Noise4Denoise: Leveraging Noise for Unsupervised Point Cloud Denoising

Weijia Wang[1], **Xiao Liu**[1], **Hailing Zhou**[2], **Lei Wei**[1], **Zhigang Deng**[3], **Manzur Murshed**[1], **and Xuequan Lu**[4](✉)

© The Author(s)

Abstract Existing deep learning-based point cloud denoising methods are generally trained in a supervised manner that requires clean data as ground-truth labels. However, in practice, it is not always feasible to obtain clean point clouds. In this paper, we introduce a novel unsupervised point cloud denoising method that eliminates the need of using clean point clouds as ground-truth labels during training. We demonstrate that it is feasible for neural networks to only take noisy point clouds as input, and learn to approximate and restore their clean versions. In particular, we generate two noise levels for the original point clouds, requiring the second noise level to be twice the amount of the first noise level. With this, we can deduce the relationship between the displacement information that recovers the clean surfaces across the two levels of noise, and thus learn the displacement of each noisy point in order to recover the corresponding clean point. Comprehensive experiments demonstrate that our method achieves outstanding denoising results across various datasets with synthetic and real-world noise, obtaining better performance than previous unsupervised methods and competitive performance to current supervised methods.

Keywords Point cloud denoising, Geometry processing, Point cloud processing, Deep learning

1 Introduction

Point clouds are used in a wide range of scenarios, such as autonomous driving, immersive reality, robotics, and remote sensing. Despite their notable advantages such as intuitive representations and lightweight characteristics, raw point clouds from sensors are usually corrupted with noise which can degrade their positional accuracy. Such noise comes in the form of displacement vectors that perturb the points and prevent them from accurately representing the objects' surfaces. Noise contamination is often caused by the sensors' quality and precision limitations or by factors from the environment, including material reflectivity and lighting conditions.

As such, point cloud denoising is an essential and fundamental research problem. However, since point clouds are unordered and lack connectivity information [1, 2], removing noise from them has been a long-standing challenge over the decades. Traditional point cloud denoising methods, based on optimisation techniques [3–5], often need non-trivial parameter tuning and thus can be burdensome for users. Recently, deep learning-based denoising methods [6–8] have alleviated the burden of parameter tuning and enhanced the robustness of the denoising performance. Most existing learning-based methods require pairs of noisy data and the corresponding ground-truth (i.e., clean) data to guide training. However, the ground-truth point clouds are not always available, as it is infeasible to obtain noise-free point clouds in some scenarios due to scanners' precision limitations or environmental constraints. In recent years, unsupervised denoising methods for images, including [9–11], have demonstrated desirable results for noise removal without training the networks using clean data. Motivated by this, we aim to design a new neural network to denoise 3D point clouds that does not require ground-truth data as training labels.

In this paper, we propose a novel unsupervised point cloud denoising framework. The primary insight of our method is that neural networks can learn to remove noise without requiring clean points as labels. During training, we generate two versions of point clouds: one with standard noise and another with double noise. Then, we deduce the relationship between the displacement information in these two noisy

1 Deakin University, Geelong, VIC 3216, Australia. E-mail: W. Wang, wangweijia@deakin.edu.au; X. Liu, xiao.liu@deakin.edu.au; L. Wei, lei.wei@deakin.edu.au; M. Murshed, m.murshed@deakin.edu.au.

2 Swinburne University of Technology, Melbourne, VIC 3122, Australia. E-mail: hailing.zhou@hotmail.com.

3 University of Houston, Houston, TX 77204-3027, USA. E-mail: zdeng4@central.uh.edu.

4 La Trobe University, Melbourne, VIC 3086, Australia. E-mail: b.lu@latrobe.edu.au (✉).

versions, enabling us to design an effective loss term for minimisation. Our network leverages an encoder-decoder architecture which takes in local neighbourhoods from the doubly-noisy point clouds, learns their features, and subsequently outputs displacement vectors to restore clean point positions. Notably, our training is accomplished using only the noisy point clouds, without reference to clean data. Extensive experiments on different datasets demonstrate that our method can be generalised to both synthetic and real-world noise, showing comparable performance with supervised methods.

In summary, the contributions of our approach are outlined as follows:

- We propose an unsupervised point cloud denoising network that eliminates the need of ground-truth point clouds as labels during training.
- Our network is designed to learn noise residuals using merely noisy data, without additional inputs beyond point clouds during training and testing.
- Our approach can achieve outstanding denoising results on synthetic and real-world noise compared to existing unsupervised and supervised methods.

2 Related work

2.1 Point cloud denoising

Optimisation-based methods. Traditional point cloud denoising methods are typically optimisation-based, removing noise by optimising a set of geometrical constraints. A main category of existing methods is based on *moving least squares (MLS)* which targets to approximate the clean underlying surfaces of the input shapes. Based on the seminal work [12], Alexa et al. [3, 13] proposed an MLS-based method to approximate the clean surfaces of the input noisy point clouds. Several other relevant variants were proposed later, including Point Set Surfaces [14], Robust MLS (RMLS) [15], and Robust Implicit MLS (RIMLS) [16]. Another category of existing methods is based on *locally optimal projection (LOP)* which projects noisy points onto the optimal surfaces. The early work, proposed by Lipman et al. [4], eliminates the burden of parameterisation. Subsequently, further works were proposed, including Weighted LOP (WLOP) [17], Edge Aware Resampling (EAR) [18], and Continuous LOP (CLOP) [19]. There are also other denoising methods including jet fitting [20], bilateral filtering [21], feature-preserving methods [22, 23], and non-local-based methods [5, 24].

Supervised learning-based methods. As deep learning techniques have been increasingly used to handle point clouds [1, 25], data-driven point cloud denoising methods, most of which are supervised, have demonstrated their robustness and generalisability. One of the early works is PointCleanNet (PCNet) [6] which requires pairs of noisy and clean point clouds for training. Luo and Hu [7] proposed Differentiable Manifold Reconstruction (DMR), an autoencoder-like network that reconstructs manifolds and re-samples denoised points. Nevertheless, the resampling process may blur details on the shapes. To address this limitation, Score-based Denoising (Score) [8] utilises gradient ascent to guide the denoising process. In recent years, other supervised methods have also been proposed [26–29].

Unsupervised learning-based methods. While supervised methods have demonstrated superior capabilities, ground-truth labels are not always available during training. To address the issue, unsupervised point cloud denoising methods have emerged over the years. Hermosilla et al. [30] proposed Total Denoising (TotalDn) which achieves noise removal by converging noisy points to their unique modes. However, it usually requires multiple denoising iterations to effectively remove noise. In the aforementioned DMR denoising method [7], Luo and Hu observed that points with denser neighbourhoods are typically closer to the ground-truth surfaces. Thus, these points can be treated as denoising objectives. With that in mind, they designed an unsupervised version of DMR which trains the network using an altered unsupervised loss function. Similarly, Score [8] exploits an unsupervised loss that shares the same training objective as the supervised one. However, the unsupervised loss tends to cluster points together, resulting in denoised point clouds that are distributed unevenly.

