# LaPDA: Latent-Space Point Cloud Denoising With Adaptivity

Peng Du, Xingce Wang, Zhongke Wu, Xudong Ru, Xavier Granier, and Ying He

***Abstract*—Point cloud denoising is a fundamental yet challenging task in computer graphics. Existing solutions typically rely on supervised training on synthesized noise. However, real-world noise often exhibits greater complexity, causing learning-based methods trained on synthetic noise to struggle when encountering unseen noise–a phenomenon we refer to as noise misalignment. To address this challenge, we propose *LaPDA (Latent-space Point cloud Denoising with Adaptivity)*, a neural network explicitly designed to mitigate noise misalignment and enhance denoising robustness. LaPDA consists of two key stages. First, we adaptively model noise in the latent space, aligning unseen noise distributions with the known training distributions or adjusting them toward distributions with lower noise scales. Training objectives at this stage are formulated based on controlled synthetic noise with varying intensity levels. Second, we introduce a gradual noise removal module that optimizes the spatial distribution of the adaptively adjusted noisy points. Extensive experiments conducted on both synthetic and scanned datasets demonstrate that LaPDA achieves enhanced accuracy and robustness compared to state-of-the-art methods.**

***Index Terms*—Point cloud denoising, 3D deep learning, noise modeling, adaptive denoising, latent space.**

## I. Introduction

POINT cloud directly output of 3D acquisition using sensors such as LiDAR. The data obtained from these sensors often contain random noise and uneven point distributions caused by inherent sensor inaccuracies, environmental interference, and reflective surface properties. Consequently, point cloud denoising has become a fundamental research topic in computer graphics and 3D vision. Producing high-quality denoised point clouds is crucial for downstream tasks such as 3D reconstruction, scene understanding, autonomous driving, and urban modeling, to name a few.

Received 28 May 2025; revised 12 October 2025; accepted 10 November 2025. Date of publication 21 November 2025; date of current version 6 February 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62072045, in part by the Beijing Municipal Science and Technology Commission, and the Zhongguancun Science Park Management Committee under Grant Z221100002722020, in part by the Natural Science Foundation of Beijing under Grant 7242167, and in part by the Ministry of Education, Singapore, under its Academic Research Fund under Grant RT19/22. Recommended for acceptance by J. Digne. *(Corresponding author: Xingce Wang.)*

Peng Du, Xingce Wang, Zhongke Wu, and Xudong Ru are with the School of Artificial Intelligence, Beijing Normal University, Beijing 100875, China (e-mail: wangxingce@bnu.edu.cn).

Xavier Granier is with Institut d'Optique Graduate School, 91127 Paris, France.

Ying He is with the College of Computing Science and Data Science, Nanyang Technological University, Singapore 639798.

Our source code is available at https://github.com/ricodp54/LaPDA.

Digital Object Identifier 10.1109/TVCG.2025.3635138

Over the past decade, significant progress in point cloud denoising has come from both traditional geometric approaches and modern deep learning-based techniques. Conventional methods typically rely on geometric priors to recover the underlying surface of noisy point clouds. These methods fall into two categories: moving least squares (MLS)-based [1], [2], [3], [4], [5] and locally optimal projection (LOP)-based [6], [7], [8], [9]. MLS-based approaches explicitly compute surface parameters to define a smooth manifold, whereas LOP-based methods iteratively project noisy points onto the underlying surface by reweighting local neighborhoods. While effective, these methods often require careful parameter tuning and may oversmooth geometric details.

Recently, deep learning has emerged as a powerful alternative, offering data-driven solutions that learn denoising directly from examples. Most learning-based methods adopt supervised training on clean point clouds corrupted by *synthetic noise*, typically Gaussian. Although this setup enables networks to learn robust mappings for controlled scenarios, it introduces a critical gap: real-world noise is far more diverse and complex than synthetic noise distributions. It is infeasible to anticipate and model all possible noise types during training, and attempting to do so can even destabilize learning. As a result, when the noise encountered during inference deviates in type or scale from the training distribution, existing methods often fail to fully restore surfaces, leaving residual noise or causing point clustering.

We refer to this gap between training assumptions and inference conditions as the noise misalignment phenomenon, illustrated in Fig. 1. Addressing this challenge requires not only better denoising models, but also mechanisms that can adapt to previously unseen noise distributions at test time, ensuring both accuracy and robustness. Similar challenges have been studied in 2D image denoising, where self-supervised approaches [10], [11], [12] learn to handle unseen noise by constructing noise pairs via downsampling. However, directly extending such techniques to point clouds is difficult, as perturbations disrupt point-to-point correspondence and alter geometry.

To overcome these challenges, we propose *LaPDA (Latent-space Point cloud Denoising with Adaptivity)*, a two-stage neural network that operates in latent space. Unlike PDLTS [13], which assumes a bijective mapping to disentangle noise into independent latent dimensions–an assumption prone to failure under high noise intensities–LaPDA introduces an adaptive noise modeling stage followed by a dedicated denoising stage. The adaptive stage reweights input noise distributions through learnable perturbations in latent space, aligning unseen noise with

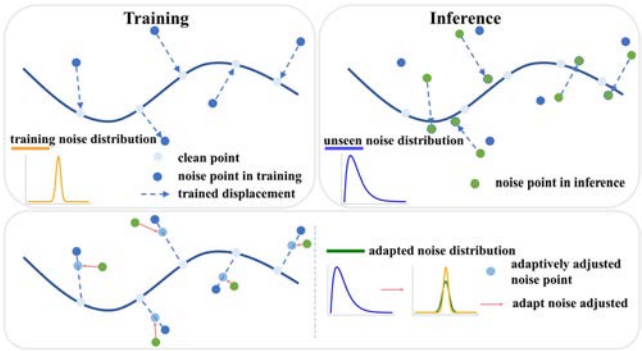


Fig. 1. Illustration of the noise misalignment phenomenon in deep learning-based point cloud denoising. *Top:* Existing methods typically learn to denoise specific synthetic noise distributions during training. At inference time, these learned displacement vectors may fail to accurately restore points when encountering unseen, complex noise distributions, resulting in residual noise or surface inaccuracies. *Bottom:* Our proposed method addresses this issue by adaptively aligning unseen noise to known training distributions or reducing its scale through adaptive latent-space noise modeling, enabling more effective and robust point cloud denoising.

known training distributions or reducing it to lower scales. The denoising stage then refines the adjusted point cloud. By training the adaptive stage with multi-scale Gaussian noise as targets, LaPDA learns to handle complex and previously unseen noise while producing well-distributed point sets. Extensive experiments on both synthetic and real-world datasets demonstrate that LaPDA effectively mitigates noise misalignment and achieves improved accuracy and robustness compared to state-of-the-art methods.

Our main contributions can be summarized as follows:

- To the best of our knowledge, we are the first to identify and explicitly formulate the *noise misalignment issue* in point cloud denoising.
- We propose LaPDA, a novel two-stage latent-space denoising method that models and adapts noise to handle complex and previously unseen noise distributions robustly.

## II. Related Work

### A. Conventional Methods

Conventional methods denoise point clouds by recovering smooth underlying surfaces [14]. Early approaches, inspired by the MLS [5], led to a series of works that incorporated feature preservation and adaptive neighborhoods to mitigate the over-smoothing inherent in MLS-based methods [1], [3], [4]. However, MLS-based methods and Jet [15] require explicit surface parameter estimation. To overcome this limitation, Lipman et al. proposed the LOP [6], which employs L1 median optimization to represent the underlying surface with a set of points and project arbitrary point sets onto target sets. Subsequent extensions, such as Weighted-LOP (WLOP) [7], [16] and Continuous-LOP (CLOP) [9], improved point uniformity and computational efficiency. Nonetheless, these point-based methods often require large neighborhoods to achieve stable smoothing at high noise levels, which can lead to a loss of fine geometric details.

Some methods incorporate normal information to guide denoising. Normals are typically estimated using principal components analysis (PCA) [17] from noisy point clouds. EAR [18] leverages initial normals to repel points located on opposite sides of edges, thereby preserving sharp features. GPF [19] replaces distance-based weighting functions with point-to-tangent distances to better maintain features. Among other normal-based methods, Fleishman et al. introduced bilateral filtering for mesh denoising [20], while Mattei et al. incorporated the similarity of normals within local neighborhoods into the weighting function to preserve sharp features [21]. However, these methods overlook the mutual influence between noise and normal quality, relying heavily on the accuracy of the initial normals.

### B. Learning-Based Methods

*Early networks:* Early approaches, built on PointNet [22] and PointNet++ [23], demonstrated the potential of deep learning with simple architectures and strong representation ability. Some works enhance geometric detail by introducing shape descriptors [24], [25], [26], [27]. For example, PointCleanNet (PCN) [28] proposed a two-stage network to separately remove outliers and surface noise. Pointfilter [29] incorporated normal into the loss function and achieved smooth denoising through bilateral filtering. These methods primarily relied on multilayer perceptrons (MLP) for feature extraction and network design, focusing on individual point information while not fully leveraging local neighborhood. In addition, the first unsupervised approaches for point cloud denoising were introduced but were found to be sensitive to large-scale noise [30]. With the advent of DGCNN [31], dynamic edge convolution became widely adopted, providing richer contextual information. Based on this, Luo et al. proposed Differentiable Manifold Reconstruction (DMR) [32], which reconstructs surfaces by selecting clean points within patches, though downsampling often caused detail loss and amplified local noise.

*Direct prediction versus iterative refinement:* A major distinction among learning-based methods lies in their denoising strategy. Direct prediction methods regress clean coordinates from noisy inputs by extracting point-wise features, often improving loss functions [29] or feature representations [32], [33]. However, these approaches generally underperform compared to iterative methods, which progressively separate noise from geometry. ScoreDenoise [34] and DeepRS [35] treat noisy points as convolutions of clean surfaces with noise, iteratively updating using score functions. IterativePFN [36] extends this idea into the network structure itself, with multiple modules that target noise at different scales, achieving strong results but at higher computational cost. Recently, PDLTS [13] introduced a latent-space noise disentanglement method using invertible operators, effectively addressing issues such as patch shrinkage and surface collapse, which commonly arise when denoising directly in the spatial domain.

*Displacement-based methods:* These methods learn local features and regress the offset between noisy and clean points. GeoDualCNN [37] introduces geometric priors to predict point coordinates through homogeneous neighborhoods, but this also

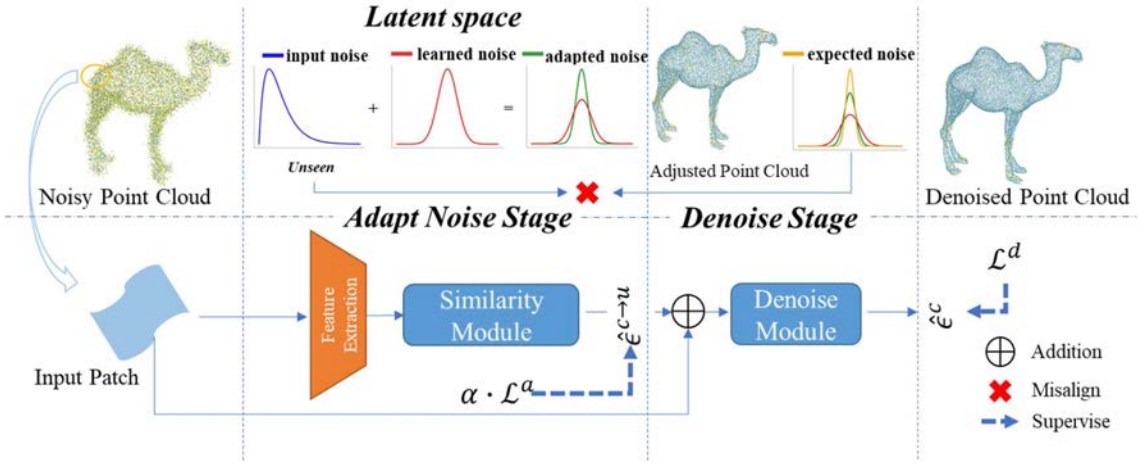


Fig. 2. Illustration of our network structure, comprising the adaptive noise phase and the denoising phase. In the adaptive noise stage, it learns the noise to derive $\hat{\epsilon}^{c\to u}$, correcting unseen noise into a certain noise distribution. The denoising stage predicts the offset $\hat{\epsilon}^c$ between the adjusted point cloud and the clean point cloud.

limits the end-to-end performance. PCDNF [33] integrates extracted point features and normal features to enhance the mutual influence between point and normal information, followed by point coordinate regression. However, the quality of the initial normal estimation impacts the denoising performance and the number of iterations required by PCDNF [33]. Similarly, Silva Edirimuni et al. proposed CFilter [38], which combines normal estimation for point cloud denoising. Subsequently, they introduced IterativePFN [36], emphasizing that iterative denoising of point clouds should not only occur during inference but also be incorporated within the network structure. IterativePFN [36] achieves competitive performance by employing target point clouds with varying noise scales in different iterative modules within the network, progressively removing noise.

