Adaptive Iterative Point Cloud Denoising with Signed Distance Function Estimation

Zihan Gu[1,2,3] Xiyan Gan[1,2,3] Linquan Yang[1,2,3*]
[1]School of Computer Science, China University of Geosciences, Wuhan 430074, China
[2]Engineering Research Center of Natural Resource Information Management and Digital Twin Engineering Software, Ministry of Education, Wuhan 430074, China
[3]Hubei Key Laboratory of Intelligent Geo-Information Processing, China University of Geosciences, Wuhan 430074, China
{guzihan, ganxiyan, yanglq}@cug.edu.cn

Abstract

Point clouds acquired by scanners frequently contain noise, which presents obstacles for downstream tasks such as surface reconstruction and analysis.In this paper, we propose a joint framework that integrates implicit signed distance function (SDF) estimation with noisy point displacement prediction and further optimizes the denoising results through an adaptive iteration mechanism.The method consists of two U-Nets with identical architectures but unshared parameters, along with a noise-level predictor.The first U-Net is used to estimate the SDF of the noisy point cloud, while the second predicts the noisy displacement vectors by combining the point cloud with the estimated SDF. We employ a noise-level predictor based on entropy theory to estimate the level of noise, deciding when to terminate.Our method effectively alleviates artifacts caused by over- or under-denoising and produces high-quality results with better preservation of geometric details. Extensive experiments on synthetic and real-world datasets demonstrate that the proposed method achieves state-of-the-art performance.

Keywords: ***Point cloud denoising, Signed distance function, U-Net architecture, Adaptive iteration.***

1. Introduction

Point clouds are used to represent the geometric structure of real-world objects and are widely applied in fields such as autonomous driving, robotics, and 3D reconstruction. However, due to limitations in sensor accuracy, optical imaging, and environmental interference, raw point cloud data often mix with noise, affecting data quality and subsequent applications. Therefore, point cloud denoising is crucial for improving 3D vision tasks. Traditional denoising methods, such as Moving Least Squares (MLS)[1], optimization-driven algorithms[2], Locally Optimal Projection (LOP)[3], and low-rank representation[4], can improve point cloud quality. However, these methods are often time-consuming, require expert knowledge, and lack robustness when dealing with complex noise, making them insufficient for practical applications.

In recent years, with the development of point cloud processing neural networks such as PointNet [5], DGCNN [6], and Point Transformer [7], learning-based methods have made rapid progress. For example, Pointfilter [8] designs a bilateral loss function that jointly considers point positions and normal information, thereby better preserving sharp edges; IterativePFN [9] simulates a progressively smoothing denoising process through an adaptive loss function and a multi-round iterative module; Score [10] trains a neural network to estimate the logarithmic density gradient of noisy point clouds and applies a gradient ascent strategy to iteratively move noisy points toward the underlying surface, thereby achieving point cloud denoising; PD-Flow [11] combines normalizing flows with noise disentanglement techniques, modeling noisy point clouds as a joint distribution of clean points and noise, and obtains denoised results by separating the noise component in the latent space, thus redefining the denoising process.

Although existing methods have achieved promising results in point cloud denoising, they all adopt a fixed number of denoising iterations, failing to strike a balance between denoising performance and inference efficiency. Furthermore, none of them considers the potentially critical role of implicit signed distance function (SDF) estimation (a scalar function that measures the shortest distance from a point to the nearest surface) in explicit point cloud displacement prediction.

To address the above issues, this paper proposes an adap-

* Corresponding author

tive iterative point cloud denoising network, which combines explicit noise displacement estimation with implicit SDF estimation, based on a dual U-Net architecture. We leverage the implicit SDF to assist explicit noise displacement estimation. Since SDF is essentially a continuous function, it can provide global distance information from points to surfaces, thereby enabling more accurate and globally consistent displacement directions in point cloud denoising. Specifically, we first employ a U-Net to estimate the implicit field of the point cloud. Then, the predicted SDF is concatenated with the noisy point cloud coordinates and fed into the second U-Net to estimate the displacement vectors from the noisy point cloud to the clean point cloud. After denoising, we evaluate the noise level of the point cloud. If the noise level is below a predefined threshold, the point cloud undergoes another round of denoising until the threshold is reached. In this way, our method is able to generate high-quality denoised point clouds with uniform distribution and well-preserved geometric features.

The main contributions of this paper can be summarized as follows:

- For the first time, we jointly perform implicit SDF estimation and explicit point cloud displacement estimation to accomplish the point cloud denoising task, achieving high-quality denoised point clouds and offering a new perspective for point cloud denoising.
- We propose an adaptive iterative dual U-Net architecture, which can perform adaptive iterations while predicting the SDF of noisy point clouds and displacement vectors.
- Both quantitative and qualitative evaluations demonstrate the effectiveness of our method on both synthetic and scanned datasets.

2. Related Work

In recent years, a large number of point cloud denoising methods have emerged. This paper provides a review only of deep-learning based point cloud denoising techniques closely relevant to our method.

