


MODNet: Multi-offset Point Cloud Denoising Network Customized for Multi-scale Patches

A. Huang¹ , Q. Xie², Z. Wang¹, D. Lu³, M. Wei¹ and J. Wang¹ †

¹Nanjing University of Aeronautics and Astronautics, Nanjing, China

²University of Oxford, Oxford, UK

³University of Waterloo, Waterloo, Canada

Abstract

The intricacy of 3D surfaces often results cutting-edge point cloud denoising (PCD) models in surface degradation including remnant noise, wrongly-removed geometric details. Although using multi-scale patches to encode the geometry of a point has become the common wisdom in PCD, we find that simple aggregation of extracted multi-scale features can not adaptively utilize the appropriate scale information according to the geometric information around noisy points. It leads to surface degradation, especially for points close to edges and points on complex curved surfaces. We raise an intriguing question – if employing multi-scale geometric perception information to guide the network to utilize multi-scale information, can eliminate the severe surface degradation problem? To answer it, we propose a Multi-offset Denoising Network (MODNet) customized for multi-scale patches. First, we extract the low-level feature of three scales patches by patch feature encoders. Second, a multi-scale perception module is designed to embed multi-scale geometric information for each scale feature and regress multi-scale weights to guide a multi-offset denoising displacement. Third, a multi-offset decoder regresses three scale offsets, which are guided by the multi-scale weights to predict the final displacement by weighting them adaptively. Experiments demonstrate that our method achieves new state-of-the-art performance on both synthetic and real-scanned datasets. Our code is publicly available at <https://github.com/hay-001/MODNet>.

CCS Concepts

• **Computing methodologies** → **Point-based models**; **Shape analysis**;

1. Introduction

With the popularity of consumer-level 3D scanners, point clouds are widely used in research areas such as robotics [HSH*20], and 3D measurement [XLH*20]. Regrettably, raw point clouds scanned by 3D scanners are often noisy. Noisy point clouds highly affect its accuracy and characteristics, making point cloud denoising a vital step before further processing.

Existing point cloud denoising methods include optimization-based methods [ABCO*01, MC17, ZCN*19, ABCO*03, LCOLTE07, HLZ*09, SSW15, HHWG21] and deep learning-based methods [RLBG*20, RÖPG18, PFVM20]. In recent years, the research of deep learning-based point cloud denoising expands substantially thanks to advanced neural network architectures proposed for point clouds [QSMG17, QYSG17, TQD*19]. Among them, pioneering works such as PointCleanNet [RLBG*20], TotalDn [HRR19] and Pointfilter [ZLQH20] are proposed. Pointfilter and PointCleanNet take as input single-scale neighboring points of

noisy points, as illustrated in Fig. 1 (a). When the scale of the patch is not appropriate, this will introduce redundant neighborhood geometric information (leading to performance degradation) or lack sufficient neighborhood geometric information (leading to remnant noise). It makes the network to be less robust to point clouds with various densities, noisy levels and complex curved surface structure. ECNet [YLF*18] and TotalDn [HRR19] employ as input multi-scale neighboring points of noisy points, while only aggregating the extracted multi-scale features simply, as shown in Fig. 1 (b). It makes the network to be less robust to deal with redundant neighborhood geometric information (leading to performance degradation). Thus, none of these methods adaptively use the appropriate scale information according to the geometric information around noisy points, leading to underutilization of the multi-scale information.

To address this issue, we design a novel multi-offset denoising network, MODNet. Our network is an encoder-decoder-based architecture which straightforwardly takes as input three scale raw neighboring points of each noisy point. It predicts three different scale denoising offsets, which are guided by the multi-scale

† J. Wang is the corresponding author.

weights to predict the final denoising displacement, as shown in Fig. 1 (c). MODNet is also the first denoising neural network customized for multi-scale patches input, which makes full use of the advantage of multi-scale information. Our MODNet consists of three parts: Patch Feature Encoder (PFE), Multi-Scale Perception Module (MSPM) and Multi-Offset Decoder (MOD). First, each patch feature encoder employs a PointNet-based feature extractor to extract the low-level features of each scale patch. Then, the outputs of patch feature encoders are fed into the multi-scale perception module for embedding multi-scale information for each scale low-level feature. And it regresses the multi-scale weights to guide multi-offset denoising. Finally, the multi-offset decoder predicts three scale denoising offsets, which are guided by the multi-scale weights to predict the final denoising displacement by weighting them adaptively. Given a noisy point cloud as input, our MODNet can automatically and robustly predict a corresponding clean point cloud, by removing noise and preserving more geometric details. Extensive experiments demonstrate the denoising capability of our MODNet under a variety of noise models, better performance than the state-of-the-art techniques, in terms of both visual quality and error metrics. The main contributions of our work include:

- We propose a novel point cloud denoising network, which takes as input multi-scale patches of noisy points. MODNet, to the best of our knowledge, is the first denoising neural network customized for multi-scale patches input, which is able to adaptively utilize appropriate scale information of each point according to the multi-scale geometric perception information.
- We design a multi-scale feature fusion perception module, named Multi-scale Perception Module (MSPM), for the efficient embedding of multi-scale information for each scale feature and guiding the multi-offset denoising displacement.
- To make better use of the advantage of multi-scale information, we introduce a Multi-offset Decoder (MOD), which is capable of regressing three different scale denoising offsets. Finally, these denoising offsets are guided by the multi-scale weights to predict the final denoising displacement by weighting them adaptively.

2. Related Work

2.1. Optimization-based denoising

Early methods of 3D point cloud denoising are mainly based on optimization. These optimization-based methods design different geometric constraints for denoising and solve them through optimization methods. Existing optimization-based denoising methods can be divided into the following categories: local surface fitting methods, sparsity-based methods and graph-based methods.

The first category methods project noisy points onto the fitted surface. Alexa et al. [ABCO*01] proposes a moving least squares (MLS) approach to calculate the optimal fitting surface of the point cloud. Its robust extensions [ABCO*03, GG07, ÖGG09] based on MLS are also widely used for point cloud denoising. Similarly, other surface fitting methods [CP05, LCOLTE07, HLZ*09, HWG*13] have been proposed for point cloud denoising. Cazals et al. [CP05] proposed jet fitting, which encodes all local geometric quantities, such as normal direction and curvatures. Lipman et al. [LCOLTE07] proposed the locally optimal projection

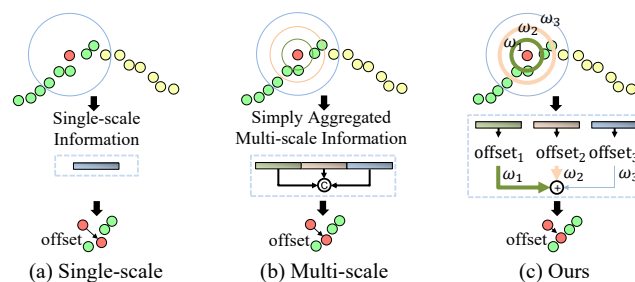


Figure 1: Illustration of different ways to use patch information on denoising network. Different ring line widths indicate the weights in the utilization of various scale information on network according to different neighboring geometric information of central points (Red points). Yellow points are the negative effect points for denoising. $offset_1$, $offset_2$ and $offset_3$ are three different scale denoising offsets, which are guided by the multi-scale weights to predict the final offset (i.e., displacement) by weighting them adaptively.

(LOP) to produce a set of points to describe the underlying surface. Huang et al. [HLZ*09, HWG*13] further advanced this technique, e.g., weighted LOP (WLOP), whereas these methods tend to over-smooth the results, while removing the high-level noise.

Sparsity-based denoising methods are based on the sparse representation theory [ASGCO10, SSW15, MC17]. These methods mainly optimize the coordinates of points by reconstructing normals. Mattei et al. [MC17] proposed Moving Robust Principal Components Analysis (MRPCA) to restore the noisy point cloud without oriented normals. Whereas, These methods are less robust to high-level noisy point cloud.