*Probability-based methods:* Another line of work models noise probabilistically. Flow-based methods use invertible mappings [13], [39] or optimal transport [40], [41], [42] to align noisy and clean distributions. Score-based methods, such as ScoreDenoise [34] and DeepRS [35], are inspired by the backward Langevin dynamics, while recent variants [43] combine score-based diffusion with boundary preservation.

*Latent-space methods:* Recent works [13], [44], [45] denoise by learning noise-free features in latent space, reducing problems like surface drift or collapse. While effective, these methods generally assume clean/noisy disentanglement is feasible, which becomes difficult under high or unseen noise levels. In contrast, our approach leverages latent space differently: instead of directly regressing clean features, we learn the noise distribution itself and adaptively align it to known or lower-scale distributions, thereby addressing the noise misalignment problem.

## III. Method

### A. Overview

Given a noisy point cloud, we begin by formulating the denoising problem in Section III-B and highlighting the issue of noise misalignment. In Section III-C, we propose a two-stage strategy comprising an adaptive noise stage, which operates in latent space to adjust input points, followed by a denoising stage. Section III-D describes the network architecture and Section III-E presents the joint loss function that supervises both stages during training. An overview of our method is illustrated in Fig. 2.

### B. Problem Formulation

For a noisy patch $X = \{x_i\}_{i=1}^N$, it is disturbed by random noise $\epsilon$ from a clean patch $Y = \{y_j\}_{j=1}^N$. In learning-based methods [34], [35], [36], $\epsilon$ is typically sampled from a certain noise distribution in training stage, such as Gaussian noise:

$$Y = X^c + \epsilon^c, \epsilon^c \sim \sigma^2 \mathcal{N}(0, I), \tag{1}$$

where $\sigma^2$ is typically related to the diagonal of the input model's bounding box. At this point, both the noisy point cloud and the noise distribution are certain, denoted as $X^c$ and $\epsilon^c$, respectively. The denoising network learns a displacement $\epsilon^c$ to pull the noisy point cloud $X^c$ back to the clean surface through the following objective function:

$$\theta^d = \underset{\theta}{\operatorname{argmin}} E\left[\left\| f_\theta^d\left(X^c\right) - \epsilon^c \right\|_2^2\right], \tag{2}$$

where $f_\theta^d(\cdot)$ is the neural network based model parameterized by $\theta$.

*Noise Misalignment in Point Cloud:* However, during inference, the complexity of noise distributions may lead to previously unseen noise $\epsilon^u$ and noisy point clouds $X^u$, where $X^u \neq X^c$. Now that, we can obtain:

$$\left(X^u + \epsilon^c\right) \neq \left(X^c + \epsilon^c\right). \tag{3}$$

This poses a challenge, the displacement $\epsilon^c$ learned from (2) during training may fail to pull the unseen noisy point cloud $X^u$ back to the expected clean surface. This misalignment of noise between training and inference reduces the generalization ability of the denoising network to some extent.

The core of this issue originates from the offset between $\epsilon^c$ and $\epsilon^u$, which are sampled from two distinct noise distributions. The noise offset can be expressed as $\epsilon^{c\to u} = (\epsilon^u - \epsilon^c)$. Our motivation is to address the misalignment in (3) by learning $\epsilon^{c\to u}$ before feeding the unseen noisy point cloud $X^u$ into the denoising network, thereby obtaining the corresponding clean point cloud:

$$\begin{aligned} Y &= X^u + \epsilon^u \\ &= X^u + \epsilon^{c\to u} + (\epsilon^u - \epsilon^{c\to u}) \\ &= (X^u + \epsilon^{c\to u}) + \epsilon^c. \end{aligned}$$

Similarly, the network for learning adaptive noise offset $\epsilon^{c\to u}$ is trained using the following objective function:

$$\theta^a = \underset{\theta}{\operatorname{argmin}} E\left[\left\| f_\theta^a\left(X^u\right) - \epsilon^{c\to u}\right\|_2^2\right], \tag{4}$$

where $f_\theta^a(\cdot)$ is a neural network parameterized by $\theta$ in the adaptive noise stage. Consequently, the objective function for training the entire adaptive noise point cloud denoising network is as follows:

$$\left(\theta^a, \theta^d\right) = \underset{\theta}{\operatorname{argmin}} E\left[\left\| f_\theta^d\left(f_\theta^a\left(X^u\right) + X^u\right) - \epsilon^c\right\|_2^2\right]. \tag{5}$$

### C. Learning Adaptive Noise

The objective function for two stages is clearly defined in form (5), but in practice, it faces the challenge of how to effectively learn the noise distribution from the noisy input. Some prior works in image denoising [10], [46], [47] address this by learning a residual noise image that is added to the input image. Through various downsampling transformations, they generate a pair of adapted image and target image, and train the residual noise image by minimizing the difference between them. When directly applied to point cloud denoising, it can be formulated as:

$$\begin{aligned} \theta^a &= \underset{\theta}{\operatorname{argmin}} E\left[\left\| T_1\left(X'\right) - T_2\left(X'\right)\right\|_2^2\right] \\ &\text{with } X' = X^u + f_\theta^a\left(X^u\right), \end{aligned} \tag{6}$$

where $T_1(\cdot)$ and $T_2(\cdot)$ represent different transformations, specifically downsampling in (6). However, for unstructured point clouds, this approach disrupts the original point-to-point correspondence. Most importantly, unlike image noise, the noise of point cloud is not cumulative. Consecutive noise perturbations at a single point do not necessarily result in increasingly severe point displacement.

*Adaptive noise stage:* Directly learning the noise distribution in the original point cloud representation space, typically $\mathbb{R}^3$, by adding coordinate offsets is challenging because the noise simultaneously affects both the range and domain of the data [30]. Recent works, such as SVCNet [44] and FCNet [45], address point cloud denoising by isolating noise-free features from noisy ones to regress clean point coordinates. This insight motivates us to learn the noise distribution in the embedding space of the point cloud rather than its original representation space, thereby avoiding the aforementioned challenges. It is important to note that while SVCNet [44] and FCNet [45] directly learn the representation of clean point clouds from noisy features, our approach focuses on learning the noise itself from the noisy point cloud.

Specifically, we first encode the unseen noisy input $X^u$ to obtain its feature representation in the latent space. Then, we add a learnable noise perturbation $\Delta h$ into the embedded features. By generating a pair of noise-perturbed features, we train the network to capture the similarity between these two noise features, enabling it to learn an adaptive understanding of the underlying noise distribution:

$$\begin{aligned} h^{c\to u} &= \text{Similarity}\left(T_1\left(X^u\right), T_2\left(X^u\right)\right) \\ \text{with } T_1\left(X^u\right) &= \text{FeatureExtraction}\left(X^u\right) \\ \text{and } T_2\left(X^u\right) &= T_1\left(X^u\right) + \Delta h, \end{aligned}$$

where, $h^{c\to u}$ refers to the noise distribution learned in the latent space and $\Delta h$ represents a learnable perturbation feature that is directly added as offset to the embedded features. Similarity$(\cdot)$ and FeatureExtraction$(\cdot)$ are the similarity function and the feature extraction function, respectively, both parameterized by neural networks.

The learned noise feature $h^{c\to u}$ in the latent space is then decoded back to the $\mathbb{R}^3$ space to match the training process defined in (4).

This brings up another challenge in implementing (4) — namely, how to construct the target noise offset $\epsilon^{c\to u}$ during training. Referring to (1), when the noise distribution $\epsilon^c$ added during training is a certain Gaussian distribution, $\sigma^2$ characterizes the noise scale. During training, the input certain noise is represented as $\epsilon^c(\sigma^2)$, which corresponds to the training objective $\epsilon^{c\to u} = \epsilon^c(s\sigma^2)$ for constructing $f_\theta^a$. Here, $s$ is a scaling parameter that controls the relative scale of the noise compared to the original noise. By training in this manner, the previously encountered noise distributions, such as Gaussian noise, are adjusted to a lower scale, or unseen random noise is aligned with Gaussian noise, thereby avoiding the misalignment phenomenon in (3).

*Denoising stage:* We require the denoising stage to reliably learn the noise displacement from a known noise distribution in order to recover clean point clouds. Therefore, we first pre-train $f_\theta^d(\cdot)$ independently on a certain noise distribution according to the objective function in (2). Finally, when jointly training both stages, the noise point cloud adjusted by the adaptive noise stage will be aligned with the expected noise distribution for the denoising stage.

### D. Structure of Network

The overall pipeline of our network is illustrated in Fig. 2, which is mainly divided into two stages: the adaptive noise stage and the denoising stage. Given a noisy input patch $X^u = \{x_i^u\}_{(i=1)}^N$, we first encode it into a latent space using a feature extraction module. Following [36], we use the Dynamic EdgeConv [31] for feature extraction. Specifically, for the $(l+1)$-th layer, we first construct a graph via k-nearest neighbors (KNN), where the neighborhood size is set to $k = 32$. Then update the

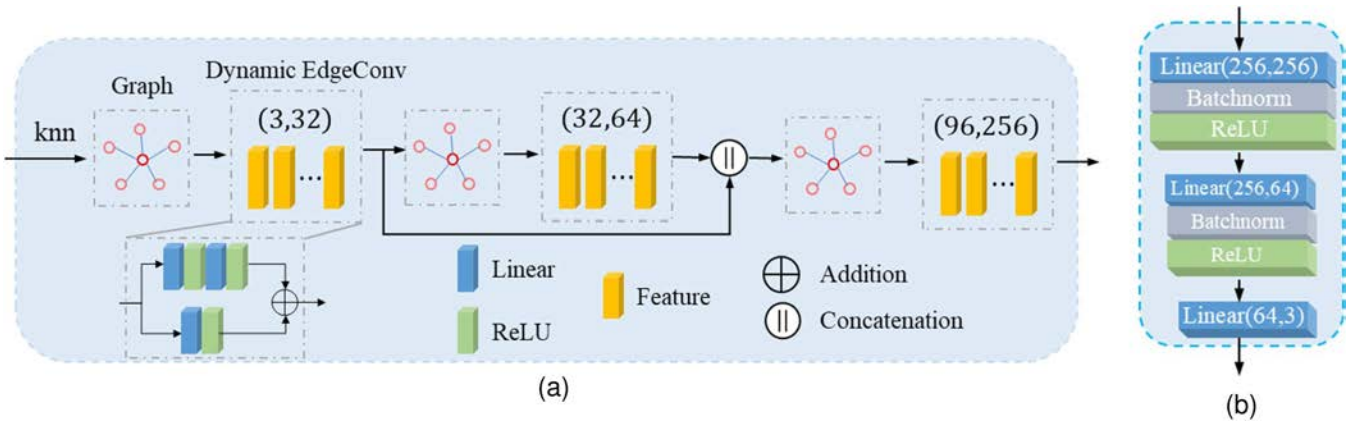


Fig. 3. Illustration of our module. (a) The FeatureExtraction module consisting of three layers of Dynamic EdgeConv, where the input to the final layer concatenates the outputs of the first two layers to enhance contextual information. (b) The Decoder module composed of three linear layers, which decodes from the high-dimensional latent space back to $\mathbb{R}^3$.