Method for estimating displacement vectors. PointCleanNet [12] achieves denoising by first removing outliers and then predicting displacement vectors for the remaining noisy points. Pointfilter [8] employs a bilateral loss function that fuses both point and normal information to denoise the point cloud. IterativePFN [9] incrementally restores noisy points onto the underlying surface through iterative denoising steps.

Methods of Multi-Task Joint Learning. Some point-cloud denoising methods accomplish this by jointly estimating surface normals and noise displacement. de Silva Edirimuni et al.[13] bundle normal estimation and point cloud denoising through contrastive learning: they first add noise to 3D patches, train the network to map noisy and clean patches to nearby points in feature space, and then use a joint normal-and-displacement loss to simultaneously refine normals and correct point positions, thereby removing noise while preserving sharp features. PCDNF[14] employs a multi-task network with two branches—one predicting per-point displacement vectors to denoise, the other finely filtering initial normals—and merges point features, normal features, and geometric priors in a tangent-space module to enhance the network's ability to distinguish fine details from noise. PNInternet[15] by first estimating normals and then projecting points based on those normals, it achieves precise point cloud denoising and normal estimation.

Methods for Inferring Surface. DMRDenoise[16] proposes performing manifold reconstruction during downsampling to reduce the impact of noise points on point clouds. Score[10] method proposes to increase the log-likelihood of each point by iteratively updating each point's position using gradient ascent to perform point cloud denoising. PD-Flow[11] combines normalizing flows and noise disentanglement techniques, constructing the denoising process from the perspective of distribution learning and feature disentanglement. P2P-Bridge[17] applies diffusion Schrödinger bridges to point clouds, treating the point cloud problem as learning the optimal transport path between noisy and clean point clouds.

3. Method

3.1. Signed Distance Function

SDF (signed distance function) is a scalar function used to measure the shortest distance from a point to the surface of an object, with the sign of the distance indicating the spatial position of the point relative to the surface. A positive value means the point is outside the surface, a negative value means the point is inside the surface, and a zero value means the point is on the surface. In point cloud denoising tasks, we consider that the SDF, by providing implicit surface information, can help predict the displacement from noisy points to clean points, thus achieving better denoising results.

Given a noisy point cloud $\tilde{P} = \{p_i\}_{i=1}^{N}$, the corresponding clean point cloud $\bar{P} = \{y_j\}_{j=1}^{N}$ can be regarded as a set of points sampled from the clean mesh surface S. The definition of the signed distance function (SDF) is as follows:

$$\mathrm{SDF}(p_i, S) = \begin{cases} d(p_i, \mathrm{S}), & \text{if } \tilde{p} \text{ is outside the surface S,} \\ -d(p_i, \mathrm{S}), & \text{if } \tilde{p} \text{ is inside the surface S,} \\ 0, & \text{if } \tilde{p} \text{ is on the surface S.} \end{cases} \tag{1}$$

$$d(p_i, \mathrm{S}) = \min_{y_i \in \mathrm{S}} || \, p_i - y_i \, ||_2 \, , \tag{2}$$

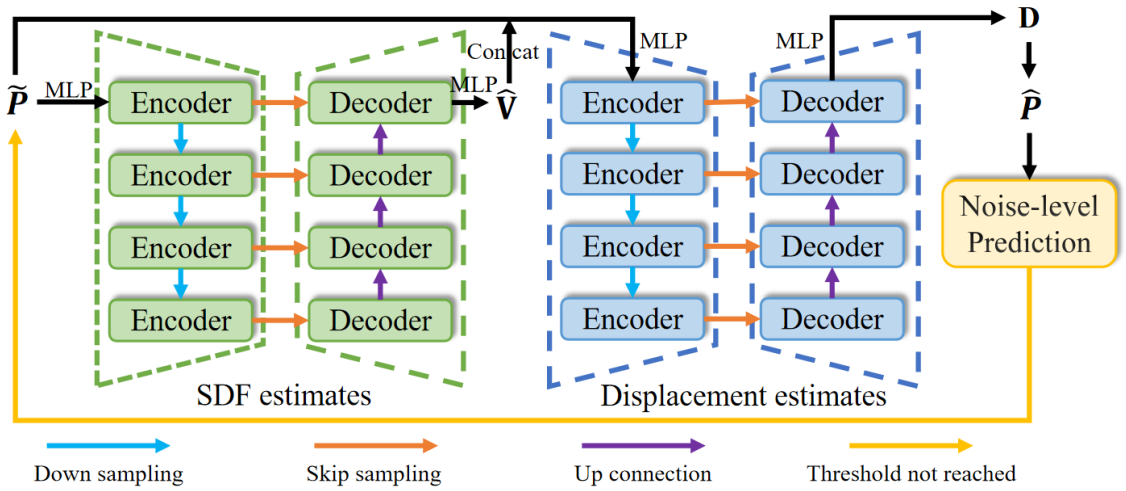


Figure 1. Illustration of our Network. Our network mainly consists of two U-Net structures and a noise-level predictor. The two U-Net structures are responsible for predicting the Signed Distance Function (SDF) and displacement vectors of the noisy point cloud, while the noise-level predictor estimates the noise level of the denoised point cloud to determine whether further denoising is required.

where, $d(p_i, \mathrm{S})$ denotes the shortest distance from the point p_i to the surface S, and the sign of the SDF indicates the spatial position of p_i relative to that surface.