Finally, graph-based denoising methods are based on the graph signal processing theory [SNF*13]. These techniques [SPV15, HPL*21, HGCG20, HHWG21] mainly employ graph filters to denoise the point cloud represented by graphs. Zeng et al. [ZCN*19] proposed patch-based graph Laplacian regularization (GLR) for denoising. However, the denoising performance of graph-based denoising methods for high-level noisy point cloud is still unstable.

All these optimization-based methods are mainly based on geometric priors, so there are often difficulties in balancing denoising performance and preserving sharp details.

2.2. Deep learning-based denoising

In recent years, the research of deep learning-based point cloud denoising expands substantially thanks to advanced neural network architectures proposed for point clouds [QSMG17, QYSG17, LXW*22]. Existing deep learning-based denoising methods mainly take as input a single-scale patch to encode the local geometric information and predict the displacement and apply the inverse displacement to each noisy point. For instance, Point-CleanNet(PCN) [RLBG*20], similar to the framework of PCP-Net [GKOM18], was proposed to estimate denoising displacement with a single patch as input and remove outliers separately in the noisy point cloud. Zhang et al. [ZLQH20] proposed the Pointfilter,

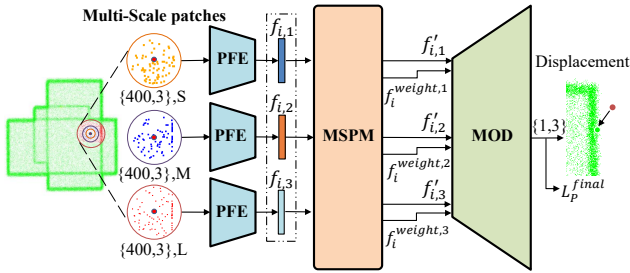


Figure 2: The architecture of MODNet. The input of MODNet are multi-scale patches of a noisy point. Then, the PFE (Section 3.3) extracts the low-level features of each scale patch (indicated by blue, orange and light blue). The MSPM (Section 3.4) is used to embed multi-scale information into each scale low-level feature and regress multi-scale weights for MOD. The MOD (Section 3.5) predicts the final displacement.

a point-wise learning encoder-decoder-based framework to take as input a patch. And it learns a latent displacement supervised by a loss function sensitive to sharp features. However, above methods are less robust to denoise point clouds with various densities and noisy levels. Recently, multi-scale neighboring information is demonstrated to be critical for point cloud denoising by several multi-scale patches based methods. Yu et al. [YLF*18] presented an edge-aware point cloud consolidation network to achieve denoising trivial low-level noise, which takes PointNet++ [QYSG17] for multi-scale feature embedding. But it cannot well preserve tiny features of surfaces. For achieving unsupervised deep learning-based denoising, TotalDn [HRR19] was proposed. In TotalDn, it introduces a spatial prior term, that steers converges to the unique closest out of the many possible modes on a manifold by using two scale information. However, denoising results of TotalDn shrink dramatically. Regretfully, these multi-scale patches based methods just simply aggregate the extracted multi-scale features. To summarize, none of these methods adaptively use the appropriate scale information according to the geometric information around noisy points, leading to underutilization of the multi-scale information.

Furthermore, some other neural network architectures were proposed for denoising, which use CNN and graph convolution to achieve point cloud denoising. Roveri et al. [RÖPG18] proposed the PointProNets, a CNN-based framework to consolidate high-quality point cloud. Francesca et al. [PFVM20] proposed GPDNet, a graph convolutional network, to achieve robust denoising.

3. Approach

3.1. Overview

Given a noisy point cloud, we aim to recover the underlying noise-free point cloud by removing the additive noise. We formulate a noisy point cloud intuitively as follows:

$$\hat{P} = P + N, \quad (1)$$

where $\hat{P} = \{\hat{p}_1, \hat{p}_2, \dots, \hat{p}_n | \hat{p}_n \in \mathbb{R}^3\}$ is a noisy point cloud, P is the corresponding clean point cloud. N is the additive noise.

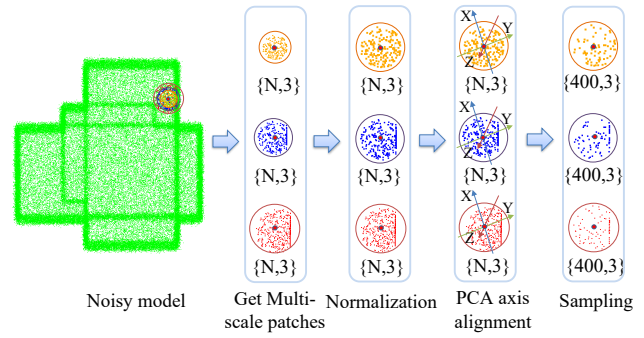


Figure 3: The preprocessing of multi-scale patches.

We solve this denoising problem in a local way, which means denoising a noisy point only depends on its neighboring points (patch). Specifically, we employ as input multi-scale neighboring points of a noisy point, defined as follows:

$$\begin{aligned} \hat{\phi}_i &= \{\hat{\phi}_{i,1}, \dots, \hat{\phi}_{i,k}, \dots, \hat{\phi}_{i,K}\}, \\ \phi_i &= \{\phi_{i,1}, \dots, \phi_{i,k}, \dots, \phi_{i,K}\}, \end{aligned} \quad (2)$$

$$\begin{aligned} \hat{\phi}_{i,k} &= \{p_{j,k} | \|p_{j,k} - \hat{p}_i\| < r_k\}, \\ \phi_{i,k} &= \{p_{j,k} | \|p_{j,k} - \hat{p}_i\| < r_k\}, \end{aligned} \quad (3)$$

where $\hat{p}_i, p_{j,k} \in \hat{P}$, $p_{j,k} \in P$ and $\hat{\phi}_i$ is a set of noisy multi-scale patches $\hat{\phi}_{i,k}$ of point \hat{p}_i and those corresponding ground truth multi-scale patches ϕ_i from a pair of point clouds \hat{P} and P . r_k is the patch radius of scale k .

Similar to PointCleanNet [RLBG*20], we aim to learn the displacement for each point, rather than directly learning clean point cloud. Therefore, we formulate the denoising model:

$$\dot{p}_i = \hat{p}_i + f(\hat{\phi}_i), \quad (4)$$

where $f(\cdot)$ represents our MODNet, and \dot{p}_i is the filtered point of the noisy point \hat{p}_i .

The overview of our MODNet is depicted in Fig. 2, consisting of three parts: Patch Feature Encoder (PFE), Multi-Scale Perception Module (MSPM) and Multi-Offset Decoder (MOD). First, each patch feature encoder employs a PointNet-based feature extractors to extract the low-level features of each scale patch. Then, the outputs of PFE are fed into the multi-scale perception module for embedding multi-scale information for each scale low-level feature. And it regresses the multi-scale weights to guide multi-offset denoising. Finally, the multi-offset decoder predicts three different scale denoising offsets. Then, These three scale denoising offsets are guided by the multi-scale weights to predict the final denoising displacement by weighting them adaptively.

