Differentiable Manifold Reconstruction for Point Cloud Denoising

Shitong Luo, Wei Hu
Wangxuan Institute of Computer Technology, Peking University
{luost,forhuwei}@pku.edu.cn

ABSTRACT

3D point clouds are often perturbed by noise due to the inherent limitation of acquisition equipments, which obstructs downstream tasks such as surface reconstruction, rendering and so on. Previous works mostly infer the displacement of noisy points from the underlying surface, which however are not designated to recover the surface explicitly and may lead to sub-optimal denoising results. To this end, we propose to learn the underlying manifold of a noisy point cloud from differentiably subsampled points with trivial noise perturbation and their embedded neighborhood feature, aiming to capture intrinsic structures in point clouds. Specifically, we present an autoencoder-like neural network. The encoder learns both local and non-local feature representations of each point, and then samples points with low noise via an adaptive differentiable pooling operation. Afterwards, the decoder infers the underlying manifold by transforming each sampled point along with the embedded feature of its neighborhood to a local surface centered around the point. By resampling on the reconstructed manifold, we obtain a denoised point cloud. Further, we design an unsupervised training loss, so that our network can be trained in either an unsupervised or supervised fashion. Experiments show that our method significantly outperforms state-of-the-art denoising methods under both synthetic noise and real world noise. The code and data are available at https://github.com/luost26/DMRDenoise.

CCS CONCEPTS

• **Computing methodologies** → **Point-based models**; *3D imaging*.

KEYWORDS

point clouds, denoising, manifold, differentiable pooling

1 INTRODUCTION

Recent advances in depth sensing, laser scanning and image processing have enabled convenient acquisition of 3D point clouds from real world scenes [1]. Point clouds consist of discrete 3D points irregularly sampled from continuous surfaces, which can be applied to a wide range of applications such as autonomous driving, robotics and immersive tele-presence. Nevertheless, they are often contaminated by noise due to the inherent limitations of scanning devices or matching ambiguities in the reconstruction from images, which significantly affects downstream understanding tasks since the underlying structures are deformed. Hence, point cloud denoising is crucial to relevant 3D vision applications, which is also challenging due to the irregular and unordered characteristics of point clouds.

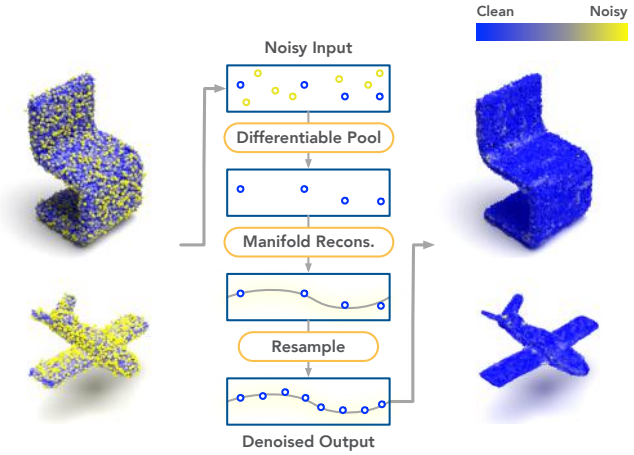

Figure 1: An overview of our method. The denoising network takes noisy point clouds as input, and then samples a subset of points with low noise via a differentiable pooling layer. Afterwards, manifolds are reconstructed based on the sampled subset of points. Finally, by sampling on the reconstructed manifold, we obtain denoised point clouds.

Previous point cloud denoising methods include non-deep-learning based methods [4, 5, 14, 17, 23, 39] and deep-learning based methods [6, 13, 26, 37]. We focus on the class of deep-learning based methods, which have achieved promising denoising results thanks to the advent of neural network architectures crafted for point clouds [1, 10, 24, 25, 31, 32, 35–38]. Neural Projection [6], PointCleanNet [26] and Total Denoising [13] are pioneers of deep-learning based point cloud denoising approaches. In general, these methods infer the displacement of noisy points from the underlying surface and reconstruct *points*, which however are not designated to recover the surface explicitly and may lead to sub-optimal denoising results.

To this end, inspired by that a point cloud is typically a representation of some underlying surface or 2D manifold over a set of sampled points, we propose to explicitly learn the underlying *manifold* of a noisy point cloud for reconstruction, aiming to capture intrinsic structures in point clouds. As demonstrated in Fig. 1, the key idea is to sample a subset of points with low noise (*i.e.*, closer to the clean surface) via differentiable pooling, and then reconstruct the underlying manifold from these points and their embedded

[1] Commercial products include Microsoft Kinect (2010-2014), Intel RealSense (2015-), Velodyne LiDAR (2007-2020), LiDAR scanner of Apple iPad Pro (2020), *etc.*

Corresponding author: Wei Hu (forhuwei@pku.edu.cn). This work was supported by National Natural Science Foundation of China [61972009] and Beijing Natural Science Foundation [4194080].

neighborhood features. By resampling on the reconstructed manifold, we obtain a denoised point cloud.

In particular, we present an autoencoder-like neural network for differentiable manifold reconstruction. At the encoder, we learn both local and non-local features of each point, which embed the representations of local surfaces. Based on the learned features, we sample points that are closer to the underlying surfaces (less noise perturbation) via the proposed adaptive *differentiable pooling* operation, which narrows down the latent space for reconstructing the underlying manifold. These sampled points are pre-filtered and retained, while the other points are discarded. At the decoder, we infer the underlying manifold by transforming each sampled point along with the embedded neighborhood feature to a local surface centered around the point—referred to as *"patch manifold"*. By sampling on such patch manifolds, we finally obtain a denoised point set which captures intrinsic structures of the underlying surface. Further, we design an unsupervised training loss, so that our network can be trained in either an unsupervised or supervised fashion. Experiments show that our method significantly outperforms state-of-the-art denoising methods especially at high noise levels.

To summarize, the contributions of our paper include

- We propose a differentiable manifold reconstruction paradigm for point cloud denoising, aiming to learn the underlying manifold of a noisy point cloud via an autoencoder-like framework.
- We propose an adaptive differentiable pooling operator on point clouds, which samples points that are closer to the underlying surfaces and thus narrows down the latent space for reconstructing the underlying manifold.
- We infer the underlying manifold by transforming each sampled point along with the embedded feature of its neighborhood to a local surface centered around the point—a patch manifold.
- We design an unsupervised training loss, so that our network can be trained in either an unsupervised or supervised fashion.