By first predicting the SDF values of the noisy point cloud, then estimating its displacement relative to the clean point cloud, and using these SDF values as auxiliary information to assist the network in predicting the displacement, the accuracy of the predicted displacement can be effectively improved, allowing the denoised point cloud to retain more geometric details.

3.2. Adaptive Iterative

Existing point cloud denoising methods typically employ a fixed number of denoising iterations, which often leads to insufficient denoising or over-denoising, thereby preventing the denoising performance from reaching the desired level. To address this issue, we propose using the entropy value to measure the noise level of the denoised point cloud, enabling us to determine whether the point cloud requires further denoising after each iteration.

For a given point set $P = \{x_i\}_{i=1}^{N}$, the K-Nearest Neighbors (KNN) algorithm is first applied to divide it into N local neighborhoods. Then, the local entropy value of each point is calculated using the following formula:

$$w_k = exp(-\frac{\|x_k - x_i\|^2}{2\sigma^2}), \tag{3}$$

$$u_k = \frac{w_k}{\sum_{l \in N(i)} w_l}, \tag{4}$$

$$H_{local}(x_i) = -\sum_{k \in N(i)} u_k log(u_k + \varepsilon), \tag{5}$$

where, x_i denotes the center point of the neighborhood, x_k represents the neighboring points, $N(i)$ is the neighborhood of x_i, σ is the distance from x_i to its k-th nearest neighbor, and $\varepsilon = 10^{-12}$ is a fixed value introduced for numerical stability. The global entropy can be obtained using the following formula:

$$H_{global} = \frac{1}{N} \sum_{i=1}^{N} H_{local}(x_i). \tag{6}$$

We take the ratio ϕ of the denoised point cloud entropy to the clean point cloud entropy as the measure of noise level.

$$\phi = \frac{H_{global}(\hat{P})}{H_{global}(\bar{P})}. \tag{7}$$

A lower ϕ indicates a higher noise-level in the point cloud, whereas a higher ϕ suggests a lower noise level. For a given threshold β (where $\beta = 0.89$), if $\phi < \beta$ the denoising process continues; if $\phi \geq \beta$, the noise is considered to have been reduced to an acceptable level, and the iteration stops.

3.3. Network Architecture

As shown in Fig. 1, the network consists of two U-Net architectures with identical structures but unshared parameters, as well as a noise-level predictor. Each U-Net has a total depth of $L = 4$. The first U-Net predicts the SDF values of the noisy point cloud. Its input is the high-dimensional features obtained by processing $\tilde{P}$ through an MLP, and its output is the predicted SDF value $\hat{V}$ for each point. The second U-Net network is used to predict the displacement of the noisy point cloud $\tilde{P}$ relative to the ground-truth point cloud $\bar{P}$, with input being the high-dimensional features obtained by processing the concatenation of $\tilde{P}$ and $\hat{V}$ through an MLP, and output being the predicted displacement D;