3.2. Multi-scale Patches Preprocessing

In order to improve the robustness of denoising, it is necessary to preprocess the multi-scale patches. The preprocessing of

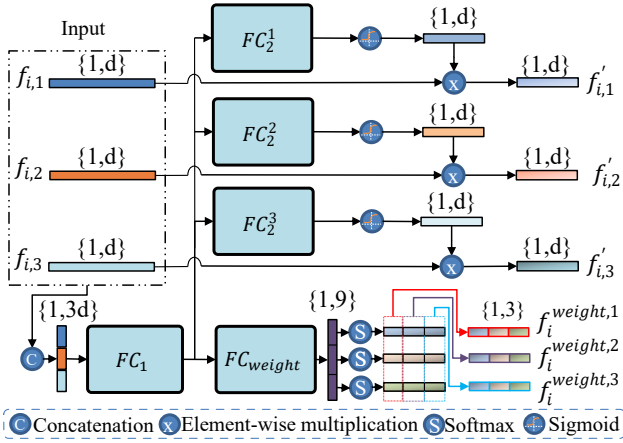


Figure 4: Illustration of MSPM in MODNet. Our method employs an attention mechanism to embed multi-scale information into each scale low-level feature and regress multi-scale weights for MOD.

multi-scale patches similar to [ZLQH20] is shown in Fig 3. The first step is to normalize the multi-scale patches, i.e., $\hat{\phi}_i = (\hat{\phi}_i - \hat{p}_i) / r_{max}$, $\hat{\varphi}_i = (\hat{\varphi}_i - \hat{p}_i) / r_{max}$, where r_{max} is radius of the max scale. For rotation invariance, we align each patch by aligning its principle axes of the PCA with the Cartesian space. More specifically, the z-axis and x-axis of the patches are aligned with the last and the second principle axis respectively. Finally, to fix the input of the MOD-Net, we set the number of scales $K = 3$, and the point number of patch $|p_{i,k}| = 400$. We pad the origin for patches with less than 400 points and randomly downsample the patches with more than 400 points. We set the radius r_k of the three patches to 3%, 4% and 5% of the model's bounding box diagonal length.

3.3. Patch Feature Encoder

Given three patches of different scales, the goal of three patch feature encoders is to extract the low-level features of each patch. For the backbone with patch as input, the encoder based on Pointnet [QSMG17] shows impressive performance compared to other backbones [ZLQH20]. We use a set of multi-layer perceptrons (MLP) with batch normalization and Relu function to encode each point of patch into per point feature. Then, we employ max pooling layers to obtain three low-level features $f_{i,1}$, $f_{i,2}$ and $f_{i,3}$.

3.4. Multi-scale Perception Module

A single-scale patch as input has some limitations in perceiving the geometric neighborhood information required for denoising. The excessively small scale has an insufficient perception of the geometric neighborhood information required for denoising, while the excessively large scale may introduce redundant geometric neighborhood information, resulting in performance degradation. Therefore, our MODNet takes as input multi-scale patches. Multi-scale patches provide more geometric neighborhood information and the choice of adaptively selecting the optimal scale information.

First, the multi-scale perception module (MSPM) aggregates the outputs of the three encoders to obtain a multi-scale fusion feature as shown in Fig. 4. The multi-scale fusion feature can fully perceive the neighborhood geometric structure information around the \hat{p}_i . Therefore, we call the fused vector as multi-scale feature vector. It can be expressed as follows:

$$f_i = \text{Concat}(f_{i,1}, f_{i,2}, f_{i,3}), \quad (5)$$

where $f_i \in \mathbb{R}^{1 \times 1536}$ is the multi-scale feature vector, and $\text{Concat}(\cdot)$ is the concatenation operation.

Then, the MSPM adopts the attention mechanism to embed the multi-scale feature to each scale low-level feature, which can be expressed as follows:

$$f'_{i,k} = f_{i,k} \otimes \text{Sig}(FC_2^k(FC_1(f_i))), \quad (6)$$

where $FC_1(\cdot)$, $FC_2^k(\cdot)$ are fully connected layers, and $\text{Sig}(\cdot)$ denotes the Sigmoid function. \otimes is the element-wise multiplication.

Meanwhile, the multi-scale perception module regresses three multi-scale weights (1×3) to guide the final multi-offset displacement, which can be expressed as follows:

$$[f_i^{x'}, f_i^{y'}, f_i^{z'}] = \text{RP}(FC_{\text{weight}}(FC_1(f_i))), \quad (7)$$

$$[f_i^{\text{weight},1}, f_i^{\text{weight},2}, f_i^{\text{weight},3}] = [\text{Soft}(f_i^{x'}), \text{Soft}(f_i^{y'}), \text{Soft}(f_i^{z'})]^T, \quad (8)$$

where $FC_{\text{weight}}(\cdot)$ is two-layer fully connected layers (FC), $\text{Soft}(\cdot)$ denotes the Softmax function, and $\text{RP}(\cdot)$ converts the vector (1×9) to a weight matrix (3×3).

3.5. Multi-offset Decoder

For noisy points with different geometric information around these points, the suitable scale information is not universal. Therefore, our method tends to adaptively use the appropriate scale information according to the perceived multi-scale information around noisy points, as illustrated in Fig. 1. Instead of directly predicting the denoising displacement like existing denoising network, we propose a multi-offset decoder, as shown in Fig. 5. It can better exploit the advantages of multi-scale information. First, multi-offset decoder employs three sub-decoders composed of a series of fully connected layers to predict denoising offsets under three different scale patches. The sub-decoder can be formulated as follows:

$$f''_{i,k} = FC_{dc1}^k(f'_{i,k}), \quad (9)$$

$$dp_{i,k}^{\text{pre}} = FC_{dc2}^k(f''_{i,k}), \quad (10)$$

$$\text{offset}_{i,k} = FC_{dc3}^k(f''_{i,k}), \quad (11)$$

where $FC_{dc1}^k(\cdot)$, $FC_{dc2}^k(\cdot)$ and $FC_{dc3}^k(\cdot)$ are fully connected layers (FC). $\text{offset}_{i,k}$ is i -th point offset of k -th scale. And three single scale denoising offsets $dp_{i,k}^{\text{pre}}$ are supervised by three different loss functions. Specifically, these denoising offsets are not used for final

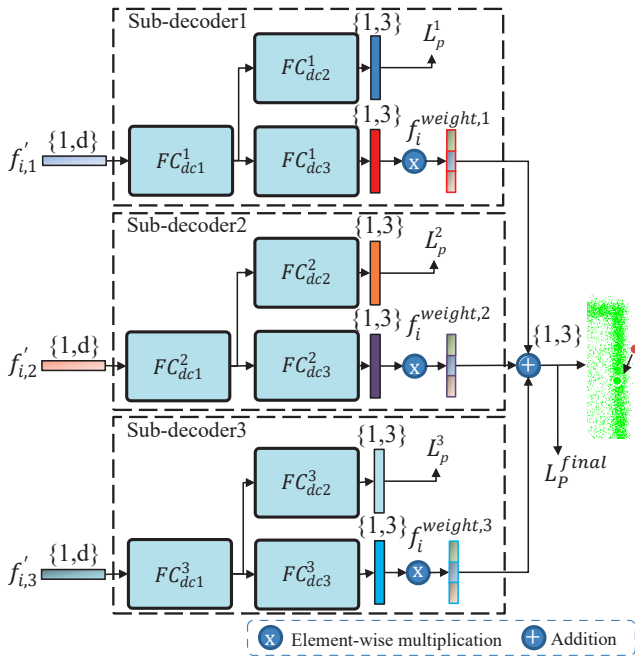


Figure 5: The structure of Multi-offset Decoder (MOD) in MODNet. It regresses three different scale denoising offsets, which are guided by the multi-scale weights to predict the final denoising displacement by weighting them adaptively.

denoising, but to lead sub-decoders of different scales to focus on learning the key feature of denoising at their respective scales. The specific introduction is shown in Section 3.6. Then these three different scale denoising offsets are guided by the multi-scale weights to predict the final denoising displacement by weighting them adaptively. It can be expressed as follows:

$$dp_i = offset_{i,1} \otimes f_i^{weight,1} + offset_{i,2} \otimes f_i^{weight,2} + offset_{i,3} \otimes f_i^{weight,3}, \quad (12)$$

where dp_i is the final denoising displacement of i -th noisy point. \otimes is the element-wise multiplication.

3.6. Loss

For the performance of point cloud denoising neural network, the choice of loss is critical. The L2 distance which has been used in PointCleanNet [RLBG*20] is the general loss for point clouds denoising. Inspired by pointfilter [ZLQH20], the loss function L_p for single scale patch consists of a projection loss function L_s^k and a repulsion term L_r^k . The function L_s^k preserves sharp features by considering the distance and the normal similarity between the current denoising point \hat{p}_i and its neighboring points of ground truth patch.