feature $h_i^{l+1}$ of each vertex to obtain the $(l+1)$-th layer features $h^{l+1}$ ($h^1 = X^u$):

$$h^{l+1} = \left\{h_i^{l+1}\right\}_{i=1}^{N} \in R^{N\times d_{l+1}},\ l = 1, 2, \ldots, L-1$$
$$\text{with } h_i^{l+1} = \text{MLP}\left(h_i^l\right)$$
$$+ \sum_{j\in NN(i)} \text{MLP}(h_i^l, h_j^l - h_i^l),$$

where, MLP stands for Multi-Layer Perceptron, $d_{l+1}$ represents the dimensionality of the $(l+1)$-th layer, and the neighborhood of the $i$-th vertex is denoted as $NN(i)$. L denotes the index of the last Dynamic EdgeConv layer. The input to the final layer of Dynamic EdgeConv is the concatenation of the output features from the previous two layers, enabling the generation of features with richer contextual information. We add a perturbation noise feature $\Delta h \in R^{N\times d_L}$ to the input noise feature $h^L$ obtained in the latent space, and denote the adapted feature as $h^A$:

$$h^A = h^L + \Delta h.$$

For the obtained noise feature pair $h^A$ and $h^L$, we learn their similarity via the Similarity module shown in Fig. 4. Specifically, we apply two cross-attention mechanisms [48], where $h^A$ and $h^L$ serve as query features, respectively. Through this process, the noise feature pair is re-weighted to capture the similarity between them:

$$h'^A = \text{CrossAttention}\left(h^L, h^A, h^A\right)$$
$$= \text{SoftMax}\left(\frac{h^L \cdot h^A}{\sqrt{d}}\right) \cdot h^A$$
$$\text{and } h'^L = \text{CrossAttention}\left(h'^A, h^L, h^L\right)$$
$$= \text{SoftMax}\left(\frac{h'^A \cdot h^L}{\sqrt{d}}\right) \cdot h^L.$$

The reweighted features integrate information from the original noise feature pairs and are further normalized using LayerNorm and refined through an MLP to produce the learned noise

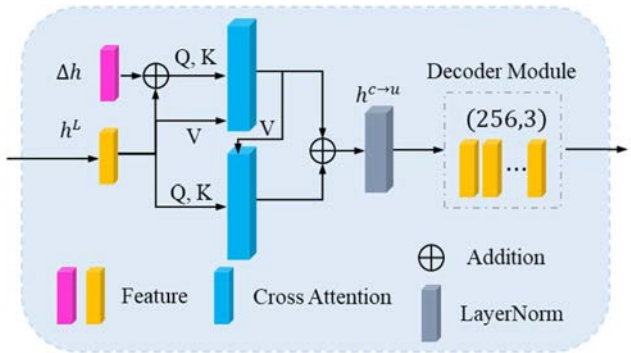


Fig. 4. Illustration of our Similarity Module. For the input noise feature $h^L$ in the latent space, we obtain a noise pair by adding a learnable perturbation feature $\Delta h$. The similarity between them is learned and reweighted to obtain the adjusted noise representation $h^{c\to u}$ in the latent space. Finally, it is decoded back to $\mathbb{R}^3$.

distribution $h^{c\to u}$ in the embedding space,

$$h^{c\to u} = \text{MLP}\left(\text{LayerNorm}\left(h'^A, h'^L\right)\right).$$

Finally, a linear layer is applied to decode the adapted noise feature $h^{c\to u}$, producing the estimated adaptive noise offset $\hat{\epsilon}^{c\to u}$ in the original point cloud space $\mathbb{R}^3$. This offset is then residually added to the input noisy point cloud $X^u$, resulting in the expected noisy point cloud for the denoising stage during pretraining. Finally, the adapted noisy point cloud is fed into the denoising stage, where Dynamic EdgeConv is used to extract features, followed by three linear layers to decode and predict the corresponding point-wise displacement $\hat{\epsilon}^c$, as illustrated in Fig. 3.

### *E. Loss Function*

Recalling (4), the training loss function for the network parameters $\theta^a$ in the adaptive noise stage can be denoted as

$$\mathcal{L}^a = E\left[\left\|\hat{\epsilon}^{c\to u} - \epsilon^{c\to u}\right\|_2^2\right]. \quad (7)$$

When the scale of the input noise $\epsilon^c$ is $\sigma^2$, the noise scale of $\epsilon^{c\to u}$ is determined by $s\sigma^2$. However, $\epsilon^{c\to u}$ is not sampled from a new distribution but instead obtained by directly scaling the input noise $\epsilon^c$ using the scaling parameter $s$. This is because, for a specific point in the point cloud, the perturbation trajectories of two noise instances may be entirely opposite. Let $\epsilon^{c\to u} = \epsilon^c(s\sigma^2) = s{\cdot}\epsilon^c(\sigma^2)$, where $s$ is a random number between 0 and 1. Thus, (7) can be rewritten as:

$$\mathcal{L}^a = E\left[\|\hat{\epsilon}^{c\to u} - s{\cdot}\epsilon^c\|_2^2\right], s{\in}(0,1).$$

After the adaptive noise stage, the original input $X$ is refined by $\hat{\epsilon}^c$. Accordingly, $\epsilon^c$ in (2) must also be refined to construct the training loss function for the network parameters $\theta^d$ in the denoising stage:

$$\mathcal{L}^d = E\left[\|\hat{\epsilon}^c - (Y - (X + \hat{\epsilon}^{c\to u}))\|_2^2\right].$$

Finally, the overall training loss function for the entire network is denoted as $\mathcal{L}$,

$$\begin{aligned}\mathcal{L} &= \alpha{\cdot}\mathcal{L}^a + \mathcal{L}^d \\ &= \alpha{\cdot}E\left[\|\hat{\epsilon}^{c\to u} - s{\cdot}\epsilon^c\|_2^2\right] + E\left[\|\hat{\epsilon}^c - (Y - (X + \hat{\epsilon}^{c\to u}))\|_2^2\right].\end{aligned}$$

We set $\alpha = 10$ to balance the two parts of the loss function, ensuring they remain on the same order of magnitude.

We did not specifically design a complex network for the denoising stage; instead, it remains small and straightforward. To enhance stability during iterative denoising, we adopt an Euler method for gradual approximation instead of directly adding the inferred displacements during inference, as demonstrated in [40]. This process is represented by

$$X_{t+1} = X_t + \frac{1}{K}\epsilon_t, t = (1, 2, \ldots, K-1). \tag{8}$$

In our experiments, setting $K = 3$ yields the best performance.

## IV. Experiments and Results

In this section, we conduct both quantitative and qualitative comparisons with state-of-the-art denoising methods on two synthetic datasets, PU-Net [49] and PCNet [28], as well as real-world scanned datasets, including Paris-Rue-Madame [50], KITTI-360 [51], Kinect v2 [52] and UrbanBIS [53].

### A. Setup

*Implementation:* All experiments were conducted using PyTorch on an NVIDIA GeForce RTX 3090 GPU. The pretraining of the denoising stage and the full network training follow the same settings. We used the Adam optimizer with a learning rate of $1 \times 10^{-4}$ . The network is trained and evaluated on the PU-Net dataset at a resolution of 10K points, with an initial noise level of 1.5%. For quantitative evaluation, we adopt standard metrics used in prior point cloud denoising works [34], [35], including Chamfer Distance (CD) [54], Point-to-Mesh distance (P2M) [55], and Hausdorff distance (HD) [39] both implemented using PyTorch3D [56]. CD and HD measure the point-to-point distance to the clean point, while P2M computes the mean distance to the underlying surface. Furthermore, we evaluated point uniformity (UNI) [13] across different local radii to assess whether denoised point clouds exhibit a well-distributed sampling.

*Datasets and Noise Models:* During the training phase, we followed the protocol of previous works [34] and train our model using 40 models from the PU-Net dataset [49]. Clean point clouds with resolutions of 10 K, 30 K, and 50 K are generated from the corresponding mesh models using Poisson disk sampling. For each training iteration, clean point clouds are randomly sampled, normalized to fit within a unit sphere, and further divided into fixed-size patches ($N = 1000$) using farthest point sampling. To adapt to a specific noise distribution during the adaptive noise stage, Gaussian noise with an intensity of $\sigma^2{\in}(0.005, 0.02)$ is added to these patches, with the scaling parameter $s{\sim}U(0,1)$.

For testing, we use 20 models from the PU-Net dataset [49] and 10 models from the PCNet dataset [28]. To evaluate the impact of different noise levels and distributions, we apply varying levels of noise scale to these models. The synthetic noise types are categorized into Gaussian noise, non-isotropic Gaussian noise, Laplace noise and uniform ball noise, where Gaussian noise is used during both training and testing, while the remaining noise types are exclusively used for testing. Following the work of DeepRS [35], these noise types are defined as follows:

1) *Gaussian noise:* Gaussian noise with an scale of $\sigma^2$ is generated from the following probability distribution:
$$p\left(x;\sigma^2\right) = \frac{1}{\sqrt{2\pi}\sigma}e^{\frac{-x^2}{2\sigma^2}}. \tag{9}$$
During testing, the noise scale $\sigma^2$ is set to 1%, 1.5%, 2%, 2.5% and 3% of the model's bounding sphere radius. The following additional noise distributions are sampled using the same noise scale settings.
2) *Non-isotropic Gaussian noise:* We scale the noise intensity $\sigma^2$ by a symmetric matrix to construct the covariance matrix $\Sigma$, replacing the variance in (9):
$$\Sigma = \sigma^2\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{4} \\ -\frac{1}{2} & 1 & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & 1 \end{bmatrix}.$$
During testing, the noise scale $\sigma^2$ is set to 2%, 2.5% and 3% of the model's bounding sphere radius. The following noise distributions, apart from Gaussian noise, are sampled with the same noise scale settings.
3) *Laplace noise:* Laplace noise with an scale of $\sigma^2$ is generated from the following probability distribution:
$$p\left(x;\sigma^2\right) = \frac{1}{2\sigma^2}e^{\frac{-|x|}{\sigma^2}}$$
4) *Uniform ball noise:* To synthesize uniform noise within a 3D sphere, we adopt the following distribution:
$$p\left(x;\sigma^2\right) = \begin{cases} \frac{3}{4\pi(\sigma^2)^3} & \|x\|_2 \le \sigma^2 \\ 0 & \text{Otherwise} \end{cases}$$
where $\sigma^2$ denotes the maximum radius of the noise ball.

TABLE I
QUANTITATIVE COMPARISON OF THE DENOISING METHODS ON THE PU-NET DATASET AND THE PCNET DATASET WITH GAUSSIAN NOISE. CD AND P2M ARE MULTIPLIED BY $10^4$. NOTABLY, OUR PARAMETERS ARE ONLY 41% OF ITERATIVEPFN.