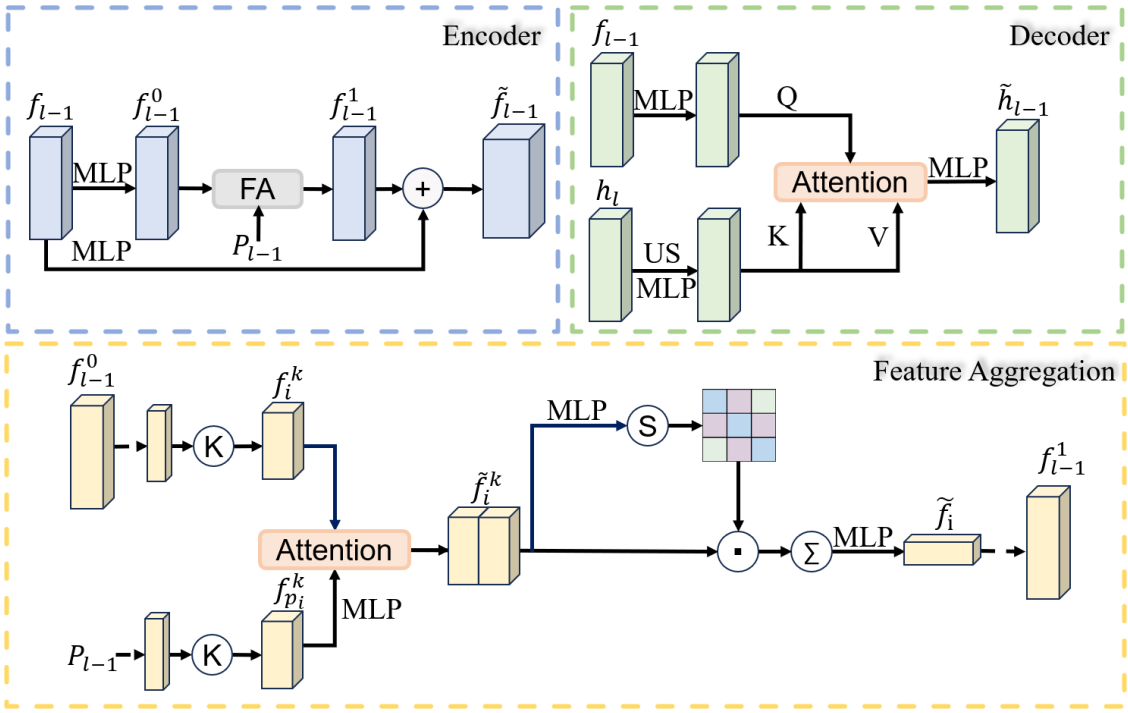


Figure 2. Illustration of our encoder and decoder.

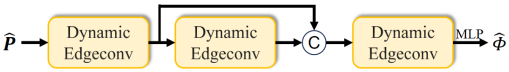


Figure 3. Illustration of our noise-level predictor.

subtracting the predicted displacement from the noisy point cloud coordinates yields the predicted clean point cloud $\hat{P}$. Subsequently, $\hat{P}$ is fed into the noise-level predictor to estimate its noise level $\hat{\phi}$, which is then used to determine whether further iterations of denoising are necessary.

As shown in Fig.2, our encoder is a residual module that uses Feature Aggregation (FA) blocks to strengthen features. In the U-Net architecture, we denote the encoder at layer l by E_l , and we write the features and point set coming from the previous layer as f_{l-1} and P_{l-1}. The operation of our encoder can be expressed as:

$$f^0_{l-1} = \text{MLP}(f_{l-1}), \quad f^1_{l-1} = \text{FA}(f^0_{l-1}, P_{l-1}), \tag{8}$$

$$\tilde{f}_{l-1} = f^1_{l-1} + \text{MLP}(f_{l-1}). \tag{9}$$

The Feature Aggregation module is shown in the Fig.2. For each point p_i, we first extract its feature vector f_i, and use the KNN algorithm to gather its neighboring features f^k_i and neighboring coordinates $f^k_{p_i}$. We then perform cross-attention interaction on these to obtain $\tilde{f}^k_i$. We then feed $\tilde{f}^k_i$ into a MLP, apply Softmax along the neighbor dimension to obtain attention weights, perform an element-wise (Hadamard) product with $\tilde{f}^k_i$, and finally compress the result into a single feature via sum-pooling (SumPooling). Formally,

$$\tilde{f}_i = \text{SumPooling}(\text{Softmax}(\text{MLP}(\tilde{f}^k_i)) \odot \tilde{f}^k_i). \tag{10}$$

After this operation for each point in the previous layer, the locally aggregated feature is obtained as f^1_{l-1}.

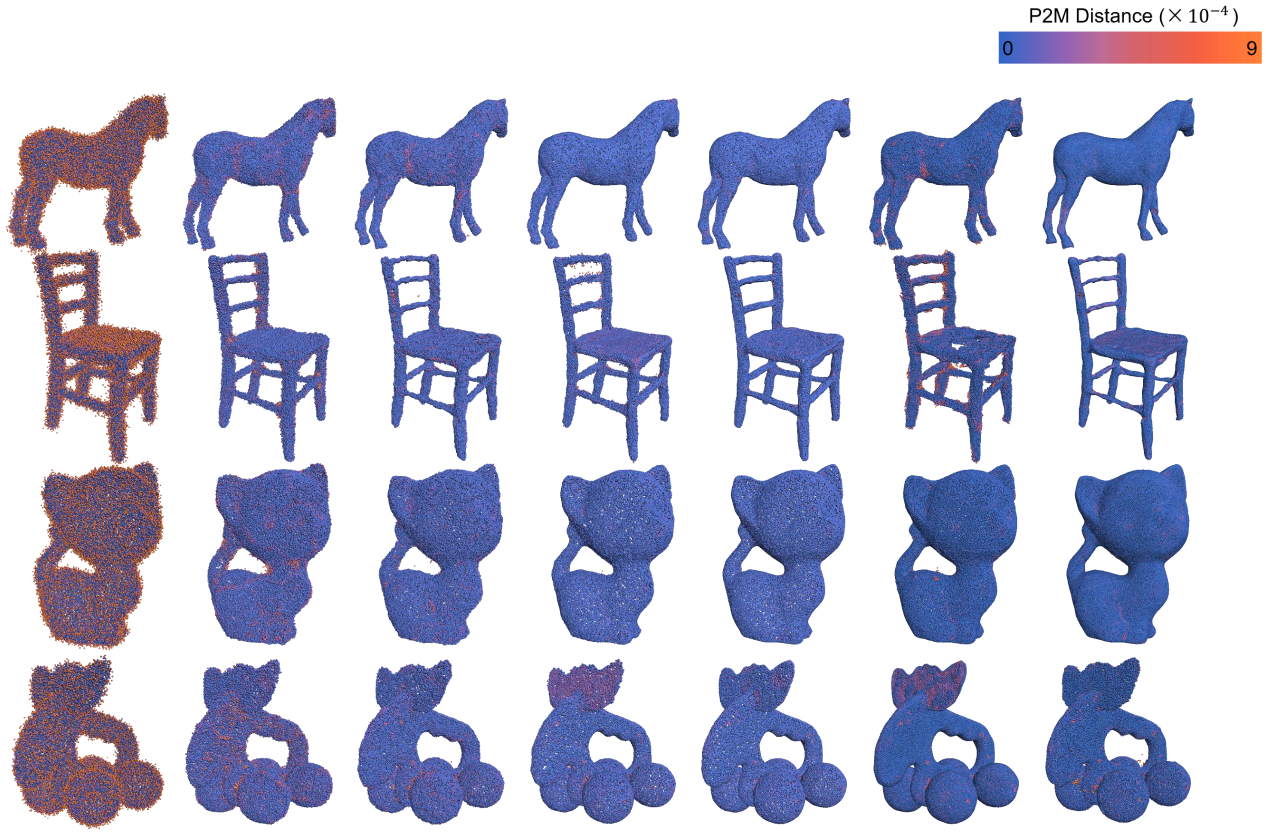
Our decoder primarily employs a cross-attention mechanism, as shown in Fig.2. We denote the decoder at layer $l-1$ by D_{l-1} and write its inputs—the encoder feature from the same layer and the decoder feature from the next deeper layer—as f_{l-1} and h_l, respectively. We apply multi-head cross-attention to capture the correlation between f_{l-1} and h_l. Finally, we use a skip connection to concatenate intermediate feature $\tilde{h}_{l-1}$ with the encoder feature f_{l-1} and pass the result through an MLP to obtain the decoder feature for layer:

$$h_{l-1} = \text{MLP}\left(\tilde{h}_{l-1} \parallel f_{l-1}\right). \tag{11}$$

As shown in Fig.3, our noise-level predictor is constructed using EdgeConv layers. The predicted point cloud $\hat{P}$ is taken as input. First, a k-nearest neighbor graph $G = (V, E_0)$ is constructed, where V and E_0 represent the set of points and the set of edges, respectively. Then, the initial features are computed according to the following formula:

$$g^0_i = \sum_{j:(i,j)\in E_0} MLP(x_i \,||\, x_i - x_j). \tag{12}$$

Subsequently, the features are enhanced using the following



(a) Noisy (b) DMR (c) Score (d) PF (e) Iterative (f) P2P (g) Ours

Figure 4. Comparison of tested approaches based on P2M distance for 50K-resolution shapes with 2% Gaussian noise on PUNet.

formula:

$$g_i^1 = \sum_{j:(i,j)\in E_0} MLP(g_i^0 \,||\, g_i^0 - g_j^0), \tag{13}$$

$$g_i = g_i^0 || g_i^1. \tag{14}$$

A dynamic graph convolution operation is then applied to the concatenated features g_i to obtain the feature representation g. Subsequently, a multi-layer perceptron (MLP) is used to predict the noise-level $\hat{\phi}$. To ensure an accurate determination of the number of iterations during training, we first pretrain our noise-level predictor. Once the noise-level predictor has been successfully trained, we then proceed to train our network's SDF and displacement prediction modules.

3.4. Loss Function

SDF prediction. In order to optimize the accuracy of our network's SDF prediction, we adopt the following loss function:

$$\mathcal{L}_{\text{SDF}} = |\hat{V} - \bar{V}|^2, \tag{15}$$

where, $\bar{V}$ denotes the ground truth of SDF values calculated from formula 1 and formula 2.

Noise level prediction. In order to optimize the accuracy of our network's noise-level prediction, we adopt the following loss function:

$$\mathcal{L}_{\text{noise}} = |\hat{\phi} - \bar{\phi}|^2, \tag{16}$$

where, $\bar{\phi}$ denotes the ground truth noise-level calculated from formula 7.

Displacement prediction. To optimize the accuracy of our network displacement prediction, we adopted the InfoCD loss function [18], which is defined as follows:

$$\mathcal{L}_{\text{InfoCD}} = \text{CD}(\hat{P}, \bar{P}) + \frac{1}{\tau}\text{Reg}(\hat{P}, \bar{P}), \tag{17}$$

where, CD denotes the Chamfer distance between the denoised point cloud and the ground-truth point cloud; Reg is the regularization term that promotes alignment between the noisy point cloud and the ground-truth distribution; and $\frac{1}{\tau}$ is a scaling factor used to adjust the weight of the regularization term.

Table 1. Quantitative evaluation on PUNet with CD and P2M metrics ($\times 10^{-4}$). The best and second-best results are highlighted in bold and underlined, respectively.