2 RELATED WORK

2.1 Non-deep-learning Based Point Cloud Denoising

Non-deep-learning based point cloud denoising methods have been extensively studied, which mainly include local-surface-fitting based methods, sparsity based methods and graph based methods.

- **Local-surface-fitting based methods.** This class of methods approximate the point cloud with a smooth surface and then project points in the noisy point cloud onto the fitted surface. [2] proposes a moving least squares (MLS) projection operator to calculate the optimal fitting surface of the point cloud. Similarly, other surface fitting methods have been proposed for point cloud denoising such as jet-fitting with re-projection [4] and bilateral filtering [17] which take into account both point coordinates and normals. However, these methods are often sensitive to outliers.
- **Sparsity based methods.** This class of methods are based on the sparse representation theory [3, 30, 34]. They generally reconstruct normal vectors by solving an optimization problem of sparse regularization and then update the position of points based on the reconstructed normals. Moving Robust Principal Component Analysis (MRPCA) [23] is a recently proposed sparsity-based method. However, the performance tends to degrade when the noise level is high due to over-smoothing or over-sharpening.
- **Graph based methods.** This class of methods represent point clouds on graphs and perform denoising via graph filters [9, 14, 15, 28, 39]. In [28], the input point cloud is represented as signal on a k-nearest-neighbor graph and then denoised via a convex optimization problem regularized by the gradient of the graph. In [39], Graph Laplacian Regularization (GLR) of low dimensional manifold models is proposed for point cloud denoising.

2.2 Deep-learning Based Point Cloud Denoising

With the advent of point-based neural networks [24, 25, 32], deep point cloud denoising has received increasing attention. Existing deep learning based methods generally involve predicting the displacement of each point in noisy point clouds via neural networks, and apply the inverse displacement to each point.

Among them, Neural Projection [6] employs PointNet [24] to predict the tangent plane at each point, and projects points to the tangent planes. However, training a Neural Projection denoiser requires the access to not only clean point clouds but also normal vectors of each point.

PointCleanNet [26] predicts displacement of points from the clean surface via PCPNet [12]—a variant of PointNet. It is trained end-to-end by minimizing the $\ell 2$ distance between the denoised point cloud and the ground truth, which does not require the access to normal vectors. PointCleanNet out-performs some classical denoising methods including bilateral filtering and jet fitting. The main defect of PointCleanNet includes outlier sensitivity and point cloud shrinking.

Total Denoising [13] is the first unsupervised deep learning method for point cloud denoising. It is based on the assumption that points with denser surroundings are closer to the underlying surface. Hence, it introduces a spatial prior that steers convergence towards the underlying surface without the supervision of ground truth point clouds. However, the unsupervised denoiser is sensitive to outliers and may shrink point clouds.

In addition to denoising networks, some other neural network architectures involve point cloud consolidation, which includes denoising but is often only applicable to trivial noise. PointProNet [27] projects patches in the point cloud into 2D height maps and leverages a 2D CNN to denoise and upsample them. EC-Net [37] and 3PU [36] mainly focus on upsampling, and have shown to be robust against trivial noise. These consolidation methods are generally prone to fail when the noise level is high [26].

3 METHOD

In this section, we present our method on learning the underlying manifold for point cloud denoising. We start with an overview of our key ideas, and then elaborate on the proposed differentiable manifold reconstruction. Finally, we present our loss functions as well as provide further analysis into our method.

3.1 Overview

Given an input point cloud $\mathbf{P} \in \mathbb{R}^{N\times 3}$ corrupted by noise, our network produces a clean point cloud $\widetilde{\mathbf{P}} \in \mathbb{R}^{N\times 3}$. As illustrated in Fig. 2, we propose an autoencoder-like network architecture for denoising.

- **Representation Encoder $\mathcal{E}$.** $\mathcal{E}$ samples a subset of M points $\mathbf{S} \in \mathbb{R}^{M\times 3}$ that are perturbed by less noise from $\mathbf{P}$ via differentiable pooling. Specifically, $\mathcal{E}$ consists of a feature extraction unit and a differentiable downsampling (pooling) unit. The feature extraction unit produces features that encode both local and non-local geometry at each point of $\mathbf{P}$. The extracted features are then fed into the differentiable pooling operator—essentially a downsampling unit that identifies points that are closer to the underlying surface, leading to a subset of points $\mathbf{S}$.
- **Manifold Reconstruction Decoder $\mathcal{D}$.** $\mathcal{D}$ first infers the underlying manifold from $\mathbf{S}$ and then samples on the inferred manifold to produce the denoised point set $\widetilde{\mathbf{P}} \in \mathbb{R}^{N\times 3}$. We transform each point in $\mathbf{S}$ along with the embedded neighborhood feature to a local surface centered around each point—a *patch manifold*. By sampling multiple times on each patch manifold, we reconstruct a clean point cloud $\widetilde{\mathbf{P}}$.

Further, we propose a dual supervised loss function as well as an unsupervised loss, so that our network can be trained end-to-end in an unsupervised or supervised fashion.

3.2 Representation Encoder with Differentiable Pooling

The representation encoder consists of a feature extraction unit and a differentiable pooling unit, which we discuss in details as follows.

3.2.1 Feature Extraction Unit. The feature extraction unit consists of multiple dynamic graph convolution layers, leveraging on the DGCNN [32]. Given features $\mathbf{X}^\ell = \{\mathbf{x}_i^\ell\}_{i=1}^N \in \mathbb{R}^{N\times F^\ell}$ in the ℓth layer, the $(\ell+1)$th layer first constructs a k-Nearest-Neighbor (k-NN) graph based on the Euclidean distance between features, and then performs edge convolution [32] on the graph:

$$\mathbf{x}_i^{\ell+1} = G_\ell(\mathbf{X}^\ell) = \mathrm{ReLU}\big(\max_{j\in\mathcal{N}(i)} H_\theta(\mathbf{x}_i^\ell, \mathbf{x}_j^\ell - \mathbf{x}_i^\ell)\big). \tag{1}$$

Here, H_θ is a densely connected multi-layer perceptron (MLP) parameterized by θ, $\mathcal{N}(i)$ denotes the neighborhood of point i, and max is the element-wise max pooling function.

To capture higher-order dependencies, multiple dynamic graph convolution layers are chained and connected densely within a feature extraction unit [16, 20, 22, 36]:

$$\mathbf{X}^\ell = G_\ell([\mathbf{X}^{\ell-1}, \ldots, \mathbf{X}^1, \mathbf{X}^0]), \tag{2}$$

where $[\ldots]$ denotes the concatenation operation, and $\mathbf{X}^0$ is the input feature.