| | Resolution | 10K | | | | | | | | | | 50K | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Noise | 1% | | 1.50% | | 2% | | 2.50% | | 3% | | 1% | | 1.50% | | 2% | | 2.50% | | 3% | |
| Dataset | Method | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ |
| PU-Net | PCN | 3.515 | 1.148 | 5.464 | 2.413 | 7.467 | 3.965 | 10.117 | 6.196 | 13.097 | 8.766 | 1.049 | 0.346 | 1.208 | 0.449 | 1.447 | 0.608 | 2.341 | 1.346 | 3.503 | 2.334 |
| | DMR | 4.482 | 1.722 | 4.721 | 1.893 | 4.982 | 2.115 | 5.377 | 2.420 | 5.892 | 2.846 | 1.162 | 0.469 | 1.317 | 0.597 | 1.566 | 0.800 | 1.927 | 1.101 | 2.432 | 1.528 |
| | SoreDenoise | 2.521 | 0.463 | 3.153 | 0.735 | 3.686 | 1.074 | 4.427 | 1.605 | 4.707 | 1.946 | 0.716 | 0.150 | 0.934 | 0.303 | 1.288 | 0.566 | 1.941 | 1.067 | 1.929 | 1.041 |
| | PCDNF | 7.120 | 4.204 | 7.761 | 4.464 | 8.250 | 4.767 | 8.759 | 5.133 | 9.436 | 5.683 | 0.713 | 0.128 | 0.925 | 0.268 | 1.243 | 0.497 | 1.625 | 0.787 | 2.023 | 1.106 |
| | DeepRS | 2.353 | 0.306 | 2.990 | 0.543 | 3.350 | 0.734 | 3.788 | 1.075 | 4.165 | 1.345 | 0.649 | 0.076 | 0.801 | 0.174 | 0.997 | 0.296 | 1.200 | 0.459 | **1.394** | **0.582** |
| | IterativePFN(HALF) | 1.998 | 0.247 | 2.573 | 0.413 | 2.918 | 0.606 | 3.201 | 0.791 | 3.618 | **1.065** | 0.581 | 0.066 | 0.692 | 0.133 | 0.852 | 0.209 | 1.125 | 0.397 | 1.955 | 1.006 |
| | IterativePFN(FULL) | 2.055 | **0.218** | 2.691 | 0.375 | 3.043 | **0.555** | 3.375 | **0.766** | 4.242 | 1.378 | 0.605 | **0.059** | 0.706 | **0.118** | 0.803 | **0.182** | 1.019 | **0.328** | 1.970 | 1.010 |
| | Ours | **1.898** | 0.233 | **2.356** | **0.373** | **2.693** | 0.593 | **3.017** | 0.834 | **3.436** | 1.155 | **0.519** | 0.082 | **0.648** | 0.179 | **0.761** | 0.250 | **1.001** | 0.407 | 1.489 | 0.755 |
| PCNet | PCN | 3.847 | 1.221 | 6.355 | 2.038 | 8.752 | 3.043 | 11.573 | 4.374 | 14.548 | 5.902 | 1.293 | 0.289 | 1.667 | 0.411 | 1.913 | 0.505 | 2.965 | 0.948 | 4.204 | 1.500 |
| | DMR | 6.602 | 2.152 | 6.848 | 2.156 | 7.145 | 2.237 | 7.528 | 2.315 | 8.087 | 2.487 | 1.566 | 0.350 | 1.737 | 0.405 | 2.009 | 0.485 | 2.423 | 0.638 | 2.993 | 0.859 |
| | SoreDenoise | 3.369 | 0.830 | 4.365 | 1.016 | 5.132 | 1.195 | 5.981 | 1.516 | 6.779 | 1.942 | 1.066 | 0.177 | 1.317 | 0.249 | 1.659 | 0.354 | 2.369 | 0.625 | 2.494 | 0.658 |
| | PCDNF | 9.879 | 4.804 | 11.084 | 5.23 | 11.744 | 5.369 | 12.532 | 5.664 | 13.329 | 5.902 | 1.194 | 0.279 | 1.509 | 0.423 | 1.829 | 0.564 | 2.211 | 0.760 | 2.591 | 0.908 |
| | DeepRS | 2.873 | 0.783 | 4.172 | 1.180 | 4.757 | 1.118 | 5.491 | 1.565 | 6.140 | 1.776 | 1.010 | 0.146 | 1.265 | 0.273 | 1.515 | 0.340 | 1.835 | 0.506 | 2.122 | 0.592 |
| | IterativePFN(HALF) | 2.754 | 0.723 | 3.696 | 0.855 | 4.471 | 1.002 | 5.071 | 1.167 | 6.103 | 1.574 | 0.903 | 0.150 | 1.077 | 0.206 | 1.282 | 0.235 | 1.605 | 0.318 | 2.479 | 0.658 |
| | IterativePFN(FULL) | **2.621** | 0.698 | 3.725 | 0.846 | 4.437 | 1.009 | 4.978 | 1.138 | 6.021 | 1.553 | 0.913 | 0.139 | 1.092 | 0.192 | 1.252 | 0.238 | 1.520 | **0.294** | 2.546 | 0.711 |
| | Ours | 2.695 | **0.525** | **3.437** | **0.607** | **4.075** | **0.808** | **4.444** | **0.921** | **5.038** | **1.111** | **0.820** | **0.118** | **1.002** | **0.191** | **1.147** | **0.234** | **1.429** | 0.304 | **2.083** | **0.512** |

It is worth noting that the noise scale parameter $\sigma^2$ for all our noise models is defined relative to the diagonal of the bounding box of the model normalized to a unit sphere. In addition to synthetic noise perturbations on PU-Net and PCNet datasets, we also conduct comparisons using real-world scanned point cloud datasets. Specifically, we select four outdoor street scenes from the Paris-Rue-Madame dataset [50], two scenes from the KITTI-360 dataset [51], and scanned point clouds of four models from the Kinect v2 dataset [52]. Additionally, there is a large-scale point cloud from the UrbanBIS dataset [53], which contains 1,770 K points. Except for point clouds in the Kinect v2 dataset, which are obtained from noisy meshes using Poisson disk sampling in MeshLab, all other real-world scanned point clouds are directly taken from the datasets.

*Baselines:* We compare the performance of our method with state-of-the-art point cloud denoising approaches, including three conventional denoising methods and seven learning-based methods.

Among the conventional methods, we include Bilateral [15], Jet [20] and MRPCA [21]. For learning-based methods, we consider displacement-based approaches such as PCN [28], DMR [32], PCDNF [33], and IterativePFN [36], as well as probability-based approaches like ScoreDenoise [34], DeepRS [35], and a most recent PDLTS [13]. Notably, PCDNF [33] requires initial normal for denoising, which we compute using PCA [17].

### B. Results on Synthetic Data

We conduct experiments on the PU-Net and PCNet datasets with various types of synthetic noise, including Gaussian noise disturbances with the different noise levels, as well as different types of noise disturbances. When faced with Gaussian noise, the CD distances in Tables I and III, along with the point distribution shown in Fig. 5(a) and (b), demonstrate that our method outperforms others. Although DeepRS and IterativePFN show strong competitiveness in the P2M distance, visual comparisons reveal that both methods exhibit varying degrees of point clustering on the surface, which explains their poorer performance about CD. It also affects the uniformity of the denoised point cloud. As shown in Table IV, our method consistently produces a better point distribution across various noise levels. On the other hand, our network has only 1.3 million trainable parameters, which is just 41% of IterativePFN. IterativePFN improves denoising performance by repeatedly applying its denoising module within the network. The full version of IterativePFN contains four repeated modules. To provide a comparison, we include a half-version of IterativePFN with two repeated modules in Table I. While the half-version improves IterativePFN's CD distance, its overall performance remains inferior to our method, despite having more parameters than our network. On the unseen PCNet dataset during training, our method demonstrates strong generalization capabilities. Except for the P2M distance at a resolution of 50 K and a noise level of 2.5%, where IterativePFN performs better, our method achieves the best results in all other cases. For complex models such as Netsuke and Column Head, our method avoids producing noticeable holes, unlike the other methods.

Table V, Fig. 5(f) and (e), presents results under unseen single noise types. In the quantitative comparison, our method outperforms the other approaches in both uniform ball noise and non-isotropic Gaussian noise. Our approach and PDLTS exhibit performance degradation under Laplace noise, as this type of noise tends to produce outliers that hinder feature learning in latent space-based methods. However, our approach outperforms PDLTS in the presence of Laplace noise at higher noise levels, due to its explicit modeling of the noise distribution. DeepRS retains fewer outliers than other methods due to its subsequent regularization process (Fig. 5(f)). However, regardless of the noise type, both DeepRS and IterativePFN suffer from similar issues, including point clustering on surfaces and artifacts along edges, which become more pronounced in sparse point clouds. While PDLTS, like our method, achieves a more uniform point distribution on the surface, quantitative analysis indicates that its overall performance in noise residual reduction is inferior to ours.

In addition to single noise types, we combined various synthetic noise types to simulate hybrid noise distributions. Table II, Fig. 5(c) and (d) present the quantitative and qualitative comparisons. In Fig. 5(c), we first combined 1% Gaussian noise with 1% non-isotropic Gaussian noise (Type 1 in Table II), and then progressively added 1% uniform ball noise (Type 2 in Table II) and 1% Laplace noise (Type 3 in Table II). In Fig. 5(d), we vary the relative intensity of different single noise types

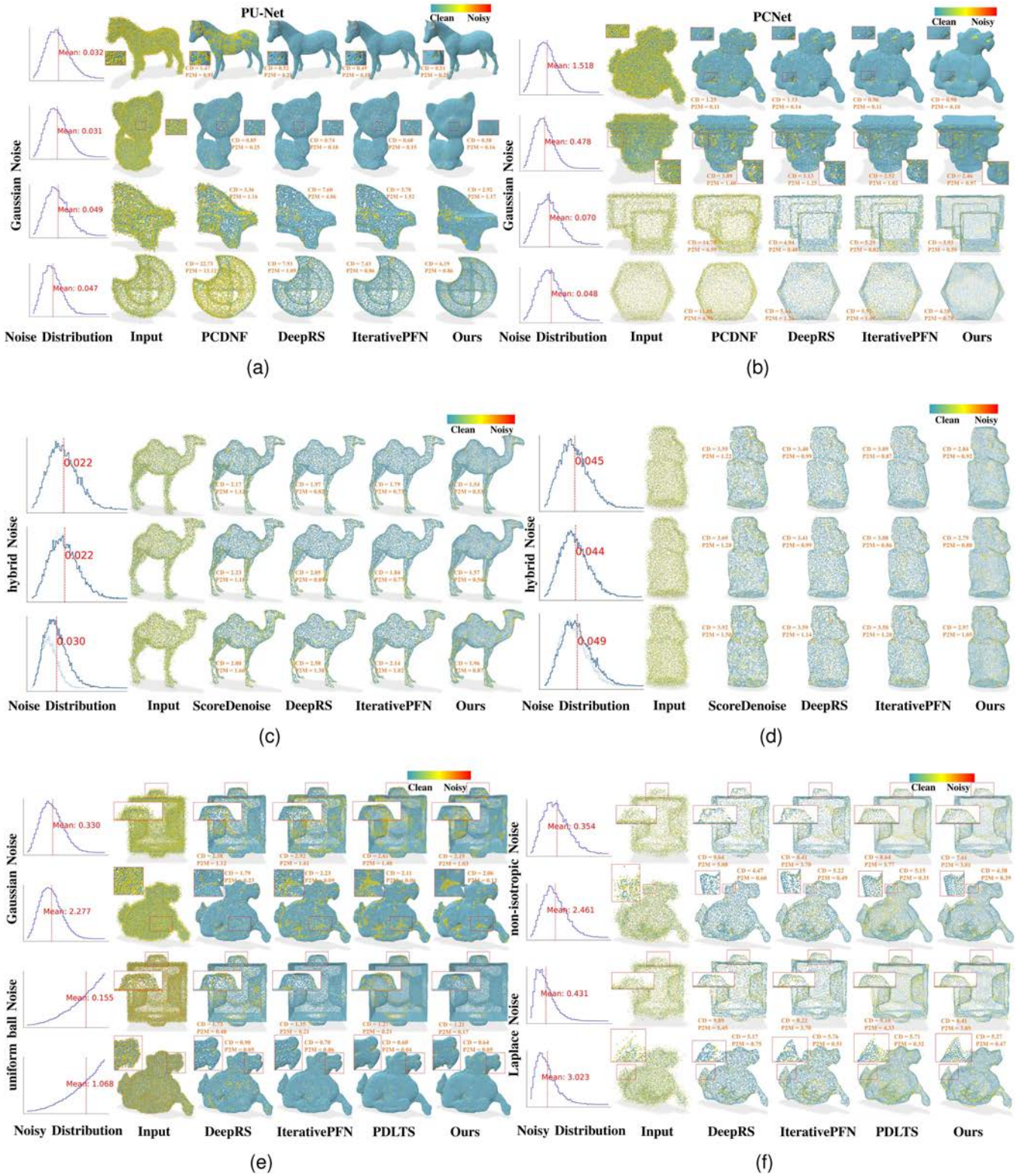


Fig. 5. Visual comparison of denoising methods under different synthetic noise settings. Noise distributions are visualized as histograms of error distances. In (a) and (b), the top two rows and bottom two rows correspond to 2% and 3% Gaussian noise, respectively. In (c), from top to bottom: 1% Gaussian noise; 1% non-isotropic Gaussian noise; then with an additional 1% uniform ball noise; and finally with an additional 1% Laplace noise. In (d), from top to bottom: 2% Gaussian noise; 2% non-isotropic Gaussian noise; with an additional 2% uniform ball noise; and finally with an additional 1% Laplace noise. In (e), Gaussian noise and uniform ball noise at 3% with 50 K resolution. In (f), non-isotropic Gaussian noise and Laplace noise at 3% with 10 K resolution.