The projection loss function L_s^k takes the form:

$$L_s^k = \frac{\sum_{p_{j,k} \in \mathcal{O}_{i,k}} |(\hat{p}_i - p_{j,k}) \cdot n_{p_{j,k}}^T| \cdot \phi_i^k \cdot \theta_i^k}{\sum_{p_{j,k} \in \mathcal{O}_{i,k}} \phi_i^k \cdot \theta_i^k}, \quad (13)$$

$$\phi_i^k = \exp\left(-\frac{\|\hat{p}_i - p_{j,k}\|^2}{\varepsilon_p^2}\right), \quad (14)$$

$$\theta_i^k = \exp\left(-\frac{1 - n_{\hat{p}_i}^T \cdot n_{p_{j,k}}}{1 - \cos(\varepsilon_n)}\right), \quad (15)$$

where \hat{p}_i is the filtered point of the noisy point \hat{p}_i , and $n_{p_{j,k}}$ is the ground-truth normal of the point $p_{j,k}$. $\phi(\|\hat{p}_i - p_{j,k}\|)$ is a Gaussian function giving larger weights to the points near \hat{p}_i . And $\theta(n_{\hat{p}_i}, n_{p_j})$ is a function giving larger weights to the neighboring points in the patch with more similar normal to \hat{p}_i . ε_p is defined as $\varepsilon_p = 4\sqrt{dobb}/m$ and $dobb$ is the length of the diagonal of the bounding box of patch $\mathcal{O}_{i,k}$. The m denotes the point number of ground truth patch. ε_n is the support angle (15° by default). Furthermore, the repulsion term L_r^k makes the denoising points be distributed uniformly. It is defined as

$$L_r^k = \max_{p_j \in \mathcal{O}_i} |\hat{p}_i - p_j|, \quad (16)$$

Overall, we formulate the loss function L_p as:

$$L_p = \alpha L_s^k + (1 - \alpha) L_r^k, \quad (17)$$

where α is a trade-off parameter to control the repulsion force, and we empirically set $\alpha = 0.97$ in our training stage.

Then the loss function L_{dp} of three scales is formulated as:

$$L_{dp} = L_p^1 + L_p^2 + L_p^3, \quad (18)$$

where L_p^1, L_p^2 and L_p^3 respectively denote the loss function L_p for offsets of three different scale. Finally, we combine all of losses:

$$L = \beta \cdot L_{dp} + L_p^{final}, \quad (19)$$

where β is a trade-off parameter to balance the different losses, and we set $\beta = 0.2$ in our training stage. The L_p^{final} denotes the projection loss function of the final denoising displacement and we set the value of ground truth $|\mathcal{O}_{i,final}| = 500, r_{final} = 5\%$. These hyperparameters are obtained according to the experimental verification.

4. Results and Discussions

4.1. Datasets

The training dataset consists of 11 CAD clean models and 11 clean non-CAD models provided by the pointfilter [ZLQH20]. And we sample 10000, 20000 and 50000 points on each clean model. Considering that the noise of depth cameras, like Kinect, is similar to Gaussian noise [NIL12], we assume noise as Gaussian distribution like most methods. Based on this, each clean model corresponds to 6 synthesized noisy models with 6 standard deviations of the clean model's bounding box diagonal length (0.0% to 1.5%) Gaussian noise. Besides, the normal information for clean models is only required in training phase.

Metrics	Model	0.5%			1%			1.5%			Average
		10K	20K	50K	10K	20K	50K	10K	20K	50K	
CD(10^{-4})	WLOP [HLZ*09]	8.68	6.33	4.19	13.2	7.26	7.04	19.0	10.7	12.55	9.88
	ECN [YLF*18]	5.12	2.85	1.29	6.40	3.74	2.08	7.90	5.14	4.11	4.29
	PCN [RLBG*20]	3.39	2.39	1.22	7.33	4.09	1.62	11.5	6.32	2.09	4.44
	GPD [PFVM20]	3.39	2.43	1.44	4.87	2.84	1.45	6.23	3.64	1.99	3.14
	PF [ZLQH20]	2.90	1.84	0.92	5.17	2.86	1.31	7.40	4.11	1.93	3.16
	Ours	2.73	1.83	0.93	4.68	2.64	1.26	5.96	3.31	1.63	2.78
MSE(10^{-2})	WLOP [HLZ*09]	4.22	2.72	2.23	4.45	2.89	2.67	4.88	3.32	3.34	3.41
	ECN [YLF*18]	3.58	2.54	1.64	3.64	2.65	1.79	3.76	2.82	2.16	2.73
	PCN [RLBG*20]	3.51	2.49	1.59	3.70	2.63	1.65	4.04	2.85	1.72	2.69
	GPD [PFVM20]	3.53	2.55	1.69	3.47	2.53	1.69	3.41	2.53	1.69	2.57
	PF [ZLQH20]	3.41	2.41	1.55	3.44	2.46	1.59	3.59	2.58	1.69	2.52
	Ours	3.40	2.40	1.54	3.39	2.44	1.58	3.46	2.50	1.64	2.48
P2M(10^{-4})	WLOP [HLZ*09]	2.88	2.66	1.394	4.79	3.19	2.49	7.71	5.25	4.92	3.92
	ECN [YLF*18]	1.30	0.68	0.29	1.84	1.07	0.63	2.54	1.70	1.49	1.28
	PCN [RLBG*20]	1.04	0.59	0.28	2.35	1.24	0.44	4.30	2.29	0.65	1.46
	GPD [PFVM20]	1.07	0.62	0.25	2.18	1.47	0.58	2.52	1.31	0.78	1.20
	PF [ZLQH20]	0.77	0.37	0.16	1.41	0.72	0.32	2.39	1.28	0.58	0.89
	Ours	0.71	0.38	0.17	1.20	0.64	0.31	1.78	0.97	0.49	0.74

Table 1: Quantitative comparisons by testing various methods on synthetic noisy models with different noise levels and point densities.

To verify the performance of our model, we perform it on synthetic models which contain 10000, 20000 and 50000 points randomly sampled from its original surface. Similar to the training set, the synthetic models consist of 96 clean models and 288 synthetic noisy models with 3 standard deviations of the clean model's bounding box diagonal length (0.5%, 1% and 1.5%) Gaussian noise. Moreover, we test a real-world dataset Paris-rue-Madame [SMGD14] to verify the effectiveness of the method in some real scenarios.

4.2. Training details

Our MODNet is implemented using PyTorch. Our model is trained on a PC with a single Nvidia GeForce RTX 2070s GPU (8GB memory). We employ the SGD optimizer to train our MODNet with a batch size of 200. We set 40 training epochs and $1e-4$ initial learning rate. And the learning rate decreases from $1e-4$ to $1e-7$ with increasing epochs. The training of MODNet takes about 12 hours.

4.3. Evaluation Metric

To evaluate the denoising performance of different methods, we adopted the Chamfer Distance (CD) to evaluate the "global similarity" between the filtered point cloud and its corresponding ground truth. The Chamfer Distance metric mutually detects the nearest neighbor in the other point cloud and sums the squared distances up. The CD can be expressed as:

$$CD(\hat{P}, P) = \frac{1}{M} \sum_{p_i \in P} \min_{\hat{p}_j \in \hat{P}} (\|p_i - \hat{p}_j\|_2^2) + \frac{1}{\hat{M}} \sum_{\hat{p}_j \in \hat{P}} \min_{p_i \in P} (\|\hat{p}_j - p_i\|_2^2), \quad (20)$$

Model	Runtime(s)		
	10K	20K	50K
WLOP	2.21	4.51	11.2
ECN	1.93	3.87	9.28
GPD	21.3	43.2	99.5
PCN	55.0	103	245
PF	5.50	10.2	24.6
Ours	16.3	32.4	81.1

Table 2: Runtime performance (in seconds) for different approaches on three test subsets with different number of points. All examples were run on the same computer configurations.