As depicted above, we adopt dense connections both within and between graph convolution layers. Within graph convolution layers, the MLP H_θ is densely connected — each fully connected (FC) layer's output is passed to all subsequent FC layers. Between graph convolution layers, the features output by each layer are fed to all subsequent layers. Dense connections reduce the number of the network's parameters and produce features with richer contextual information [22, 36].

In addition, we assemble multiple feature extraction units with different k-NN values in parallel to obtain features of different scales, and concatenate them before passing them to downstream networks. The final output is a feature matrix $\mathbf{X} \in \mathbb{R}^{N\times F}$, where N denotes the number of points and F denotes the dimension of features.

3.2.2 Differentiable Pooling Operator. Having extracted multi-scale features from the input point cloud $\mathbf{P}$, we propose a differentiable pooling operator on point clouds to sample a subset of points $\mathbf{S}$ from $\mathbf{P}$ adaptively. Ideally, the operator will learn to identify points that are closer to the underlying surface, which capture the surface structure better and thus will be used to reconstruct the underlying manifold at the decoder. Different from existing pooling techniques that often employ hand-crafted rules such as random sampling and farthest point sampling [25], our differentiable pooling learns the optimal downsampling strategy adaptively during the training process.

Now we formulate the differentiable pooling operator. Given the learned feature $\mathbf{X} \in \mathbb{R}^{N\times F}$ of the input point cloud $\mathbf{P}$ obtained from the feature extraction unit, our pooling operator first computes a score for each point:

$$\mathbf{s} = \mathrm{Score}(\mathbf{X}), \tag{3}$$

where $\mathrm{Score}(\cdot)$ is the score function implemented by an MLP that produces a score vector $\mathbf{s} \in \mathbb{R}^{N\times 1}$. The score function will learn a higher score for points closer to the underlying surface and a lower score for points perturbed with large noise during the end-to-end training process.

Points in $\mathbf{P}$ that have top-M ($M < N$) scores will be retained, while the others will be discarded:

$$\mathbf{i} = \arg\mathrm{top}_M(\mathbf{s}), \tag{4}$$

$$\mathbf{S} = \mathbf{P}[\mathbf{i}], \tag{5}$$

where $\mathbf{i}$ is the index vector of the top-M points and $\mathbf{S} \in \mathbb{R}^{M\times 3}$ is the downsampled point set. In the experiments, we set $M = \frac{N}{2}$ without loss of generality.

To make the score function differentiable so as to be trained by back propagation [8], we deploy the following gate operation on the features of the sampled point set $\mathbf{X}[\mathbf{i}]$ to acquire the features $\mathbf{Y}$ of $\mathbf{S}$:

$$\mathbf{Y} = \mathbf{X}[\mathbf{i}] \odot \mathrm{sigmoid}(\mathbf{s}[\mathbf{i}] \cdot \mathbf{1}^{1\times F}), \tag{6}$$

where $\mathbf{Y} \in \mathbb{R}^{M\times F}$ is the feature matrix of $\mathbf{S}$ after the above gate operation, $\mathbf{X}[\mathbf{i}] \in \mathbb{R}^{M\times F}$ is the feature matrix of $\mathbf{S}$ before the gate operation, $\mathbf{s}[\mathbf{i}] \in \mathbb{R}^{M\times 1}$ is the score vector of the retained points, and $\odot$ denotes element-wise multiplication.

To further reduce the noise variance of the sampled point set $\mathbf{S}$, we perform pre-filtering on $\mathbf{S}$:

$$\hat{\mathbf{S}} = \mathbf{S} + \Delta\mathbf{S}, \tag{7}$$

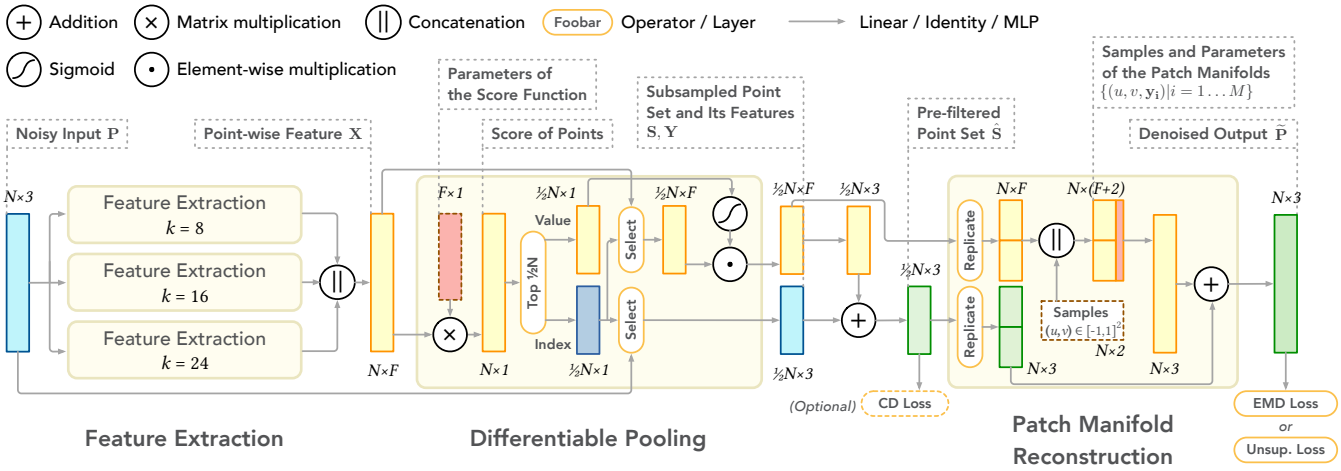

Figure 2: Illustration of the proposed point cloud denoising framework.

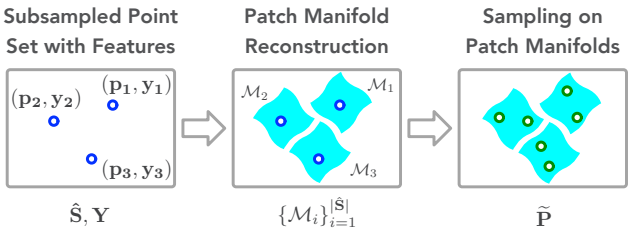

Figure 3: Illustration of the patch manifold reconstruction and resampling. Note that $\widetilde{\mathbf{P}}$ is resampled from the manifolds, so there is no strict point-to-point correspondence between $\hat{\mathbf{S}}$ and $\widetilde{\mathbf{P}}$.

$$\Delta \mathbf{S} = \mathrm{MLP}(\mathbf{Y}), \tag{8}$$

where $\hat{\mathbf{S}}, \Delta\mathbf{S} \in \mathbb{R}^{M\times 3}$, and $\Delta\mathbf{S}$ is produced by an MLP that takes the feature matrix $\mathbf{Y}$ as input. The pre-filtering term $\Delta\mathbf{S}$ moves each point in $\mathbf{S}$ closer to the underlying surface, which will lead to more accurate manifold reconstruction at the decoder to be discussed.