TABLE II
QUANTITATIVE COMPARISON OF DENOISING METHODS UNDER COMPLEX NOISE (PU-NET DATASET, 10K) AND THE KINECT V2 DATASET. HD IS MULTIPLIED BY $10^4$.

| Method | Type 1 | | | Type 2 | | | Type 3 | | | Cone | Boy | Girl | Pyramid | Mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Gaussian+non-isotropic | | | Type 1+uniform ball | | | Type 2+Laplace | | | Kinect v2 (P2M↓) | | | | |
| | CD↓ | P2M↓ | HD↓ | CD↓ | P2M↓ | HD↓ | CD↓ | P2M↓ | HD↓ | | | | | |
| PCN | 5.403 | 2.374 | 8.443 | 5.407 | 2.379 | 8.41 | 7.742 | 4.232 | 15.543 | | | — | | |
| DMR | 4.714 | 1.889 | 6.942 | 4.686 | 1.86 | 6.852 | 4.997 | 2.101 | 7.586 | | | — | | |
| SoreDenoise | 3.118 | 0.72 | 4.693 | 3.126 | 0.719 | 4.695 | 3.705 | 1.097 | 5.475 | 1.256 | 0.987 | 0.620 | 1.132 | **0.999** |
| PCDNF | 7.742 | 4.46 | 9.977 | 7.738 | 4.463 | 9.958 | 8.287 | 4.807 | 11.707 | 1.317 | **0.962** | 0.648 | 1.157 | 1.021 |
| DeepRS | 2.974 | 0.543 | 4.941 | 2.977 | 0.536 | 4.966 | 3.455 | 0.848 | 5.846 | **1.230** | 1.133 | 0.707 | 1.199 | 1.067 |
| IterativePFN | 2.671 | 0.373 | 4.251 | 2.665 | 0.367 | 4.239 | 3.042 | **0.554** | 5.027 | 1.334 | 0.987 | 0.635 | 1.196 | 1.038 |
| PDLTS | — | — | — | — | — | — | — | — | — | 1.311 | 0.986 | 0.621 | 1.136 | 1.014 |
| Ours | **2.337** | **0.371** | **2.839** | **2.337** | **0.366** | **2.846** | **2.684** | 0.592 | **3.379** | 1.288 | 0.984 | **0.613** | **1.123** | 1.002 |

TABLE III
QUANTITATIVE COMPARISON OF THE CONVENTIONAL METHODS ON THE PU-NET DATASET AND THE PCNET DATASET WITH GAUSSIAN NOISE

| Resolution | | 10K | | | | 50K | | | |
|---|---|---|---|---|---|---|---|---|---|
| Noise | | 1% | | 2% | | 1% | | 2% | |
| Dataset | Method | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ |
| PU-Net | Bilateral | 3.646 | 1.342 | 5.007 | 2.018 | 0.877 | 0.234 | 2.376 | 1.389 |
| | MRPCA | 2.972 | 0.922 | 3.728 | 1.117 | 0.669 | 0.099 | 2.008 | 1.033 |
| | Jet | 2.712 | 0.613 | 4.155 | 1.347 | 0.851 | 0.207 | 2.432 | 1.403 |
| | Ours | **1.898** | **0.233** | **2.693** | **0.593** | **0.519** | **0.082** | **0.761** | **0.250** |
| PCNet | Bilateral | 4.320 | 1.351 | 6.171 | 1.646 | 1.172 | 0.198 | 2.478 | 0.634 |
| | MRPCA | 3.323 | 0.931 | 4.874 | 1.178 | 0.966 | 0.140 | 2.153 | 0.478 |
| | Jet | 3.032 | 0.830 | 5.298 | 1.372 | 1.091 | 0.180 | 2.582 | 0.700 |
| | Ours | **2.695** | **0.525** | **4.075** | **0.808** | **0.820** | **0.118** | **1.147** | **0.234** |

TABLE IV
UNIFORMITY COMPARISON OF THE DENOISING METHODS ON THE PU-NET DATASET, WITH 10 K RESOLUTION AND GAUSSIAN NOISE. THE UNIFORMITY IS ESTIMATED IN THE LOCAL AREA OF DIFFERENT RADDI AND IS MULTIPLIED BY 10.

| Local area of radii | | 0.4% | 0.6% | 0.8% | 1.0% | 1.2% |
|---|---|---|---|---|---|---|
| Noise | Method | UNI↓ | UNI↓ | UNI↓ | UNI↓ | UNI↓ |
| 1% | ScoreDenoise | 1.004 | 1.210 | 1.440 | 1.717 | 2.055 |
| | DeepRS | 0.654 | 0.719 | 0.840 | 1.015 | 1.248 |
| | IterativePFN | 0.572 | 0.699 | 0.822 | 0.964 | 1.150 |
| | Ours | **0.222** | **0.260** | **0.291** | **0.347** | **0.426** |
| 2% | ScoreDenoise | 1.973 | 2.453 | 2.961 | 3.518 | 4.125 |
| | DeepRS | 1.562 | 1.634 | 1.821 | 2.030 | 2.297 |
| | IterativePFN | 0.934 | 0.983 | 1.120 | 1.374 | 1.633 |
| | Ours | **0.309** | **0.379** | **0.441** | **0.519** | **0.647** |
| 3% | ScoreDenoise | 4.388 | 5.534 | 6.487 | 7.589 | 8.530 |
| | DeepRS | 2.351 | 2.409 | 2.412 | 2.545 | 2.883 |
| | IterativePFN | 1.132 | 1.129 | 1.260 | 1.463 | 1.719 |
| | Ours | **0.465** | **0.517** | **0.562** | **0.648** | **0.784** |

to synthesize another hybrid noise for visual comparison. As the complexity of the noise increased, both ScoreDenoise and DeepRS exhibited higher levels of residual noise. Moreover, DeepRS and IterativePFN showed point clustering on the surface after denoising. Our results demonstrate superior performance in handling complex synthetic noise compared to other methods, showcasing comprehensive advantages in both lower residual noise levels and better uniform point distribution.

### C. Results on Real-World Scan Data

Compared to synthetic noise, real-world scanned data exhibit random noise, uneven density, and more complex scenes. Figs. 6 and 7 provide a visual comparison on the Paris-Rue-Madame dataset, focusing on two regions: building exteriors and parked vehicles. The zoomed-in views highlight denoising details in these areas. Our method demonstrates better smoothness and uniform point distribution compared to other approaches. In contrast, other methods either produce artifacts (IterativePFN, DeepRS) or fail to effectively remove noise (ScoreDenoise, PCDNF, PDLTS).

Fig. 8 shows a visual comparison on the autonomous driving dataset KITTI-360. While IterativePFN and DeepRS handle outliers more effectively than our method, our approach outperforms others in terms of surface smoothness on parked vehicles and noise removal near windows. Other methods tend to produce noise artifacts in these regions. Although PDLTS achieves a more uniform point distribution, it retains more residual noise. The Kinect v2 dataset includes ground-truth meshes, allowing us to conduct quantitative P2M evaluations (Table II). ScoreDenoise performs worse than ours on all models except for the Cone model.

Fig. 9 shows our denoising performance on large-scale point cloud. Existing SOTA methods [34], [35], [36] handle large-scale point cloud by first clustering them into smaller patches before feeding them into the network. We follow the same preprocessing steps as IterativePFN [36]. Due to point clustering and artifacts at patch edges, IterativePFN introduces holes and gaps in the entire denoised point cloud. In contrast, our method effectively removes noise while maintaining a well-distributed point, mitigating these issues.

### D. Noise Robust Analysis and Discussion

Finally, We summarize and compare the performance of each method in Table VI. We evaluate the performance across six aspects: noise level, noise type, resolution, dataset, residual, and point distribution. Our method demonstrates robust performance, achieving superior results in point distribution

In terms of noise levels, PCN exhibits a significantly higher rate of change starting from 2% noise scales on the PU-Net dataset at a 10 K resolution (Table I). Similarly, IterativePFN and PDLTS becomes more sensitive when the noise rises from 2.5% to 3% (Table V). Regarding noise types, PCN demonstrates notable sensitivity to Laplace noise, while DMR does not show the same ease of handling uniform ball noise as other methods (Table V). Fig. 10(a) compares the CD distances at 10 K and 50 K resolutions for noise scale ranging from 1% to 3%. After normalizing the values using min-max scaling, we plot them as

TABLE V
QUANTITATIVE COMPARISON OF THE DENOISING METHODS ON THE PCNET DATASET WITH DIFFERENT NOISE TYPES

| Resolution | | 10K | | | | | | | | | 50K | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Noise scale | | 2.0% | | | 2.5% | | | 3.0% | | | 2.0% | | | 2.5% | | | 3.0% | | |
| Noise type | Method | CD↓ | P2M↓ | HD↓ | CD↓ | P2M↓ | HD↓ | CD↓ | P2M↓ | HD↓ | CD↓ | P2M↓ | HD↓ | CD↓ | P2M↓ | HD↓ | CD↓ | P2M↓ | HD↓ |
| non-isotropic | PCN | 9.995 | 3.582 | 19.633 | 13.141 | 5.127 | 29.447 | 16.659 | 6.948 | 41.114 | 2.503 | 0.749 | 4.389 | 3.665 | 1.245 | 7.063 | 5.494 | 2.068 | 11.247 |
| | DMR | 7.286 | 2.232 | 11.514 | 7.876 | 2.420 | 13.169 | 8.620 | 2.663 | 15.212 | 2.219 | 0.576 | 4.539 | 2.804 | 0.805 | 6.481 | 3.642 | 1.164 | 9.321 |
| | SoreDenoise | 5.706 | 1.553 | 9.484 | 6.300 | 1.759 | 10.442 | 7.153 | 2.060 | 11.542 | 1.774 | 0.460 | 3.127 | 2.269 | 0.624 | 3.968 | 3.177 | 0.941 | 6.317 |
| | PCDNF | 12.100 | 5.482 | 17.842 | 12.962 | 5.768 | 20.844 | 14.088 | 6.210 | 24.634 | 1.988 | 0.628 | 3.451 | 2.423 | 0.838 | 4.383 | 2.932 | 1.080 | 5.653 |
| | DeepRS | 5.157 | 1.444 | 8.379 | 5.838 | 1.682 | 9.461 | 6.683 | 2.026 | 10.637 | 1.732 | 0.480 | 3.093 | 2.079 | 0.610 | 3.871 | **2.558** | **0.775** | **4.920** |
| | IterativePFN | 4.668 | 1.069 | 7.234 | 5.540 | 1.386 | 8.216 | 6.246 | 1.689 | 9.510 | 1.773 | 0.313 | 2.988 | 1.914 | **0.479** | 3.289 | 3.986 | 1.450 | 6.915 |
| | PDLTS | **4.219** | **0.848** | **5.796** | 4.951 | 1.139 | 7.030 | 6.306 | 1.699 | 9.763 | 1.313 | **0.294** | **2.083** | 1.978 | 0.539 | 3.691 | 3.618 | 1.231 | 7.398 |
| | Ours | 4.233 | 0.870 | 5.834 | **4.808** | **1.096** | **6.730** | **5.749** | **1.476** | **8.539** | **1.307** | 0.301 | 2.148 | **1.868** | 0.490 | **3.283** | 3.106 | 0.968 | 6.521 |
| uniform ball | PCN | 3.655 | 1.129 | 3.638 | 4.913 | 1.454 | 5.370 | 6.100 | 1.832 | 7.305 | 1.325 | 0.282 | 1.726 | 1.495 | 0.344 | 2.031 | 1.677 | 0.419 | 2.402 |
| | DMR | 6.531 | 2.107 | 9.947 | 6.601 | 2.093 | 9.979 | 6.724 | 2.109 | 10.193 | 1.555 | 0.356 | 2.499 | 1.613 | 0.364 | 2.638 | 1.701 | 0.391 | 2.885 |
| | SoreDenoise | 3.925 | 1.101 | 6.016 | 4.398 | 1.183 | 6.739 | 4.787 | 1.265 | 7.520 | 1.139 | 0.238 | 1.941 | 1.251 | 0.277 | 2.200 | 1.349 | 0.305 | 2.413 |
| | PCDNF | 9.672 | 4.714 | 13.689 | 10.352 | 4.942 | 14.331 | 10.886 | 5.122 | 14.876 | 1.179 | 0.260 | 1.913 | 1.317 | 0.317 | 2.156 | 1.450 | 0.375 | 2.392 |
| | DeepRS | 3.064 | 1.010 | 4.517 | 3.671 | 1.070 | 5.605 | 4.153 | 1.149 | 6.505 | 0.989 | 0.153 | 1.596 | 1.123 | 0.199 | 1.811 | 1.315 | 0.274 | 2.102 |
| | IterativePFN | **2.512** | 0.670 | **3.386** | 3.210 | 0.733 | 4.452 | 3.668 | 0.820 | 5.171 | 0.914 | 0.127 | 1.474 | 0.998 | 0.150 | 1.629 | 1.036 | 0.183 | 1.707 |
| | PDLTS | 2.826 | **0.475** | 4.021 | 3.220 | **0.513** | 4.418 | 3.486 | **0.561** | 4.746 | **0.786** | **0.104** | **1.119** | **0.862** | 0.133 | 1.262 | 0.958 | 0.170 | 1.431 |
| | Ours | 2.609 | 0.514 | 3.541 | **3.085** | 0.542 | **4.155** | **3.414** | 0.579 | **4.628** | 0.807 | 0.111 | 1.160 | 0.862 | **0.125** | **1.250** | **0.926** | **0.139** | **1.340** |
| Laplace | PCN | 12.509 | 5.046 | 32.502 | 18.409 | 8.064 | 52.792 | 21.956 | 10.056 | 68.098 | 3.707 | 1.355 | 5.050 | 6.831 | 2.879 | 10.078 | 9.370 | 4.156 | 13.920 |
| | DMR | 7.646 | 2.325 | 12.578 | 8.690 | 2.672 | 15.782 | 9.248 | 2.891 | 17.397 | 2.417 | 0.605 | 5.302 | 3.411 | 0.994 | 8.886 | 4.173 | 1.284 | 11.984 |
| | SoreDenoise | 6.196 | 1.809 | 10.150 | 8.681 | 2.617 | 15.659 | 8.606 | 2.720 | 13.721 | 1.897 | 0.506 | 3.354 | 2.760 | 0.772 | 5.070 | 3.355 | 0.993 | 6.489 |
| | PCDNF | 12.740 | 5.627 | 20.094 | 14.927 | 6.561 | 26.279 | 16.029 | 7.005 | 29.225 | 2.122 | 0.695 | 3.728 | 2.774 | 0.978 | 5.160 | 3.080 | 1.109 | 5.886 |
| | DeepRS | 5.414 | 1.565 | 8.949 | 6.565 | 2.033 | 10.661 | 7.042 | 2.175 | 11.549 | 1.814 | 0.490 | 3.389 | 2.350 | 0.672 | 4.638 | **2.607** | **0.749** | **5.166** |
| | IterativePFN | 4.861 | 1.151 | 7.550 | 6.806 | 1.893 | 9.856 | 6.555 | **1.729** | 9.997 | 1.854 | 0.322 | 3.309 | 2.487 | **0.614** | 4.395 | 3.366 | 0.933 | 6.553 |
| | PDLTS | **4.491** | **0.943** | **6.240** | 5.840 | **1.456** | 8.616 | 6.737 | 1.832 | 10.480 | **1.414** | **0.314** | **2.103** | 2.633 | 0.713 | 5.184 | 3.943 | 1.247 | 8.644 |
| | Ours | 4.528 | 0.983 | 6.394 | **5.653** | 1.465 | **7.979** | **6.405** | 1.760 | **9.654** | 1.442 | 0.345 | 2.278 | **2.342** | 0.642 | **4.209** | 3.035 | 0.924 | 6.009 |