2.2 Unsupervised image denoising

Conventional image denoising operations, such as Non-local Means [31] and Bilateral Filter [32], as well as supervised methods [33, 34], have demonstrated promising capabilities and have been widely adopted. However, noise-free images are not always available as ground-truth labels during the training process. Over the years, a range of unsupervised image denoising methods have been proposed, demonstrating results comparable to those of conventional and supervised methods. For instance, Deep Image Prior [35] can learn image priors and produce high-quality images without taking ground-truth data as input. Lehtinen et al. [9] proposed Noise2Noise where the network only observes noisy images and learns to restore their clean versions. Krull et al. [10] proposed Noise2Void where the objective pixels' values are masked, and the network attempts to predict those values from neighboring patches. Similarly, Batson and Royer [36] proposed Noise2Self which aims to predict residual noise from noisy images alone. Moran et al. [11] proposed Noisier2Noise to restore clean images from additional noise that is added to noisy images. In summary, the aforementioned unsupervised methods showcase superior noise removal capabilities on images.

3 Our method

3.1 Problem formulation

In this section, we formulate the relationship of displacement information across different versions of noisy point clouds, and establish a structured approach to approximate the clean points that is inspired by unsupervised image denoising [11]. We start with the basic definitions: a clean point cloud $\mathcal{P}$ with T points can be defined as a set of 3D coordinates as

$$\mathcal{P} = \{\mathbf{p}_i \mid i = 1, \ldots, T\}, \tag{1}$$

where $\mathbf{p}_i = (x_i, y_i, z_i)$ stands for each point in $\mathcal{P}$. We also define the noisy version of $\mathcal{P}$ (denoted as $\hat{\mathcal{P}}$) as

$$\hat{\mathcal{P}} = \{\hat{\mathbf{p}}_i \mid i = 1, \ldots, T\} = \mathcal{P} + \mathcal{N}, \tag{2}$$

where $\hat{\mathbf{p}}_i$ stands for each noisy point in $\hat{\mathcal{P}}$, $\mathcal{N}$ is the noise component that perturbs each point, and we denote $\mathcal{N} \sim \mathcal{A}$ where $\mathcal{A}$ stands for the noise distribution.

We then define another noise component $\mathcal{M}$ sampled from the same noise distribution, such that $\mathcal{M} \sim \mathcal{A}$. *Note that the two noise components $\mathcal{M}$ and $\mathcal{N}$ are independent and identically distributed (i.i.d.), but are NOT identical to each other.* We add $\mathcal{M}$ to $\hat{\mathcal{P}}$ to obtain another noisier point cloud $\hat{\mathcal{P}}'$ where

$$\begin{aligned}\hat{\mathcal{P}}' = \{\hat{\mathbf{p}}'_i \mid i = 1, \ldots, T\} &= \hat{\mathcal{P}} + \mathcal{M} \\ &= \mathcal{P} + \mathcal{N} + \mathcal{M},\end{aligned} \tag{3}$$

such that $\hat{\mathcal{P}}'$ is doubly-noisy and contains twice the amount of noise in $\hat{\mathcal{P}}$. The concepts of clean, noisy and doubly-noisy point clouds are shown in Fig. 1. *Our network has no access to the ground-truth point clouds during training.*

We argue that it is feasible to estimate the clean point cloud $\mathcal{P}$ based on the noisy inputs. Given $\hat{\mathcal{P}}'$ as the prior, which represents the observed doubly-noisy point cloud, we denote the estimation of the overall surface of the noisy point cloud $\hat{\mathcal{P}}$ in an expected value format $\mathbb{E}(\hat{\mathcal{P}}|\hat{\mathcal{P}}')$. Based on Eq. (2), we have

$$\begin{aligned}\mathbb{E}(\hat{\mathcal{P}}|\hat{\mathcal{P}}') &= \mathbb{E}(\mathcal{P} + \mathcal{N}|\hat{\mathcal{P}}') \\ &= \mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') + \mathbb{E}(\mathcal{N}|\hat{\mathcal{P}}').\end{aligned} \tag{4}$$

Since $\mathcal{M}$ and $\mathcal{N}$ are i.i.d. and are both sampled from the distribution $\mathcal{A}$, their expectation values are equivalent, i.e., $\mathbb{E}(\mathcal{M}|\hat{\mathcal{P}}') = \mathbb{E}(\mathcal{N}|\hat{\mathcal{P}}')$. With that established, we can derive the following:

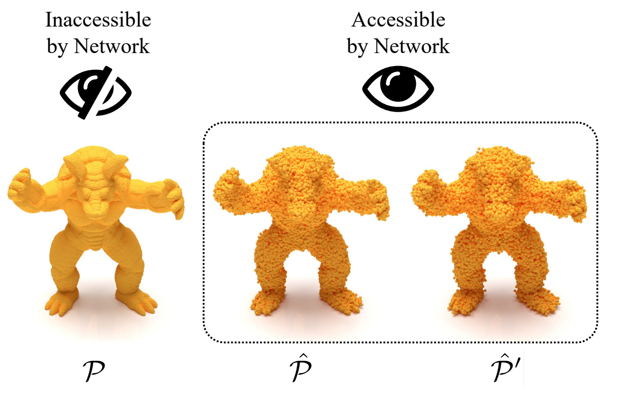

Fig. 1 Point clouds $\mathcal{P}$, $\hat{\mathcal{P}}$ and $\hat{\mathcal{P}}'$ and their corresponding visibilities to the network. Our network can only access the noisy versions of the point clouds.

$$\begin{aligned}2\mathbb{E}(\hat{\mathcal{P}}|\hat{\mathcal{P}}') &= 2\mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') + 2\mathbb{E}(\mathcal{N}|\hat{\mathcal{P}}') \\ &= \mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') + \mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') + \mathbb{E}(\mathcal{N}|\hat{\mathcal{P}}') + \mathbb{E}(\mathcal{N}|\hat{\mathcal{P}}') \\ &= \mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') + \mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') + \mathbb{E}(\mathcal{N}|\hat{\mathcal{P}}') + \mathbb{E}(\mathcal{M}|\hat{\mathcal{P}}') \\ &= \mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') + \mathbb{E}(\mathcal{P} + \mathcal{N} + \mathcal{M}|\hat{\mathcal{P}}') \\ &= \mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') + \mathbb{E}(\hat{\mathcal{P}}'|\hat{\mathcal{P}}'),\end{aligned} \tag{5}$$

which thus leads to

$$\begin{aligned}\mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') &= 2\mathbb{E}(\hat{\mathcal{P}}|\hat{\mathcal{P}}') - \mathbb{E}(\hat{\mathcal{P}}'|\hat{\mathcal{P}}') \\ \mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') - \mathbb{E}(\hat{\mathcal{P}}|\hat{\mathcal{P}}') &= \mathbb{E}(\hat{\mathcal{P}}|\hat{\mathcal{P}}') - \mathbb{E}(\hat{\mathcal{P}}'|\hat{\mathcal{P}}').\end{aligned} \tag{6}$$

