

Learning Signed Hyper Surfaces for Oriented Point Cloud Normal Estimation

Qing Li *Member, IEEE*, Huifang Feng, Kanle Shi, Yue Gao *Senior Member, IEEE*, Yi Fang, Yu-Shen Liu *Member, IEEE*, Zhizhong Han

Abstract—We propose a novel method called SHS-Net for point cloud normal estimation by learning signed hyper surfaces, which can accurately predict normals with global consistent orientation from various point clouds. Almost all existing methods estimate oriented normals through a two-stage pipeline, i.e., unoriented normal estimation and normal orientation, and each step is implemented by a separate algorithm. However, previous methods are sensitive to parameter settings, resulting in poor results from point clouds with noise, density variations and complex geometries. In this work, we introduce signed hyper surfaces (SHS), which are parameterized by multi-layer perceptron (MLP) layers, to learn to estimate oriented normals from point clouds in an end-to-end manner. The signed hyper surfaces are implicitly learned in a high-dimensional feature space where the local and global information is aggregated. Specifically, we introduce a patch encoding module and a shape encoding module to encode a 3D point cloud into a local latent code and a global latent code, respectively. Then, an attention-weighted normal prediction module is proposed as a decoder, which takes the local and global latent codes as input to predict oriented normals. Experimental results show that our algorithm outperforms the state-of-the-art methods in both unoriented and oriented normal estimation.

Index Terms—Point clouds, normal estimation, normal orientation, hyper surfaces, surface reconstruction

1 INTRODUCTION

IN computer vision and graphics, estimating normals for point clouds is a prerequisite for many techniques. As an important geometric property of point clouds, normals with consistent orientation, i.e., *oriented normals*, clearly reveal the geometric structures and make significant contributions in downstream applications, such as rendering and surface reconstruction [1], [2], [3]. Generally, the estimation of oriented normals requires a two-stage paradigm (see Fig. 1): (1) the unoriented normal estimation from the local neighbors of the query point, (2) the normal orientation to make the normal directions to be globally consistent, e.g., facing outward of the surface. While unoriented normals can be estimated by plane or surface fitting of the local neighborhood, determining whether the normals are facing outward or inward is ambiguous. In recent years, many excellent algorithms [4], [5], [6], [7], [8] have been proposed for unoriented normal estimation, while there are few methods that have reliable performance for normal orientation

or directly estimating oriented normals. Estimating oriented normals from point clouds with noise, density variations, and complex geometries in an end-to-end manner is still a challenge.

The classic normal orientation methods rely on simple greedy propagation, which selects a seed point as the start and diffuses its normal orientation to the adjacent points via a minimum spanning tree (MST) [9]. These methods are limited by error accumulation, where an incorrect orientation may degenerate all subsequent steps during the iterative propagation. Furthermore, they heavily rely on a smooth and clean assumption, which makes them easily fail in the presence of sharp edges or corners, density variations and noise. Meanwhile, their accuracy is sensitive to the neighborhood size of propagation. For example, a large size is usually used to smooth out outliers and noise, but can also erroneously include nearby surfaces. Considering that local information is usually not sufficient to guarantee robust orientation, some improved methods [10], [11], [12], [13], [14], [15] try to formulate the propagation process as a global energy optimization by introducing various constraints. Since their constraints are mainly derived from local consistency, the defects are inevitably inherited, and they also suffer from cumulative errors. Moreover, their data-specific parameters are difficult to generalize to new input types and topologies.

Different from the propagation-based methods, which only consider the adjacent normal orientation, the volume-based approaches exploit volumetric representation, such as signed distance functions [16], [17] and variational formulations [18], [19], [20]. They aim to divide the space into interior/exterior and determine whether point normals are facing inward or outward. Despite improvements in accuracy and robustness, these methods cannot scale to large

- Qing Li, Yue Gao and Yu-Shen Liu are with the School of Software, Tsinghua University, Beijing, China. E-mail: {gaoyue, liuyushen}@tsinghua.edu.cn
- Qing Li is with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. E-mail: qingli@swjtu.edu.cn
- Huifang Feng is with the School of Computer and Software Engineering, Xihua University, Chengdu, China. E-mail: fhf@xhu.edu.cn
- Kanle Shi is with Kuaishou Technology, Beijing, China. E-mail: shikanle@kuaishou.com
- Yi Fang is with the 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, Detroit, USA. E-mail: h312h@wayne.edu

The corresponding author is Yu-Shen Liu. This work was supported by the National Key R&D Program of China (2022YFC3800600), the National Natural Science Foundation of China (62272263, 62072268), and in part by Tsinghua-Kuaishou Institute of Future Media Data. The source code, data and pretrained models are available at <https://github.com/LeoQLi/SHS-Net>.

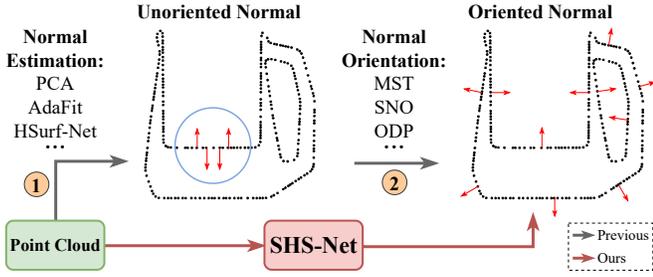


Fig. 1. We propose SHS-Net to estimate oriented normals directly from point clouds. In contrast, previous studies usually achieve this process through a two-stage paradigm using different algorithms, *i.e.*, (1) unoriented normal estimation (*e.g.*, PCA [9], AdaFit [6] and HSurf-Net [8]) and (2) normal orientation (*e.g.*, MST [9], SNO [12] and ODP [15]).

point clouds due to their computational complexity. In general, propagation-based methods have difficulty with sharp features, while volume-based methods have difficulty with open surfaces. Furthermore, the above-mentioned methods are usually complex and require a two-stage operation, their performance heavily depends on the parameter tuning in each separated stage. Recently, several learning-based methods [21], [22], [23] have been proposed to deliver oriented normals from point clouds and have exhibited promising performance. Since they focus on learning an accurate local feature descriptor and do not fully explore the relationship between the surface’s normal orientation and the underlying surface, their performance cannot be guaranteed across different noise levels and geometric structures.

In this work, we propose to estimate oriented normals from point clouds by implicitly learning *signed hyper surfaces*, which are represented by MLP layers to interpret the geometric property in a high-dimensional feature space. We learn this new geometry representation from both local and global shape properties to directly estimate normals with consistent orientation in an end-to-end manner. The insight of our method is that determining a globally consistent normal orientation should require a global context to eliminate the orientation ambiguity in local regions since orientation should be related to the global structure. We evaluate our method by conducting a series of qualitative and quantitative experiments on a range of point clouds with different sampling densities, noise levels, and thin and sharp structures. We reported our original method in [24] and extended our method with unoriented normal estimation, unoriented normal orientation, more applications, and experimental results.

Our main contributions can be summarized as follows.

- We introduce a new technique to represent point cloud geometric properties as signed hyper surfaces in a high-dimensional feature space.
- We show that the signed hyper surfaces can be used to estimate normals with consistent orientations directly from point clouds, rather than through a two-stage paradigm.
- We also show that the modules we designed can be used to build a novel highly efficient pipeline with fewer parameters to estimate accurate unoriented normals, which can be combined with oriented normals to further improve our performance by using a new normal orientation strategy.

- We experimentally demonstrate that our method is able to estimate normals with high accuracy and achieves the state-of-the-art results in both unoriented and oriented normal estimation.
- We apply our method to downstream applications, such as surface reconstruction and point cloud filtering, and show that our estimated normals can effectively improve their performance. We also provide more analysis of the algorithm and experimental results on real-world indoor datasets based on the conference version.

2 RELATED WORK

2.1 Unoriented Normal Estimation

Traditional Methods. Over the past few decades, many algorithms have been proposed for point cloud normal estimation, such as the classic Principle Component Analysis (PCA) [9] and its improvements [25], [26], [27], [28], [29]. Generally, according to Singular Value Decomposition (SVD) [30], the covariance matrix of a local patch is decomposed and the eigenvector with the smallest eigenvalue is perpendicular to the plane defined by the patch. Thanks to its simplicity and efficiency, PCA is widely used in various point cloud processing tasks. However, it is always difficult to determine the data-specific parameter, *e.g.*, patch size, which is crucial to the accuracy of estimation. To find an optimal size for different data, Mitra *et al.* [27] propose to costly investigate the effect of local curvature and point density of the underlying surface. Later, some works introduce Hough transform [31] and Voronoi-based paradigms [20], [32], [33], [34] to improve the robustness of normal estimation and deal with sharp features. Furthermore, the pattern description of local patches is not limited to planes, various complex surfaces [35], [36], [37], [38], [39] are adopted to more accurately fit the surface represented by the point cloud, such as moving least squares [35], truncated Taylor expansion (*n*-jet fitting) [36] and spherical surface fitting [37]. These traditional methods are usually sensitive to noise and various data types, and have limited accuracy even with heavy fine-tuned parameters.

Learning-based Methods. More recently, learning-based methods have been proposed to improve performance in this area and can be mainly divided into two categories: regression-based and surface fitting-based.

(1) *Regression-based methods.* The regression-based methods try to directly predict normals from structured data [40], [41], [42] or raw point clouds [8], [21], [22], [43], [44], [45], [46], [47] in a data-driven manner. For example, HoughCNN [40] uses the Hough transform to convert 3D points into 2D grid representations, and then trains a simple neural network to select a normal from a Hough image-accumulator. PCPNet [21] is regarded as the prior work that adopts the PointNet architecture [48] to extract patch features and predict point normals and curvatures. Based on PCPNet, Zhou *et al.* [43] introduce a local plane constraint and a multi-scale neighborhood selection strategy. NestiNet [44] aims to learn a multi-scale feature vector, and tries to costly find the optimal neighborhood scale for each point. HSurf-Net [8] achieves good performance by learning hyper surfaces from local patches, but the learned surfaces

have no sign and cannot determine the normal orientation. NeAF [47] selects normals from randomly sampled vectors by predicting the angular offset of the query vector. MSEC-Net [49] improves normal estimation in areas with drastic normal changes by introducing edge detection technology. CMG-Net [50] proposes a metric of Chamfer Normal Distance to address the issue of normal direction inconsistency in noisy point clouds.

(2) *Surface fitting-based methods.* The surface fitting-based methods integrate the traditional surface fitting techniques, such as plane fitting [4], [51] and jet fitting [5], [6], [7], [52], [53], [54], into the end of the learning pipeline. They usually carefully design a network to predict pointwise weights, and then use a weighted surface formulation to solve the normal of the fitted surface. For example, Lenssen *et al.* [4] propose to iteratively refine a weighted least squares plane fitting by introducing an adaptive anisotropic kernel. MTR-Net [51] aims to fit a latent tangent plane by designing a differentiable RANSAC-like module. DeepFit [5], AdaFit [6], GraphFit [7], Zhang *et al.* [53], Zhou *et al.* [52] and Du *et al.* [54] predict pointwise weights of local neighborhoods through a PointNet or graph convolutional network, and then apply a weighted polynomial surface fitting to calculate the surface normal. The unoriented normals estimated by the above-mentioned methods randomly face both sides of the surface and cannot be used in many downstream applications without normal orientation.

2.2 Consistent Normal Orientation

To make the unoriented normals have globally consistent orientations, early approaches mainly focus on local consistency and use the orientation propagation strategy upon a minimum spanning tree (MST) to let the adjacent points have the same orientations, such as the pioneering work of [9] and its improved methods [10], [11], [12], [13], [14], [55]. These methods have many limitations in real applications as we introduced earlier. Typical examples are that noisy and sharp features may lead to incorrect orientation propagation, and local errors will spread to larger regions and eventually result in severe performance degradation. In addition, the orientation step cannot correct the wrong directions of the initial unoriented normal (*e.g.*, orthogonal to the true normal). Wang *et al.* [11] present a variational model that makes normals perpendicular to and consistent along the shape surface by minimizing a combination of the Dirichlet energy and the coupled-orthogonality deviation. Their method requires parameter fine-tuning for complex features and may fail on data with outliers. Schertler *et al.* [12] formulate the orientation process as a graph-based energy minimization problem, which is solved by improved quadratic pseudo-Boolean optimization [56]. However, the accuracy of their orientation depends heavily on the chosen normal flipping criterion. Jakob *et al.* [14] perform a graph-based energy optimization on the GPU for the entire point cloud. It uses a parallel greedy solver to achieve faster speed than previous works. Although the above works propose different improved flip criteria or formulate orientation as various global optimization problems to reduce the failure rate of orientation inversion, it is still difficult to guarantee robustness to different inputs. ODP [15] aims to achieve

global consistency by introducing a dipole propagation strategy across the partitioned patches, but its robustness may suffer from the patch partition of nearby gaps or nested structures. GCNO [57] proposes to characterize the requirements of an acceptable winding-number field. The oriented normals are found by utilizing these requirements to ensure global consistency and relying on the Voronoi diagram to estimate normals. However, its optimization is extremely time-consuming for a large number of points since it needs to repeatedly evaluate the winding number of each data point and each query point. NGLO [58] first predicts coarse normals with global consistency from the whole point cloud by learning implicit functions, and then refines the normals based on local information to improve their accuracy. In contrast, some other approaches [21], [22], [23] explore to gather information of different scales and directly predict oriented normals through end-to-end deep networks. These methods focus on learning a general mapping from point clouds to normals and neglect the underlying surface distribution for normal orientation, leading to a sub-optimal solution. In conclusion, the global orientation of point cloud normals is still an open problem with much room for improvement.

The task of implicit unoriented reconstruction, *i.e.*, reconstructing surfaces from point clouds without normals, is also closely related to normal orientation, where the orientation and the reconstruction are bridged in implicit space [59]. Specifically, some approaches propose to solve the consistent normal orientation through volumetric representation. They are usually developed for reconstructing surfaces from unoriented points by various techniques, such as signed distance functions [16], [17], variational formulations [18], [19], [20], visibility [60], [61] and active contours [62]. Xiao *et al.* [59] propose to incorporate isovalue constraints to the Poisson equation, and optimize implicit functions and point normals simultaneously. iPSR [63] runs Poisson reconstruction in an iterative manner and updates normals using the generated surface of the last iteration. PGR [64] takes the point normal and surface element in the Gauss formula as unknown parameters, and then optimizes the parametric function space. In addition to the traditional methods mentioned above, some other works [65], [66], [67], [68], [69] use deep neural networks to learn implicit surfaces directly from raw point clouds without using training labels. We know that the gradient determines the direction of function convergence, and the gradient of the iso-surface can be used as the normal of the surface. Some methods add normals to constraints during optimization to assist surface reconstruction. For example, SAP [70] proposes a differentiable Poisson solver to represent shape surfaces as oriented point clouds, and the point positions and normals are updated during the optimization of surface. Neural-Pull [67] predicts the signed distance field to move a point along or against the gradient for finding its nearest path to the surface, and its gradient is equivalent to normal. IGR [66] proposes an implicit geometric regularization to encourage unit norm gradients and favor a smooth zero-level set of an implicit function. Experimental results show that these methods have limited ability to deal with noise. If the gradient is properly guided, the convergence can be robust and efficient, avoiding local extremum caused by noise or outliers. To handle different types of data, effective

gradient constraints need to be further explored.