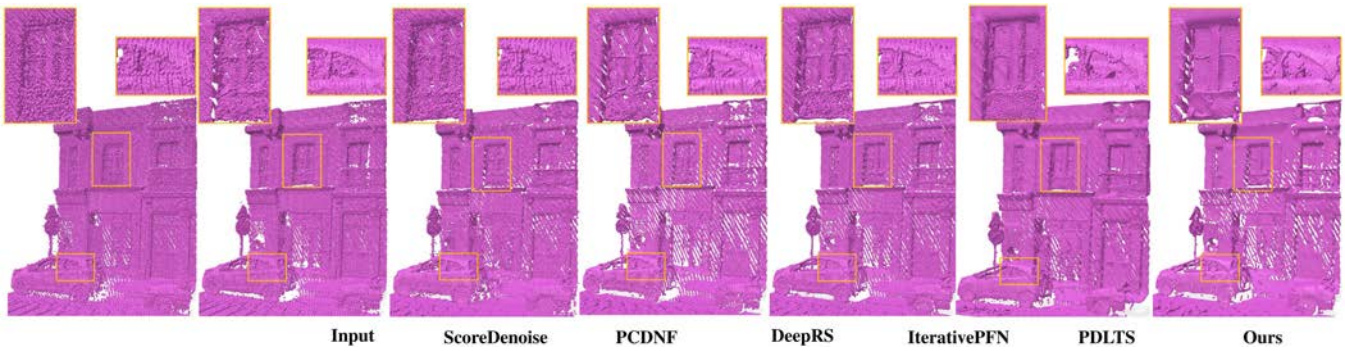


Fig. 6. Visual comparison of the denoising methods on one scene in the Paris-Rue-Madame dataset.

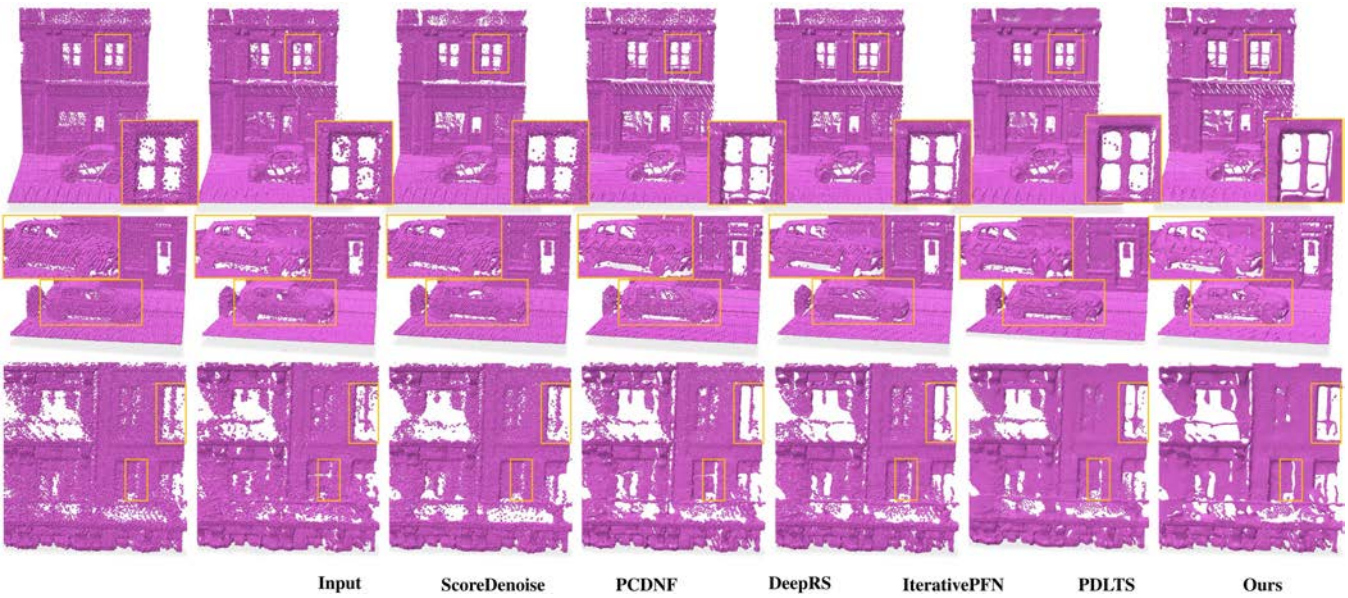


Fig. 7. Visual comparison of denoising methods on the other scenes in the Paris-Rue-Madame dataset.

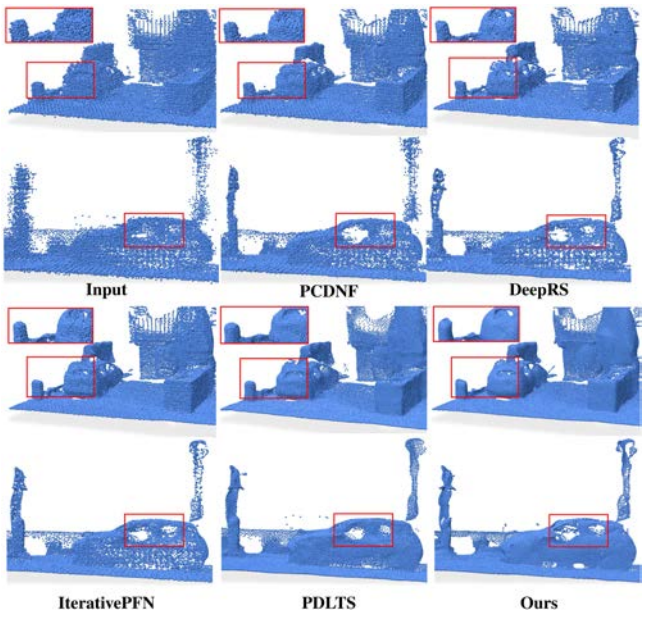


Fig. 8. Visual comparison of denoising methods on the KITTI-360 dataset.

TABLE VI
COMPARISON OF THE ROBUSTNESS OF DIFFERENT DENOISING METHODS. CELLS HIGHLIGHTED IN PINK INDICATE STRONGER PERFORMANCE

| Method | Level | Type | Resolution | Dataset | Residual | Distribution |
|---|---|---|---|---|---|---|
| PCN | sensitive | partial | sensitive | dataset-specific | high | non-uniform |
| DMR | robust | partial | sensitive | generalizable | high | non-uniform |
| ScoreDenoise | robust | comprehensive | sensitive | generalizable | medium | non-uniform |
| PCDNF | robust | comprehensive | sensitive | dataset-specific | medium | non-uniform |
| DeepRS | robust | comprehensive | robust | generalizable | low | non-uniform |
| IterativePFN | sensitive | comprehensive | robust | generalizable | low | non-uniform |
| PDLTS | sensitive | comprehensive | robust | generalizable | low | uniform |
| Ours | robust | comprehensive | robust | generalizable | low | uniform |

TABLE VII
TESTING TIME COMPARISON OF THE DENOISING METHODS ON THE PU-NET DATASET WITH 10 K RESOLUTIONS

| Method | PCN | SoreDenoise | PCDNF | DeepRS | IterativePFN | PDLTS | Ours |
|---|---|---|---|---|---|---|---|
| Time (s) | 7.25 | 1.97 | 7.53 | 1.74 | 11.22 | 2.41 | 3.83 |

paired coordinates in a scatter plot to observe whether the trends across noise scales are consistent for different resolutions (with the expectation of aligning closely with the blue dashed line). Our method predominantly appears at the origin, as its CD distances are generally the smallest. With the exception of DeepRS and IterativePFN, other methods show points significantly deviating from the blue dashed line, with PCDNF exhibiting the strongest sensitivity to resolution. A similar plotting is used in Fig. 10(b), where PCN and PCDNF demonstrate higher sensitivity to the dataset compared to the other methods.

*Computational Efficiency:* The overall training process of our model took about 32 hours, including 19 hours for pretraining and 13 hours for full training. In Table VII, we compare the average inference time of 20 models from the PU-Net dataset at a 10 K resolution. IterativePFN, with 3.2 million trainable parameters, has the longest inference time. PDLTS has 1.4 M parameters; although its inference time is slightly faster than ours, the parameter count is comparable. ScoreDenoise and DeepRS perform better in single inference time. The Euler method used in the (8) increases ours inference time. In Section IV-E, we further compare the computational efficiency across different configurations of our model, demonstrating that our approach achieves the best performance within a relatively advantageous time frame.

*Reconstruction:* The nice point distribution generated by our method benefits downstream point cloud tasks, such as reconstruction. Fig. 11 illustrates a visual comparison of reconstruction results under 2% Gaussian noise. For a fair comparison, meshes were generated using iterative Poisson Surface Reconstruction [57] with same parameters. Our method produces significantly smoother surfaces with clearer geometric details.

TABLE VIII
ABLATION STUDIES. QUANTITATIVE COMPARISON ON THE PU-NET DATASET AT 10 K POINT RESOLUTION

| Configuration | 1% | | 2% | | 3% | |
|---|---|---|---|---|---|---|
| | CD↓ | P2M↓ | CD↓ | P2M↓ | CD↓ | P2M↓ |
| C1 | 2.16 | 0.42 | 3.06 | 0.92 | 3.73 | 1.42 |
| C2 | 1.97 | 0.32 | 3.01 | 0.92 | 3.71 | 1.45 |
| C3 | 1.99 | 0.31 | 2.76 | 0.67 | 3.56 | 1.24 |
| FULL | 1.90 | 0.23 | 2.69 | 0.59 | 3.44 | 1.17 |

## E. Ablation Studies

We conducted ablation experiments to evaluate the contribution of each key component in our method. Specifically, we designed three comparative configurations by systematically removing critical parts from the full model (FULL):

- C1: Remove the adaptive noise stage, retaining only the denoising stage.
- C2: Remove adaptive noise learning in the adaptive noise stage, directly passing the extracted features into the decoder. This effectively makes the adaptive stage similar to a repeated denoising stage.
- C3: Remove the Euler method during inference.
- FULL: Our complete model.