Method	10K points						50K points					
	1% noise		2% noise		2.5% noise		1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M
PCN[12]	3.686	1.599	7.926	4.759	10.486	6.987	1.103	0.646	1.978	1.370	3.203	2.486
GPDNet[19]	2.310	0.714	4.284	1.855	5.837	3.066	1.049	0.635	3.288	2.503	5.085	4.134
DMR[16]	4.712	2.196	5.085	2.523	5.277	2.669	1.205	0.762	1.443	0.970	1.696	1.190
Pointfilter[8]	2.461	0.730	3.534	1.155	4.099	1.505	0.758	0.432	0.907	0.507	1.099	0.629
Score[10]	2.522	0.754	3.683	1.380	4.232	1.904	0.716	0.400	1.288	0.833	1.445	0.958
PDFlow[11]	2.126	0.674	3.246	1.324	3.627	1.702	0.651	0.416	1.270	0.921	1.874	1.426
IterativePFN[9]	2.055	0.501	3.043	0.843	3.353	1.046	0.605	0.302	0.803	**0.436**	1.015	**0.588**
P2P-Bridge[17]	2.284	0.686	3.202	1.114	3.531	1.386	0.586	0.330	0.902	0.580	1.165	0.803
Ours	**1.803**	**0.487**	**2.873**	**0.791**	**2.956**	**1.019**	**0.564**	**0.298**	**0.757**	0.439	**1.001**	0.608

4. Experiments

4.1. Datasets and Settings

Training and Testing Dataset. We train our framework using the PUNet dataset [20], which consists of 40 meshes. Point clouds are generated from these meshes at resolutions of 10K, 30K, and 50K, yielding a total of 120 training point clouds. Noisy data is introduced by applying Gaussian noise with a standard deviation ranging from 0.05 to 0.2 times the radius of the bounding sphere. We conduct comparison experiments on the PUNet dataset [20] at 10K and 50K resolutions, resulting in 40 point clouds.To evaluate the generalization capability of our method, we also used the real-scanned Kinect dataset [21], as well as the Paris-rue-Madame dataset, which contains real Paris street scenes captured with a 3D mobile laser scanner [22].

Implementation. Our method is implemented using PyTorch and trained on an NVIDIA GeForce RTX 4070 Ti Super GPU. We use the Adam optimizer with a learning rate of 1×10^{-4}. The model is trained for a maximum of 200 epochs, with a parameter count of 833K. During training, the batch size is set to 16. During both training and testing, we use the Farthest Point Sampling (FPS) and K-Nearest Neighbors (KNN) algorithms to divide the full point cloud into patches with a resolution of 1K. For patch stitching, we adopt the same method as [9].

4.2. Quantitative Results

For quantitative evaluation, we adopt Chamfer Distance (CD) and Point-to-Mesh (P2M) distance as error metrics to assess denoising accuracy and completeness, where lower values indicate better denoising performance. We compared our method with other competing approaches on the synthetic dataset [20], and the results are shown in Table 1. From the table, it can be seen that our method achieves excellent denoising performance on both sparse 10K point clouds and dense 50K point clouds. The experimental results demonstrate that our method outperforms all competing approaches in most cases, especially under high noise levels in dense point clouds, where the advantage of our method becomes more significant. This clearly validates the effectiveness of our approach in high-noise scenarios.

Table 2. Quantitative evaluation on Kinect data with CD and P2M metrics ($\times 10^{-4}$).

Method	Kinect	
	CD	P2M
PCN	13.73	8.75
GPDNet	14.83	8.69
Pointfilter	13.77	7.91
Score	13.22	8.18
PDFlow	13.34	8.69
IterativePFN	13.20	8.43
P2P-Bridge	13.12	8.64
Ours	**13.03**	**7.80**

To demonstrate the adaptability of our method, we further showcase its effectiveness on a real-scanned dataset [21], with the results compared to other methods recorded in Table 2. As shown, our method achieves significantly lower values in both CD and P2M metrics, indicating that our method outperforms all the competing approaches.

To validate the robustness of the proposed method, we conduct denoising experiments on the PUNet dataset under anisotropic Gaussian noise, Laplace noise, uniform noise, and discrete noise, and compare our approach with other methods. The experimental results are summarized in the Table 3. As shown by the results, our method achieves con-

Table 3. Numerical comparison among competing methods under various noise types.

Noise Type	Method	10K points				50K points			
		1% noise		2% noise		1% noise		2% noise	
		CD	P2M	CD	P2M	CD	P2M	CD	P2M
Anisotropic	IterativePFN	2.603	0.611	4.932	2.315	0.611	**0.309**	0.825	**0.456**
	P2P-Bridge	2.871	0.915	**4.206**	**1.939**	0.608	0.348	0.947	0.621
	Ours	**2.515**	**0.607**	4.887	2.924	**0.570**	0.312	**0.806**	0.476
Laplace	IterativePFN	2.429	0.618	3.486	**1.182**	0.662	**0.343**	**1.051**	**0.618**
	P2P-Bridge	2.613	0.843	3.883	1.721	0.709	0.422	1.286	0.884
	Ours	**2.399**	**0.600**	**3.444**	1.801	**0.646**	0.358	1.113	0.726
Uniform	IterativePFN	1.355	0.399	2.644	0.584	0.444	0.254	0.598	0.297
	P2P-Bridge	1.768	0.588	2.829	0.827	0.465	0.277	0.571	0.315
	Ours	**1.341**	**0.380**	**2.630**	**0.554**	**0.401**	**0.253**	**0.550**	**0.288**

sistently strong performance across different noise conditions, which clearly demonstrates its robustness.