3.3 Manifold Reconstruction Decoder

The manifold reconstruction decoder transforms each point in the pre-filtered low-noise point set $\hat{\mathbf{S}}$ along with its embedded neighborhood feature matrix $\mathbf{Y}$ into a local surface centered around the point—referred to as a *patch manifold*. Afterwards, we upsample $\hat{\mathbf{S}}$ to a denoised point cloud $\widetilde{\mathbf{P}} \in \mathbb{R}^{N\times 3}$ based on the inferred patch manifolds. The whole process is illustrated in Fig. 3.

As discussed in Sec. 3.2, a feature vector $\mathbf{y}_i \in \mathbb{R}^F$ encodes the geometry of the *neighborhood surface* surrounding the point $\mathbf{p}_i \in \hat{\mathbf{S}}$, so that $\mathbf{y}_i$ can be transformed into a manifold that describes the local underlying surface around $\mathbf{p}_i$. We refer to such locally defined manifold as a *patch manifold* around $\mathbf{p}_i$.

Formally, we first define a 2D manifold $\mathcal{M}$ embedded in the 3D space parameterized by some feature vector $\mathbf{y}$ as:

$$\mathcal{M}(u, v; \mathbf{y}) : [-1, 1] \times [-1, 1] \to \mathbb{R}^3, \tag{9}$$

where (u, v) is some point in the 2D rectangular area $[-1, 1]^2$. Eq. (9) maps the 2D rectangle to an arbitrarily shaped patch manifold parameterized by $\mathbf{y}$. Such mapping allows us to draw samples from the arbitrarily shaped patch manifold $\mathcal{M}$ in the following way: we firstly draw samples from the uniform distribution over $[-1, 1]^2$ and then transform them into the 3D space via the mapping.

Having defined a mapping to manifold $\mathcal{M}$, it is natural to define the patch manifold $\mathcal{M}_i$ around each point $\mathbf{p}_i$ in $\hat{\mathbf{S}}$ as:

$$\mathcal{M}_i(u, v; \mathbf{y}_i) = \mathbf{p}_i + \mathcal{M}(u, v; \mathbf{y}_i), \tag{10}$$

which moves the constructed manifold $\mathcal{M}(u, v; \mathbf{y}_i)$ to a local surface centering at $\mathbf{p}_i$.

Now we have a set of patch manifolds $\{\mathcal{M}_i | \mathbf{p}_i \in \hat{\mathbf{S}}\}_{i=1}^{M}$, which characterize the underlying surface of the point cloud. By sampling on these M patch manifolds, we can obtain the denoised point set $\widetilde{\mathbf{P}}$.

Specifically, we assume the number of points in the subsampled point set is the half of that in the input point set, *i.e.*, $\mathrm{M}{=}|\hat{\mathbf{S}}| = \frac{1}{2}|\mathbf{P}|$. In order to acquire a denoised point set $\widetilde{\mathbf{P}}$ that has the same size as the input point set $\mathbf{P}$, we need to sample twice on each patch manifold. Hence, it is essentially an upsampling process.

In practice, the parameterized patch manifold $\mathcal{M}_i(u, v; \mathbf{y}_i)$ is implemented by an MLP:

$$\mathcal{M}_i(u, v; \mathbf{y}_i) = \mathrm{MLP}_{\mathcal{M}}([u, v, \mathbf{y}_i]). \tag{11}$$

We choose the MLP implementation because it is a universal function approximator [19] which is expressive enough to approximate arbitrarily shaped manifolds.

Then, we sample two points from each patch manifold $\mathcal{M}_i([u, v, \mathbf{y}_i])$, leading to a denoised point cloud :

$$\widetilde{\mathbf{P}} = \begin{pmatrix} p_1 + \mathrm{MLP}_{\mathcal{M}}([u_{11}, v_{11}, \mathbf{y}_1]) \\ p_1 + \mathrm{MLP}_{\mathcal{M}}([u_{12}, v_{12}, \mathbf{y}_1]) \\ \vdots \\ p_M + \mathrm{MLP}_{\mathcal{M}}([u_{M1}, v_{M1}, \mathbf{y}_M]) \\ p_M + \mathrm{MLP}_{\mathcal{M}}([u_{M2}, v_{M2}, \mathbf{y}_M]) \end{pmatrix}. \tag{12}$$

To summarize, by learning a parameterized patch manifold $\mathcal{M}(u, v; \mathbf{y}_i)$, $i = 1, \ldots, M = |\hat{\mathbf{S}}|$ from each point i in $\hat{\mathbf{S}}$ and sampling on each patch manifold, we reconstruct a clean point cloud from the noisy input.

3.4 Loss Functions

We present loss functions for supervised training and unsupervised training respectively.

3.4.1 Supervised Training Loss. We consider *dual loss* in the setting of supervised training to measure the quality of both subsampling and final point cloud reconstruction. That is, we have two parts in the supervised loss function, including 1) a loss function $\mathcal{L}_{\text{sample}}$ to quantify the distance between the subsampled and pre-filtered set $\hat{\mathbf{S}}$ and the ground truth point cloud $\mathbf{P}_{\text{gt}}$, which explicitly reduces the noise in $\hat{\mathbf{S}}$ but is not required for the convergence of the training; 2) a loss function $\mathcal{L}_{\text{rec}}$ to quantify the distance between the finally reconstructed point cloud $\widetilde{\mathbf{P}}$ and the ground truth $\mathbf{P}_{\text{gt}}$.

Formally, our network can be trained supervisedly end-to-end by minimizing

$$\min_{\Theta} \mathcal{L}_{\text{sample}} + \mathcal{L}_{\text{rec}}, \tag{13}$$

where Θ denotes the learnable parameters in the network.