Quantitative results for each ablation configuration are presented in Table VIII, and qualitative comparisons on the Elephant model (with 10 K points under 2% Gaussian noise) are shown in Fig. 12. Overall, the full version of our method consistently outperforms all ablated configurations.

We observed that C1 exhibits the most significant performance degradation, highlighting the critical contribution of the adaptive noise stage in addressing noise misalignment. By comparing C1 and C2, we observed that as noise scale increases, omitting adaptive noise learning significantly reduces the network's effectiveness in handling diverse noise distributions. The results from C3 indicate that employing the Euler method during inference contributes further improvement, especially at lower noise levels, enhancing the already effective output produced by our adaptive framework.

In addition, we analyze the trade-off between inference time and accuracy for each configuration in the Fig. 13. While FULL achieves the highest accuracy, it also requires the longest inference time. C1, which removes the noise adaptation stage, attains the shortest inference time but suffers a substantial accuracy

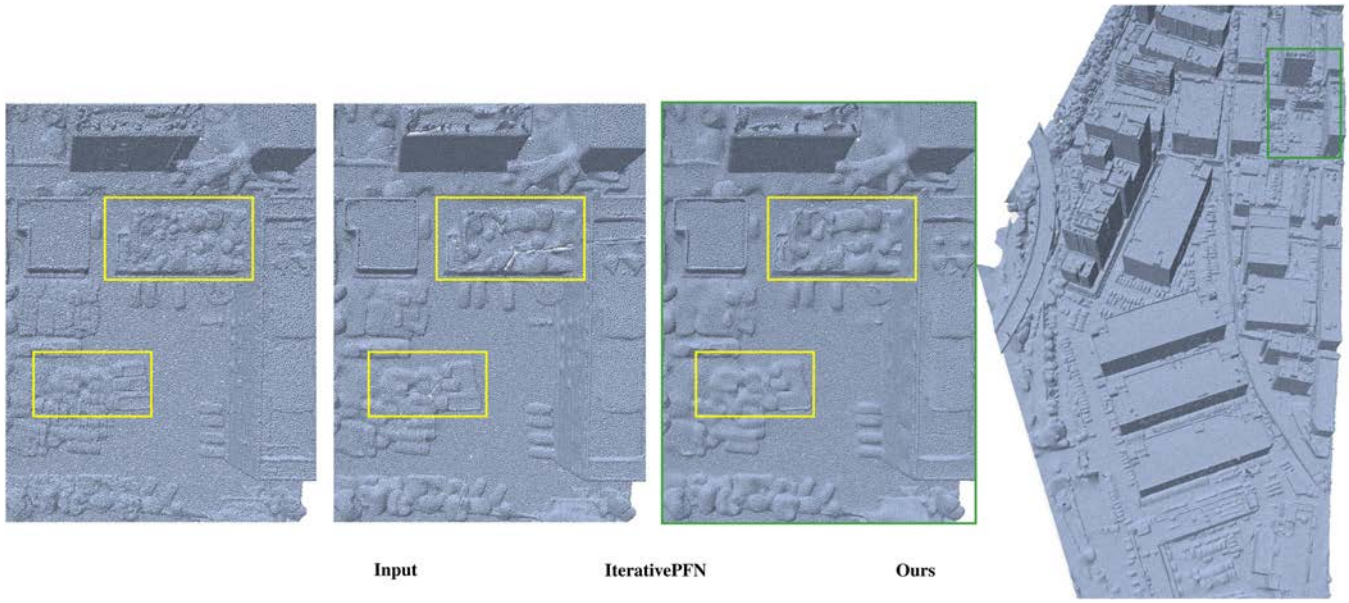


Fig. 9. Denoising results on large-scale point clouds from the UrbanBIS dataset, with 1770 K resolution. The rightmost shows the denoised result of our method on the entire point cloud, with the green region zoomed in to highlight local details for visual comparison.

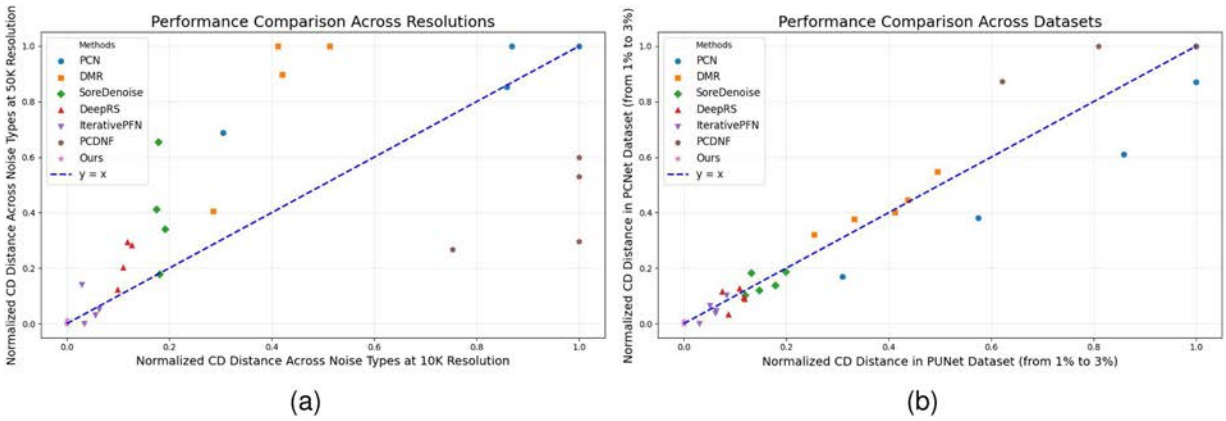


Fig. 10. Noise robust analysis of the denoising methods, with the expectation of aligning closely with the blue dashed line. (a) Across resolutions. (b) Across datasets.

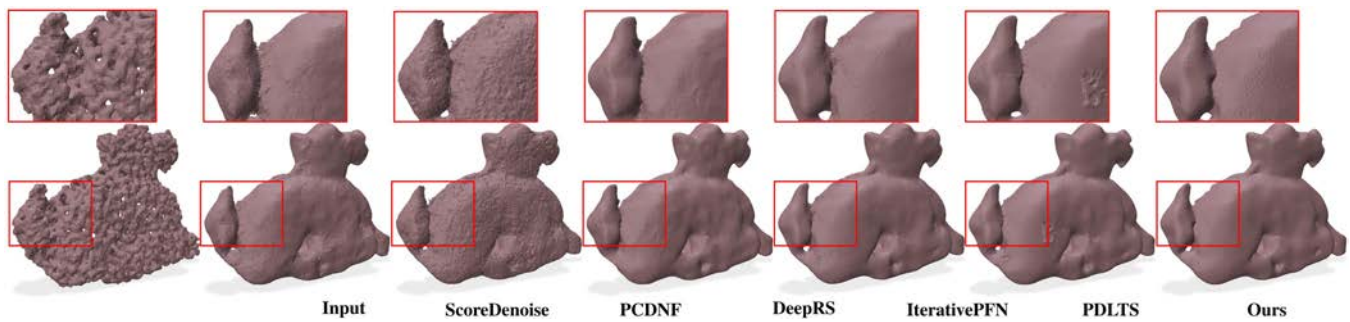


Fig. 11. Visual comparison of reconstructed surfaces from denoised points.

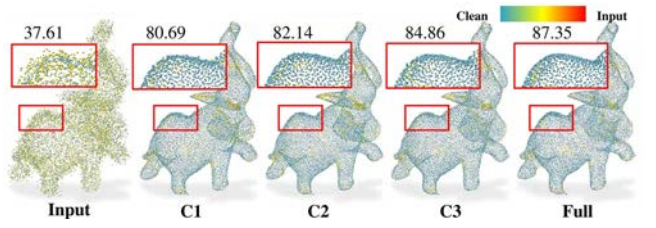


Fig. 12. Ablation studies: Visual comparison of different method configurations on the Elephant model (10 K points) contaminated with 2% Gaussian noise. Points colored from yellow to red indicate increasing distances from the ground-truth surface. Numerical values represent the percentage of filtered points lying on the ground-truth mesh.

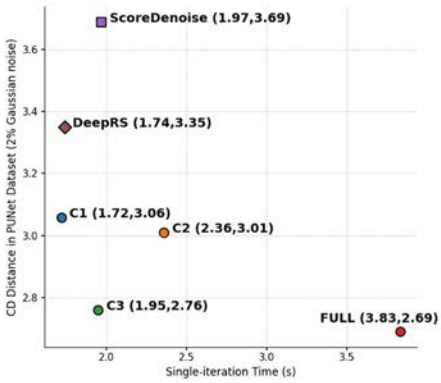


Fig. 13. Ablation studies: Efficiency analysis of different method configurations on the PU-Net dataset (10 K point resolution) contaminated with 2% Gaussian noise.

drop. In contrast, C3, which removes the Euler method during inference, effectively balances accuracy and efficiency, producing near-FULL, second-best results with an inference time comparable to ScoreDenoise and DeepRS. By comparing C3 and FULL, we provide alternative choices for applying our model in different scenarios. FULL is preferable for relief models to preserve more geometric details, while C3 is better suited for large-scale point clouds where higher computational efficiency is required. For instance, when processing the large-scale point cloud with 1770 K points shown in the Fig. 9, C3 accelerates inference by 1.77× (from 472 seconds to 266 seconds).

### *F. Limitations*

While LaPDA demonstrates strong generalization across diverse noise types, it still struggles with severe outliers near thin structures. This limitation stems from the assumption in our adaptive noise stage that noise distributions correlate with surface proximity. In practice, LaPDA learns to adapt noise by targeting scaled versions of known training distributions. However, outliers that lie far from the underlying surface often belong to distinct and independent noise distributions that are not explicitly modeled. As a result, the adaptive stage has difficulty aligning such points, and the denoising stage, designed primarily for surface noise, cannot reliably remove them. For instance, when processing outliers around window regions in

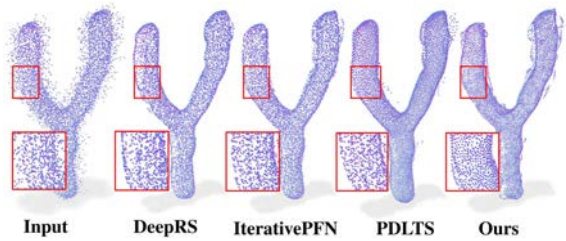


Fig. 14. Limitation when handling outliers near thin structures. While our method preserves a uniform point distribution, it struggles to effectively remove distant outliers. Other state-of-the-methods also encounter similar difficulties in such scenarios.

Fig. 7, LaPDA produces spurious detail recovery that distorts boundaries. Fig. 14 highlights such a failure case.

We observe similar limitations in other state-of-the-art methods. PDLTS, for example, also struggles to disentangle isolated outliers in latent space; IterativePFN reduces some outliers but still leaves residual points outside the surface; and DeepRS aggressively projects nearly all outliers back to the surface, which eliminates isolated noise but introduces surface tearing, point clustering, and artificial boundaries.

## V. Conclusion

In this paper, we identified the problem of *noise misalignment*, showing that recent learning-based point cloud denoising methods, trained on specific synthetic noise distributions, often fail when exposed to complex or unseen real-world noise. To address this challenge, we introduced *LaPDA*, a two-stage denoising framework that adaptively learns and adjusts noise distributions directly from noisy inputs. The adaptive noise modeling stage aligns unseen inference-time noise either with known training distributions or with reduced-scale noise, enabling the subsequent denoising stage to operate more effectively.

Our progressive approach eliminates the need for explicit post-processing to handle surface drift, which is commonly required by displacement-based methods, and produces point clouds that are both clean and uniformly distributed. Extensive experiments on synthetic and real-world datasets demonstrate that LaPDA achieves improved robustness and accuracy compared to state-of-the-art methods.