Here \hat{P}, P are the filtered point cloud and its corresponding ground truth point cloud. \hat{M} and M are the cardinalities of the filtered \hat{P} and ground truth P point clouds.

Moreover, to evaluate the "local similarity" between the filtered point cloud and its corresponding ground truth, we employ the mean square distance error (MSE) [LWC*17]. The MSE metric evaluates the distance between each point in ground truth and its closest points in the filtered point cloud. It takes the form:

$$D(p_i) = \frac{1}{N} \sum_{\hat{p}_j \in NN(p_i)} \|p_i - \hat{p}_j\|_2^2 \quad (21)$$

where p_i is the ground truth point and \hat{p}_j is one of its neighboring point in the filtered point cloud. $NN(*)$ represents the nearest neighbors, and $N = |NN(p_i)|$. we set N to 10.

Besides the two above metrics, we also introduce the point-to-mesh (P2M) [RRN*20] to evaluate the error between the filtered point cloud and its corresponding ground truth mesh.

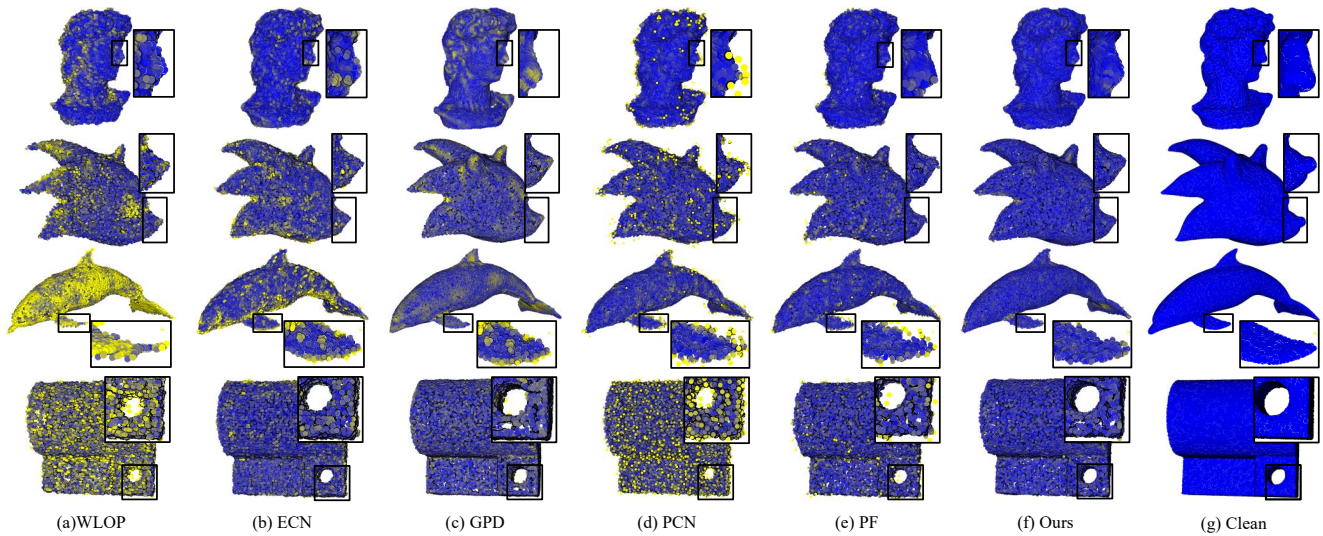


Figure 6: Visual comparisons of denoising methods under Gaussian noise. Yellower Points are farther away from the ground truth surface. Obviously, our method outperforms others in terms of both noise removal and feature preservation; see particularly the blown-up views.

4.4. Quantitative Comparison

Errors. We compare our MODNet quantitatively to the state-of-the-art denoising methods, including WLOP [HLZ*09], ECN [YLF*18], GPD [PFVM20], PCN [RLBG*20] and PF [ZLQH20]. We first calculate the errors over the test set, in terms of the Chamfer distance (CD), mean square error (MSE) and point-to-mesh distance (P2M). As presented in Table 1, our MODNet significantly outperforms previous existing methods. In particular, our method still has the promising performance whether input is low-density point cloud with 10000 points lacking local geometric information or point cloud with a 1.5% high-level noise.

Runtime. Table 2 summarizes the runtime of each method on test set. As for optimization-based methods, the WLOP is fast than most of deep learning-based methods. However, in order to acquire better results, WLOP needs multi-step preprocessing and trial-and-error efforts to tune parameters, which take a much longer extra time in practice. On the contrary, ECN is the fastest method among the deep-learning-based methods (GPD, PCN, PF and ours) and even faster than optimized-based methods (WLOP). And PCN is the slowest method due to its complex backbone structure. Our MODNet is faster than PCN and GPD, but slower than PF. This is because our MODNet requires to search multi-scale patches for each noisy point and extract the feature of each scale's patch.

Robustness. To test the generalizability of our MODNet, We use a variety of other noise models to test our models, including discrete noise (DN), Laplace noise (LN) and uniform noise (UN). Moreover, the performance of our model is then evaluated using 3% high-level Gaussian noise (HLN). From Table. 3, we can observe that our MODNet achieves superior performance of generalizability for other noise models and high-level noise. As such, our MODNet is more robust than other compared methods.

4.5. Visual Comparison

Besides the quantitative comparisons on synthetic dataset, we also show the visual comparisons on both synthetic noisy point clouds and raw-scan point clouds. Fig. 6 shows the denoising results from the proposed method and competitive baselines under 4 sets of 1.5% Gaussian noisy point clouds. In order to test these methods on point clouds of different geometric complexity, these point clouds include both CAD models and non-CAD models. From the results, the WLOP can not balance the denoising and the feature preservation. ECN and PCN have relatively weak denoising performance for point clouds with high-level noise. GPD tends to shape shrinkage. As a contrast, our MODNet significantly outperforms previous methods on both the simple parts point clouds and the complex curved surface point clouds. Our method not only achieves high performance denoising, but also does not lead to shape shrinkage and over-smooth geometric features.

As for raw-scan point clouds, we also verify the effectiveness of our method on the real-world dataset Paris-rue-Madame [SMGD14] corrupted with raw noise. Since the ground truth models of these raw-scan point sets are not available, it may make the results difficult to objectively quantify. However, the performance of our method can still be partially demonstrated through visual comparison. And it is meaningful and essential to apply the proposed denoising method to raw scanned point clouds. Therefore, we still demonstrate the visual comparisons (Fig. 7) with other methods, as suggested by previous techniques [ZLQH20, HRR19].

WLOP, ECN tend to oversmooth the results, As shown in Fig. 7 (b, c). The shape of the point sets after GPD denoising are shrunk a bit. The PCN and PF seem not robust enough on raw-scan point clouds. As shown in Fig. 7 (e, f), there is still a lot of noise after PCN or PF denoising. As a contrast, our MODNet can balance the denoising and the feature preservation well.