Here, we denote $\mathbb{E}(\mathcal{P}|\hat{\mathcal{P}}') = \bar{\mathcal{P}}$, $\mathbb{E}(\hat{\mathcal{P}}|\hat{\mathcal{P}}') = \bar{\hat{\mathcal{P}}}$ and $\mathbb{E}(\hat{\mathcal{P}}'|\hat{\mathcal{P}}') = \bar{\hat{\mathcal{P}}}'$, where $\bar{\mathcal{P}}$, $\bar{\hat{\mathcal{P}}}$ and $\bar{\hat{\mathcal{P}}}'$ stand for the predictions of $\mathcal{P}$, $\hat{\mathcal{P}}$ and $\hat{\mathcal{P}}'$, respectively. We substitute the notations back to Eq. (6) and have

$$\bar{\mathcal{P}} - \bar{\hat{\mathcal{P}}} = \bar{\hat{\mathcal{P}}} - \bar{\hat{\mathcal{P}}}'. \tag{7}$$

It is worth noting that $\bar{\hat{\mathcal{P}}}' = \mathbb{E}(\hat{\mathcal{P}}'|\hat{\mathcal{P}}')$ is equivalent to $\hat{\mathcal{P}}'$ itself based on the rules of conditional expectation and point clouds' discrete characteristics. We also assume the points in $\bar{\mathcal{P}}$, $\bar{\hat{\mathcal{P}}}$ and $\hat{\mathcal{P}}'$ have one-to-one correspondences. Based on this assumption, we can estimate a noise displacement vector for each point in $\hat{\mathcal{P}}'$, and use these vectors to build the approximated clean point cloud $\bar{\mathcal{P}}$. Here, we denote the relationship as

$$\bar{\mathcal{P}} = \{\bar{\mathbf{p}}_i \mid i = 1, \ldots, T\} = \{\hat{\mathbf{p}}'_i + \bar{\mathbf{d}}'_i \mid i = 1, \ldots, T\}, \tag{8}$$

where $\bar{\mathbf{d}}'_i$ represents the predicted displacement vector for point $\hat{\mathbf{p}}'_i$ and directly estimates the position of the corresponding clean point.

The prediction $\bar{\hat{\mathcal{P}}}$ can also be obtained by adding another set of displacement vectors on $\hat{\mathcal{P}}'$. We denote that as

$$\bar{\hat{\mathcal{P}}} = \{\bar{\hat{\mathbf{p}}}_i \mid i = 1, \ldots, T\} = \{\hat{\mathbf{p}}'_i + \bar{\mathbf{d}}_i \mid i = 1, \ldots, T\}. \tag{9}$$

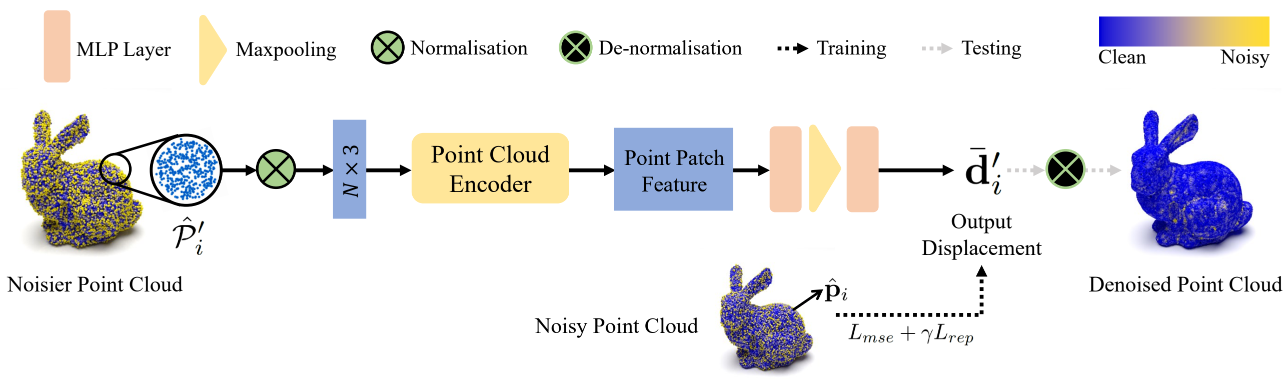

Fig. 2 The network architecture of our method. We employ normalisation before inputting the patch into our encoder, and do de-normalisation during testing phase.

Based on our one-to-one correspondence assumption, we finally demonstrate that the direct estimation $\bar{\mathbf{d}}'_i$ is equivalent to twice $\bar{\mathbf{d}}_i$ by substituting the values back into Eq. (7):

$$\begin{aligned} \bar{\mathbf{p}}_i - \bar{\hat{\mathbf{p}}}_i &= \bar{\hat{\mathbf{p}}}_i - \hat{\mathbf{p}}'_i \\ (\hat{\mathbf{p}}'_i + \bar{\mathbf{d}}'_i) - (\hat{\mathbf{p}}'_i + \bar{\mathbf{d}}_i) &= (\hat{\mathbf{p}}'_i + \bar{\mathbf{d}}_i) - \hat{\mathbf{p}}'_i \\ \bar{\mathbf{d}}'_i - \bar{\mathbf{d}}_i &= \bar{\mathbf{d}}_i \\ \bar{\mathbf{d}}'_i &= 2\bar{\mathbf{d}}_i, \end{aligned} \tag{10}$$

which guides us in designing our loss function for training.

3.2 Denoising framework

Our network architecture is shown in Fig. 2. Similar to previous literature [6, 26, 28], we design a point cloud denoising pipeline based on local neighbourhoods as patches. The noisier point cloud $\hat{\mathcal{P}}'$ is the input source of our network, and we define the local patch $\hat{\mathcal{P}}'_i$ around each query point $\hat{\mathbf{p}}'_i$ as

$$\hat{\mathcal{P}}'_i = \{\hat{\mathbf{p}}'_j \mid \left\|\hat{\mathbf{p}}'_j - \hat{\mathbf{p}}'_i\right\|_2 < r,\ \hat{\mathbf{p}}'_j \in \hat{\mathcal{P}}'\}, \tag{11}$$

where r is the query ball's radius, $\|\cdot\|_2$ represents L2-norm, and $\hat{\mathbf{p}}'_j$ stands for the neighbour points within the radius r. During training, we empirically set r to be equivalent to 5% of the bounding box diagonal length of $\hat{\mathcal{P}}'$, which is a common setting in prior literature [26, 28]. Meanwhile, we also obtain $\hat{\mathbf{p}}_i$, the point with the corresponding index i in point cloud $\hat{\mathcal{P}}$, for training purposes.