3 PRELIMINARY

In mathematics, an explicit representation in Euclidean space expresses the z coordinate of a point p in terms of the x and y , *i.e.*, $z = f(x, y)$. Such a surface is called an explicit surface, also called a height field. Another symmetric representation is $F(x, y, z) = 0$, where F implicitly defines a locus called an implicit surface, also called a scalar field [71]. The implicit surface is a zero iso-surface of F , *i.e.*, the point set $\{p \in \mathbb{R}^3 : F(p) = 0\}$ is a surface implicitly defined by F . Sampling points from an implicit surface is difficult, but the relationship between points and surfaces can be easily determined. On the contrary, it is easy to sample points from an explicit surface, but it is difficult to determine the relationship between the points and the surface. The explicit surface is usually used in surface fitting-based normal estimation, such as jet fitting [36], while the implicit surface is widely used in surface reconstruction. Generally, an explicit surface, *i.e.*, $z = f(x, y)$, can always be rewritten as an implicit surface, *i.e.*, $F(x, y, z) = z - f(x, y) = 0$. These two surface representations have the same tangent plane at a given point, where the normal is defined.

Explicit Surface Fitting. We employ the widely used n -jet surface model [36] to briefly review the explicit surface fitting for normal estimation. It represents the surface by a polynomial function $J_n : \mathbb{R}^2 \rightarrow \mathbb{R}$, which maps a coordinate (x, y) to its height z that is not in the tangent space by

$$z \doteq J_{\alpha, n}(x, y) = \sum_{k=0}^n \sum_{j=0}^k \alpha_{k-j, j} x^{k-j} y^j, \quad (1)$$

where α is the coefficient vector that defines the surface function. In order to find the optimal solution, the least squares approximation strategy is usually adopted to minimize the sum of the square errors between the (ground truth) height and the jet value over a point set $\{p_i\}_{i=1}^N$,

$$J_{\alpha, n}^* = \operatorname{argmin}_{\alpha} \sum_{i=1}^N \|z_i - J_{\alpha, n}(x_i, y_i)\|^2. \quad (2)$$

If $\alpha = (\alpha_{0,0}, \alpha_{1,0}, \alpha_{0,1}, \dots, \alpha_{0,n})$ is solved, then the normal at point p on the fitted surface is computed by

$$\mathbf{n}_p = h(\alpha) = (-\alpha_{1,0}, -\alpha_{0,1}, 1) / \sqrt{1 + \alpha_{1,0}^2 + \alpha_{0,1}^2}. \quad (3)$$

Implicit Surface Learning. In recent years, many learning-based approaches have been proposed to represent surfaces by implicit functions, such as signed distance function (SDF) [72] and occupancy function [73]. The signed (or oriented) distance function is the shortest distance of a given point $p = (x_0, y_0, z_0)$ to the closest surface \mathcal{S} in a metric space, with the sign determined by whether the point is inside ($F(p) < 0$) or outside ($F(p) > 0$) of the surface. The underlying surface is implicitly represented by the iso-surface of $F(p) = 0$. In the surface reconstruction task, a deep network is usually adopted to encode a 3D shape into a latent code, which is fed into a decoder together with query points to predict signed distances. If an implicit surface function is continuous and differentiable, the formula of tangent plane at a regular point p (gradient is non-null) is $F_x(p)(x - x_0) + F_y(p)(y - y_0) + F_z(p)(z - z_0) = 0$ and its normal (*i.e.*, perpendicular) is $\mathbf{n}_p = \nabla F(p) / \|\nabla F(p)\|$.

4 METHOD

As shown in Fig. 2, we propose to implicitly learn signed hyper surfaces in the feature space for estimating oriented normals. In the following sections, we first introduce the representation of signed hyper surfaces by combining the characteristics of the above two surface representations. Then, we design an attention-weighted normal prediction module to solve the oriented normals of query points from signed hyper surfaces. Finally, we introduce how to learn this new surface representation from patch encoding and shape encoding using our designed loss functions.

4.1 Signed Hyper Surface

Similar to the learning of implicit surface, the signed hyper surface is implicitly learned by taking the latent encodings of point clouds as inputs and outputting an approximation of the surface in feature space,

$$f_S(\chi) \approx \mathcal{E}_{\theta}(\chi | z_1, z_2), \quad z_1 = e_{\varphi}(\mathbf{P}_{\chi}^1), \quad z_2 = e_{\psi}(\mathbf{P}_{\chi}^2), \quad (4)$$

where \mathcal{E} is implemented by a neural network with parameter θ that is conditioned on two latent vectors $z_1, z_2 \in \mathbb{R}^c$, which are extracted from point clouds by encoders e_{φ} and e_{ψ} , respectively. \mathbf{P}_{χ}^1 and \mathbf{P}_{χ}^2 are subsample sets of the raw point cloud \mathcal{P} , *e.g.*, point patches around a given point χ .

Similar to existing unoriented normal estimation methods [5], [6], [8], [21], we use a local patch \mathbf{p}_q to capture the local geometry for accurately describing the surface pattern around a query point q ,

$$f_{\mathbf{P}}^{\mathbf{n}}(q) = \mathcal{E}_{\theta}^{\mathbf{n}}(q | z_q^{\mathbf{n}}), \quad z_q^{\mathbf{n}} = e_{\varphi}(\mathbf{P}_q). \quad (5)$$

Since the interior/exterior of a surface cannot be determined reliably from a local patch, we take a global subsample set \mathbf{P}_q from the point cloud \mathcal{P} to provide additional information to estimate the sign at point q ,

$$f_{\mathbf{P}}^{\mathbf{s}}(q) = \operatorname{sgn}(g^{\mathbf{s}}(q)) = \operatorname{sgn}(\mathcal{E}_{\theta}^{\mathbf{s}}(q | z_q^{\mathbf{s}})), \quad z_q^{\mathbf{s}} = e_{\psi}(\mathbf{P}_q), \quad (6)$$

where $\operatorname{sgn}(\cdot)$ is signum function, $g^{\mathbf{s}}(q)$ denotes logit of the probability that q has a positive sign. Thus, the signed hyper surface function at point q is formulated as

$$f_S(q) = f_{\mathbf{P}}^{\mathbf{n}}(q) \cdot f_{\mathbf{P}}^{\mathbf{s}}(q) = \mathcal{E}_{\theta}^{\mathbf{n}, \mathbf{s}}(q | z_q^{\mathbf{n}}, z_q^{\mathbf{s}}). \quad (7)$$

Different from the surface reconstruction task that learns SDF by representing a surface as the zero-set of the SDF, we do not learn a distance field of points with respect to the underlying surface.

4.2 Oriented Normal Estimation

To simplify notations, we denote $\mathcal{E}_{\theta}^{\mathbf{n}, \mathbf{s}}(q | z_q^{\mathbf{n}}, z_q^{\mathbf{s}})$ as $\mathcal{S}_{\theta}(\mathcal{X}, \mathcal{Y})$, where $z_q^{\mathbf{n}} = \mathcal{X} \in \mathbb{R}^c$ and $z_q^{\mathbf{s}} = \mathcal{Y} \in \mathbb{R}^c$ are high dimensional latent vectors. According to the explicit surface fitting, we formulate the signed hyper surface $\mathcal{S}_{\theta} : \mathbb{R}^{2c} \rightarrow \mathbb{R}^c$ as a feature-based polynomial function [8]

$$\mathcal{S}_{\theta, \mu}(\mathcal{X}, \mathcal{Y}) = \sum_{k=0}^{\mu} \sum_{j=0}^k \theta_{k-j, j} \mathbf{x}_{k-j} \mathbf{y}_j = \theta [\mathcal{X} : \mathcal{Y}], \quad (8)$$

where $[\ :]$ means the feature fusion through concatenation, μ denotes the number of fused items.

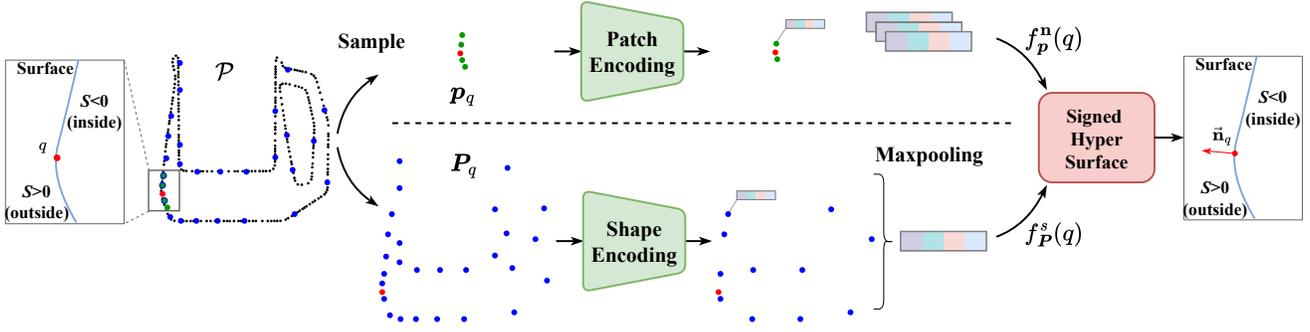


Fig. 2. The learning pipeline of the signed hyper surfaces for oriented normal estimation. It consists of two parallel branches, *i.e.*, patch encoding and shape encoding, which have similar network architectures (see Fig. 3), to extract local and global latent codes, respectively. In both branches, the number of point clouds is downsampled relative to the query point q . Finally, an embedding of the signed hyper surface is used to regress the oriented normal of the query point, which points to the outside of the shape surface.

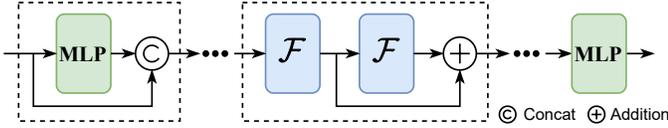


Fig. 3. Feature encoding network in patch and shape encoding. \mathcal{F} is the latent code extraction layer. Black dots indicate the repetition of blocks (dashed box).

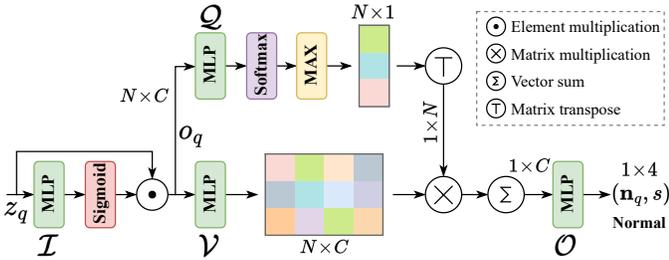


Fig. 4. Attention-weighted normal prediction module $\mathcal{H}(\cdot)$. After we obtain the surface embedding z_q from the fused local and global latent code $[z_q^n : z_q^s]$, we can predict the normal \mathbf{n}_q of the query point q and the sign s to determine its orientation.

Similar to Eq. (2), the bivariate function $\mathcal{S}_{\theta, \mu}(\mathcal{X}, \mathcal{Y})$ aims to map a feature pair $(\mathcal{X}_i, \mathcal{Y}_i)$ to their ground truth value $\mathcal{Z}_i = \mathcal{S}(\mathcal{X}_i, \mathcal{Y}_i) \in \mathbb{R}^c$ in the feature space, *i.e.*,

$$\mathcal{S}_{\theta, \mu}^* = \operatorname{argmin}_{\theta, \mu} \sum_{i=1}^N \|\mathcal{Z}_i - \mathcal{S}_{\theta, \mu}(\mathcal{X}_i, \mathcal{Y}_i)\|^2. \quad (9)$$

To solve the oriented normal $\vec{\mathbf{n}}$ from signed hyper surfaces, we introduce a normal prediction module $\mathcal{H}(\cdot)$, thus

$$\mathcal{S}_{\theta, \mu}^* = \operatorname{argmin}_{\theta, \mu} \sum_{i=1}^N \|\mathcal{H}(\mathcal{Z}_i) - \mathcal{H}(\mathcal{S}_{\theta, \mu}(\mathcal{X}_i, \mathcal{Y}_i))\|^2. \quad (10)$$

Finally, the oriented normal is optimized by

$$\mathcal{S}_{\theta, \mu}^* = \operatorname{argmin}_{\theta, \mu} \sum_{i=1}^N \|\hat{\mathbf{n}}_i - \vec{\mathbf{n}}_i\|^2. \quad (11)$$

Attention-weighted Normal Prediction $\mathcal{H}(\cdot) : \mathbb{R}^c \rightarrow \mathbb{R}^4$. As shown in Fig. 4, we use an attention mechanism to recover the oriented normal $\vec{\mathbf{n}}_q$ of the query point q from c -dimensional fused surface embedding z_q ,

$$(\hat{\mathbf{n}}_q, s) = \mathcal{O}(\mathcal{V}(o_q) \otimes \operatorname{MAX}\{\operatorname{softmax}_{\mathcal{N}_q}(\mathcal{Q}_j(o_q)_{j=1}^m)\}), \quad (12)$$

where $o_q = \tau \cdot z_q$, $\tau = \operatorname{sigmoid}(\mathcal{I}(z_q))$. $\mathcal{O}, \mathcal{V}, \mathcal{Q}$ and \mathcal{I} are MLPs. $m = 64$ is the feature dimension size. First, a multi-head strategy is adopted to deliver m relative weights $\mathcal{Q}_j(o_q)$, which are normalized by softmax over neighbors \mathcal{N}_q into positive interpolation weights. Then, the feature maxpooling $\operatorname{MAX}\{\cdot\}$ is performed to produce attention weights for each point. Meanwhile, the feature embedding o_q is refined through another branch \mathcal{V} and modulated as the weighted sum through matrix multiplication. Finally, the normal and its sign (*i.e.*, orientation) $\vec{\mathbf{n}}_q = (\mathbf{n}_q \in \mathbb{R}^3, s \in \mathbb{R})$ is predicted as a 4D vector by \mathcal{O} , and $\mathbf{n}_q = \hat{\mathbf{n}}_q / \|\hat{\mathbf{n}}_q\|$.

4.3 Feature Encoding

Patch Encoding. Given a neighborhood point patch p_q of the query point q , our local latent code extraction layer \mathcal{F} is formulated as

$$z_i^n = \mathcal{A} \left(\mathcal{B} \left(\operatorname{MAX}\{\mathcal{C}(w_j \cdot z_j^n)\}_{j=1}^{N_l} \right), z_i^n \right), \quad (13)$$

where $i = 1, \dots, N_{l+1}$, l is the neighborhood scale index and $N_{l+1} \leq N_l$. $z_i^n = \mathcal{D}(p_i)$, $p_i \in p_q$ is the per-point feature in the patch. $\mathcal{A}, \mathcal{B}, \mathcal{C}$ and \mathcal{D} are MLPs. $\operatorname{MAX}\{\cdot\}$ denotes the feature maxpooling over N_l -nearest neighbors of the query point q . w is a distance-based weight given by

$$w_j = \frac{\beta_j}{\sum_{i=1}^N \beta_i}, \quad \beta_i = \operatorname{sigmoid}(\gamma_1 - \gamma_2 \|p_i - q\|_2), \quad (14)$$

where γ_1 and γ_2 are learnable parameters with an initial value of 1.0. We use the weight w to make the layer focus on the points p_i that are closer to the query point q in areas where the geometry changes drastically, thereby improving the robustness of feature encoding. As shown in Fig. 3, we stack two layers \mathcal{F} to form a block, which is further stacked to build our patch feature encoder e_φ .