We choose the Chamfer distance (CD) [7] as $\mathcal{L}_{\text{sample}}$, since $\hat{\mathbf{S}}$ and $\mathbf{P}_{\text{gt}}$ exhibit different number of points, *i.e.*, $|\hat{\mathbf{S}}| < |\mathbf{P}_{\text{gt}}|$. It is defined as

$$\mathcal{L}_{\text{sample}} = \mathcal{L}_{\text{CD}}(\hat{\mathbf{S}}, \mathbf{P}_{\text{gt}}) = \frac{1}{|\hat{\mathbf{S}}|} \sum_{\mathbf{p} \in \hat{\mathbf{S}}} \min_{\mathbf{q} \in \mathbf{P}_{\text{gt}}} \|\mathbf{p}-\mathbf{q}\|_2^2 + \frac{1}{|\mathbf{P}_{\text{gt}}|} \sum_{\mathbf{q} \in \mathbf{P}_{\text{gt}}} \min_{\mathbf{p} \in \hat{\mathbf{S}}} \|\mathbf{q}-\mathbf{p}\|_2^2. \tag{14}$$

This loss term improves the denoising quality by explicitly optimizing the sampled and pre-filtered set $\hat{\mathbf{S}}$, but is optional for the network training.

We choose the Earth Mover's distance (EMD) [7] as $\mathcal{L}_{\text{rec}}$, which is shown superior to the Chamfer distance in terms of the visual quality [1, 21]. The Earth Mover's distance is defined when two point clouds have the *same* number of points. Fortunately, the denoising task naturally satisfies this requirement. The EMD loss measuring the distance between the denoised point cloud $\widetilde{\mathbf{P}}$ and the ground truth point cloud $\mathbf{P}_{\text{gt}}$ is given by:

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\text{EMD}}\left(\widetilde{\mathbf{P}}, \mathbf{P}_{\text{gt}}\right) = \min_{\varphi: \widetilde{\mathbf{P}} \to \mathbf{P}_{\text{gt}}} \frac{1}{N} \sum_{\mathbf{p} \in \widetilde{\mathbf{P}}} \|\mathbf{p} - \varphi(\mathbf{p})\|_2^2, \tag{15}$$

where $N = |\widetilde{\mathbf{P}}| = |\mathbf{P}_{\text{gt}}|$, and φ is a bijection.

Note that, previous works on denoising [13, 26] often suffer from the clustering effect of points, which is alleviated by introducing a repulsion loss. Our architecture does not suffer from this problem thanks to the one-to-one correspondence of points in $\mathcal{L}_{\text{EMD}}$.

3.4.2 Unsupervised Training Loss. Our network can also be trained in an unsupervised fashion. Leveraging the unsupervised denoising loss in [13], we design an unsupervised loss tailored for our manifold reconstruction based denoising. The key observation is that points with a denser neighborhood are closer to the underlying clean surface, which may be regarded as ground truth points for training a denoiser.

In [13], the unsupervised denoising loss is defined as

$$\mathcal{L}_{\text{U}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{\mathbf{q} \sim P(\mathbf{q}|\mathbf{p}_i)} \|f(\mathbf{p}_i) - \mathbf{q}\|, \tag{16}$$

where $P(\mathbf{q}|\mathbf{p}_i)$ is a prior capturing the probability that a point $\mathbf{q}$ from the noisy point cloud is the underlying clean point of the given $\mathbf{p}_i$ in the noisy point cloud. Empirically, $P(\mathbf{q}|\mathbf{p}_i)$ is defined as $P(\mathbf{q}|\mathbf{p}_i) \propto \exp(-\frac{\|\mathbf{q}-\mathbf{p}_i\|_2^2}{2\sigma^2})$, so that sampling points from the noisy input point cloud $\mathbf{P}$ according to $P(\mathbf{q}|\mathbf{p}_i)$ produces points that are closer to the underlying clean surface with high probability [13]. $f(\cdot)$ represents the denoiser that maps a noisy point $\mathbf{p}_i$ to a denoised point $\mathbf{q}_i$. It is a bijection between the noisy point cloud $\mathbf{P}$ and the output point cloud $\widetilde{\mathbf{P}}$.

This bijection can be naturally established in previous deep-learning based denoising methods such as PointCleanNet [26], Total Denoising [13] *etc.*, because these methods predict the displacement of each point. However, there is no natural one-to-one correspondence between $\mathbf{P}$ and $\widetilde{\mathbf{P}}$ in our method based on patch manifolds. Hence, we seek to construct one by

$$f = \arg \min_{f: \mathbf{P} \to \widetilde{\mathbf{P}}} \sum_{\mathbf{p} \in \mathbf{P}} \|f(\mathbf{p}) - \mathbf{p}\|_2. \tag{17}$$

Having established the bijection f between $\mathbf{P}$ and $\widetilde{\mathbf{P}}$, the unsupervised loss $\mathcal{L}_{\text{U}}$ in Eq. (16) can be computed.

3.5 Analysis

Intuitively, our method can be regarded as a generalization of local-surface-fitting based denoising methods. As discussed in Section 2.1, local-surface-fitting based methods divide point cloud into patches and fit each patch via approximations. The *patch manifold* defined in our method is essentially a local surface surrounding some point in the subsampled point set $\hat{\mathbf{S}}$, which is analogous to patches in local-surface-fitting based methods. Our manifold reconstruction decoder leverages on neural networks to infer the shape of patch manifolds, which is analogous to patch fitting.

Another intuitive interpretation of our method is that, our differentiable pooling layer is analogous to a low-pass filter which removes high-frequency components (*i.e.*, noise), while the manifold reconstruction is similar to high-pass filtering which recovers details from the embedded neighborhood features to avoid over-smoothing.

4 EXPERIMENTAL RESULTS

In this section, we compare our method quantitatively and qualitatively with state-of-the-art denoising methods.

4.1 Experimental Setup

Dataset. For training, we have collected 13 different classes with 7 different meshes, each from ModelNet-40 [33]. We use Poisson disk sampling to sample points from the meshes, at resolution levels ranging from 10K to 50K points. The point clouds are then perturbed by Gaussian noise with standard deviation from 1% to 3% of the bounding box diagonal. Due to the limit of GPU memory, we split the point clouds into patches consisting of 1024 points and feed them into the neural network.

Table 1: Comparison of denoising algorithms. Each resolution and noise level is evaluated by 60 point clouds of different shapes from our collected test dataset, which is a subset of ModelNet-40.