We then follow the normalisation process in prior literature [26, 28] to minimise the arbitrary degrees of freedom in the queried patches. We first centre $\hat{\mathcal{P}}'_i$ by translating it to its query point and normalise its size using r, such that $\hat{\mathcal{P}}'_i = (\hat{\mathcal{P}}'_i - \hat{\mathbf{p}}'_i)/r$. Then, we align $\hat{\mathcal{P}}'_i$ with the canonical space using the rotation matrix $\mathbf{R}$ which was obtained via Principal Component Analysis (PCA) decomposition on $\hat{\mathcal{P}}'_i$. We also process $\hat{\mathbf{p}}_i$ following the same transformation procedure for $\hat{\mathcal{P}}'_i$. Next, as the number of points may be inconsistent with each other in raw patches, we set an empirical threshold $N = 500$ to regularise the patch size for $\hat{\mathcal{P}}'_i$ and ensure they can be grouped into mini-batches. Specifically, following previous works, we downsample patches with more than N points, and fill extra points for patches with fewer points than N. Note that during testing phase, we require the inverse of the aforementioned transformations (rotation, scaling and translation) to map the denoised point back to its original position.

We input the normalised patch $\hat{\mathcal{P}}'_i$ (as an $N \times 3$ matrix) into our point cloud encoder. It uses graph-based Dense Block modules [37] as backbone and encodes each point with its neighbours' information. To further enhance the network's generalisation performance, we adopt the Global Shift module in [38] to fuse patch features into fewer points, in order to improve the robustness of our network and increase the inference efficiency. The output of our encoder is a feature matrix which goes through our output module to form the final predicted displacement $\bar{\mathbf{d}}'_i$, a 1×3 vector. During training, we utilise the noisy point $\hat{\mathbf{p}}_i$ to train the estimation of the displacement vector.

3.3 Loss functions

We exploit $\hat{\mathbf{d}}_i = \hat{\mathbf{p}}_i - \hat{\mathbf{p}}'_i$, the actual value of prediction $\bar{\mathbf{d}}_i$ in Eq. (10), to guide the estimation of displacement $\bar{\mathbf{d}}'_i$. We design a Mean Squared Error (MSE) loss function which is defined as

$$L_{mse} = \left\|\bar{\mathbf{d}}'_i - 2\hat{\mathbf{d}}_i\right\|_2^2, \tag{12}$$

and we further discuss our choice regarding noise prediction in Sec. 5.

We also require our denoised point cloud to be spaced out to avoid potential clustering issues. To achieve this, we design a repulsion loss that is inspired by prior literature [6, 26, 28] to assist with the distribution of the denoised points. First of

Table 1 Quantitative results on PUNet's test set, where the best results are marked in **bold**. CD and P2M are both multiplied by 10^4.

Type	Density	$10k$						$50k$					
	Noise level	1%		2%		3%		1%		2%		3%	
	Method	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M	CD	P2M
Op.-based	Bilateral [21]	3.646	1.342	5.007	2.018	6.998	3.557	0.877	0.234	2.376	1.389	6.304	4.730
	Jet [20]	2.712	0.613	4.155	1.347	6.262	2.921	0.851	0.207	2.432	1.403	5.788	4.267
	Non-local [5]	3.506	1.331	4.720	1.985	7.405	4.489	0.966	0.337	2.240	1.317	3.623	2.566
Supervised	PCNet [6]	3.515	1.148	7.467	3.965	13.067	8.737	1.049	0.346	1.447	0.608	2.289	1.285
	DMR [7]	4.482	1.722	4.982	2.115	5.892	2.846	1.162	0.469	1.566	0.800	2.432	1.528
	Score [8]	2.521	0.463	3.686	1.074	4.708	1.942	0.716	0.150	1.288	0.566	1.928	1.041
Unsupervised	TotalDn [30]	3.390	0.826	7.251	3.485	13.385	8.740	1.024	0.314	2.722	1.567	7.474	5.729
	DMR-U [7]	5.313	2.522	6.455	3.317	8.134	4.647	1.226	0.521	2.138	1.251	2.496	1.520
	Score-U [8]	3.107	0.888	4.675	1.829	7.225	3.762	0.918	0.265	2.439	1.411	5.303	3.841
	Ours	**2.204**	**0.354**	**3.449**	**0.930**	**4.153**	**1.554**	**0.700**	**0.134**	**1.056**	**0.374**	**1.282**	**0.563**

all, we define a pseudo-clean point cloud $\tilde{\mathcal{P}}$ as

$$\tilde{\mathcal{P}} = \{\hat{\mathbf{p}}'_i + 2\hat{\mathbf{d}}_i \mid i = 1, \dots, T\}, \tag{13}$$

where $\tilde{\mathcal{P}}$ does not actually have a clean appearance. Next, we query pseudo-clean patches from it where the process is defined as

$$\tilde{\mathcal{P}}_i = \{\tilde{\mathbf{p}}_j \mid \|\tilde{\mathbf{p}}_j - \hat{\mathbf{p}}_i\|_2 < r,\ \tilde{\mathbf{p}}_j \in \tilde{\mathcal{P}}\}. \tag{14}$$

Here, we use $\hat{\mathbf{p}}_i \in \hat{\mathcal{P}}$ to perform queries for $\tilde{\mathcal{P}}_i$ as we assume $\hat{\mathbf{p}}_i$ is less impacted by noise, and normalise patch $\tilde{\mathcal{P}}_i$ using the same normalisation process introduced in Sec. 3.2. Then, we use these patches to formulate our repulsion loss as

$$L_{rep} = \max_{\tilde{\mathbf{p}}_j \in \tilde{\mathcal{P}}_i} \left\|(\hat{\mathbf{p}}'_i + \bar{\mathbf{d}}'_i) - \tilde{\mathbf{p}}_j\right\|_2^2. \tag{15}$$

Our final loss L is thus formulated as

$$L = L_{mse} + \gamma L_{rep}, \tag{16}$$

where γ is the factor controlling the repulsion loss and is set to 0.0005. We discuss the setting of γ in Sec. 5.

4 Experimental results

4.1 Training configurations

We carried out our experiments on an NVIDIA RTX 3080 GPU with 10 GB memory. We implemented our network model with PyTorch, and set the batch size for training to 128. We trained the network for 200 epochs with a single Adam optimiser. The learning rate was set to 0.0001 and was multiplied by a factor of 0.1 in the 40^{th}, 80^{th}, 120^{th}, 160^{th}, and 180^{th} epochs, respectively.