4.3. Qualitative Comparisons

In Fig. 4, we present the denoised results on the PUNet dataset [20] with 50K point clouds at a noise level of 2%. Through a visual comparison with all other methods, we can observe that our method's denoising results are more aesthetically pleasing, with a more uniform point distribution and more accurate recovery of geometric details.

In Fig. 5, we present the visual experimental results of our method compared to other approaches on the Paris street scene dataset [22]. As shown, our method effectively restores the details of both the car and the walls.

4.4. Ablation Studies

To validate the effectiveness of each module design in our method, we conducted ablation experiments on variants of our method using the PUNet dataset with a point cloud resolution of 50K, as shown in Table 4.

Effectiveness of adaptive iteration. To validate the impact of adaptive iteration on the denoising performance of our method, we removed the adaptive iteration from our approach and fixed the number of iterations to two. As shown in Table 4, the inclusion of adaptive iteration significantly improves the denoising performance of our method.

Network depth selection. Networks with insufficient depth cannot fully learn the geometric features of the point clouds, while excessively deep networks result in unnecessary computational cost. To assess the impact of different network depths on the denoising performance of our method, we compared the denoising results for various network depths. As shown in Table 4, the network with four layers yields the best results.

Effectiveness of the SDF auxiliary. To verify the effectiveness of the SDF in assisting point cloud denoising, we removed the SDF estimation network and only estimated the displacement of the noisy point clouds for denoising. From Table 4, we can observe that the inclusion of the SDF auxiliary significantly improves the point cloud denoising performance.

Table 4. Numerical results of ablation studies on PUNet.

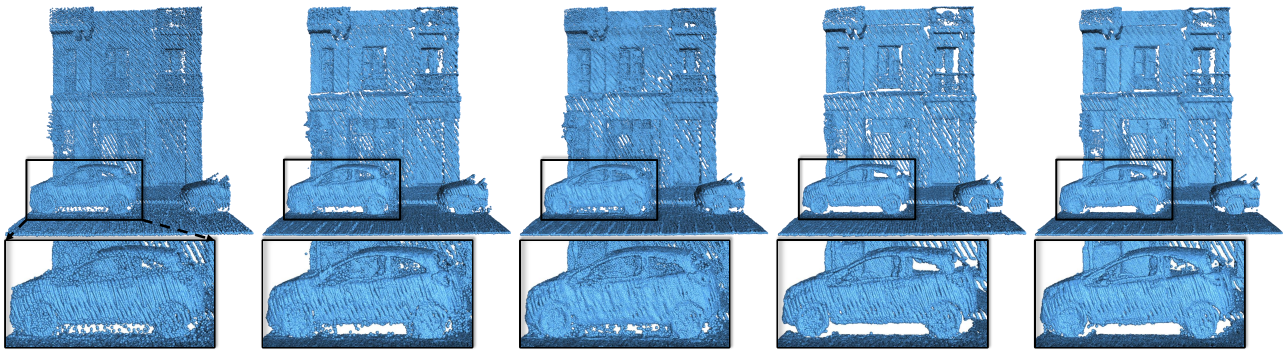
Ablation	**50K points**					
	1% noise		2% noise		2.5% noise	
	CD	P2M	CD	P2M	CD	P2M
w/o adative iterative	0.588	0.323	0.786	0.467	1.024	0.619
3 layers	0.583	0.314	0.772	0.451	1.021	0.617
5 layers	**0.551**	0.300	0.760	0.442	1.018	**0.601**
w/o SDF estimate	0.665	0.341	0.842	0.489	1.125	0.637
Default	0.564	**0.298**	**0.757**	**0.439**	**1.001**	0.608

4.5. Model Complexity and Inference Time Comparison

In Table 5, we compare the proposed method with other approaches in terms of model complexity and inference efficiency. As shown, Score and DMR have the smallest number of parameters and the fastest inference speed, respectively, but their denoising performance is relatively limited. Since our method adaptively determines the number of denoising iterations according to the noise level of the input point cloud, it requires a longer inference time during testing. Nevertheless, this additional computational cost results in a significant improvement in denoising quality, enabling our method to clearly outperform other competing approaches in terms of denoising performance, and thus achieve a more favorable balance between inference efficiency and denoising effectiveness.

5. Conclusion

In this paper, we propose an adaptive iterative point cloud denoising method based on a dual U-Net architecture, which jointly estimates the SDF and noisy displace-



(a) Noisy (b) Score (c) IterativePFN (d) P2P-Bridge (e) Ours

Figure 5. Comparison of tested approaches on real-world scenarios.

Table 5. Comparison of parameter count and inference time between our method and competing approaches.