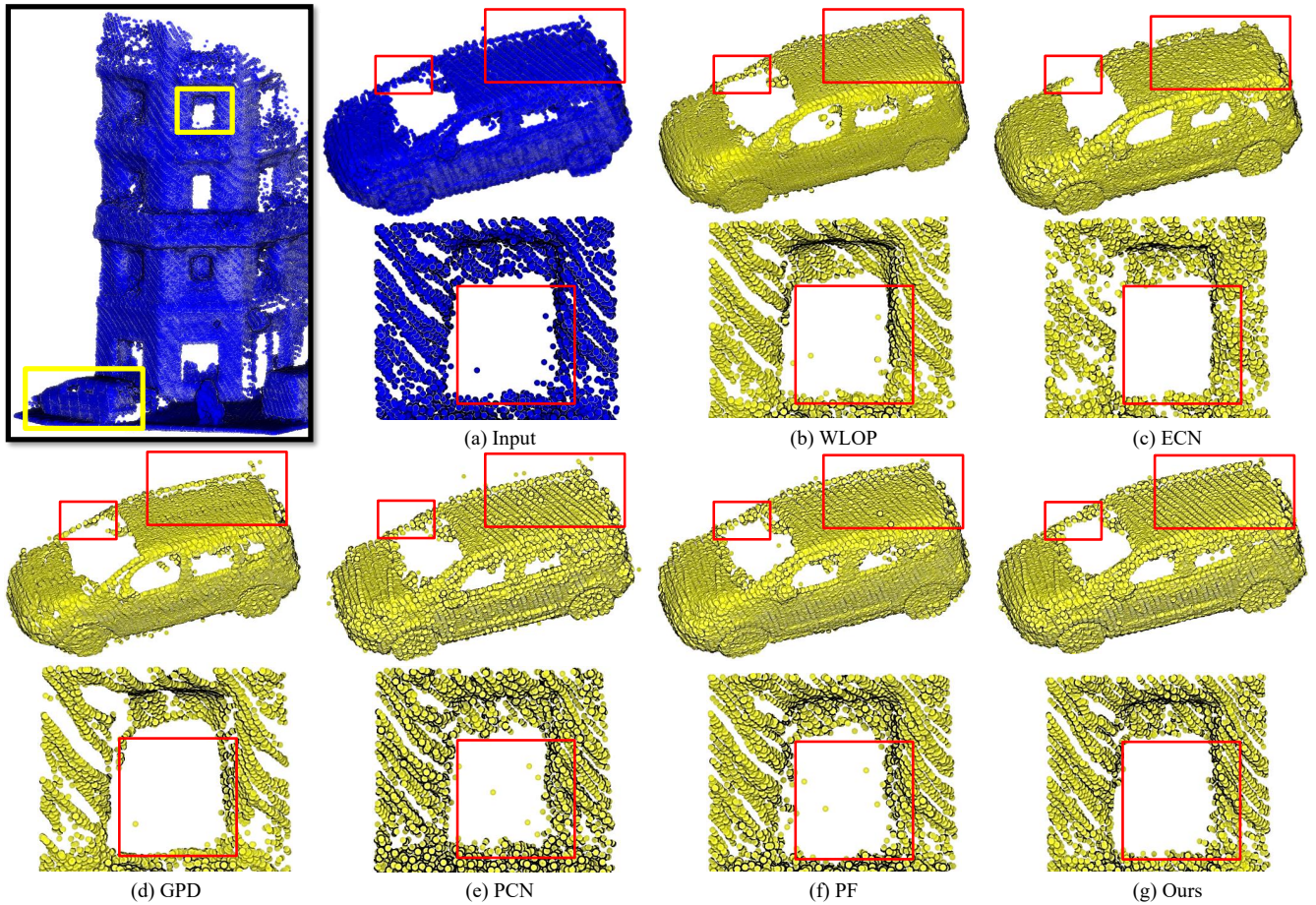


Figure 7: Visual results of denoising methods on the real-world dataset Paris-rue-Madame.

In order to visualize the denoising results more intuitively, we reconstruct meshes from denoised point clouds by Poisson reconstruction. Fig. 8 shows the meshes of denoising results from the proposed method and competitive baselines under 3 sets of 1.0% Gaussian noisy point clouds. Since the reconstructed surface of WLOP is poor, it is not shown. Only deep learning-based methods are compared here. From Fig. 8 (b), although GPD outputs smoothing reconstructed meshes, it fails to preserve sharp features. The reconstructed mesh surface of ECN, PF, and PCN is relatively rough. The reconstructed mesh surface of ECN is the roughest, even taking as input multi-scale patches. Therefore, we believe that simply aggregation multi-scale features is not necessarily effective in denoising task. In contrast, the meshes of the point clouds filtered by our method are cleaner and preserve more geometric details.

Furthermore, in order to demonstrate that MODNet can adaptively use the appropriate scale information according to the geometric information around noisy points, we sum the weights of the three offsets respectively. Then we visualize these weights. As shown in Fig. 9 (a,b,c), small-scale information has the greatest effect on denoising. For those edge points, MODNet tends to

use more medium-scale and large-scale information. As for those points on plane, MODNet tends to use more small-scale information. This coincides with our priori knowledge. For points on simple geometric structures such as planes, small-scale information is sufficient. Middle-scale and large-scale information instead introduces redundant information. It sometimes even causes surface degradation. For points on complex geometric structures such as edges, the addition of middle-scale and large-scale information leads to better denoising results. As shown in Fig. 9 (d, e, f), the higher noise level of point cloud require more middle-scale and large-scale information for denoising. From the visualization results of Fig. 9 (g, h, i), higher density of point cloud need less middle-scale and large-scale information. The above results also demonstrate the adaptive process of our MODNet for different point cloud noise levels and point cloud densities.

4.6. Hyperparameter selection

In order to achieve better performance of our network, we do some comparative experiments on following hyperparameters, as shown in Table 4. The first hyperparameter β is a trade-off parameter

Metrics	Model	Noise			
		DN	LN	UN	HLN
CD(10^{-4})	WLOP	8.81	9.38	12.5	15.2
	ECN	5.31	6.12	5.84	12.9
	PCN	3.45	3.79	7.12	12.7
	GPD	3.42	3.89	4.25	12.1
	PF	2.88	3.38	4.07	11.9
	Ours	2.71	3.19	3.19	7.99
MSE(10^{-2})	WLOP	4.43	4.35	3.41	4.99
	ECN	3.49	3.66	2.89	4.39
	PCN	3.56	3.61	2.89	4.21
	GPD	3.58	3.59	2.53	4.12
	PF	3.40	3.42	3.47	4.01
	Ours	3.39	3.41	3.44	3.66
P2M(10^{-4})	WLOP	3.02	4.94	3.25	9.21
	ECN	1.53	1.95	1.71	3.24
	PCN	1.36	2.57	1.45	4.78
	GPD	2.28	2.32	2.41	3.02
	PF	0.81	1.31	0.83	4.77
	Ours	0.73	1.08	0.77	2.74

Table 3: Quantitative comparisons by testing various methods on synthetic noisy models with various noise models, including discrete noise (DN), Laplace noise (LN), uniform noise (UN) and 3% high-level Gaussian noise (HLN).

to balance L_{dp} and L_p^{final} . When our MODNet is trained without L_{dp} ($\beta = 0$), it achieves worse denoising result than $\beta = 0.2$. And the denoising performance will also decrease when our network pays too much attention to these three single scale denoising displacements ($\beta > 0.2$). According to the results of our experiments, we set $\beta = 0.2$ in our training stage. $|p_{i,k}^{\wedge}|, r_k, K$ are hyperparameters of input patches. According to the results of our experiments, we set $|p_{i,k}^{\wedge}| = 400, K = 3, r_k = \{3\%, 4\%, 5\%\}$ of the model's bounding box diagonal length in our MODNet. In particular, we noticed that too large $|p_{i,k}^{\wedge}|, r_k$ and K degrade the denoising performance instead. Therefore, selecting the optimal local neighborhood information is crucial for denoising, so it is a meaningful work to do in the future.