4.2 Training dataset

We adopted PUNet [39] for training, which contains 40 clean triangular mesh shapes and the corresponding noise-free point clouds in three different resolutions ($10k$, $30k$ and $50k$, respectively). All clean point clouds were taken from the surfaces of the mesh shapes, and we sample 1,000 patches per point cloud in each epoch. During training, we followed the noise addition method in [8, 27], i.e., the additive noise is zero-mean Gaussian noise with a standard deviation ranging from 0.5% to 2.0% of each shape's bounding sphere radius. To achieve this, the clean point cloud should first be normalised into a unit sphere, and then added with noise $\mathcal{Z} \sim \mathcal{D}(\mu_{\mathcal{D}}, \sigma^2_{\mathcal{D}})$ where $\mathcal{D}$ is the objective noise distribution, $\mu_{\mathcal{D}} = 0$, and $\sigma_{\mathcal{D}} \in [0.005, 0.02]$.

We define the distribution of each level of the additive noise during training as $\mathcal{A}(\mu_{\mathcal{A}}, \sigma^2_{\mathcal{A}})$, and recall that $\mathcal{N} \sim \mathcal{A}$ and $\mathcal{M} \sim \mathcal{A}$. To ensure $\mathcal{N} + \mathcal{M} = \mathcal{Z}$ holds, we need $2\mu_{\mathcal{A}} = \mu_{\mathcal{D}}$ and $2\sigma^2_{\mathcal{A}} = \sigma^2_{\mathcal{D}}$. Thus, we set $\mu_{\mathcal{A}} = 0$ and $\sigma_{\mathcal{A}} \in [\frac{0.005}{\sqrt{2}}, \frac{0.02}{\sqrt{2}}]$ for noise sampling during our training process.

4.3 Quantitative results

PUNet dataset. First, we evaluated the performance of our method on the PUNet test dataset. It contains point clouds sampled from 20 mesh shapes in 2 resolutions (sparse, $10k$ points per shape; and dense, $50k$ points per shape), and has 3 Gaussian noise levels (1%, 2% and 3% of the bounding sphere's radius) for each resolution. Following prior works [8, 27], we ran 1 denoising iteration for 1% and 2% noise and ran 2 denoising iterations for 3% noise. We employed two metrics, Chamfer Distance (CD) and Point-to-mesh Distance (P2M), to measure the quality of the denoised point clouds. For both metrics, smaller values indicate more accurate results. We denote the unsupervised versions of DMR [7] and Score [8] using DMR-U and Score-U, respectively. The denoising results are shown in Table 1 where our method outperforms all others on each setting.

Kinect dataset. We also tested our method on real-world scanned data captured by Kinect v1 and v2 [40], which is inherently noisy. The datasets provide reconstructed clean data

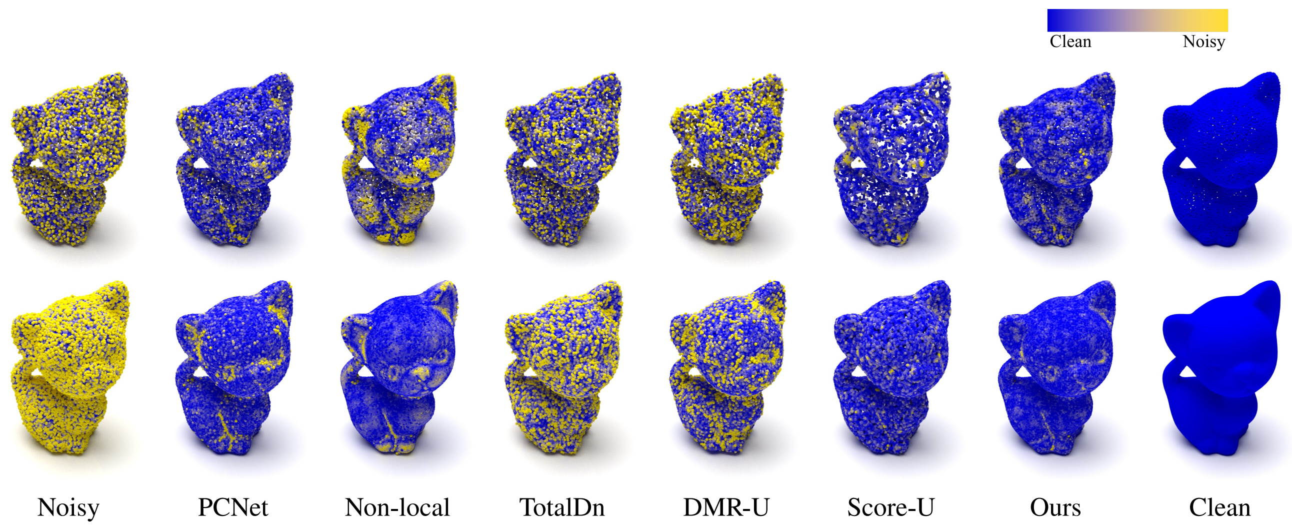

Fig. 3 The visual comparison results on a shape from PUNet dataset (1% noise) where yellow points are further from the ground-truth mesh surfaces. The top row shows the sparse ($10k$) configuration and the bottom row shows the dense ($50k$) configuration.

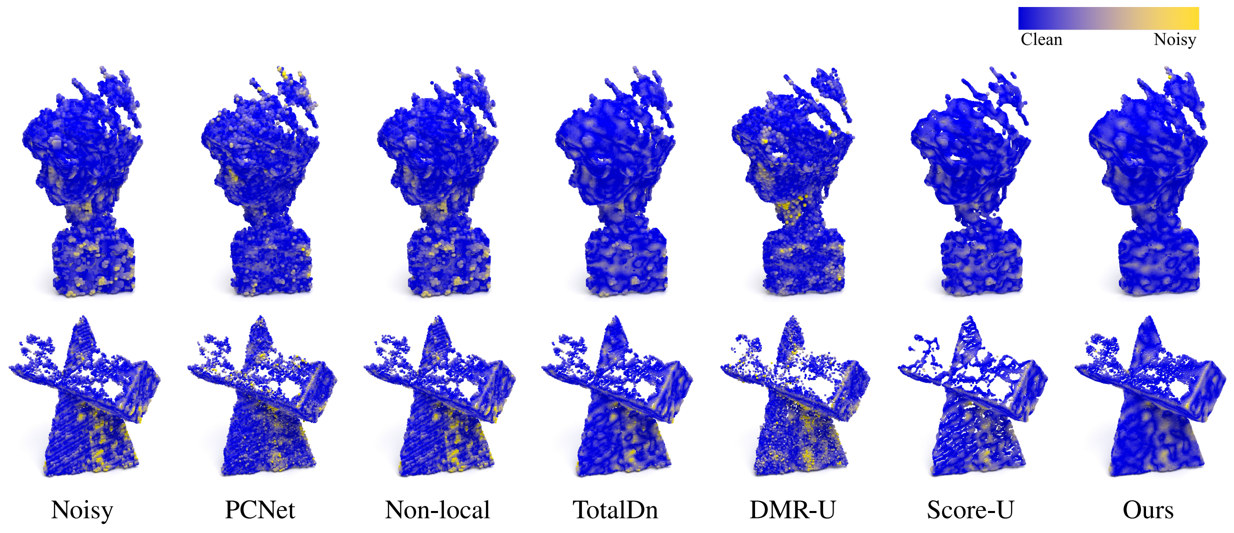

Fig. 4 The visual denoising effects on Kinect dataset where yellow points are further from the ground-truth mesh surfaces.