Shape Encoding. Since the global subsample set $P_q = \{p_i\}_{i=1}^{N_P}$ can be seen as a patch with points distributed globally on the shape surface, we adopt a similar network architecture with the patch feature encoder to get the global latent code z_q^s . To obtain P_q , we use a probability-based sampling strategy [74], which brings more points closer to the query point q . It samples points according to a density gradient that decreases with increasing distance from the point q . Moreover, we find that adding some points from uniform sampling can bring better results in structures with

different densities and concavities. Then, the gradient of a point is calculated by

$$v(p_i) = \begin{cases} \left[1 - 1.5 \frac{\|p_i - q\|_2}{\max_{p_j \in \mathcal{P}} \|p_j - q\|_2} \right]_{0.05}^1; \\ 1 \text{ if } i \in \mathcal{R}, \end{cases} \quad (15)$$

where $[\cdot]_{0.05}^1$ indicates value clamping. \mathcal{R} is a random sample index set of \mathcal{P} with $N_{\mathcal{P}}/1.5$ items. Finally, the sampling probability of a point $p_i \in \mathcal{P}$ is $\rho(p_i) = v(p_i) / \sum_{p_j \in \mathcal{P}} v(p_j)$.

Feature Fusion. In order to allow each point in the local patch to have global information and determine the normal orientation, we first use the maxpooling and repetition operation to make the output global latent code has the same dimension as the local latent code. Then, the two kinds of codes are fused by concatenation, *i.e.*, $[z_q^n : z_q^s]$ in Eq. (8).

4.4 Loss Functions

For the query point q , we constrain its unoriented normal and normal sign (*i.e.*, orientation), respectively. To learn an accurate unoriented normal, we employ the ground truth $\hat{\mathbf{n}}_q$ to calculate a normal vector *sin* loss [5]

$$\mathcal{L}_{sin} = \|\mathbf{n}_q \times \hat{\mathbf{n}}_q\|. \quad (16)$$

For the normal orientation, we adopt the binary cross entropy H [74] to calculate a sign classification loss

$$\mathcal{L}_{sgn} = H(\sigma(g^s(q)), [f_S(q) > 0]), \quad (17)$$

where σ is a logistic function that converts the sign logits to probabilities. $[f_S(q) > 0]$ is 1 if the estimated normal faces the outward of surface S and 0 otherwise. Our method achieves a significant performance boost by dividing the oriented normal estimation into unoriented normal regression and its sign classification, instead of directly regressing the oriented normals of query points (see ablations in Sec. 5.5).

To facilitate the local feature learning and make the model also pay attention to the orientation consistency of neighboring points $p_i \in \mathcal{P}_q$, we compute a weighted mean square error (MSE)

$$\mathcal{L}_{mse} = \frac{1}{N} \sum_{i=1}^N \tau_i \|\bar{\mathbf{n}}_i - \hat{\mathbf{n}}_i\|^2, \quad (18)$$

where the neighborhood point normals $\bar{\mathbf{n}} = \delta(z_q)$ are predicted from the surface embedding z_q by an MLP layer $\delta : \mathbb{R}^c \rightarrow \mathbb{R}^3$. Moreover, we add a loss term according to coplanarity [53] to facilitate the learning of τ in Eq.(12),

$$\mathcal{L}_\tau = \frac{1}{N} \sum_{i=1}^N (\tau_i - \hat{\tau}_i)^2, \quad \hat{\tau}_i = \exp\left(-\frac{(p_i \cdot \hat{\mathbf{n}}_q)^2}{\xi^2}\right), \quad (19)$$

where $\xi = \max(0.0025, 0.3 \sum_{i=1}^N (p_i \cdot \hat{\mathbf{n}}_q)^2 / N)$. In summary, our final training loss for oriented normal estimation is

$$\mathcal{L} = \lambda_1 \mathcal{L}_{sin} + \lambda_2 \mathcal{L}_{sgn} + \lambda_3 \mathcal{L}_{mse} + \lambda_4 \mathcal{L}_\tau, \quad (20)$$

where $\lambda_1 = 0.1$, $\lambda_2 = 0.1$, $\lambda_3 = 0.5$ and $\lambda_4 = 1.0$ are weighting factors that are first set empirically and then fine-tuned based on experiments.

4.5 Unoriented Normal Estimation

In the previous sections, we introduce to estimate oriented normals by implicitly learning signed hyper surfaces in the feature space. The network model extracts local and global feature representations through patch and shape encoding respectively, and the global features from shape encoding help determine the normal orientation. In this section, we show that the modules we designed in the patch encoding can be reorganized to estimate unoriented normals, and their orientations are solved in the next section.

To estimate unoriented normals, *i.e.*, the local property of point clouds, whose orientations are not guaranteed to be globally consistent, the global information from shape encoding is not needed. Thus, we build an unoriented normal estimation pipeline by using the local latent code extraction layer \mathcal{F} in Eq. (13). The input of this new network pipeline is the local point cloud patch and its output is the unoriented normal of the query point. The layer \mathcal{F} is stacked recursively, enabling the network model to learn increasingly rich representations of the point cloud patch. For that, the features \mathcal{X} from two layers are aggregated by add operation and passed to the next layer, *i.e.*,

$$\mathcal{X}_{k+1} = [\mathcal{X}_k]_{N_{k+1}} + \mathcal{F}_2(\mathcal{X}_k), \quad \mathcal{X}_k = \mathcal{F}_1(\mathcal{X}_{k-1}), \quad (21)$$

where $[\cdot]_{N_{k+1}}$ denotes the neighborhood scale of size N_{k+1} with respect to query point. By continuously reducing the number of k-nearest neighbors of the query point, our layers extract features from different scales of the query point in order from large to small. With the recursive utilization of our layers, the large scales of earlier layers give more robust information about the underlying geometries, while the small scales of the latter layers lead to a more accurate description of the local details. In this manner, the features from different scales of the local patch around a query point are fused to obtain its optimal geometric description.

After obtaining the final output feature \mathcal{X}_o with N_o neighboring points in the patch, the unnormalized normal \mathbf{n}_q of the query point is predicted by a weighted maxpooling of its neighboring features $x_i \in \mathcal{X}_o$, that is

$$\mathbf{n}_q = \mathcal{O}(\text{MAX}\{w_i \cdot \tau_i \cdot x_i | i=1, \dots, N_o\}), \quad (22)$$

where $\tau_i = \text{sigmoid}(\mathcal{I}(x_i))$ is the point weight. \mathcal{O} and \mathcal{I} are MLPs. Furthermore, the neighboring point normals \mathbf{n}_i are predicted from \mathcal{X}_o by another MLP.

Training Loss. To constrain the predicted normal of the query point, we calculate the *sin* distance d_{sin} and squared Euclidean distance d_{euc} between the predicted normal \mathbf{n} and the ground truth normal $\hat{\mathbf{n}}$, *i.e.*,

$$\mathcal{L}_q = \|\mathbf{n} \times \hat{\mathbf{n}}\| + \min(\|\mathbf{n} - \hat{\mathbf{n}}\|^2, \|\mathbf{n} + \hat{\mathbf{n}}\|^2). \quad (23)$$

Meanwhile, we calculate a weighted neighborhood consistency loss based on the ground truth normals $\hat{\mathbf{n}}_i$ of neighboring points, then we have

$$\mathcal{L}_{con} = \frac{1}{N_o} \sum_{i=1}^{N_o} \tau_i (\|\mathbf{n}_i \times \hat{\mathbf{n}}_i\| + \min(|\mathbf{n}_i - \hat{\mathbf{n}}_i|^2, |\mathbf{n}_i + \hat{\mathbf{n}}_i|^2)). \quad (24)$$

Thus, we obtain the loss \mathcal{L}_q for query point normal \mathbf{n}_q and the mean loss \mathcal{L}_{con} for neighboring point normals \mathbf{n}_i . The

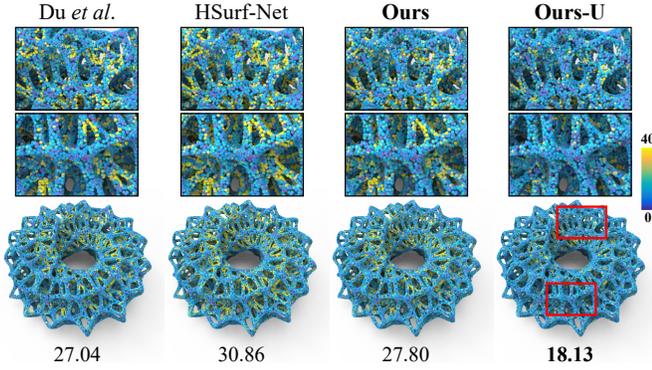


Fig. 5. Visual comparison of unoriented normal errors on a point cloud with complex geometry. The normal RMSE is mapped to a heatmap ($0^\circ - 40^\circ$). We provide the average RMSE over shape for each method.

distance-based weight w in Eq. (14) and the loss function of τ in Eq. (19) are also adopted in this section.

In summary, the final training loss function for unoriented normal estimation is given by

$$\mathcal{L} = \kappa_1 \mathcal{L}_q + \kappa_2 \mathcal{L}_{con} + \kappa_3 \mathcal{L}_\tau, \quad (25)$$

where the weighting factors are set to $\kappa_1 = 0.1$, $\kappa_2 = 0.4$ and $\kappa_3 = 1.0$. We first select their initial values empirically and then fine-tune the parameters experimentally.

4.6 Unoriented Normal Orientation

Another issue is that the global consistency of the estimated unoriented normals in Sec. 4.5 cannot be guaranteed since they are obtained by regression from local features only. We know that unoriented normal is a local property of the point cloud, while oriented normal estimation requires additional information to determine its orientation. Thus, the unoriented normal estimation in Sec. 4.5 solely uses the local information, and the oriented normal estimation in Sec. 4.2 further incorporates global information to learn signed hyper surfaces to predict the normal and its sign. Here we show that we can tune the orientation of unoriented normal \mathbf{n} to achieve its consistent direction by using the oriented normal result $\vec{\mathbf{n}}$ as the reference. For this, we introduce a normal orientation strategy to transfer the normal sign of oriented normal to the unoriented normal and re-orientate its direction. Different from the local orientation propagation using MST [9], our strategy is based on the angle distance between the corresponding oriented and unoriented normal vectors at the same point, and the new oriented normal $\vec{\mathbf{n}}'$ is obtained by

$$\vec{\mathbf{n}}' = \begin{cases} \mathbf{n}, & \text{if } \mathbf{n} \cdot \vec{\mathbf{n}} > 0; \\ -\mathbf{n}, & \text{if } \mathbf{n} \cdot \vec{\mathbf{n}} < 0, \end{cases} \quad (26)$$

where the reference normal $\vec{\mathbf{n}}$ can be the oriented normal estimated in this work or obtained through other methods. We will experimentally show that we can employ the unoriented normals to further improve the accuracy of oriented normal estimation results.

5 EXPERIMENTS

Implementation. We only train our network model on the PCPNet shape dataset [21], which provides the ground

truth normals with consistent orientation (outward of the surface). We follow the same train/test data split and data processing as in [5], [6], [8], [21]. For patch encoding, we randomly select a query point from the shape point cloud and search its 700 neighbors to form a patch. For shape encoding, we sample $N_P = 1200$ points from the shape point cloud according to the sampling probability. The Adam optimizer is adopted with an initial learning rate of 9×10^{-4} which is decayed to $1/5$ of the latest value at epochs $\{400, 600, 800\}$. The model is trained on an NVIDIA 2080 Ti GPU with a batch size of 145 and epochs of 800.

FamousShape Dataset. Due to the lack of relevant datasets and the relatively simple test shapes of the PCPNet dataset [21], we further collect shapes with complex structures from other public datasets, such as the Famous dataset [74] and the Stanford 3D Scanning Repository [75]. We sample 100K points from each shape and follow the same preprocessing steps as the PCPNet dataset to conduct data augmentation, *e.g.*, adding Gaussian noise with different levels (0.12%, 0.6% and 1.2%) and non-uniform sampling (stripe and gradient). The ground truths of oriented normals are extracted from mesh data and used for evaluation. We call this dataset *FamousShape*, and it is available along with our code. In the supplementary material, we visualize the point cloud shapes of the FamousShape dataset, which has more complex geometries than the PCPNet dataset.

Evaluation Metrics. We adopt the same evaluation metrics as in [5], [6], [8], [21] to evaluate the estimated normals. More specifically, Root Mean Squared Error (RMSE) measures normal angles between the ground truth normals $\hat{\mathbf{n}}$ and the predicted normals \mathbf{n} , while the curve of Percentage of Good Point (PGP) shows the overall quality of results by counting points whose normal errors are less than the given thresholds. They are computed by

$$\text{RMSE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\arccos(\phi))^2}, \quad (27)$$

$$\text{PGP}(\tau) = \frac{1}{N} \sum_{i=1}^N \mathcal{I}(\arccos(\phi) < \tau), \quad (28)$$

where N is the number of evaluated normals in a point cloud. ϕ is the angle cosine of two vectors, and $\phi_{\text{unoriented}} = |\hat{\mathbf{n}}_i \odot \mathbf{n}_i|$ and $\phi_{\text{oriented}} = \hat{\mathbf{n}}_i \odot \mathbf{n}_i$ are used in unoriented and oriented normal evaluation, respectively. $|\cdot|$ represents the absolute value of the inner product \odot of two normal vectors. Therefore, the normal angle error $\text{RMSE}_{\text{unoriented}}$ is bounded between 0° and 90° in unoriented normal evaluation, and $\text{RMSE}_{\text{oriented}}$ is bounded between 0° and 180° in oriented normal evaluation. \mathcal{I} represents an indicator function that measures whether the error is less than a given threshold τ . The ground truth normals in the benchmark datasets face outward of the shape surface. For the baseline methods, we flip their estimated normals if more than half of the normals face inward during oriented normal evaluation.

In the following experiments, we use 'Ours' to denote the result of our oriented normals, 'Ours-U' to denote the result of our unoriented normals, 'Ours-U+O' to denote the result of our unoriented normals being re-orientated by our oriented normal, and 'Ours-U+NGL' to denote the result of our unoriented normals being re-orientated using the NGL module [58].

TABLE 1

Unoriented normal evaluation on datasets PCPNet and FamousShape. We report RMSE results under different noise levels and different sampling ways, and rank the methods according to the average RMSE on the PCPNet dataset. * means the code is uncompleted or unavailable, and the result comes from its public paper. 'Ours' denotes the result of our oriented normals and 'Ours-U' denotes the result of our unoriented normals.

