boulch and R. Marlet, "Deep learning for robust normal estimation in unstructured point clouds," in *Computer Graphics Forum*, vol. 35, no. 5. Wiley Online Library, 2016, pp. 281–290.
[81] J. E. Lenssen, C. Osendorfer, and J. Masci, "Deep iterative surface normal estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 247–11 256.
[82] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 728–737.
[83] X. Yu, Y. Rao, Z. Wang, J. Lu, and J. Zhou, "AdaPoinTr: Diverse point cloud completion with adaptive geometry-aware transformers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
[84] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2790–2799.
[85] D. de Silva Edirimuni, X. Lu, Z. Shao, G. Li, A. Robles-Kelly, and Y. He, "IterativePFN: True iterative point cloud filtering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 530–13 539.
[86] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-GAN: a point cloud upsampling adversarial network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7203–7212.
[87] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.
[88] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.
[89] J. Huang, H.-X. Chen, and S.-M. Hu, "A neural galerkin solver for accurate surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 6, pp. 1–16, 2022.

Junsheng Zhou received the B.S. degree in software engineering from Xiamen University, China, in 2021. He is currently the graduate student with the School of Software, Tsinghua University. His research interests include deep learning, 3D shape analysis and 3D reconstruction.

Baorui Ma received the B.S. degree in computer science and technology from Jilin University, China, in 2018. He is currently pursuing the Ph.D. degree with the School of Software, Tsinghua University. His research interests include deep learning and 3D reconstruction.

Shujuan Li received the B.S. degree in software engineering from Tsinghua University, Beijing, China, in 2022. She is currently working toward the Ph.D. degree in the School of Software, Tsinghua University, Beijing, China. Her research interests include deep learning and 3D computer vision.

Yu-Shen Liu (M'18) received the B.S. degree in mathematics from Jilin University, China, in 2000, and the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2006. From 2006 to 2009, he was a Post-Doctoral 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.

Yi Fang received B.S. and M.S. degrees in biomedical engineering from Xian Jiaotong University, Xian, China, in 2003 and 2006, respectively, and a Ph.D. in mechanical engineering from Purdue University, West Lafayette, IN, USA, in 2011. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, New York University Abu Dhabi, Abu Dhabi, United Arab Emirates. His research interests include 3D computer vision and pattern recognition, large-scale visual computing, deep visual computing, deep cross-domain and crossmodality multimedia analysis, and computational structural biology.

Zhizhong Han received the Ph.D. degree from Northwestern Polytechnical University, China, 2017. He was a Post-Doctoral Researcher with the Department of Computer Science, at the University of Maryland, College Park, USA. Currently, he is an Assistant Professor of Computer Science at Wayne State University, USA. His research interests include 3D computer vision, digital geometry processing and artificial intelligence.