4.7. Ablation Studies

We further illustrate the effectiveness of different modules in MODNet, as shown in Table 5. Pointfilter [ZLQH20] is employed as the baseline, and we analyze the effects of using different module settings on network performance. Based on PF, we add the remaining modules back one at a time to obtain a new model, which is denoted as M1 to M4 respectively. M4 is our full pipeline with all the modules. First, Using the PFE module alone gains lower CD, MSE and P2M, compared with the Pointfilter. Compared with the single-scale patch, multi-scale patches information leads to a notable performance improvement of denoising. Based on M1, we add the MOD module to obtain model M2. The MOD of M2 uses as input the output of PFE, and the multi-scale weights are set to an all-one matrix. And the MOD of M2 doesn't predicts $dp_{i,k}^{pre}$. As for M3, we add the MSPM module in front of the MOD, and the

β	Metrics		
	CD(10^{-4})	MSE(10^{-2})	P2M(10^{-4})
0.0	2.82	2.49	0.77
0.2	2.74	2.48	0.74
0.4	2.86	2.49	0.78
0.6	2.81	2.49	0.76
0.8	2.81	2.49	0.76
K	Metrics		
	CD(10^{-4})	MSE(10^{-2})	P2M(10^{-4})
2	2.89	2.49	0.80
3	2.78	2.48	0.74
4	2.81	2.48	0.74
5	2.79	2.49	0.74
$ p_{i,k}^{\wedge} $	Metrics		
	CD(10^{-4})	MSE(10^{-2})	P2M(10^{-4})
200	2.82	2.49	0.77
400	2.74	2.48	0.74
600	2.87	2.49	0.78
800	2.90	2.50	0.79
1000	2.93	2.51	0.80
r_k	Metrics		
	CD(10^{-4})	MSE(10^{-2})	P2M(10^{-4})
2%, 3%, 4%	2.94	2.49	0.81
3%, 4%, 5%	2.78	2.48	0.74
4%, 5%, 6%	2.79	2.50	0.74
5%, 6%, 7%	2.78	2.50	0.74
6%, 7%, 8%	2.96	2.53	0.79
2%, 5%, 8%	2.93	2.50	0.80
3%, 5%, 7%	2.81	2.49	0.75

Table 4: The comparisons with different hyper-parameter $\beta, K, |p_{i,k}^{\wedge}|, r_k$ on test set.

Model	PFE	MOD	MSPM	L_{dp}	Metrics		
					CD	MSE	P2M
PF					3.16	2.52	0.89
M1	✓				2.92	2.50	0.80
M2	✓	✓			2.85	2.49	0.77
M3	✓	✓	✓		2.79	2.49	0.77
M4*	✓	✓	✓	✓	2.80	2.49	0.75
M4	✓	✓	✓	✓	2.78	2.48	0.74

Table 5: Effectiveness of different sub-module of our model. CD is multiplied by 10^4 , MSE is multiplied by 10^2 and P2M is multiplied by 10^4 .

multi-scale weights were acquired by MSPM. M4* doesn't predicts $dp_{i,k}^{pre}$ like M2, but directly supervises $offset_{i,k}$ with L_{dp} . M4 predicts $dp_{i,k}^{pre}$ and employs the loss L_{dp} to supervise $dp_{i,k}^{pre}$. From the results, directly supervising $offset_{i,k}$ seems to limit the learning ability of the network and instead leads to denoising performance degradation. Therefore, PFE, MSPM, MOD and L_{dp} all contribute positively to the performance.

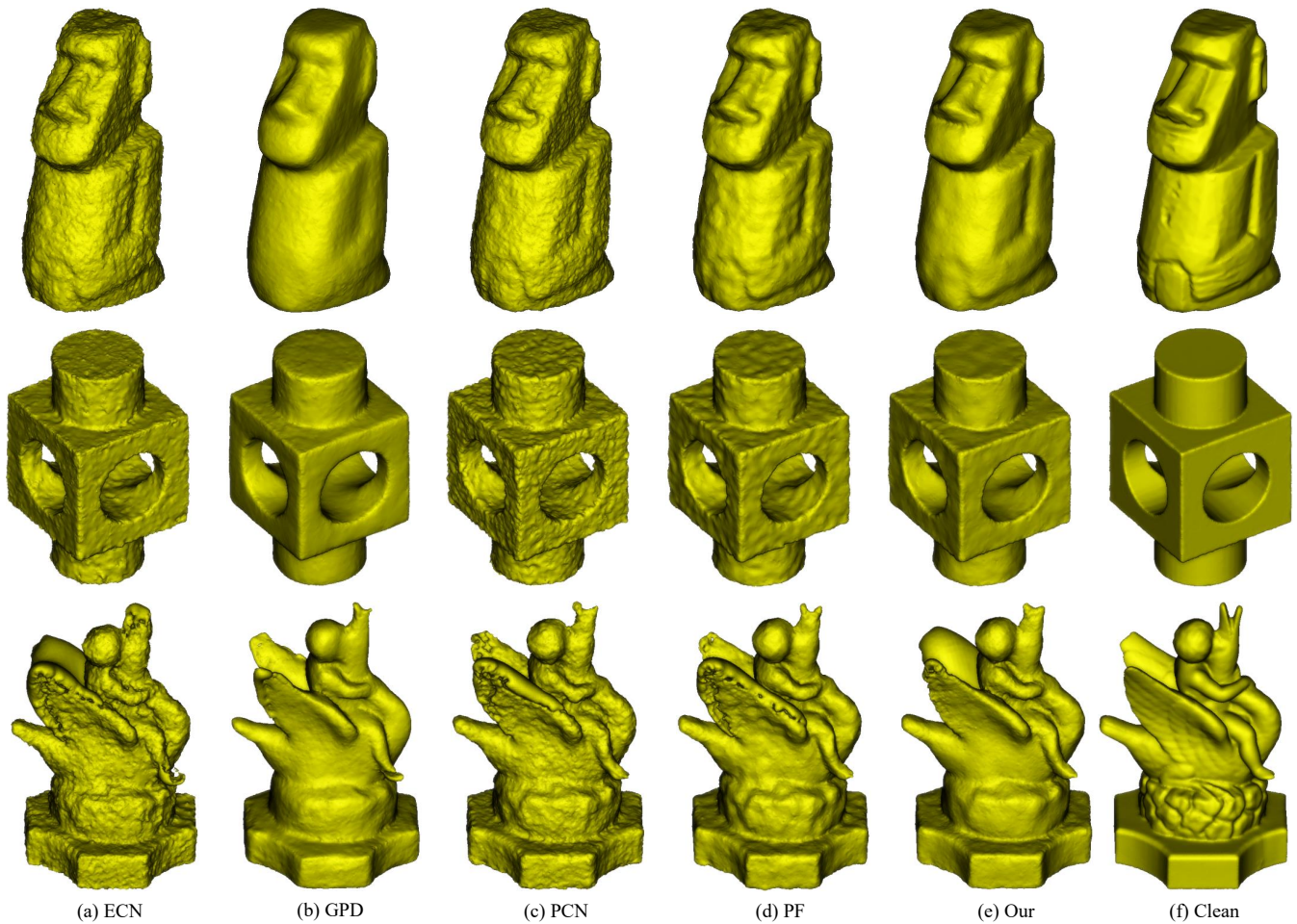


Figure 8: Comparisons of Poisson reconstructed meshes of denoised results. From the Comparisons, the reconstructed meshes of the point cloud filtered by our method are cleaner and preserve more geometric details.

5. Conclusion

In this paper, we propose a multi-offset denoising network customized for multi-scale patches input, which employs multi-scale geometric perception information to guide the network to utilize multi-scale information. It is a semi-soft neighborhood selecting mechanism. More specifically, the MODNet is employed to extract the features of multi-scale patches separately. And it regresses three different scale denoising offsets. And three different scale denoising offsets are guided by the multi-scale weights to predict the final denoising displacement by weighting them adaptively. A variety of experiments demonstrate that our method can yield state-of-the-art denoising performance.

Limitation. Our method is a new paradigm for using multi-scale patches information on denoising networks. It is a semi-soft neighborhood selecting mechanism. However, the number of scales and patch scale are still hyper-parameters that need to be determined. And the running time of our MODNet is not very fast.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2020YFB2010702, No. 2019YFB1707504, 2019YFB1707501) and Natural Science Foundation of Jiangsu Province (No.BK20190016).