for quantitative measurement. We present the results in Table 2 where our method outperforms other unsupervised denoising techniques as well as the supervised method PCNet [6] and the optimisation-based method Non-local [5] in terms of CD and P2M errors.

Table 2 Results on Kinect dataset. CD and P2M are both multiplied by 10^4.

Dataset	Kinect v1		Kinect v2	
Method	CD	P2M	CD	P2M
PCNet [6]	1.540	0.780	2.304	1.234
Non-local [5]	1.442	0.750	2.264	1.305
TotalDn [30]	1.356	0.703	2.094	1.200
DMR-U [7]	1.668	0.743	2.997	1.494
Score-U [8]	1.664	0.738	2.547	1.092
Ours	**1.213**	**0.623**	**1.918**	**1.088**

4.4 Qualitative results

PUNet dataset. We first demonstrate the visual denoising effects on PUNet dataset. We use a colour scale to indicate the distances of the denoised points from the ground-truth mesh surfaces, where blue indicates more accurate positions and yellow indicates inaccurate positions. The results are listed in Fig. 3 which includes both the sparse and the dense resolutions. As illustrated, the downsampling-resampling strategy of DMR-U does not effectively remove the residual noise, leading to substantial noisy points that are marked in yellow. Score-U tends to cluster the points together, resulting in uneven distributions. Other methods cannot effectively

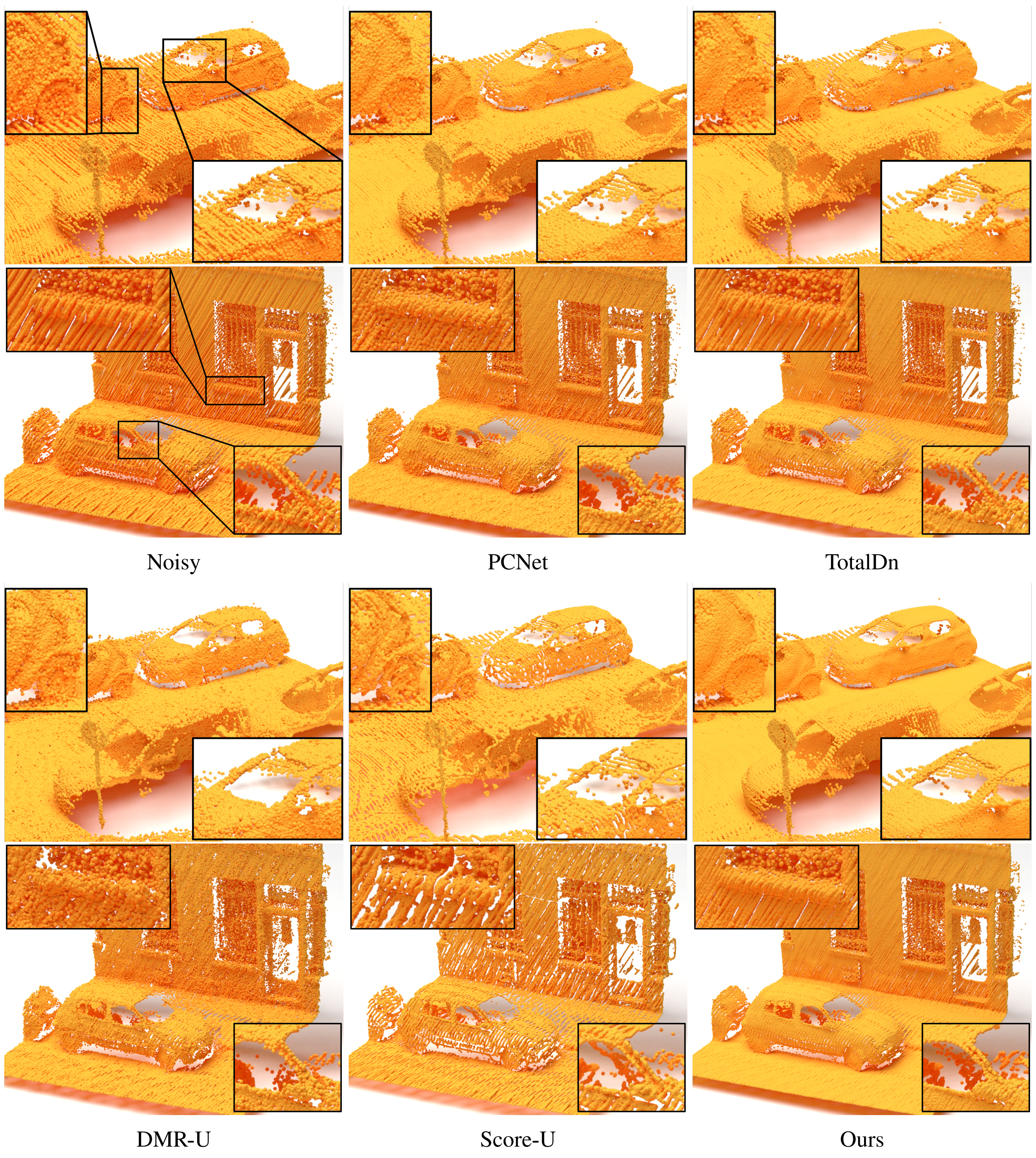

Fig. 5 Comparison of the denoising results on Paris-rue-Madame dataset.

preserve small details, resulting in yellow regions in those areas. By contrast, our method achieves more accurate results overall on both sparse and dense configuration settings.

Kinect dataset. We also present the visual denoising effects on Kinect dataset in Fig. 4 where the denoised shapes are taken from the results presented in Table 2. It demonstrates that our method can produce smoother point cloud surfaces compared with other methods.

Paris-rue-Madame dataset. We show denoising results on Paris-rue-Madame [41], a street scene point cloud dataset collected by laser scanners. It is contaminated with severe noise due to outdoor environmental factors. No ground-truth data is associated with this dataset, so we compare the visual denoising effects in terms of smoothness and feature preservation. The results are shown in Fig. 5 where our method outputs smoother point clouds and effectively restores small details, as illustrated in the close-up windows.