Category	PCPNet Dataset							FamousShape Dataset						
	Noise				Density		Average	Noise				Density		Average
	None	0.12%	0.6%	1.2%	Stripe	Gradient		None	0.12%	0.6%	1.2%	Stripe	Gradient	
Jet [36]	12.35	12.84	18.33	27.68	13.39	13.13	16.29	20.11	20.57	31.34	45.19	18.82	18.69	25.79
PCA [9]	12.29	12.87	18.38	27.52	13.66	12.81	16.25	19.90	20.60	31.33	45.00	19.84	18.54	25.87
PCPNet [21]	9.64	11.51	18.27	22.84	11.73	13.46	14.58	18.47	21.07	32.60	39.93	18.14	19.50	24.95
Zhou <i>et al.</i> * [43]	8.67	10.49	17.62	24.14	10.29	10.66	13.62	-	-	-	-	-	-	-
Nesti-Net [44]	7.06	10.24	17.77	22.31	8.64	8.95	12.49	11.60	16.80	31.61	39.22	12.33	11.77	20.55
Lenßen <i>et al.</i> [4]	6.72	9.95	17.18	21.96	7.73	7.51	11.84	11.62	16.97	30.62	39.43	11.21	10.76	20.10
DeepFit [5]	6.51	9.21	16.73	23.12	7.92	7.31	11.80	11.21	16.39	29.84	39.95	11.84	10.54	19.96
MTRNet* [51]	6.43	9.69	17.08	22.23	8.39	6.89	11.78	-	-	-	-	-	-	-
Refine-Net [46]	5.92	9.04	16.52	22.19	7.70	7.20	11.43	-	-	-	-	-	-	-
Zhang <i>et al.</i> * [53]	5.65	9.19	16.78	22.93	6.68	6.29	11.25	9.83	16.13	29.81	39.81	9.72	9.19	19.08
Zhou <i>et al.</i> * [52]	5.90	9.10	16.50	22.08	6.79	6.40	11.13	-	-	-	-	-	-	-
AdaFit [6]	5.19	9.05	16.45	21.94	6.01	5.90	10.76	9.09	15.78	29.78	38.74	8.52	8.57	18.41
GraphFit [7]	5.21	8.96	16.12	21.71	6.30	5.86	10.69	8.91	15.73	29.37	38.67	9.10	8.62	18.40
NeAF [47]	4.20	9.25	16.35	21.74	4.89	4.88	10.22	7.67	15.67	29.75	38.76	7.22	7.47	17.76
HSurf-Net [8]	4.17	8.78	16.25	21.61	4.98	4.86	10.11	7.59	15.64	29.43	38.54	7.63	7.40	17.70
NGLO [58]	4.06	8.70	16.12	21.65	4.80	4.56	9.98	7.25	15.60	29.35	38.74	7.60	7.20	17.62
Du <i>et al.</i> [54]	3.85	8.67	16.11	21.75	4.78	4.63	9.96	6.92	15.05	29.49	38.73	7.19	6.92	17.38
CMG-Net [50]	3.87	8.45	16.08	21.89	4.85	4.45	9.93	7.07	14.83	29.04	38.93	7.43	7.03	17.39
MSENet [49]	3.84	8.74	16.10	21.05	4.34	4.51	9.76	-	-	-	-	-	-	-
Ours	3.95	8.55	16.13	21.53	4.91	4.67	9.96	7.41	15.34	29.33	38.56	7.74	7.28	17.61
Ours-U	3.49	8.43	15.73	21.05	4.45	4.17	9.55	6.79	15.04	29.03	38.40	7.21	6.67	17.19

TABLE 2

Comparison of unoriented normal PGP(20°) on the datasets PCPNet and FamousShape under the highest noise. The higher the better.

(%)	GraphFit	NeAF	HSurf-Net	NGLO	Du <i>et al.</i>	CMG-Net	Ours	Ours-U
PCPNet	77.71	77.44	77.77	77.76	77.79	77.35	77.94	77.80
Famous.	43.99	44.05	44.62	44.19	43.97	43.18	44.67	44.69

TABLE 3

Unoriented normal RMSE on the datasets SceneNN and ScanNet.

	GraphFit	NeAF	HSurf-Net	NGLO	Du <i>et al.</i>	CMG-Net	Ours	Ours-U
SceneNN	8.59	7.88	7.55	7.73	7.68	7.64	7.93	7.30
ScanNet	18.31	15.48	15.83	15.44	15.37	15.14	15.75	15.02

5.1 Unoriented Normal Comparison

We use our estimation results of *oriented* and *unoriented* normal to compare with baseline methods that are designed for estimating *unoriented* normals, such as the traditional methods PCA [9] and Jet [36], the learning-based surface fitting methods DeepFit [5], AdaFit [6] and GraphFit [7], and the learning-based regression methods PCPNet [21], Nesti-Net [44] and HSurf-Net [8]. As shown in Table 1, we report quantitative comparison results with the baselines in terms of normal angle RMSE on two datasets, PCPNet and FamousShape. On the PCPNet dataset, our method achieves the best performance under almost all noise levels and density variations. On our FamousShape dataset, our method achieves the best performance under most metrics and has the lowest average RMSE result. In Fig. 5, we provide visual comparisons of the unoriented normal error of various methods, and the results show that our method can handle complex geometries better. In Table 2, we use the metric of PGP(20°) to quantitatively evaluate the accuracy of normal results, *i.e.*, the percentage of points whose normal errors are less than 20°. The evaluation results show that our method can obtain accurate normals for more points than baseline methods.

Evaluation on Scene Point Clouds. To evaluate the generalization ability of our method, we use the network models (including models for oriented and unoriented normal estimation) trained on the PCPNet shape dataset to test on real-scanned scene data of datasets SceneNN [76] and ScanNet [77]. For these two indoor scene datasets, we only report the quantitative evaluation results for unoriented normals, rather than oriented normals, because it is ambiguous to judge the internal or external orientation of normals of objects and walls inside a room. Unless these objects and walls are segmented very precisely, which is not easy to achieve. The ground truth normal is obtained from the provided mesh data. In Table 3, we provide the evaluation results of unoriented normals on these two datasets, and our method achieves significant improvements compared to baseline methods. In Fig. 6 and Fig. 7, we provide visual comparisons of the unoriented normal error on some indoor room scenes of datasets, SceneNN and ScanNet, respectively. These comparison results demonstrate the good generalization ability and outstanding performance of our method. In the supplementary material, we provide more visual comparisons of the unoriented normal error on real-scanned indoor scenes.

5.2 Oriented Normal Comparison

We compare our approach for *oriented* normal estimation with various baseline methods, such as PCPNet [21], DPGO [23] and NGLO [58]. The trained model of PCPNet is available. The source code of DPGO is uncompleted and its results on the PCPNet dataset are taken from its paper. In addition, we choose three *unoriented* normal estimation methods (PCA [9], AdaFit [6] and HSurf-Net [8]) and three normal orientation methods (MST [9], SNO [12] and ODP [15]), and make different combinations of them to form two-stage pipelines for estimating oriented normals,

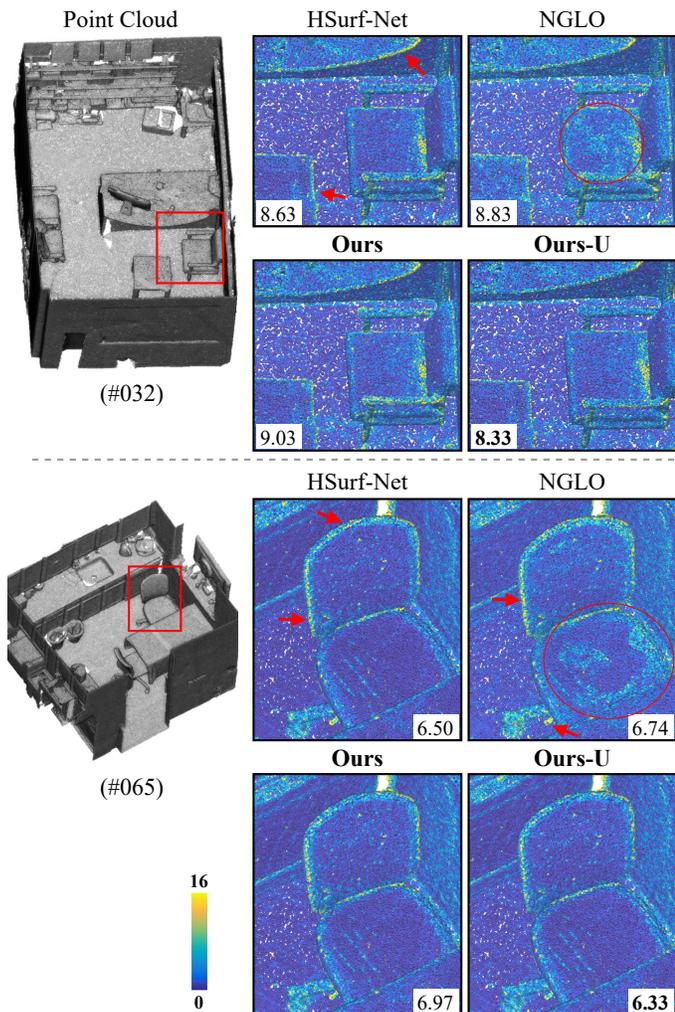


Fig. 6. Visual comparison of unoriented normal errors on the SceneNN dataset. The normal RMSE is mapped to a heatmap ($0^\circ - 16^\circ$). We provide the average RMSE of each method over the entire point cloud.

such as PCA+MST and HSurf-Net+ODP. Among the baselines, PCA is a widely used traditional method, AdaFit is a representative surface fitting-based method, and HSurf-Net is a regression-based method and has the state-of-the-art performance for *unoriented* normal estimation. We use the original implementation of SNO and ODP, and the implementation of MST in [78]. In Table 4, we show the quantitative comparison results on datasets PCPNet and FamousShape. We can see that our method provides the most accurate normals under almost all noise levels and density variations for both datasets, and achieves huge performance gains in terms of average results compared to all baselines. From the experimental results, we find that the propagation-based normal orientation methods have significantly varied results when dealing with unoriented normal inputs from different estimation methods, such as PCA+MST and AdaFit+MST. The overall error distributions of various methods on datasets PCPNet and FamousShape are illustrated in Fig. 8, and our method achieves excellent performance at different thresholds. A visual comparison result of the normal errors on a point cloud with sharp corners is shown in Fig. 9, which shows the superior capability of our method. As shown in Fig. 10, we provide an example

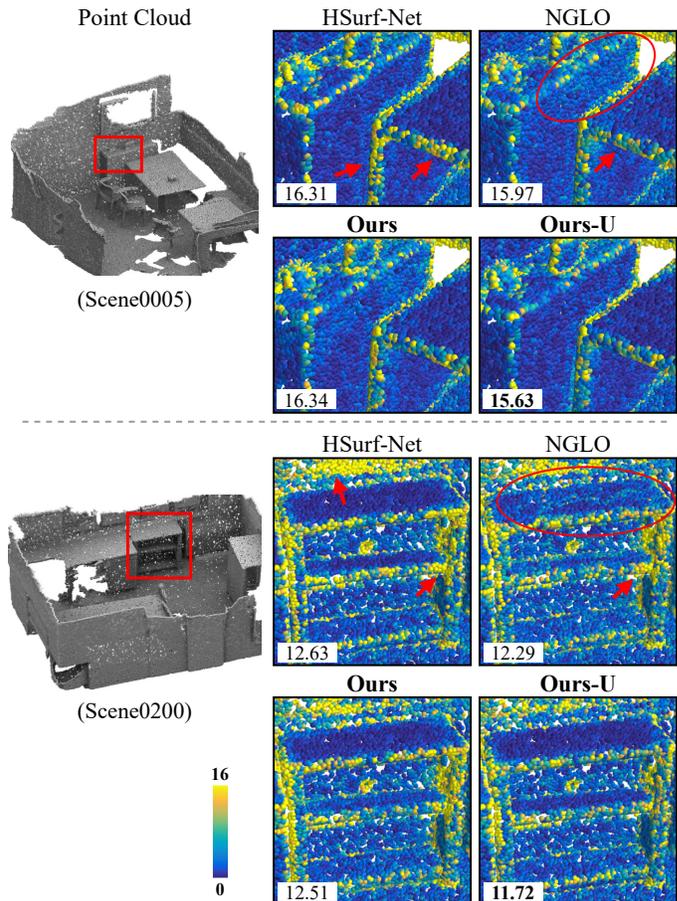


Fig. 7. Visual comparison of unoriented normal errors on the ScanNet dataset. The normal RMSE is mapped to a heatmap ($0^\circ - 16^\circ$). We provide the average RMSE of each method over the entire point cloud.

of a point cloud sampled from a thin sheet. It has two planes that are very close together, which can easily affect the accuracy of the unoriented normal and the orientation of the oriented normal, *i.e.*, which side of the sheet it points to. The quantitative and qualitative comparison results show that our method has a huge advantage over the baseline methods in the oriented normal estimation task.