# Points	20K								50K							
Noise	1%		2%		2.5%		3%		1%		2%		2.5%		3%	
10^{-2}	CD	P2S	CD	P2S	CD	P2S	CD	P2S	CD	P2S	CD	P2S	CD	P2S	CD	P2S
Bilateral [17]	1.54	1.27	1.84	1.82	2.11	2.26	2.43	2.78	1.04	0.94	1.61	1.90	1.97	2.49	2.37	3.17
Jet [4]	1.25	0.96	2.11	2.32	2.55	3.04	2.99	3.78	1.11	1.10	2.01	2.61	2.44	3.35	2.86	4.09
MRPCA [23]	**1.13**	**0.72**	2.12	2.18	2.66	3.02	3.16	3.84	1.03	0.91	2.12	2.63	2.58	3.42	3.02	4.18
GLR [39]	1.16	0.88	1.78	1.87	2.20	2.55	2.65	3.30	0.94	0.88	1.79	2.28	2.24	3.05	2.68	3.83
TotalDn [13]	1.51	1.23	2.57	2.97	3.02	3.75	3.46	4.51	1.13	1.03	2.20	2.80	2.66	3.60	3.09	4.37
PCNet [26]	1.45	1.20	2.25	2.41	2.79	3.23	3.37	4.12	0.95	**0.74**	1.41	1.37	2.03	2.19	2.86	3.28
Ours (Supervised)	**1.14**	**0.85**	**1.40**	**1.16**	**1.50**	**1.37**	**1.79**	**1.67**	**0.84**	**0.74**	**1.09**	**1.11**	**1.39**	**1.48**	**1.92**	**2.32**
Ours (Unsupervised)	1.45	1.35	1.82	1.92	2.07	2.22	2.32	2.71	1.14	1.23	1.65	2.14	1.89	2.59	2.22	3.21

Table 2: Comparison of different denoising methods on the point clouds generated by simulated LiDAR scanning with realistic LiDAR noise. LiDAR noise is an unseen noise pattern to our denoiser since we train our denoiser only on Gaussian noise. Results show that our denoiser is *effective in generalizing to unseen noise pattern*, and its generalizability is better than other denoisers.

10^{-2}	Bilat.	Jet	MRPCA	GLR	TotalDn	PCN	Ours
CD	1.23	1.18	1.10	**1.06**	1.25	1.09	**1.06**
P2S	1.13	1.17	0.99	0.99	1.17	0.93	**0.88**

For testing, we have collected 20 classes with 3 meshes each, which are different from the training set. Similarly, we use Poisson disk sampling at resolution levels of 20K and 50K points to generate point clouds and perturb them by Gaussian noise with standard deviation of 1%, 2%, 2.5% and 3% of the bounding box diagonal, leading to 8 classes, each with 60 point clouds.

Furthermore, to examine the generalizability of our method to unseen noise patterns, we also generate 60 noisy point clouds via LiDAR simulators. The simulator we use is the simulation package Blensor [11], which can produce more realistic point clouds and noise. We use Velodyne HDL-64E2 as the scanner model in simulations. Similar to training, we split the point clouds into patches using the K-means algorithm, and feed them separately into the denoiser.

For qualitative evaluation, we additionally use the *Paris-rue-Madame* dataset [29], which is obtained from the real world via laser scanners.

Metrics. We use the Chamfer distance (CD) [7] between the ground truth point cloud $\mathbf{P}_{\text{gt}}$ and the output point cloud $\widetilde{\mathbf{P}}$ as an evaluation metric:

$$C(\mathbf{P}_{\text{gt}}, \widetilde{\mathbf{P}}) = \frac{1}{\left|\widetilde{\mathbf{P}}\right|} \sum_{\mathbf{q}\in\widetilde{\mathbf{P}}} \min_{\mathbf{p}\in\mathbf{P}_{\text{gt}}} \|\mathbf{q}-\mathbf{p}\|_2 + \frac{1}{\left|\mathbf{P}_{\text{gt}}\right|} \sum_{\mathbf{p}\in\mathbf{P}_{\text{gt}}} \min_{\mathbf{q}\in\widetilde{\mathbf{P}}} \|\mathbf{p}-\mathbf{q}\|_2, \quad (18)$$

where the first term measures a distance from each output point to the target surface, and the second term intuitively rewards an even distribution on the target surface of the output point cloud [26]. Note that, in Eq. (18), we use the $\ell 2$ distance, different from the *squared* $\ell 2$ distance used in Eq. (14) which is a term in the loss function. This is because, computing the $\ell 2$ distance involves square root operation, which is not preferable in the loss function due to its numerical instability.

As our method aims to reconstruct the underlying surface, we also use the point-to-surface distance (P2S):

$$\mathcal{P}(\widetilde{\mathbf{P}}, \mathcal{S}) = \frac{1}{|\widetilde{\mathbf{P}}|} \sum_{\mathbf{p}\in\widetilde{\mathbf{P}}} \min_{\mathbf{q}\in\mathcal{S}} \|\mathbf{p}-\mathbf{q}\|_2, \quad (19)$$

where $\mathcal{S}$ is the underlying surface of the ground truth point cloud $\mathbf{P}_{\text{gt}}$.

These two metrics measure the distance between the denoised point cloud and the ground truth one, with smaller values indicating better results.

Iterative denoising. For point clouds at higher noise levels, best possible results are obtained by iterative denoising (*i.e.*, feeding the output of the network as the input again), which is similar to previous neural denoising methods such as PCNet and TotalDn. However, compared to them, our method requires much fewer iterations to get the best possible results. We tune the number of iterations for PCNet, TotalDn and our denoiser, and find that for 1% Gaussian noise, only 1 iteration is required for our denoiser, while 8 iterations are required for PCNet and TotalDn.

4.2 Quantitative Results

We compare both supervised and unsupervised versions of our method quantitatively to state-of-the-art deep-learning based denoising methods as well as non-deep-learning based methods, including PointCleanNet (PCNet) [26], TotalDenoising (TotalDn) [13], bilateral filter [17], Jet fitting [4], MRPCA [23] and GLR [39]. For each resolution and noise level, we compute the Chamfer distances (CD) and the point-to-surface (P2S) distances based on the 60 point clouds.