References

- [ABCO*01] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Point set surfaces. In *Proceedings Visualization, 2001. VIS'01.* (2001), IEEE, pp. 21–29. 1, 2
- [ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *TVCG* 9, 1 (2003), 3–15. 1, 2
- [ASGC010] AVRON H., SHARF A., GREIF C., COHEN-OR D.: L1-sparse reconstruction of sharp point set surfaces. *TOG* 29, 5 (2010), 1–12. 2
- [CP05] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. *CAGD* 22, 2 (2005), 121–146. 2

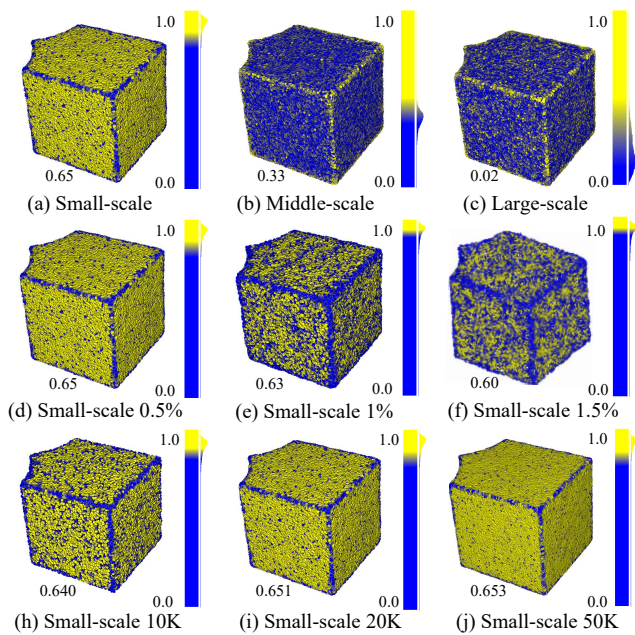


Figure 9: Visual comparisons of the multi-scale weights. The models in the top row are three different scale weights, the models in the bottom two rows are the small-scale weights of models with different noise levels and densities. These numbers of the lower left corners of the models are the mean value of models' scale weights.

[GG07] GUENNEBAUD G., GROSS M.: Algebraic point set surfaces. *Acm TOG* 26, 3 (2007), 23–1–23–23–923–23–9. 2

[GKOM18] GUERRERO P., KLEIMAN Y., OVSJANIKOV M., MITRA N. J.: Pcpnet learning local shape properties from raw point clouds. In *CGF* (2018), vol. 37, Wiley Online Library, pp. 75–85. 2

[HGCG20] HU W., GAO X., CHEUNG G., GUO Z.: Feature graph learning for 3d point cloud denoising. *IEEE Transactions on Signal Processing* 68 (2020), 2841–2856. 2

[HHWG21] HU W., HU Q., WANG Z., GAO X.: Dynamic point cloud denoising via manifold-to-manifold distance. *IEEE TIP* 30 (2021), 6168–6183. 1, 2

[HLZ*09] HUANG H., LI D., ZHANG H., ASCHER U., COHEN-OR D.: Consolidation of unorganized point clouds for surface reconstruction. *TOG* 28, 5 (2009), 1–7. 1, 2, 6, 7

[HPL*21] HU W., PANG J., LIU X., TIAN D., LIN C.-W., VETRO A.: Graph signal processing for geometric data and beyond: Theory and applications. *IEEE Transactions on Multimedia* (2021). 2

[HRR19] HERMOSILLA P., RITSCHER T., ROPINSKI T.: Total denoising: Unsupervised learning of 3d point cloud cleaning. In *ICCV* (2019), pp. 52–60. 1, 3, 7

[HSH*20] HE Y., SUN W., HUANG H., LIU J., FAN H., SUN J.: Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *CVPR* (2020), pp. 11632–11641. 1

[HWG*13] HUANG H., WU S., GONG M., COHEN-OR D., ASCHER U., ZHANG H.: Edge-aware point set resampling. *TOG* 32, 1 (2013), 1–12. 2

[LCOLTE07] LIPMAN Y., COHEN-OR D., LEVIN D., TAL-EZER H.: Parameterization-free projection for geometry reconstruction. *TOG* 26, 3 (2007), 22–es. 1, 2

[LWC*17] LU X., WU S., CHEN H., YEUNG S.-K., CHEN W., ZWICKER M.: Gpf: Gmm-inspired feature-preserving point set filtering. *TVCG* 24, 8 (2017), 2315–2326. 6

[LXW*22] LU D., XIE Q., WEI M., XU L., LI J.: Transformers in 3d point clouds: A survey. *arXiv preprint arXiv:2205.07417* (2022). 2

[MC17] MATTEI E., CASTRODAD A.: Point cloud denoising via moving rpca. In *CGF* (2017), vol. 36, Wiley Online Library, pp. 123–137. 1, 2

[NIL12] NGUYEN C. V., IZADI S., LOVELL D.: Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *2012 second international conference on 3D imaging, modeling, processing, visualization & transmission* (2012), IEEE, pp. 524–530. 5

[ÖGG09] ÖZTIRELI A. C., GUENNEBAUD G., GROSS M.: Feature preserving point set surfaces based on non-linear kernel regression. In *CGF* (2009), vol. 28, Wiley Online Library, pp. 493–501. 2

[PFVM20] PISTILLI F., FRACASTORO G., VALSESIA D., MAGLI E.: Learning graph-convolutional representations for point cloud denoising. In *ECCV* (2020), Springer, pp. 103–118. 1, 3, 6, 7

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR* (2017). 1, 2, 4

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413* (2017). 1, 2, 3

[RLBG*20] RAKOTOSAONA M.-J., LA BARBERA V., GUERRERO P., MITRA N. J., OVSJANIKOV M.: Pointcleannet: Learning to denoise and remove outliers from dense point clouds. In *CGF* (2020), vol. 39, Wiley Online Library, pp. 185–203. 1, 2, 3, 5, 6, 7

[RÖPG18] ROVERI R., ÖZTIRELI A. C., PANDELE I., GROSS M.: Pointprons: Consolidation of point clouds with convolutional neural networks. In *CGF* (2018), vol. 37, Wiley Online Library, pp. 87–99. 1, 3

[RRN*20] RAVI N., REIZENSTEIN J., NOVOTNY D., GORDON T., LO W.-Y., JOHNSON J., GKIOXARI G.: Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501* (2020). 6

[SMGD14] SERNA A., MARCOTEGUI B., GOULETTE F., DESCHAUD J.-E.: Paris-rue-madame database: a 3d mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In *ICPRAM* (2014). 6, 7

[SNF*13] SHUMAN D. I., NARANG S. K., FROSSARD P., ORTEGA A., VANDERHEYNST P.: The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE SIGNAL PROC MAG* 30, 3 (2013), 83–98. 2

[SPV15] SCHOENENBERGER Y., PARATTE J., VANDERHEYNST P.: Graph-based denoising for time-varying point clouds. In *2015 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)* (2015), IEEE, pp. 1–4. 2

[SSW15] SUN Y., SCHAEFER S., WANG W.: Denoising point sets via l0 minimization. *CAGD* 35 (2015), 2–15. 1, 2

[TQD*19] THOMAS H., QI C. R., DESCHAUD J.-E., MARCOTEGUI B., GOULETTE F., GUIBAS L. J.: Kpconv: Flexible and deformable convolution for point clouds. In *ICCV* (2019), pp. 6411–6420. 1

[XLH*20] XIE Q., LU D., HUANG A., YANG J., LI D., ZHANG Y., WANG J.: Rrnet: Rivet region classification network for rivet flush measurement based on 3-d point cloud. *IEEE TIM* 70 (2020), 1–12. 1

[YLF*18] YU L., LI X., FU C.-W., COHEN-OR D., HENG P.-A.: Ecnnet: an edge-aware point set consolidation network. In *ECCV* (2018), pp. 386–402. 1, 3, 6, 7

[ZCN*19] ZENG J., CHEUNG G., NG M., PANG J., YANG C.: 3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model. *IEEE TIP* 29 (2019), 3474–3489. 1, 2

[ZLQH20] ZHANG D., LU X., QIN H., HE Y.: Pointfilter: Point cloud filtering via encoder-decoder modeling. *TVCG* 27, 3 (2020), 2015–2027. 1, 2, 4, 5, 6, 7, 9