Evaluation on Sparse Point Clouds. To evaluate the generalization ability of our method, we conduct evaluations on two sets of point clouds that have the same shapes as the FamousShape dataset but each shape in these two sets contains only 3000 and 5000 points, respectively. We first evaluate our method for unoriented normal estimation. As shown in Table 5, we report quantitative comparison results of unoriented normals, and our method achieves significant performance improvements. Then, we evaluate for oriented normal estimation. The recently proposed algorithm, GCNO [57], can estimate point cloud normals with globally consistent orientations. The problem is that its running time increases so drastically with the number of points in the point cloud that we cannot fully test it on existing benchmark datasets, such as PCPNet and FamousShape where each shape has 100K points. Therefore, in order to complete the evaluation in a reasonable amount of time, we compare with GCNO on sparse data. As shown in Table 6, we report quantitative comparison results of oriented normal estimation on sparse point clouds. The traditional

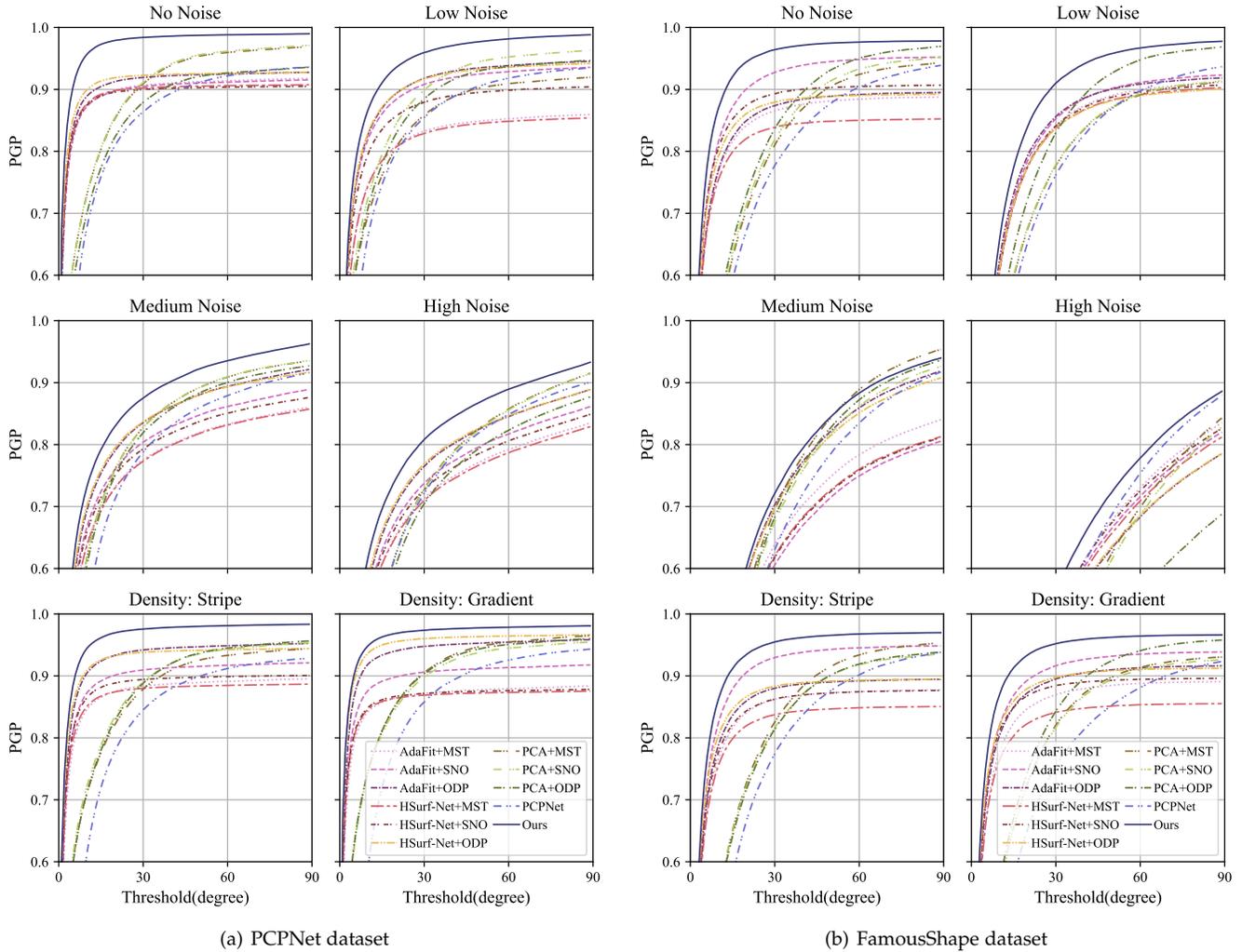


Fig. 8. PGP of oriented normals on the PCPNet dataset and the FamousShape dataset. It shows the percentage of correctly estimated point normals for a given angle threshold. Our method has the best PGP result for almost all thresholds.

baseline algorithms, including GCNO and PCA+MST, are implemented in C++ on the Windows platform and run on an Intel i9-11900K CPU. And other learning-based methods are implemented in Pytorch on the Linux platform and run on the GPU. From Table 6, we can see that our method has the best RMSE result and good execution efficiency. The high time consumption of GCNO limits its application to data with a large number of points. Furthermore, we show a visual comparison of oriented normal errors in Fig. 11. These evaluation experimental results demonstrate the excellent performance of our method on sparse point cloud data.

5.3 Complexity and Efficiency

In this evaluation experiment, we compare the learning-based methods on the same machine with an NVIDIA 2080 Ti GPU. The comparative experiment is first conducted on the task of unoriented normal estimation, and the learning-based methods that have good performance on the PCPNet dataset are selected as baselines, such as GraphFit [7], NeAF [47] and Du *et al.* [54]. As shown in Table 7, we report the average RMSE of unoriented normal on the PCPNet dataset, the number of learnable network parameters, and

the execution time for unoriented normal estimation. Our method achieves significant performance improvement with minimal parameters and running time. It is worth noting that our running efficiency is dozens or even hundreds of times faster than other baseline methods, such as 9.77 (Ours-U) vs. 295.69 (Du *et al.* [54]).

In the oriented normal evaluation experiments, we compare our method to PCPNet [21], DPGO [23], NGLO [58] and other methods that are based on a two-stage paradigm. We make different combinations of existing works to estimate oriented normals, such as AdaFit+ODP and HSurf-Net+ODP. Among the baseline methods, AdaFit [6], HSurf-Net [8] and ODP [15] are learning-based methods, and the others are traditional methods. In Table 8, we report the average RMSE of oriented normal on the PCPNet dataset, the number of learnable network parameters, and the execution time of each method for oriented normal estimation. Our method achieves a large performance improvement with relatively fewer parameters and less running time.

5.4 Visualization of Weight and Attention

As shown in Fig. 12, we visualize the learned attention weight in the normal prediction module \mathcal{H} , weight τ in

TABLE 4

Oriented normal evaluation results on datasets PCPNet and FamousShape. Among the baseline methods, there are schemes that directly estimate the oriented normal, as well as schemes based on two stages. * means the source code is uncompleted and the result is from its paper. 'Ours' denotes the result of our oriented normals, 'Ours-U+O' denotes the result of our unoriented normals being re-orientedated by our oriented normal, and 'Ours-U+NGL' denotes the result of our unoriented normals being re-orientedated using the NGL module [58].

Category	PCPNet Dataset							FamousShape Dataset						
	None	Noise		Density		Average	None	Noise		Density		Average		
	0.12%	0.6%	1.2%	Stripe	Gradient			0.12%	0.6%	1.2%	Stripe		Gradient	
PCA [9]+MST [9]	19.05	30.20	31.76	39.64	27.11	23.38	28.52	35.88	41.67	38.09	60.16	31.69	35.40	40.48
PCA [9]+SNO [12]	18.55	21.61	30.94	39.54	23.00	25.46	26.52	32.25	39.39	41.80	61.91	36.69	35.82	41.31
PCA [9]+ODP [15]	28.96	25.86	34.91	51.52	28.70	23.00	32.16	30.47	31.29	41.65	84.00	39.41	30.72	42.92
AdaFit [6]+MST [9]	27.67	43.69	48.83	54.39	36.18	40.46	41.87	43.12	39.33	62.28	60.27	45.57	42.00	48.76
AdaFit [6]+SNO [12]	26.41	24.17	40.31	48.76	27.74	31.56	33.16	27.55	37.60	69.56	62.77	27.86	29.19	42.42
AdaFit [6]+ODP [15]	26.37	24.86	35.44	51.88	26.45	20.57	30.93	41.75	39.19	44.31	72.91	45.09	42.37	47.60
HSurf-Net [8]+MST [9]	29.82	44.49	50.47	55.47	40.54	43.15	43.99	54.02	42.67	68.37	65.91	52.52	53.96	56.24
HSurf-Net [8]+SNO [12]	30.34	32.34	44.08	51.71	33.46	40.49	38.74	41.62	41.06	67.41	62.04	45.59	43.83	50.26
HSurf-Net [8]+ODP [15]	26.91	24.85	35.87	51.75	26.91	20.16	31.07	43.77	43.74	46.91	72.70	45.09	43.98	49.37
PCPNet [21]	33.34	34.22	40.54	44.46	37.95	35.44	37.66	40.51	41.09	46.67	54.36	40.54	44.26	44.57
DPGO* [23]	23.79	25.19	35.66	43.89	28.99	29.33	31.14	-	-	-	-	-	-	-
NGLO [58]	12.52	12.97	25.94	33.25	16.81	9.47	18.49	13.22	18.66	39.70	51.96	31.32	11.30	27.69
Ours	10.28	13.23	25.40	35.51	16.40	17.92	19.79	21.63	25.96	41.14	52.67	26.39	28.97	32.79
Ours-U+O	10.26	13.47	25.85	36.04	16.54	17.95	20.02	21.69	26.09	41.91	52.87	26.44	29.00	33.00
Ours-U+NGL	12.05	12.87	25.95	33.43	16.44	8.97	18.29	12.30	18.20	39.96	51.57	31.11	10.56	27.28

TABLE 5

Unoriented normal RMSE on sparse point clouds with 3000 and 5000 points. Our approach has significant advantages.

	GraphFit	NeAF	HSurf-Net	NGLO	Du <i>et al.</i>	CMG-Net	GCNO	Ours	Ours-U
3K	29.56	28.64	27.74	26.78	27.18	26.13	27.87	27.22	24.91
5K	25.47	25.10	23.93	23.07	23.30	22.40	31.54	23.55	21.20

TABLE 6

Oriented normal RMSE on sparse point clouds. The algorithms of GCNO and PCA+MST run on the CPU, and the running time (seconds per 5000 points) of GCNO is much longer than other methods.

	PCA +MST	HSurf-Net +ODP	PCPNet	GCNO	NGLO	Ours	Ours-U +O	Ours-U +NGL
3K	51.62	63.88	53.13	33.40	32.65	37.31	36.52	30.80
5K	45.40	62.51	48.48	41.24	28.34	32.64	31.85	26.97
Time	0.01+0.71	3.87+31.75	3.34	822.60	0.10+2.82	3.54	1.44+3.54	1.44+0.10

Eq. (12) and weight w in Eq. (14). They illustrate the points that the model focuses on at different stages of the normal estimation process. The weight w indicates that the model focuses on points closer to the center during the patch and shape encoding. The weight τ indicates that the model focuses on points coplanar with the query point during the final local feature modulation for normal prediction. The attention weight indicates that the model focuses on the query point during the final oriented normal prediction of the query point.

5.5 Ablation Studies

We provide ablation results for oriented normal estimation in Table 9 (a)-(d), which are discussed as follows.

(a) Feature Encoding. (i) We realize the oriented normal estimation without using the patch encoding or the shape encoding. (ii) The distance-based weight w is not used in both patch encoding and shape encoding.

(b) Module \mathcal{H} . The attention-weighted normal prediction module \mathcal{H} is replaced with simple MLP layers.

(c) Losses \mathcal{L}_{sin} , \mathcal{L}_{sgn} and \mathcal{L}_{mse} . In our pipeline, we regress the unoriented normal and its orientation sign of the query point q , and constrain them in loss functions \mathcal{L}_{sin} and \mathcal{L}_{sgn} , respectively. Here, we do not use \mathcal{L}_{sin} and \mathcal{L}_{sgn} , and directly predict the oriented normal \vec{n}_q of the point q from surface

TABLE 7

Comparison of the *unoriented* normal RMSE, the learnable network parameter (million) and the average inference time (seconds per 100K points) of different learning-based methods on the PCPNet dataset.

	GraphFit	NeAF	HSurf-Net	NGLO	Du <i>et al.</i>	CMG-Net	Ours	Ours-U
RMSE	10.69	10.22	10.11	9.98	9.96	9.93	9.96	9.55
Param.	4.26	6.74	2.16	0.46+1.92	4.46	2.70	3.27	1.76
Time	292.12	400.81	72.47	0.56+70.77	295.69	109.98	65.89	9.77

TABLE 8

Comparison of the *oriented* normal RMSE, the learnable network parameter (million) and the average inference time (seconds per 100K points) of different learning-based methods on the PCPNet dataset.

	HSurf-Net +ODP	AdaFit +ODP	PCPNet	NGLO	Ours	Ours-U +O	Ours-U +NGL
RMSE	31.07	30.93	37.66	18.49	19.79	20.02	18.29
Param.	2.16+0.43	4.87+0.43	22.36	0.46+1.92	3.27	1.76+3.27	1.76+0.46
Time	72.47+236.35	56.23+248.54	63.02	0.56+70.77	65.89	9.77+65.89	9.77+0.56

embedding and compute its MSE loss. Moreover, we also conduct an experiment by removing the weighted mean square error loss of neighboring point normals in Eq. (18) to show its effect on the estimation of query point normal, *i.e.*, without using \mathcal{L}_{mse} .

(d) Point Sampling. In the shape encoding, we obtain a global point set P_q by a probability-based sampling strategy as in Eq. (15), which includes density gradient term and random sample term. The point set P_q includes $N_P = 1200$ points, and the ratio of randomly sampled points is $\zeta = 1/1.5$. (i) We only adopt one of the two terms in Eq. (15) for point sampling, *e.g.*, "w/o density gradient" means all points are randomly sampled and "w/o random sample" means all points are sampled by the density gradient. (ii) The ratio ζ is changed to 1/2 and 1/3. (iii) The number of points N_P is set to 1100 and 1300.

From Table 9, we can conclude that both patch encoding and shape encoding are vital for learning accurate oriented and unoriented normals in our pipeline. The adoption of the weight w and the attention-weighted normal prediction module \mathcal{H} effectively improves the algorithm's performance. Compared to directly predicting the oriented normal \vec{n}_q , solving it separately (normal and its sign) by learning signed hyper surfaces is significantly better. The

TABLE 9

Ablation studies for oriented normals on the PCPNet dataset. The average results under unoriented metric are also provided in the last column.

Ablation	Feat. Enco.	Module \mathcal{H}	Loss	Point Samp.	Noise				Density		Oriented Average	Unoriented Average
					None	0.12%	0.6%	1.2%	Stripe	Gradient		
(a)	w/o patch encoding	✓	✓	✓	35.19	42.23	55.59	61.38	38.92	41.49	45.80	18.83
	w/o shape encoding	✓	✓	✓	69.72	64.37	81.87	77.07	74.84	90.35	76.37	14.94
	w/o weight w	✓	✓	✓	11.15	14.32	26.49	36.03	17.99	26.03	22.00	10.48
(b)	w/o module \mathcal{H}	✓	✓	✓	12.08	14.53	25.87	35.88	18.45	31.84	23.11	10.24
(c)	w/o $\mathcal{L}_{sin}, \mathcal{L}_{sgn}$	✓	✓	✓	23.86	25.55	34.13	42.48	32.42	41.30	33.29	20.23
	w/o \mathcal{L}_{mse}	✓	✓	✓	18.89	29.83	35.61	43.53	24.41	33.71	30.99	10.03
(d)	w/o density gradient	✓	✓	✓	12.10	18.25	28.05	38.15	19.79	28.09	24.07	10.00
	w/o random sample	✓	✓	✓	11.01	13.79	25.64	35.86	17.22	25.71	21.54	9.94
	$\zeta = 1/2$	✓	✓	✓	10.99	14.04	25.66	35.78	17.73	37.82	23.67	9.92
	$\zeta = 1/3$	✓	✓	✓	13.27	15.42	26.82	37.16	17.52	28.11	23.05	9.95
	$N_P = 1100$	✓	✓	✓	10.67	14.21	25.54	35.97	16.80	26.98	21.69	9.99
	$N_P = 1300$	✓	✓	✓	12.44	14.53	25.93	35.79	18.40	19.85	21.16	9.98
	Final	✓	✓	✓	✓	10.28	13.23	25.40	35.51	16.40	17.92	19.79

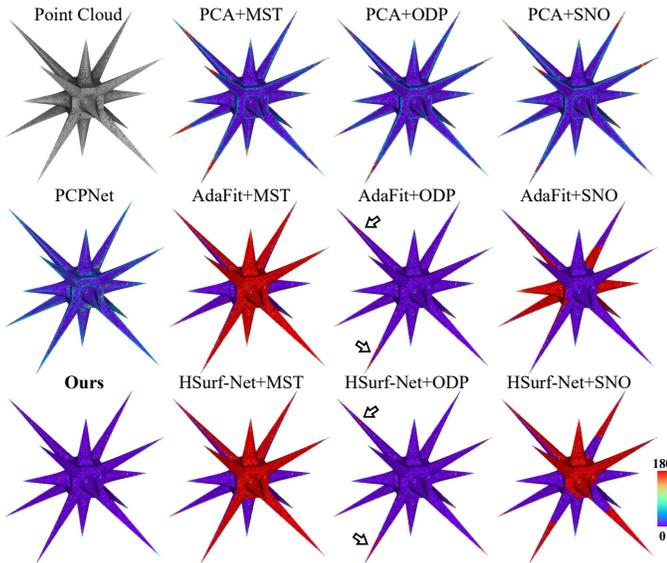


Fig. 9. Visualization of the oriented normal error on a point cloud with sharp features. The angle error is mapped to a heatmap ranging from 0° to 180°. Our method has less error than other baseline methods.

constraints on neighboring point normal help the network model fully explore the local and global geometry of point cloud patches, and ensure the local normal consistency, especially in oriented normal estimation. The combination of the density gradient term and the random sample term in sampling can produce better results than either one alone. The proportion and number of points in the sampling of shape encoding have a significant positive effect on oriented normals, but very little on unoriented normals.