Method	Param. (M)	Time (s)
PCN	27.91	13.17
DMR	0.23	**1.23**
Pointfilter	1.39	8.66
Score	**0.18**	1.29
PDFlow	0.47	4.68
IterativePFN	3.22	4.81
P2P-Bridge	26.44	2.63
Ours	0.83	4.58

ments. Our method fully considers the inherent relationship between noise displacement estimation and SDF estimation in point cloud denoising tasks. Unlike previous methods, our approach effectively addresses the issues of over-denoising and under-denoising during the denoising process. Extensive experiments on both synthetic and real-world datasets demonstrate that our proposed method achieves excellent denoising performance. Our method not only provides a new approach to point cloud denoising but also holds great potential for various tasks, including point cloud completion, normal vector estimation, and mesh reconstruction.

References

[1] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Comput. Graph. Forum*, 28(2):493–501, 2009. 1

[2] Zheng Liu, Xiaowen Xiao, Saishang Zhong, Weina Wang, Yanlei Li, Ling Zhang, and Zhong Xie. A feature-preserving framework for point cloud denoising. *Comput-Aided Des.*, 127:102857, 2020. 1

[3] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph.*, 28(5):176:1–7, 2009. 1

[4] Honghua Chen, Mingqiang Wei, Yangxing Sun, Xingyu Xie, and Jun Wang. Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint. *IEEE Trans. Vis. Comput. Graph.*, 26(11):3255–3270, 2019. 1

[5] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 652–660, 2017. 1

[6] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic Graph CNN for learning on point clouds. *ACM Trans. Graph.*, 38(5):1–12, 2019. 1

[7] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 16239–16248, 2021. 1

[8] Dongbo Zhang, Xuequan Lu, Hong Qin, and Ying He. Pointfilter: Point cloud filtering via encoder-decoder modeling. *IEEE Trans. Vis. Comput. Graph.*, 27(3):2015–2027, 2021. 1, 2, 6

[9] Dasith de Silva Edirimuni, Xuequan Lu, Zhiwen Shao, Gang Li, Antonio Robles-Kelly, and Ying He. IterativePFN: True iterative point cloud filtering. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 13530–13539, 2023. 1, 2, 6

[10] Shitong Luo and Wei Hu. Score-based point cloud denoising. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 4583–4592, 2021. 1, 2, 6

[11] Aihua Mao, Zihui Du, Yu-Hui Wen, Jun Xuan, and Yong-Jin Liu. PD-Flow: A point cloud denoising framework with normalizing flows. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, page 13663, 2022. 1, 2, 6

[12] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. PointCleanNet: Learning to denoise and remove outliers from dense point clouds. *Comput. Graph. Forum*, 39(1):185–203, 2020. 2, 6

[13] Dasith de Silva Edirimuni, Xuequan Lu, Gang Li, and Antonio Robles-Kelly. Contrastive learning for joint normal estimation and point cloud filtering. *IEEE Trans. Vis. Comput. Graph.*, 30(8):4527–4541, 2024. 2

[14] Zheng Liu, Yaowu Zhao, Sijing Zhan, Yuanyuan Liu, Renjie Chen, and Ying He. PCDNF: Revisiting learning-based point cloud denoising via joint normal filtering. *IEEE Trans. Vis. Comput. Graph.*, 30(8):5419–5436, 2023. 2

[15] Cheng Yi, Zeyong Wei, Jingbo Qiu, Honghua Chen, Jun Wang, and Mingqiang Wei. PN-Internet: Point-and-normal interactive network for noisy point clouds. *IEEE Trans. Geosci. Remote Sens.*, 62:1–11, 2024. 2

[16] Shitong Luo and Wei Hu. Differentiable manifold reconstruction for point cloud denoising. In *Proc. ACM Int. Conf. Multimedia (ACM MM)*, pages 1330–1338, 2020. 2, 6

[17] Mathias Vogel, Keisuke Tateno, Marc Pollefeys, Federico Tombari, Marie-Julie Rakotosaona, and Francis Engelmann. P2P-Bridge: Diffusion bridges for 3d point cloud denoising. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 184–201, 2024. 2, 6

[18] Fangzhou Lin, Yun Yue, Ziming Zhang, Songlin Hou, Kazunori Yamada, Vijaya B Kolachalama, and Venkatesh Saligrama. InfoCD: A contrastive chamfer distance loss for point cloud completion. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 5

[19] Francesca Pistilli, Giulia Fracastoro, Diego Valsesia, and Enrico Magli. Learning graph-convolutional representations for point cloud denoising. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 103–118, 2020. 6

[20] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proc. IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 2790–2799, 2018. 6, 7

[21] Peng-Shuai Wang, Yang Liu, and Xin Tong. Mesh denoising via cascaded normal regression. *ACM Trans. Graph.*, 35(6):232–1, 2016. 6

[22] Andrés Serna, Beatriz Marcotegui, François Goulette, and Jean-Emmanuel Deschaud. Paris-rue-madame database: a 3d mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In *International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, pages 819–24, 2014. 6, 7