Table 1 shows that the supervised version of our method significantly outperforms previous deep-learning based methods as well as non-deep-learning denoisers. The unsupervised version is inferior to our supervised counterpart, but still outperforms Total

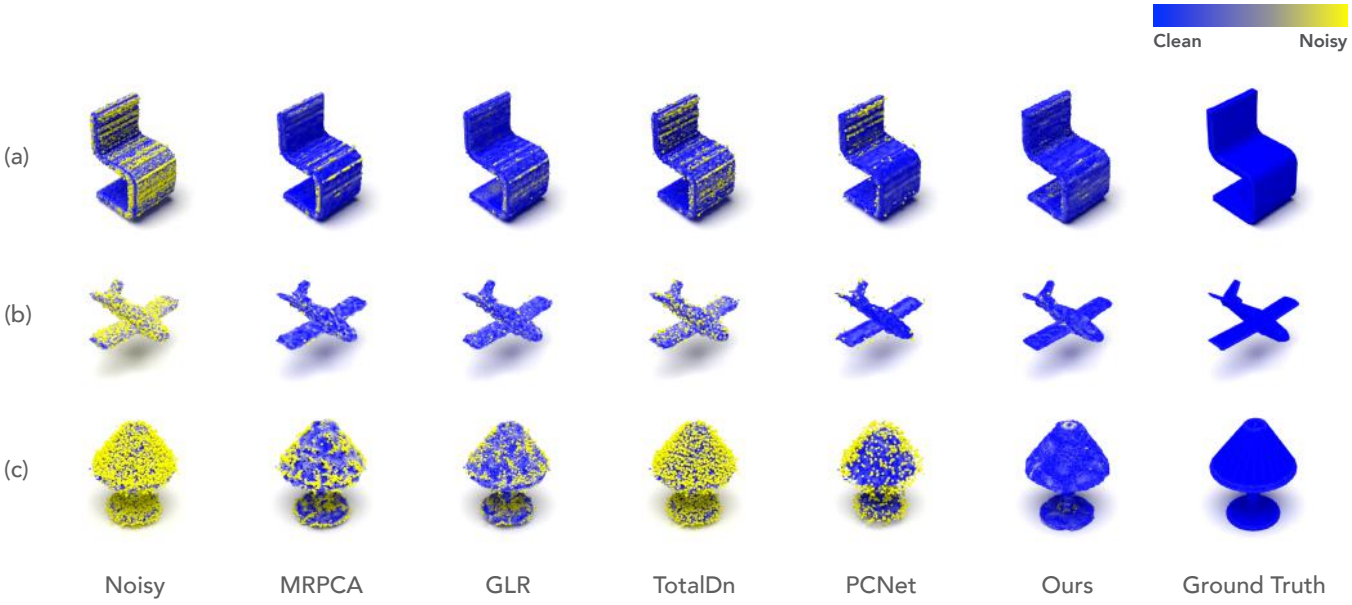

Figure 4: Visual comparison of denoising methods. (a) Simulated scanner noise. (b) 1% Gaussian noise. (c) 2% Gaussian noise.

Denoising, which is also unsupervised, at higher noise levels. Also, the unsupervised version performs better than non-deep learning denoisers at 2%, 2.5% and 3% noise levels. In general, our method outperforms previous denoising methods, and is more robust to high noise levels.

To examine our method's generalizability, we also conduct evaluations on point clouds perturbed by simulated LiDAR noise. Table 2 shows that while our denoiser is trained on Gaussian noise, it is effective in generalizing to the unseen LiDAR noise pattern and performs much better than previous methods.

Discussion on results under differnet metrics. We notice that the superiority of our method is more significant when measured by the point-to-surface distance (P2S), compared to the Chamfer distance (CD), which is essentially a point-to-point distance. This is because our denoiser reconstructs the underlying manifold of the point cloud and resamples on it. Resampling on the manifold does not guarantee that the newly sampled points are close to the points from the original point cloud, which may lead to comparatively larger point-to-point distances. However, the point-to-point distance may not reflect the quality of surface reconstruction well, while the point-to-surface distance generally provides a better measurement as point clouds are representations of 3D surfaces.

Also, [18] finds that the point-to-surface distance is more correlated with subjective evaluation of denoising results. The significant superiority in the point-to-surface distance of our method indicates that our method is more visually preferable than previous methods, which is discussed below.

4.3 Qualitative Results

We demonstrate the comparison of visual denoising results under simulated scanner noise and Gaussian noise with different noise levels in Figure 4. The reconstruction error of each point is measured by the point-to-surface distance. Points with smaller error are colored more blue, and otherwise colored yellow, as indicated in the color bar. The figure shows that our results are much cleaner and exhibit more visually pleasing surfaces than other methods, especially at higher noise levels. Specifically, our method is more robust to outliers compared to the other two deep-learning based methods TotalDn and PCNet. Compared to the two state-of-the-art non-deep-learning methods MRPCA and GLR, our method explicitly reconstructs the geometry of the underlying surface and thus can produce results with lower bias. In summary, the qualitative results in Figure 4 are in line with the quantitative results in Table 1 and 2.

Further, we conduct qualitative studies on the real world dataset *Paris-rue-Madame*. Note that, the ground truth point cloud is unknown, so the error of each point cannot be visualized as the synthesized datasets. As demonstrated in Figure 5, our denoising result is much cleaner and smoother than that of PCNet, while details are well preserved. This validates that our method is effective in generalizing to real world datasets.

In addition, we visualize the intermediate subsampled point set output by the differentiable pooling layer in Fig. 6. The figure reveals that our differentiable pooling layer is effective in sampling points with lower noise, which provides a good initialization for the reconstruction of patch manifolds.

4.4 Ablation Studies

We conduct progressive ablation studies to evaluate the contribution of each component:

(1) **Differentiable pooling.** We replace the differentiable pooling layer with a static pooling layer, which downsamples point clouds by random sampling.

Figure 5: Qualitative results of our denoiser on the real world dataset *Paris-rue-Madame*.

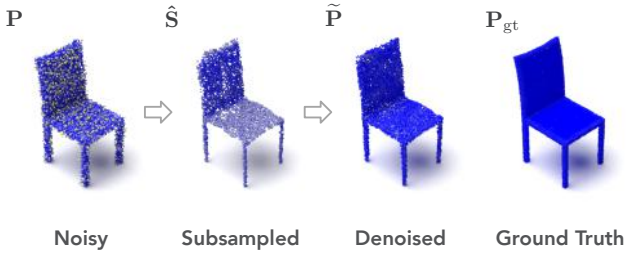

Figure 6: Visualization of the intermediate subsampled point set. Our differentiable pooling operator is effective in sampling points with lower noise.

Table 3: Ablation studies. All the proposed components contribute positively to the performance.

#	Baseline	Diff. Pool	Dual Loss	CD (10^{-2})	P2S (10^{-2})
1	✓			1.41	1.72
2	✓	✓		1.36	1.63
3	✓		✓	1.15	1.22
4	✓	✓	✓	**1.09**	**1.11**

(2) **Dual loss functions.** We remove the Chamfer loss ($\mathcal{L}_{\text{CD}}$) that explicitly measures the quality of pooling (sampling) and pre-filtering, and employ only the EMD loss ($\mathcal{L}_{\text{EMD}}$) for final reconstruction.