(e) Epoch of Model Training. To determine how many epochs a model needs to train, we evaluate our method on the PCPNet test set using models trained for 100 to 1000 epochs. The estimated normals are measured using the evaluation metrics $RMSE_{unoriented}$ and $RMSE_{oriented}$ of normal angles. The evaluation results are shown in Fig. 13. We provide the results at different noise levels and different density variations along with their average results. It can be seen from the curves in the figure that the errors of unoriented normal evaluation keep decreasing, while the errors of oriented normal evaluation fluctuate greatly. In our training, we observed that the model is harder to converge in oriented normal estimation than in unoriented normal

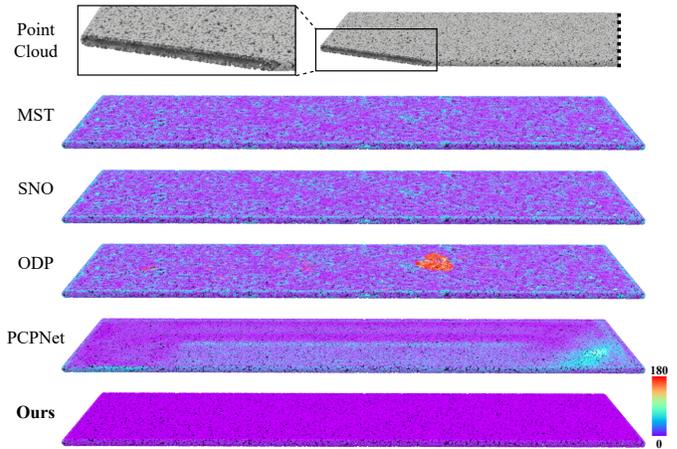


Fig. 10. Visualization of the oriented normal error on a thin sheet with a hollow structure. The initial unoriented normals for methods MST [9], SNO [12] and ODP [15] are provided by PCA. MST and SNO have very similar orientation results, and our method has the least error.

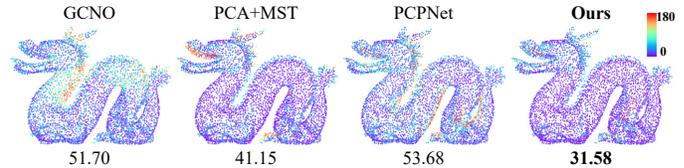


Fig. 11. Visualization of the oriented normal error on a sparse point cloud with 5000 points. The RMSE of the estimated normal is provided for quantitative comparison, and our method has the lowest error.

estimation. After about 800 epochs of training, the errors of oriented normal evaluation reach a minimum value, and the errors of unoriented normal evaluation also reach the lowest value and remain unchanged. Therefore, in all experiments of the paper, we use the model trained in 800 epochs.

6 APPLICATIONS

In the following experiments, we will demonstrate that our method can accurately estimate normals on point clouds with noise, density variations, and complex geometries, thereby facilitating downstream tasks, such as surface reconstruction and point cloud filtering.

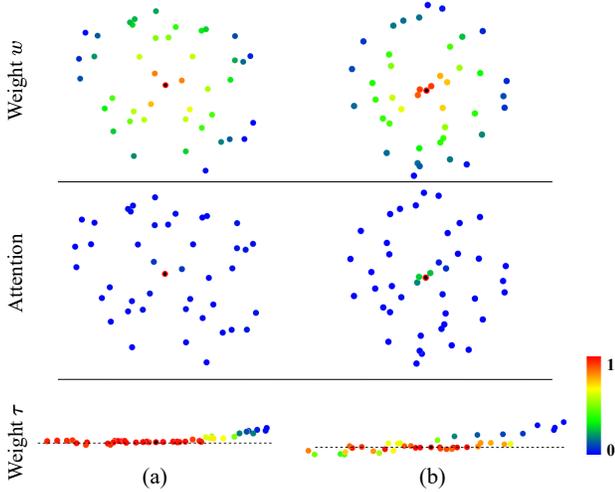


Fig. 12. Visualization of the learned weight w , attention weight, and weight τ in two point cloud patches (a)(b). They show the points that the model focuses on at different stages of the normal estimation process. Specifically, these points are the following three types: (top) the points closer to the center, (middle) the query point and its neighbors, and (bottom) the points coplanar with the query point. The red color indicates that the point has a large value, while the blue color indicates that it has a small value. The black point is the query point of the patch. The first two rows are top views of the patch, and the third row is a side view. The viewing angle of the third row is changed for better visualization.

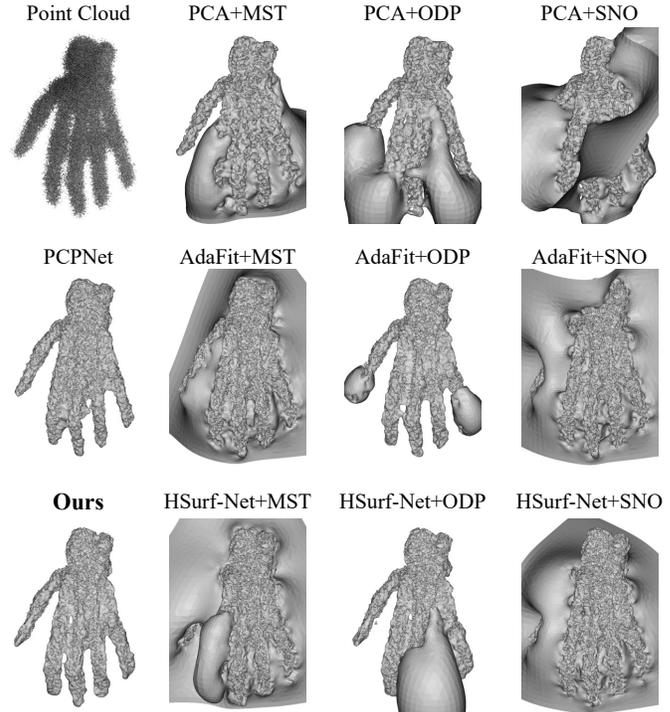


Fig. 14. Comparison of surface reconstruction results from a noisy point cloud using oriented normals estimated by different methods.

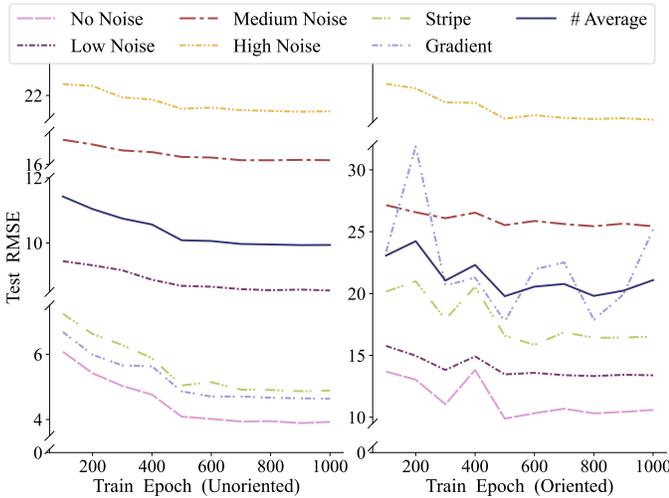


Fig. 13. Unoriented and oriented normal evaluation results on the PCP-Net test set using our models trained for 100 to 1000 epochs. We report the RMSE results under different noise levels and density variations along with their average values. Note that the normals used in the unoriented and oriented normal evaluation are the same, but evaluated using different metrics, *i.e.*, $RMSE_{unoriented}$ and $RMSE_{oriented}$.

6.1 Surface Reconstruction

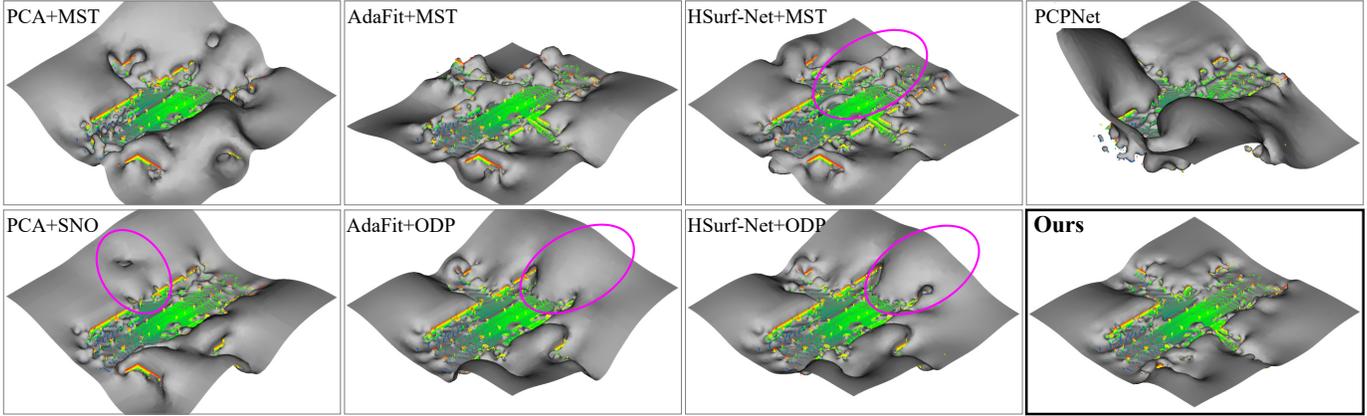
6.1.1 Oriented Normal Estimation Methods

For surface reconstruction, we first compare our method with oriented normal estimation methods. Based on the oriented normals estimated by different methods, we use the Poisson reconstruction algorithm [3] to reconstruct surfaces from point clouds. In Fig. 14, we show a visual comparison of the reconstructed surfaces on a noisy point cloud. We can see that our method helps the Poisson algorithm to reconstruct better surfaces from point clouds with noise and complex geometries compared to the baseline methods.

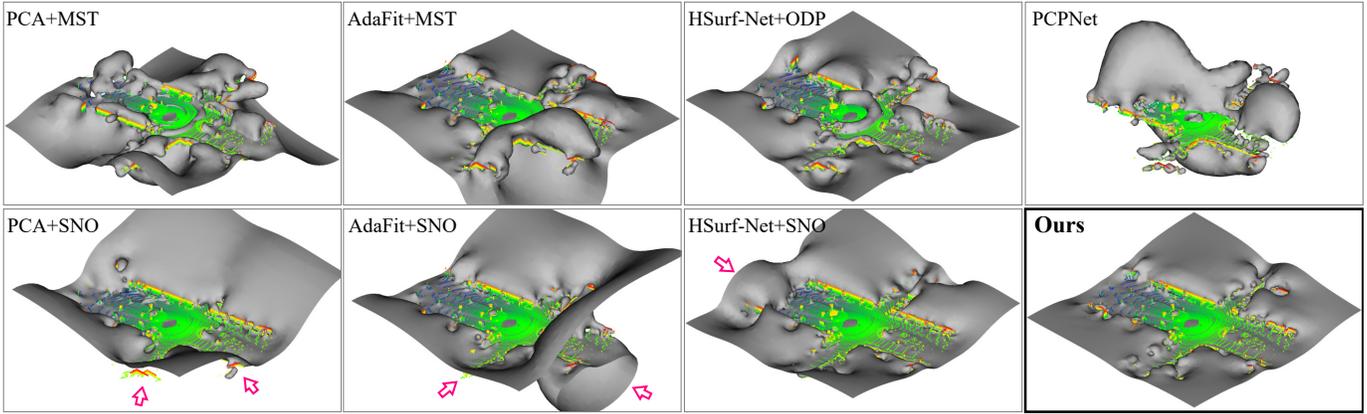
Real-world LiDAR Data. To verify the generalization ability

of our method on LiDAR point clouds of outdoor scenes, we test directly on the KITTI dataset [79] with the network model trained on the PCPNet dataset. The point clouds in this dataset have non-uniform density and open surface structure, which pose a great challenge for oriented normal estimation. We only report qualitative results on this dataset as it does not provide the ground truth normals or surfaces. As shown in Fig. 15, we use the Poisson surface reconstruction algorithm [3] to generate surfaces using oriented normals estimated by different methods. As can be seen in the figures, compared to the baselines, our estimated normals facilitate the algorithm to reconstruct surfaces that can more accurately depict the spatial structure and distribution of real scenes.

Wireframe Point Clouds. The point cloud used above can accurately describe the details of objects or scenes. In addition, we can also use wireframe point clouds to describe the outline of objects, which provide a compact skeletal representation with a very small number of points. Due to the extremely sparse and non-uniform distribution of the data, restoring 3D surfaces from such point cloud data has always been very challenging. As shown in Fig. 16, we visualize the reconstructed surfaces of the baseline methods, such as PCA+MST [9], PCPNet [21] and GCNO [57]. They represent three typical classes of oriented normal estimation methods, namely two-stage pipeline, learning-based single-stage, and traditional scheme-based single-stage. Both Iso-value Constraint (IC) [59] and iPSR [63] can serve as a kind of improved Poisson surface reconstruction, and their optimization of implicit surfaces can also solve oriented normals with globally consistent orientation. The initial unoriented normals of IC [59] are estimated by PCA, *i.e.*, PCA+IC in Fig. 16, and iPSR [63] uses randomly initialized point normals. The experimental results show that our method



(a) Street scene 1 of the KITTI dataset.



(b) Street scene 2 of the KITTI dataset.

Fig. 15. Comparison of reconstructed surfaces using oriented normals estimated by different methods on the sparse and non-uniformly distributed point clouds of the KITTI dataset. The raw point cloud has an open surface structure and is colored with height values. The reconstructed surface of the street scene is shown in gray.

is capable of handling the wireframe-type inputs, and is superior to some latest competitors in certain details.