Looking forward, we see two promising directions. One is to explicitly model outliers as a separate noise category, for example by incorporating robust statistical estimators or dedicated outlier-removal modules into the pipeline. Another is to introduce multi-modal noise priors during the adaptive noise stage, enabling LaPDA to adapt to a broader range of noise behaviors and further improve robustness on challenging real-world scans.

## References

[1] Z. Xu and A. Foi, "Anisotropic denoising of 3D point clouds by aggregation of multiple surface-adaptive estimates," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 6, pp. 2851–2868, Jun. 2021.
[2] V. Katkovnik, "A new method for varying adaptive bandwidth selection," *IEEE Trans. Signal Process.*, vol. 47, no. 9, pp. 2567–2571, Sep. 1999.
[3] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 544–552, 2005.

[4] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," *Comput. Graph. Forum*, vol. 28, no. 2, pp. 493–501, 2009.

[5] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *IEEE Trans. Vis. Comput. Graphics*, vol. 9, no. 1, pp. 3–15, First Quarter 2003.

[6] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 22–es, 2007.

[7] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 1–7, 2009.

[8] B. Liao, C. Xiao, L. Jin, and H. Fu, "Efficient feature-preserving local projection operator for geometry reconstruction," *Comput-Aided Des.*, vol. 45, no. 5, pp. 861–874, 2013.

[9] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, and M. Wimmer, "Continuous projection for fast L1 reconstruction," *ACM Trans. Graph.*, vol. 33, no. 4, 2014, Art. no. 47.

[10] J. Batson and L. Royer, "Noise2Self: Blind denoising by self-supervision," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 524–533.

[11] C. Kim, T. H. Kim, and S. Baik, "LAN: Learning to adapt noise for image denoising," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 25193–25202.

[12] J. Wang, S. Di, L. Chen, and C. W. W. Ng, "Noise2Info: Noisy image to information of noise for self-supervised image denoising," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2023, pp. 16034–16043.

[13] A. Mao, B. Yan, Z. Ma, and Y. He, "Denoising point clouds in latent space via graph convolution and invertible neural network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 5768–5777.

[14] J. Wang, B. Fei, D. de Silva Edirimuni, Z. Liu, Y. He, and X. Lu, "A survey of deep learning-based point cloud denoising," 2025. [Online]. Available: https://arxiv.org/abs/2508.17011

[15] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *Comput. Aided Geometric Des.*, vol. 22, no. 2, pp. 121–146, 2005.

[16] P. Du, X. Wang, Y. Fang, X. Ru, H. Zhao, and Z. Wu, "Feature-preserving point cloud filtering via mixture family manifold," *Comput. Aided Geometric Des.*, vol. 119, 2025, Art. no. 102453.

[17] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proc. 19th Annu. Conf. Comput. Graph. Interactive Techn.*, 1992, pp. 71–78.

[18] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," *ACM Trans. Graph.*, vol. 32, no. 1, pp. 1–12, 2013.

[19] X. Lu, S. Wu, H. Chen, S.-K. Yeung, W. Chen, and M. Zwicker, "GPF: GMM-inspired feature-preserving point set filtering," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 8, pp. 2315–2326, Aug. 2018.

[20] S. Fleishman, I. Drori, and D. Cohen-Or, "Bilateral mesh denoising," in *Proc. ACM SIGGRAPH Papers*, 2003, pp. 950–953.

[21] E. Mattei and A. Castrodad, "Point cloud denoising via moving RPCA," *Comput. Graph. Forum*, vol. 36, no. 8, pp. 123–137, 2017.

[22] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[23] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5099–5108.

[24] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "PCPNet learning local shape properties from raw point clouds," *Comput. Graph. Forum*, vol. 37, no. 2, pp. 75–85, 2018.

[25] C.-E. Himeur, T. Lejemble, T. Pellegrini, M. Paulin, L. Barthe, and N. Mellado, "PCEDNet: A lightweight neural network for fast and interactive edge detection in 3D point clouds," *ACM Trans. Graph.*, vol. 41, no. 1, Nov. 2021, Art. no. 10.

[26] Y. Yuan, Y. Wu, X. Fan, M. Gong, W. Ma, and Q. Miao, "EGST: Enhanced geometric structure transformer for point cloud registration," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 9, pp. 6222–6234, Sep. 2024.

[27] M. Zhao, X. Huang, J. Jiang, L. Mou, D.-M. Yan, and L. Ma, "Accurate registration of cross-modality geometry via consistent clustering," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 7, pp. 4055–4067, Jul. 2024.

[28] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov, "PointCleanNet: Learning to denoise and remove outliers from dense point clouds," *Comput. Graph. Forum*, vol. 39, no. 1, pp. 185–203, 2020.

[29] D. Zhang, X. Lu, H. Qin, and Y. He, "Pointfilter: Point cloud filtering via encoder-decoder modeling," *IEEE Trans. Vis. Comput. Graphics*, vol. 27, no. 3, pp. 2015–2027, Mar. 2021.

[30] P. Hermosilla, T. Ritschel, and T. Ropinski, "Total denoising: Unsupervised learning of 3D point cloud cleaning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 52–60.

[31] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.

[32] S. Luo and W. Hu, "Differentiable manifold reconstruction for point cloud denoising," in *Proc. ACM Int. Conf. Multimedia*, 2020, pp. 1330–1338.

[33] Z. Liu, Y. Zhao, S. Zhan, Y. Liu, R. Chen, and Y. He, "PCDNF: Revisiting learning-based point cloud denoising via joint normal filtering," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 8, pp. 5419–5436, Aug. 2024.

[34] S. Luo and W. Hu, "Score-based point cloud denoising," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2021, pp. 4583–4592.

[35] H. Chen, B. Du, S. Luo, and W. Hu, "Deep point set resampling via gradient fields," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 2913–2930, Mar. 2023.

[36] D. de Silva Edirimuni, X. Lu, Z. Shao, G. Li, A. Robles-Kelly, and Y. He, "IterativePFN: True iterative point cloud filtering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 13530–13539.

[37] M. Wei, H. Chen, Y. Zhang, H. Xie, Y. Guo, and J. Wang, "GeoDualCNN: Geometry-supporting dual convolutional neural network for noisy point clouds," *IEEE Trans. Vis. Comput. Graphics*, vol. 29, no. 2, pp. 1357–1370, Feb. 2023.

[38] D. de Silva Edirimuni, X. Lu, G. Li, and A. Robles-Kelly, "Contrastive learning for joint normal estimation and point cloud filtering," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 8, pp. 4527–4541, Aug. 2024.

[39] A. Mao, Z. Du, Y.-H. Wen, J. Xuan, and Y.-J. Liu, "PD-Flow: A point cloud denoising framework with normalizing flows," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2022, pp. 398–415.

[40] D. de Silva Edirimuni, X. Lu, G. Li, L. Wei, A. Robles-Kelly, and H. Li, "StraightPCF: Straight point cloud filtering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 20721–20730.

[41] X. Liu, C. Gong, and Q. Liu, "Flow straight and fast: Learning to generate and transfer data with rectified flow," in *Proc. 11th Int. Conf. Learn. Representations*, Kigali, Rwanda, May 2023.

[42] D. de Silva Edirimuni et al., "Hybrid long and short range flows for point cloud filtering," 2025. [Online]. Available: https://arxiv.org/abs/2508.08542

[43] Z. Wang, M. Li, S. Xin, and C. Tu, "Adaptive and iterative point cloud denoising with score-based diffusion model," *Comput. Graph. Forum*, vol. 44, no. 6, 2025, Art. no. e70149.

[44] T. Zhao, P. Gao, T. Tian, J. Ma, and J. Tian, "From noise addition to denoising: A self-variation capture network for point cloud optimization," *IEEE Trans. Vis. Comput. Graphics*, vol. 30, no. 7, pp. 3413–3426, Jul. 2024.

[45] X. Wang, W. Cui, R. Xiong, X. Fan, and D. Zhao, "FCNet: Learning noise-free features for point cloud denoising," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 11, pp. 6288–6301, Nov. 2023.

[46] T. Huang, S. Li, X. Jia, H. Lu, and J. Liu, "Neighbor2Neighbor: Self-supervised denoising from single noisy images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 14781–14790.

[47] J. Lehtinen et al., "Noise2Noise: Learning image restoration without clean data," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2971–2980.

[48] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[49] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2790–2799.

[50] A. Serna, B. Marcotegui, F. Goulette, and J.-E. Deschaud, "Paris-rue-madame database: A 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods," in *Proc. 4th Int. Conf. Pattern Recognit. Appl. Methods*, 2014, pp. 819–824.

[51] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3292–3310, Mar. 2023.

[52] P.-S. Wang, Y. Liu, and X. Tong, "Mesh denoising via cascaded normal regression," *ACM Trans. Graph.*, vol. 35, no. 6, 2016, Art. no. 232.

[53] G. Yang, F. Xue, Q. Zhang, K. Xie, C.-W. Fu, and H. Huang, "UrbanBIS: A large-scale benchmark for fine-grained urban building instance segmentation," in *Proc. ACM SIGGRAPH Conf.*, 2023, pp. 16:1–16:11.

[54] H. Fan, H. Su, and L. J. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 605–613.
[55] R. Li, X. Li, P.-A. Heng, and C.-W. Fu, "Point cloud upsampling via disentangled refinement," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 344–353.
[56] N. Ravi et al., "Accelerating 3D deep learning with PyTorch3D," 2020. [Online]. Available: https://arxiv.org/abs/2007.08501
[57] F. Hou, C. Wang, W. Wang, H. Qin, C. Qian, and Y. He, "Iterative poisson surface reconstruction (iPSR) for unoriented points," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 1–13, 2022.

**Peng Du** received the BS and MS degrees in information and computational science and computational mathematics from the Kunming University of Science and Technology, in 2015 and 2021, respectively. He is currently working toward the PhD degree in computer application technology with Beijing Normal University. His research interest include point cloud processing.

**Xingce Wang** became a post-doctoral fellow with the Institute of Brain and Cognitive Sciences, Beijing Normal University, Beijing, in 2010. She is a full professor with the School of Artificial Intelligence, Beijing Normal University, PR China. She is major in the geometric machine learning and computer graphics. Her current research interests include 3D model analysis, medical image processing, and artificial intelligence.

**Zhongke Wu** received the BS degree from the Department of Mathematics, Peking University, Beijing, in 1988, and the MS and PhD degrees from the Department of Aircraft Design, Beihang University, Beijing, in 1991 and 1995. From August 1999 to April 2006, he worked with the National Institute of High Performance Computing (IHPC) in Singapore, the National Institute of Information and Automation (INRIA) in France, and the Department of Computer Engineering, Nanyang Technological University (NTU) in Singapore. His research interests include the computer graphics and computer-aided geometric design, computer animation, virtual reality, medical image processing, etc.

**Xudong Ru** received the bachelor of science degree in computer science and technology from Xinxiang University, and the master of engineering degree in materials engineering from the North University of China. He is currently working toward the doctor of philosophy degree in computer application technology with Beijing Normal University. His research interests include medical image processing, computer graphics, and geometric deep learning.

**Xavier Granier** received the engineer and MS degrees from the Grenoble Institute of Technology, and the PhD diploma from the University Joseph Fourier, Grenoble, France. As a research scientist at Inria, he has created the research team MANAO, dedicated to computer graphics and optics. He is now a full professor with Institut d'Optique Graduate School - University of Paris Saclay. He is a member of the LP2N (Laboratory for Photonics, Numerics and Nanosciences). He is co-leading a national consortium dedicated to the use of 3D in digital humanities. He is research thus spans from computer graphics and optics to cultural heritage.

**Ying He** is an associate professor with the College of Computing and Data Science and the director with the Centre for Augmented and Virtual Reality, Nanyang Technological University, Singapore. His research focuses on geometric computing and analysis. He has served on the editorial boards of *IEEE Transactions on Visualization and Computer Graphics*, *Computer Graphics Forum*, and *Computational Visual Media*, and has been an active member of the technical program committees of leading conferences in geometric modeling. He has also held leadership roles as General or Program co-chair for Shape Modeling International (2022), the Symposium on Solid and Physical Modeling (2022 and 2023), the Geometric Modeling and Processing Conference (2014 and 2021), the Conference on Computational Visual Media (2020), and Pacific Graphics (2026).