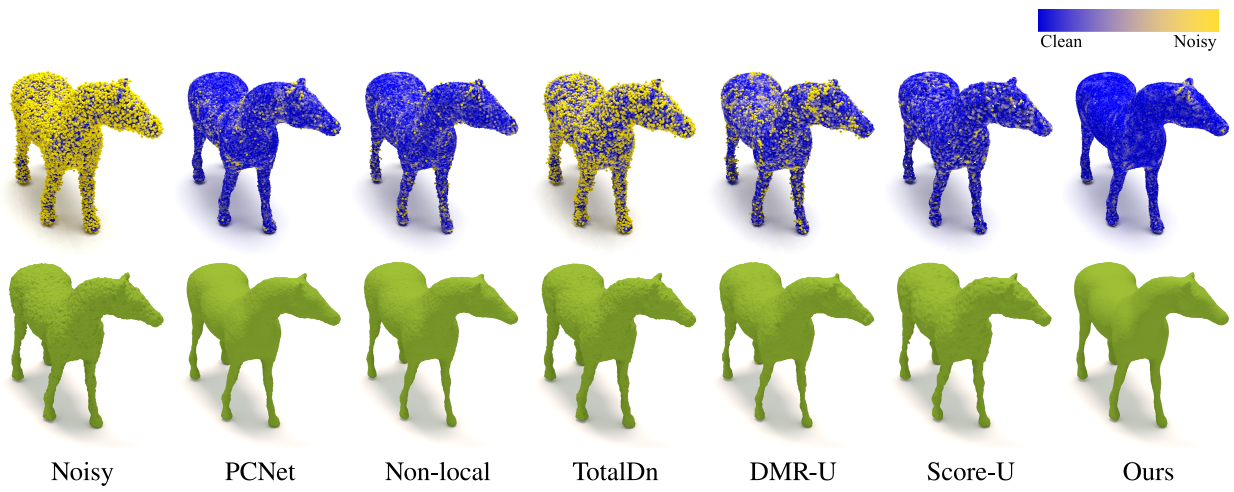

Fig. 6 Mesh reconstruction results on the denoised point clouds.

5 Ablation study

We performed our ablation study on the validation dataset following the settings in [8] and measured CD and P2M metrics. In this section, we explore the effect of repulsion loss and our displacement regression technique.

Repulsion loss. We first validate the effects with and without the repulsion term L_{rep}, and additionally evaluate with different γ values. The results are shown in configuration number 1 to 6 in Table 3, where setting $\gamma = 0.0005$ achieves the best results among the options.

Displacement regression technique. An alternative training technique is regressing the displacement for noise $\mathcal{M}$ only and doubling it during the evaluation phase. Consequently, we regress $\bar{\mathbf{d}}_i$ in this setting and the altered training objective becomes

$$L' = L'_{mse} + \gamma L'_{rep}, \quad (17)$$

where we define L'_{mse} as

$$L'_{mse} = \left\| \bar{\mathbf{d}}_i - \hat{\mathbf{d}}_i \right\|_2^2, \quad (18)$$

and define L'_{rep} as

$$L'_{rep} = \max_{\tilde{\mathbf{p}}_j \in \tilde{\mathcal{P}}_i} \left\| (\hat{\mathbf{p}}'_i + 2\bar{\mathbf{d}}_i) - \tilde{\mathbf{p}}_j \right\|_2^2. \quad (19)$$

We set γ to be consistent with our experiments. The results, displayed in configuration number 7 in Table 3, cannot achieve comparable results on the validation dataset.

6 Application

Denoised point clouds can be utilised for mesh reconstruction. To achieve this, we first exploit HSurf-Net [38] to estimate normals for the denoised point clouds produced by different methods, and then perform Poisson surface reconstruction [42]. The denoised point clouds and the reconstructed meshes are shown in Fig. 6 where our method achieves smoother results on the mesh surface.

Table 3 Ablation study on validation set with different training configurations. CD and P2M are both multiplied by 10^4.

Config.	Loss	γ	CD	P2M
1	L	0.0	2.923	0.619
2	L	0.0001	2.916	0.615
3	L	0.0005	**2.880**	**0.606**
4	L	0.001	2.903	0.610
5	L	0.01	2.909	0.617
6	L	0.05	2.977	0.673
7	L'	0.0005	3.056	0.668

7 Discussion and conclusion

In this paper, we presented a novel unsupervised point cloud denoising framework that learns to restore clean point clouds from noisy inputs only. During training, we leverage two levels of noise, where the second noise level is twice that of the first, and train our network to predict clean points without using ground-truth data as labels. Extensive experimental results demonstrate that our method generalises well across synthetic and real-world noise, outperforming state-of-the-art unsupervised methods and achieving comparable performance to supervised methods.

Our method unavoidably demonstrates some limitations during denoising. For instance, since our method is trained using point-based loss functions only, it may occasionally blur the sharp features and produce smooth edges and corners. In order to enhance the capability of preserving sharp features, it is necessary to utilise normal information as demonstrated in the prior literature [26, 43]. For our future works, we aim to explore further in terms of sharp feature preservation by incorporating normal information in our denoising process.

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Charles RQ, Su H, Kaichun M, Guibas LJ. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, 77–85, doi:10.1109/CVPR.2017.16.

[2] Guerrero P, Kleiman Y, Ovsjanikov M, Mitra NJ. PCPNet: Learning Local Shape Properties from Raw Point Clouds. *Computer Graphics Forum*, 2018, 37(2): 75–85, doi:10.1111/cgf.13343.

[3] Alexa M, Behr J, Cohen-Or D, Fleishman S, Levin D, Silva C. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 2003, 9(1): 3–15, doi:10.1109/TVCG.2003.1175093.

[4] Lipman Y, Cohen-Or D, Levin D, Tal-Ezer H. Parameterization-Free Projection for Geometry Reconstruction. *ACM Trans. Graph.*, 2007, 26(3): 22–es, doi:10.1145/1276377.1276405.

[5] Wang J, Jiang J, Lu X, Wang M. Rethinking Point Cloud Filtering: A Non-Local Position Based Approach. *Computer-Aided Design*, 2022, 144: 103162, doi:https://doi.org/10.1016/j.cad.2021.103162.

[6] Rakotosaona MJ, La Barbera V, Guerrero P, Mitra NJ, Ovsjanikov M. POINTCLEANNET: Learning to denoise and remove outliers from dense point clouds. In *Computer Graphics Forum*, volume 39, 2020, 185–203.

[7] Luo S, Hu W. Differentiable Manifold Reconstruction for Point Cloud Denoising. In *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[8] Luo S, Hu W. Score-Based Point Cloud Denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, 4583–4592.