Complex Topological Data. As shown in Fig. 17, we provide several point clouds with nested structures whose highly complex topology/geometry poses great challenges for normal orientation. Our method shows significant performance improvement over some baseline methods, especially the deep learning-based counterpart, PCPNet. The propagation-based methods do not use globally sampled points to determine the orientation, but instead rely on the orientation propagation of neighboring point normals. Their scheme does not have advantages in some regions with adjacent surfaces or large curvature changes.

6.1.2 Surface Reconstruction Methods

To further evaluate the effect of estimated normals in surface reconstruction, we compare our method with other methods that are designed for surface reconstruction from point clouds. These methods determine the zero-level set of the learned implicit function via the signed distance field, and use the marching cubes algorithm [80] to extract a surface of the point cloud. The baseline methods include Neural-Pull [67], CAP-UDF [68] [81], OSP [69], PCP [82], SAL [65], IGR [66] and Shape As Points (SAP) [70] and their distance fields are predicted through a learning-based pipeline. A visual comparison of the extracted surfaces on point clouds

with different noise levels is shown in Fig. 18. We can see that, based on the accurate normals estimated by our method, the Poisson reconstruction algorithm [3] generates more complete and detailed geometry from noisy point clouds than baseline methods.

6.2 Point Cloud Filtering

Point cloud data collected from the real world are often noisy due to sensors and environments. Therefore, point cloud filtering is often an important preprocessing step before further processing of point clouds. In this evaluation, we use the algorithm proposed in [83] to perform point cloud filtering using the estimated point normals. As shown in Fig. 19, we provide the qualitative and qualitative comparison results of the filtered point clouds and their surfaces reconstructed by Poisson surface reconstruction algorithm [3]. It can be seen that the filtering algorithm can benefit from our estimated normals, and it smooths the surfaces in flat areas while still keeping detailed structures at sharp edges.

7 CONCLUSION

In this work, we formulate the oriented normal estimation of point clouds as the learning of signed hyper surfaces. We first review the explicit surface fitting and the implicit

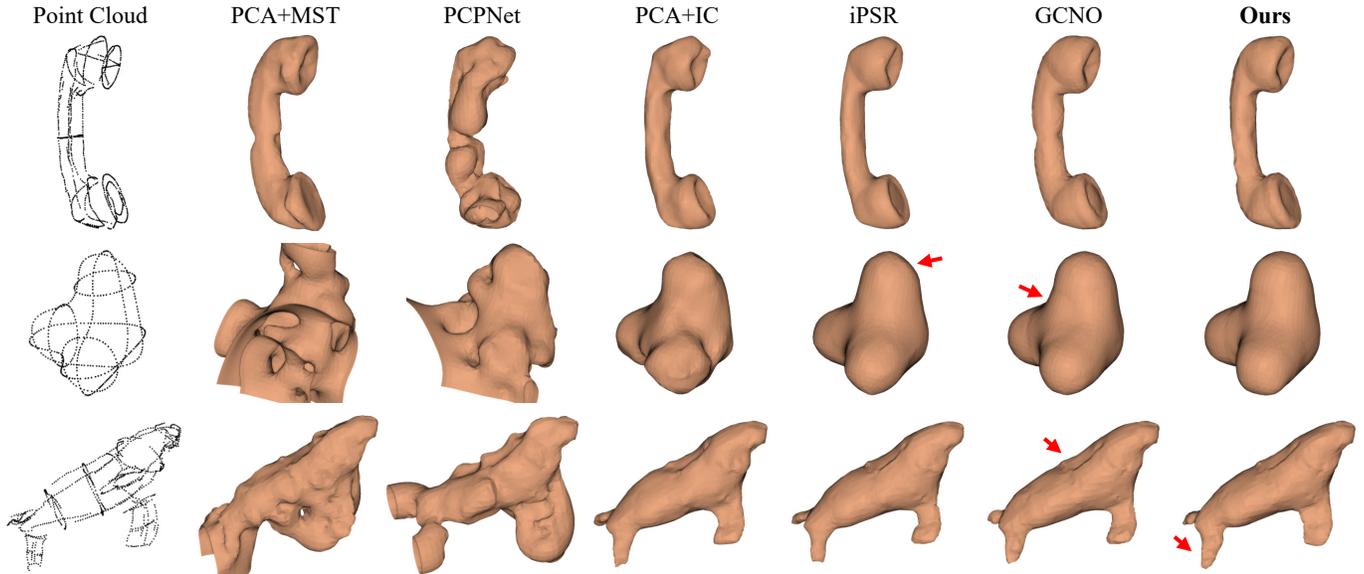


Fig. 16. Oriented normal estimation on wireframe point clouds with sparse and non-uniform sampling. We compare shape surfaces generated by Poisson surface reconstruction. The number of points in the input point cloud is less than 1000. The ground truth is not available.

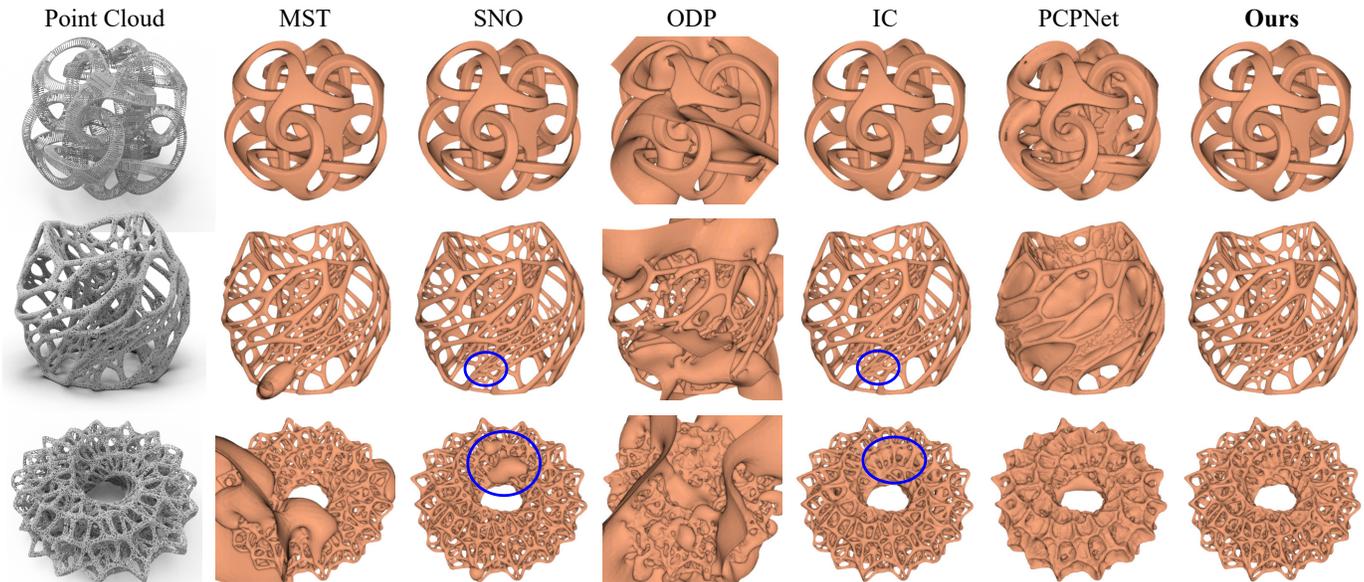


Fig. 17. Oriented normal estimation on point clouds with highly complex topology/geometry. The initial unoriented normals for methods MST [9], SNO [12], ODP [15] and IC [59] are provided by PCA. The point cloud in the first row has about 70K points, while the point clouds in the second and third rows have 100K points. The surfaces are reconstructed using the estimated oriented normals. Our method can provide explicit shape structures from these three point clouds and achieves good results on the first two point clouds. However, all methods fail to distinguish the details inside the model in the third point cloud.

surface learning, and derive the formulation of the signed hyper surfaces from their inspiration. Then, we propose to use an attention-weighted normal prediction module to recover the normal and its sign of the query point from the embedding of the signed hyper surfaces. Finally, we introduce how such surfaces can be learned from the patch encoding and shape encoding using the designed loss functions. We conduct extensive evaluation and ablation experiments to report the state-of-the-art performance and justify the effectiveness of our designs. We show that oriented normal estimation is tightly coupled with surface reconstruction, and that our estimated normals can facilitate surface reconstruction algorithms to generate better object structures.

For the normal estimation task, we make a comprehen-

sive analysis of the problem in theory and make specific designs in technology. It is these important innovative designs that enable our method to outperform the state-of-the-art methods in both unoriented and oriented normal estimation on the widely used benchmarks. In summary, we explore a new idea for learning local features and geometric properties from point clouds. It can better serve the community for point cloud processing and has a positive impact on the performance improvement of downstream tasks using normals. Future work includes developing noise-adaptive techniques to handle more diverse point clouds and integrating our method into recently developed surface reconstruction methods. In addition, the transfer of contextual information between adjacent points or patches is a research direction worth exploring. The limitations of our approach

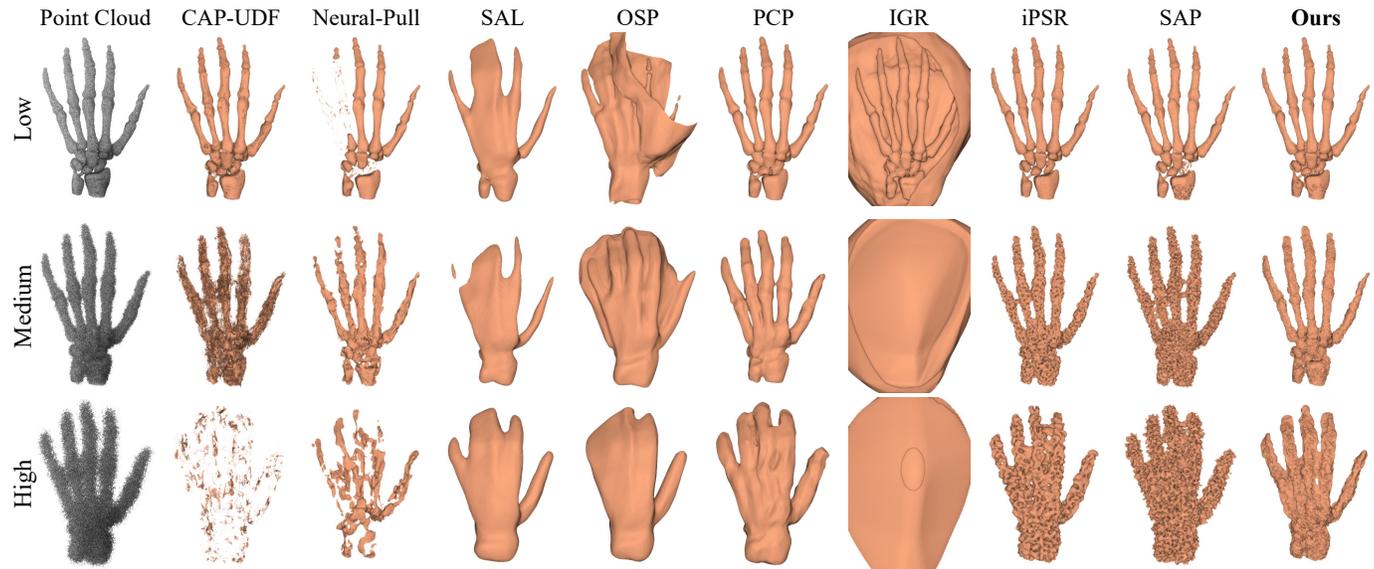


Fig. 18. Comparison with surface reconstruction methods. The input point clouds are with different noise levels (low, medium and high). As the noise increases, our method has the advantage of better performance.

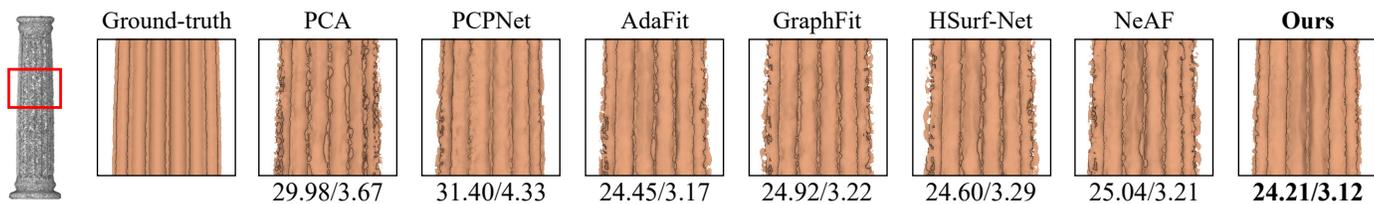


Fig. 19. Comparison of point cloud filtering using normals estimated by different methods. The first column shows the raw point cloud (grey), the surface reconstructed using its filtered point cloud is shown behind with a local zoomed-in view. For a quantitative comparison, the average RMSE of the estimated normals for each point cloud and the Chamfer Distance (CD, $\times 10^{-5}$) of the filtered point cloud are reported separately below the shape in RMSE/CD format.

are discussed in the supplementary material, and some failure cases are also provided.

REFERENCES

- [1] M. Kazhdan, "Reconstruction of solid models from oriented point sets," in *Proceedings of the third Eurographics Symposium on Geometry Processing*, 2005. 1
- [2] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics Symposium on Geometry Processing*, vol. 7, 2006. 1
- [3] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 1–13, 2013. 1, 13, 14
- [4] 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. 1, 3, 8
- [5] 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. 1, 3, 4, 6, 7, 8
- [6] R. Zhu, Y. Liu, Z. Dong, Y. Wang, T. Jiang, W. Wang, and B. Yang, "AdaFit: Rethinking learning-based normal estimation on point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6118–6127. 1, 2, 3, 4, 7, 8, 10, 11
- [7] K. Li, M. Zhao, H. Wu, D.-M. Yan, Z. Shen, F.-Y. Wang, and G. Xiong, "GraphFit: Learning multi-scale graph-convolutional representation for point cloud normal estimation," in *European Conference on Computer Vision*. Springer, 2022, pp. 651–667. 1, 3, 8, 10
- [8] 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," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35. Curran Associates, Inc., 2022, pp. 4218–4230. 1, 2, 4, 7, 8, 10, 11
- [9] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, 1992, pp. 71–78. 1, 2, 3, 7, 8, 11, 12, 13, 15
- [10] L. M. Seversky, M. S. Berger, and L. Yin, "Harmonic point cloud orientation," *Computers & Graphics*, vol. 35, no. 3, pp. 492–499, 2011. 1, 3
- [11] J. Wang, Z. Yang, and F. Chen, "A variational model for normal computation of point clouds," *The Visual Computer*, vol. 28, no. 2, pp. 163–174, 2012. 1, 3
- [12] N. Schertler, B. Savchynskyy, and S. Gumhold, "Towards globally optimal normal orientations for large point clouds," in *Computer Graphics Forum*, vol. 36, no. 1. Wiley Online Library, 2017, pp. 197–208. 1, 2, 3, 8, 11, 12, 15
- [13] M. Xu, S. Xin, and C. Tu, "Towards globally optimal normal orientations for thin surfaces," *Computers & Graphics*, vol. 75, pp. 36–43, 2018. 1, 3
- [14] J. Jakob, C. Buchenau, and M. Guthe, "Parallel globally consistent normal orientation of raw unorganized point clouds," in *Computer Graphics Forum*, vol. 38, no. 5. Wiley Online Library, 2019, pp. 163–173. 1, 3
- [15] G. Metzger, R. Hanocka, D. Zorin, R. Giryes, D. Panozzo, and D. Cohen-Or, "Orienting point clouds with dipole propagation," *ACM Transactions on Graphics*, vol. 40, no. 4, pp. 1–14, 2021. 1, 2, 3, 8, 10, 11, 12, 15
- [16] P. Mullen, F. De Goes, M. Desbrun, D. Cohen-Steiner, and P. Alliez, "Signing the unsigned: Robust surface reconstruction from raw pointsets," in *Computer Graphics Forum*, vol. 29, no. 5. Wiley Online Library, 2010, pp. 1733–1741. 1, 3
- [17] V. c. Mello, L. Velho, and G. Taubin, "Estimating the in/out function of a surface represented by points," in *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*, 2003,

- pp. 108–114. [1](#), [3](#)
- [18] C. Walder, O. Chapelle, and B. Schölkopf, “Implicit surface modelling as an eigenvalue problem,” in *Proceedings of the 22nd International Conference on Machine Learning*, 2005, pp. 936–939. [1](#), [3](#)
- [19] Z. Huang, N. Carr, and T. Ju, “Variational implicit point set surfaces,” *ACM Transactions on Graphics*, vol. 38, no. 4, pp. 1–13, 2019. [1](#), [3](#)
- [20] P. Alliez, D. Cohen-Steiner, Y. Tong, and M. Desbrun, “Voronoi-based variational reconstruction of unoriented point sets,” in *Symposium on Geometry Processing*, vol. 7, 2007, pp. 39–48. [1](#), [2](#), [3](#)
- [21] 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. [2](#), [3](#), [4](#), [7](#), [8](#), [10](#), [11](#), [13](#)
- [22] T. Hashimoto and M. Saito, “Normal estimation for accurate 3D mesh reconstruction with point cloud model incorporating spatial structure,” in *CVPR Workshops*, 2019, pp. 54–63. [2](#), [3](#)
- [23] S. Wang, X. Liu, J. Liu, S. Li, and J. Cao, “Deep patch-based global normal orientation,” *Computer-Aided Design*, p. 103281, 2022. [2](#), [3](#), [8](#), [10](#), [11](#)
- [24] Q. Li, H. Feng, K. Shi, Y. Gao, Y. Fang, Y.-S. Liu, and Z. Han, “SHS-Net: Learning signed hyper surfaces for oriented normal estimation of point clouds,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, June 2023, pp. 13 591–13 600. [2](#)
- [25] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, “Point set surfaces,” in *Proceedings Visualization, 2001. VIS’01.* IEEE, 2001, pp. 21–29. [2](#)
- [26] M. Pauly, M. Gross, and L. P. Kobbelt, “Efficient simplification of point-sampled surfaces,” in *IEEE Visualization, 2002. VIS 2002.* IEEE, 2002, pp. 163–170. [2](#)
- [27] N. J. Mitra and A. Nguyen, “Estimating surface normals in noisy point cloud data,” in *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, 2003, pp. 322–328. [2](#)
- [28] C. Lange and K. Polthier, “Anisotropic smoothing of point sets,” *Computer Aided Geometric Design*, vol. 22, no. 7, pp. 680–692, 2005. [2](#)
- [29] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, “Consolidation of unorganized point clouds for surface reconstruction,” *ACM Transactions on Graphics*, vol. 28, no. 5, pp. 1–7, 2009. [2](#)
- [30] G. W. Stewart, “On the early history of the singular value decomposition,” *SIAM Review*, vol. 35, no. 4, pp. 551–566, 1993. [2](#)
- [31] A. Bouchl and R. Marlet, “Fast and robust normal estimation for point clouds with sharp features,” in *Computer Graphics Forum*, vol. 31, no. 5. Wiley Online Library, 2012, pp. 1765–1774. [2](#)
- [32] N. Amenta and M. Bern, “Surface reconstruction by voronoi filtering,” *Discrete & Computational Geometry*, vol. 22, no. 4, pp. 481–504, 1999. [2](#)
- [33] Q. Mérigot, M. Ovsjanikov, and L. J. Guibas, “Voronoi-based curvature and feature estimation from point clouds,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 6, pp. 743–756, 2010. [2](#)
- [34] T. K. Dey and S. Goswami, “Provable surface reconstruction from noisy samples,” *Computational Geometry*, vol. 35, no. 1-2, pp. 124–141, 2006. [2](#)
- [35] D. Levin, “The approximation power of moving least-squares,” *Mathematics of Computation*, vol. 67, no. 224, pp. 1517–1531, 1998. [2](#)
- [36] 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. [2](#), [4](#), [8](#)
- [37] G. Guennebaud and M. Gross, “Algebraic point set surfaces,” in *ACM SIGGRAPH 2007 papers*, 2007. [2](#)
- [38] S. Aroudj, P. Seemann, F. Langguth, S. Guthe, and M. Goesele, “Visibility-consistent thin surface reconstruction using multi-scale kernels,” *ACM Transactions on Graphics*, vol. 36, no. 6, pp. 1–13, 2017. [2](#)
- [39] A. C. Öztireli, G. Guennebaud, and M. Gross, “Feature preserving point set surfaces based on non-linear kernel regression,” in *Computer Graphics Forum*, vol. 28, no. 2. Wiley Online Library, 2009, pp. 493–501. [2](#)
- [40] A. Bouchl 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. [2](#)
- [41] R. Roveri, A. C. Öztireli, I. Pandele, and M. Gross, “PointProNets: Consolidation of point clouds with convolutional neural networks,” in *Computer Graphics Forum*, vol. 37, no. 2. Wiley Online Library, 2018, pp. 87–99. [2](#)
- [42] D. Lu, X. Lu, Y. Sun, and J. Wang, “Deep feature-preserving normal estimation for point cloud filtering,” *Computer-Aided Design*, vol. 125, p. 102860, 2020. [2](#)
- [43] J. Zhou, H. Huang, B. Liu, and X. Liu, “Normal estimation for 3D point clouds via local plane constraint and multi-scale selection,” *Computer-Aided Design*, vol. 129, p. 102916, 2020. [2](#), [8](#)
- [44] 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 Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 112–10 120. [2](#), [8](#)
- [45] H. Zhou, H. Chen, Y. Feng, Q. Wang, J. Qin, H. Xie, F. L. Wang, M. Wei, and J. Wang, “Geometry and learning co-supported normal estimation for unstructured point cloud,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13 238–13 247. [2](#)
- [46] H. Zhou, H. Chen, Y. Zhang, M. Wei, H. Xie, J. Wang, T. Lu, J. Qin, and X.-P. Zhang, “Refine-Net: Normal refinement neural network for noisy point clouds,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 946–963, 2022. [2](#), [8](#)
- [47] 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. [2](#), [3](#), [8](#), [10](#)
- [48] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660. [2](#)
- [49] H. Xiu, X. Liu, W. Wang, K.-S. Kim, and M. Matsuoka, “MSECNet: Accurate and robust normal estimation for 3D point clouds by multi-scale edge conditioning,” in *Proceedings of the 31st ACM International Conference on Multimedia*, 2023, pp. 2535–2543. [3](#), [8](#)
- [50] Y. Wu, M. Zhao, K. Li, W. Quan, T. Yu, J. Yang, X. Jia, and D.-M. Yan, “CMG-Net: Robust normal estimation for point clouds via chamfer normal distance and multi-scale geometry,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 6, 2024, pp. 6171–6179. [3](#), [8](#)
- [51] J. Cao, H. Zhu, Y. Bai, J. Zhou, J. Pan, and Z. Su, “Latent tangent space representation for normal estimation,” *IEEE Transactions on Industrial Electronics*, vol. 69, no. 1, pp. 921–929, 2021. [3](#), [8](#)
- [52] J. Zhou, W. Jin, M. Wang, X. Liu, Z. Li, and Z. Liu, “Improvement of normal estimation for point clouds via simplifying surface fitting,” *Computer-Aided Design*, p. 103533, 2023. [3](#), [8](#)
- [53] J. Zhang, J.-J. Cao, H.-R. Zhu, D.-M. Yan, and X.-P. Liu, “Geometry guided deep surface normal estimation,” *Computer-Aided Design*, vol. 142, p. 103119, 2022. [3](#), [6](#), [8](#)
- [54] H. Du, X. Yan, J. Wang, D. Xie, and S. Pu, “Rethinking the approximation error in 3D surface fitting for point cloud normal estimation,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. [3](#), [8](#), [10](#)
- [55] S. König and S. Gumhold, “Consistent propagation of normal orientations in point clouds,” in *International Symposium on Vision, Modeling, and Visualization (VMV)*, 2009, pp. 83–92. [3](#)
- [56] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer, “Optimizing binary MRFs via extended roof duality,” in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007, pp. 1–8. [3](#)
- [57] R. Xu, Z. Dou, N. Wang, S. Xin, S. Chen, M. Jiang, X. Guo, W. Wang, and C. Tu, “Globally consistent normal orientation for point clouds by regularizing the winding-number field,” *ACM Transactions on Graphics (TOG)*, 2023. [3](#), [9](#), [13](#)
- [58] Q. Li, H. Feng, K. Shi, Y. Fang, Y.-S. Liu, and Z. Han, “Neural gradient learning and optimization for oriented point normal estimation,” in *SIGGRAPH Asia 2023 Conference Papers*, ser. SA ’23. New York, NY, USA: Association for Computing Machinery, 2023. [3](#), [7](#), [8](#), [10](#), [11](#)
- [59] D. Xiao, Z. Shi, S. Li, B. Deng, and B. Wang, “Point normal orientation and surface reconstruction by incorporating isovalue constraints to poisson equation,” *Computer Aided Geometric Design*, p. 102195, 2023. [3](#), [13](#), [15](#)
- [60] S. Katz, A. Tal, and R. Basri, “Direct visibility of point sets,” in *ACM SIGGRAPH*, 2007, pp. 24–es. [3](#)
- [61] Y.-L. Chen, B.-Y. Chen, S.-H. Lai, and T. Nishita, “Binary orientation trees for volume and surface reconstruction from unoriented point clouds,” in *Computer Graphics Forum*, vol. 29, no. 7. Wiley Online Library, 2010, pp. 2011–2019. [3](#)

- [62] H. Xie, K. T. McDonnell, and H. Qin, "Surface reconstruction of noisy and defective data sets," in *IEEE Visualization*. IEEE, 2004, pp. 259–266. [3](#)
- [63] F. Hou, C. Wang, W. Wang, H. Qin, C. Qian, and Y. He, "Iterative poisson surface reconstruction (iPSR) for unoriented points," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–13, 2022. [3](#), [13](#)
- [64] S. Lin, D. Xiao, Z. Shi, and B. Wang, "Surface reconstruction from point clouds without normals by parametrizing the gauss formula," *ACM Transactions on Graphics*, vol. 42, no. 2, pp. 1–19, 2022. [3](#)
- [65] 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. [3](#), [14](#)
- [66] 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. [3](#), [14](#)
- [67] B. Ma, Z. Han, Y.-S. Liu, and M. Zwicker, "Neural-Pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces," *International Conference on Machine Learning*, 2021. [3](#), [14](#)
- [68] J. Zhou, B. Ma, Y.-S. Liu, Y. Fang, and Z. Han, "Learning consistency-aware unsigned distance functions progressively from raw point clouds," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. [3](#), [14](#)
- [69] 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, pp. 6315–6325. [3](#), [14](#)
- [70] 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, pp. 13 032–13 044, 2021. [3](#), [14](#)
- [71] J. Bloomenthal, C. Bajaj, J. Blinn, M.-P. Cani, B. Wyvill, A. Rockwood, and G. Wyvill, *Introduction to implicit surfaces*. Morgan Kaufmann, 1997. [4](#)
- [72] 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. [4](#)
- [73] 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. [4](#)
- [74] P. Erler, P. Guerrero, S. Ohrhallinger, N. J. Mitra, and M. Wimmer, "Points2Surf: learning implicit surfaces from point clouds," in *European Conference on Computer Vision*. Springer, 2020, pp. 108–124. [5](#), [6](#), [7](#)
- [75] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 1996, pp. 303–312. [7](#)
- [76] B.-S. Hua, Q.-H. Pham, D. T. Nguyen, M.-K. Tran, L.-F. Yu, and S.-K. Yeung, "SceneNN: A scene meshes dataset with annotations," in *2016 Fourth International Conference on 3D Vision*. IEEE, 2016, pp. 92–101. [8](#)
- [77] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5828–5839. [8](#)
- [78] C. (version 2.12) [GPL software], "User documentation," <http://www.cloudcompare.org>, 2022. [9](#)
- [79] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361. [13](#)
- [80] 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. [14](#)
- [81] J. Zhou, B. Ma, S. Li, Y.-S. Liu, Y. Fang, and Z. Han, "CAP-UDF: Learning unsigned distance functions progressively from raw point clouds with consistency-aware field optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. [14](#)
- [82] 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, pp. 6326–6337. [14](#)
- [83] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He, "Low rank matrix approximation for 3D geometry filtering," *IEEE Transactions on Visualization and Computer Graphics*, 2020. [14](#)



Qing Li (Member, IEEE) received his Ph.D. degree from Xiamen University, Xiamen, China in 2021. From 2021 to 2023, he was a Postdoctoral Researcher at the School of Software, Tsinghua University, Beijing, China. He is currently an Associate Researcher at the School of Computing and Artificial Intelligence, Southwest Jiaotong University, China. His research interests include 3D computer vision and point cloud processing.



Huifang Feng received her Ph.D. degree from Xiamen University, Xiamen, China in 2023. She is currently an Associate Professor at the School of Computer and Software Engineering, Xihua University, Chengdu, China. Her research interests include 3D computer vision and point cloud processing.



Kanle Shi received his B.E. and Ph.D. degrees from Tsinghua University, China. He is now working for Kuaishou Technology, Beijing, China, mainly involved in research and development related to computer graphics. His research interests include computer graphics and computer vision.



Yue Gao (Senior Member, IEEE) received his B.E. degree from the Harbin Institute of Technology, Harbin, China, and his M.E. and Ph.D. degrees from Tsinghua University, China. He is currently an Associate Professor at the School of Software, Tsinghua University. His research interests include hypergraph computing, visual perception and augmentation, and data analysis and mining.



Yi Fang received his Ph.D. degree from Purdue University, West Lafayette in 2011. He is currently an Assistant Professor of Electrical and Computer Engineering, New York University Abu Dhabi. His research interests include 3D computer vision and machine learning.



Yu-Shen Liu (Member, IEEE) received his B.S. degree in mathematics from Jilin University in 2000, and his Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University in 2006. From 2006 to 2009, he was a Postdoctoral Researcher at Purdue University. He is currently an Associate Professor at the School of Software, Tsinghua University. His research interests include shape analysis, pattern recognition and machine learning.



Zhizhong Han received his Ph.D. degree from Northwestern Polytechnical University, China in 2017. He was a Postdoctoral 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.