[9] Lehtinen J, Munkberg J, Hasselgren J, Laine S, Karras T, Aittala M, Aila T. Noise2Noise: Learning Image Restoration without Clean Data. In J Dy, A Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 2018, 2965–2974.

[10] Krull A, Buchholz TO, Jug F. Noise2Void - Learning Denoising From Single Noisy Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[11] Moran N, Schmidt D, Zhong Y, Coady P. Noisier2Noise: Learning to Denoise From Unpaired Noisy Data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[12] Levin D. The Approximation Power of Moving Least-Squares. *Math. Comput.*, 1998, 67(224): 1517–1531, doi:10.1090/S0025-5718-98-00974-0.

[13] Alexa M, Behr J, Cohen-Or D, Fleishman S, Levin D, Silva C. Point set surfaces. In *Proceedings Visualization, 2001. VIS '01.*, 2001, 21–29, 537, doi:10.1109/VISUAL.2001.964489.

[14] Amenta N, Kil YJ. Defining Point-Set Surfaces. *ACM Trans. Graph.*, 2004, 23(3): 264–270, doi:10.1145/1015706.1015713.

[15] Fleishman S, Cohen-Or D, Silva CT. Robust Moving Least-Squares Fitting with Sharp Features. *ACM Trans. Graph.*, 2005, 24(3): 544–552, doi:10.1145/1073204.1073227.

[16] Öztireli AC, Guennebaud G, Gross M. Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression. *Computer Graphics Forum*, 2009, 28(2): 493–501, doi:https://doi.org/10.1111/j.1467-8659.2009.01388.x.

[17] Huang H, Li D, Zhang H, Ascher U, Cohen-Or D. Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2009)*, 2009, 28: 176:1–176:7.

[18] Huang H, Wu S, Gong M, Cohen-Or D, Ascher U, Zhang H. Edge-Aware Point Set Resampling. *ACM Transactions on Graphics*, 2013, 32: 9:1–9:12.

[19] Preiner R, Mattausch O, Arikan M, Pajarola R, Wimmer M. Continuous Projection for Fast L1 Reconstruction. *ACM Trans. Graph.*, 2014, 33(4), doi:10.1145/2601097.2601172.

[20] Cazals F, Pouget M. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 2005, 22(2): 121–146, doi:https://doi.org/10.1016/j.cagd.2004.09.004.

[21] Fleishman S, Drori I, Cohen-Or D. Bilateral Mesh Denoising. *ACM Trans. Graph.*, 2003, 22(3): 950–953, doi:10.1145/882262.882368.

[22] Lu X, Wu S, Chen H, Yeung SK, Chen W, Zwicker M. GPF: GMM-Inspired Feature-Preserving Point Set Filtering. *IEEE Transactions on Visualization and Computer Graphics*, 2018, 24(8): 2315–2326, doi:10.1109/TVCG.2017.2725948.

[23] Chen S, Wang J, Pan W, Gao S, Wang M, Lu X. Towards uniform point distribution in feature-preserving point cloud filtering. *Computational Visual Media*, 2023, 9(2): 249–263, doi:10.1007/s41095-022-0278-4.

[24] Chen H, Wei M, Sun Y, Xie X, Wang J. Multi-Patch Collaborative Point Cloud Denoising via Low-Rank Recovery with Graph Constraint. *IEEE Transactions on Visualization and Computer Graphics*, 2020, 26(11): 3255–3270, doi:10.1109/TVCG.2019.2920817.

[25] Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics (TOG)*, 2019.

[26] Zhang D, Lu X, Qin H, He Y. Pointfilter: Point cloud filtering via encoder-decoder modeling. *IEEE Transactions on Visualization and Computer Graphics*, 2020.

[27] Mao A, Du Z, Wen YH, Xuan J, Liu YJ. PD-Flow: A Point Cloud Denoising Framework with Normalizing Flows. In *The European Conference on Computer Vision (ECCV)*, 2022.

[28] Wang W, Pan W, Liu X, Su K, Rolfe B, Lu X. Random screening-based feature aggregation for point cloud denoising. *Computers & Graphics*, 2023, 116: 64–72, doi:https://doi.org/10.1016/j.cag.2023.08.013.

[29] De Silva Edirimuni D, Lu X, Shao Z, Li G, Robles-Kelly A, He Y. IterativePFN: True Iterative Point Cloud Filtering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, 13530–13539.

[30] Hermosilla P, Ritschel T, Ropinski T. Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[31] Buades A, Coll B, Morel JM. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, 2005, 60–65 vol. 2, doi:10.1109/CVPR.2005.38.

[32] Tomasi C, Manduchi R. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998, 839–846, doi:10.1109/ICCV.1998.710815.

[33] Burger HC, Schuler CJ, Harmeling S. Image denoising: Can plain neural networks compete with BM3D? In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, 2392–2399, doi:10.1109/CVPR.2012.6247952.

[34] Zhang K, Zuo W, Chen Y, Meng D, Zhang L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on Image Processing*, 2017, 26(7): 3142–3155, doi:10.1109/TIP.2017.2662206.

[35] Lempitsky V, Vedaldi A, Ulyanov D. Deep Image Prior. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, 9446–9454, doi:10.1109/CVPR.2018.00984.

[36] Batson J, Royer L. Noise2Self: Blind Denoising by Self-Supervision. *CoRR*, 2019, abs/1901.11365.

[37] Huang G, Liu Z, van der Maaten L, Weinberger KQ. Densely Connected Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[38] Li Q, Liu YS, Cheng JS, Wang C, Fang Y, Han Z. HSurf-Net: Normal Estimation for 3D Point Clouds by Learning Hyper Surfaces. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[39] Yu L, Li X, Fu CW, Cohen-Or D, Heng PA. PU-Net: Point Cloud Upsampling Network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[40] Wang PS, Liu Y, Tong X. Mesh Denoising via Cascaded Normal Regression. *ACM Trans. Graph.*, 2016, 35(6), doi:10.1145/2980179.2980232.

[41] Serna A, Marcotegui B, Goulette F, Deschaud JE. Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In *4th International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014*, 2014.

[42] Kazhdan M, Bolitho M, Hoppe H. Poisson Surface Reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, 2006, 61–70.

[43] Lu D, Lu X, Sun Y, Wang J. Deep feature-preserving normal estimation for point cloud filtering. *Computer-Aided Design*, 2020: 102860.

Author biography

Weijia Wang is currently pursuing his Ph.D. at Deakin University. Prior to that, he was awarded Master of Information Technology and Bachelor of Science at the University of Melbourne. His research interests include 3D geometry processing and deep learning.

Xuequan Lu is a Senior Lecturer at La Trobe University. He received his Ph.D. in the College of Computer Science and Technology at Zhejiang University. His research interests include geometry modelling, processing and analysis.