The evaluation is based on point clouds of 50K points with 2% Gaussian noise in our test set. As shown in Table 3, all components contribute positively to the full model.

The differentiable pooling enables the denoiser to sample points with lower noise perturbation, relieving the stress of pre-filtering, since the noise level of the input to the pre-filtering layer is lowered.

The dual loss functions explicitly guide the pre-filtering layer to learn to denoise, leading to a more accurate subset of points that characterizes the underlying manifold, eventually improving the quality of manifold reconstruction.

In summary, the above two components boost the performance of manifold reconstruction, resulting in better denoising output.

5 CONCLUSION

In this paper, we propose a novel paradigm of learning the underlying manifold of a noisy point cloud from differentiably subsampled points. We sample points that tend to be closer to the underlying surfaces via an adaptive differentiable pooling operation. Then, we infer patch manifolds by transforming each sampled point along with its embedded neighborhood feature to a local surface. By sampling on each patch manifold, we reconstruct a clean point cloud that captures the intrinsic structure. Our network can be trained end-to-end in either a supervised or unsupervised fashion. Extensive experiments demonstrate the superiority of our method compared to the state-of-the-art methods under both synthetic noise and real-world noise.

REFERENCES

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. 2018. Learning Representations and Generative Models for 3D Point Clouds. In *International Conference on Machine Learning*. 40–49.
[2] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T Silva. 2001. Point set surfaces. In *Proceedings Visualization, 2001. VIS'01.* IEEE, 21–29.
[3] Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. 2010. ℓ1-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics (TOG)* 29, 5 (2010), 1–12.
[4] Frédéric Cazals and Marc Pouget. 2005. Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design* 22, 2 (2005), 121–146.
[5] Chaojing Duan, Siheng Chen, and Jelena Kovacevic. 2018. Weighted multi-projection: 3d point cloud denoising with tangent planes. In *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP).* IEEE, 725–729.
[6] Chaojing Duan, Siheng Chen, and Jelena Kovacevic. 2019. 3D Point Cloud Denoising via Deep Neural Network Based Local Surface Estimation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 8553–8557.
[7] Haoqiang Fan, Hao Su, and Leonidas J Guibas. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 605–613.
[8] Hongyang Gao and Shuiwang Ji. 2019. Graph U-Nets. In *International Conference on Machine Learning*. 2083–2092.
[9] Xiang Gao, Wei Hu, and Zongming Guo. 2018. Graph-Based Point Cloud Denoising. In *2018 IEEE Fourth International Conference on Multimedia Big Data (BigMM).* IEEE, 1–6.

[10] Xiang Gao, Wei Hu, and Guo-Jun Qi. 2020. GraphTER: Unsupervised Learning of Graph Transformation Equivariant Representations via Auto-Encoding Node-wise Transformations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[11] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. 2011. BlenSor: Blender sensor simulation toolbox. In *International Symposium on Visual Computing*. Springer, 199–208.

[12] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. 2018. PCPNet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 75–85.

[13] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. 2019. Total Denoising: Unsupervised Learning of 3D Point Cloud Cleaning. In *Proceedings of the IEEE International Conference on Computer Vision*. 52–60.

[14] Wei Hu, Xiang Gao, Gene Cheung, and Zongming Guo. 2020. Feature graph learning for 3D point cloud denoising. *IEEE Transactions on Signal Processing* 68 (2020), 2841–2856.

[15] Wei Hu, Jiahao Pang, Xianming Liu, Dong Tian, Chia-Wen Lin, and Anthony Vetro. 2020. Graph Signal Processing for Geometric Data and Beyond: Theory and Applications. *arXiv preprint arXiv:2008.01918* (2020).

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[17] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. 2013. Edge-aware point set resampling. *ACM transactions on graphics (TOG)* 32, 1 (2013), 1–12.

[18] Alireza Javaheri, Catarina Brites, Fernando Pereira, and Joao Ascenso. 2017. Subjective and objective quality evaluation of 3D point cloud denoising algorithms. (2017), 1–6.

[19] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks* 6, 6 (1993), 861–867.

[20] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In *Proceedings of the IEEE International Conference on Computer Vision*. 9267–9276.

[21] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. 2019. Morphing and Sampling Network for Dense Point Cloud Completion. *arXiv preprint arXiv:1912.00280* (2019).

[22] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. 2019. DensePoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE International Conference on Computer Vision*. 5239–5248.

[23] Enrico Mattei and Alexey Castrodad. 2017. Point cloud denoising via moving rpca. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 123–137.

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.

[25] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*. 5099–5108.

[26] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. 2020. POINTCLEANNET: Learning to denoise and remove outliers from dense point clouds. In *Computer Graphics Forum*, Vol. 39. Wiley Online Library, 185–203.

[27] Riccardo Roveri, A Cengiz Öztireli, Ioana Pandele, and Markus Gross. 2018. Pointpronets: Consolidation of point clouds with convolutional neural networks. In *Computer Graphics Forum*, Vol. 37. Wiley Online Library, 87–99.

[28] Yann Schoenenberger, Johan Paratte, and Pierre Vandergheynst. 2015. Graph-based denoising for time-varying point clouds. In *2015 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*. IEEE, 1–4.

[29] Andrés Serna, Beatriz Marcotegui, François Goulette, and Jean-Emmanuel Deschaud. 2014. Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods.

[30] Yujing Sun, Scott Schaefer, and Wenping Wang. 2015. Denoising point sets via L0 minimization. *Computer Aided Geometric Design* 35 (2015), 2–15.

[31] Gusi Te, Wei Hu, Amin Zheng, and Zongming Guo. 2018. RGCNN: Regularized graph CNN for point cloud segmentation. In *Proceedings of the 26th ACM International Conference on Multimedia*. 746–754.

[32] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)* 38, 5 (2019), 1–12.

[33] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1912–1920.

[34] Linlin Xu, Ruimin Wang, Juyong Zhang, Zhouwang Yang, Jiansong Deng, Falai Chen, and Ligang Liu. 2015. Survey on sparsity in geometric modeling and processing. *Graphical Models* 82 (2015), 160–180.

[35] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. 2018. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 206–215.

[36] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. 2019. Patch-based progressive 3d point set upsampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5958–5967.

[37] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. Ec-net: an edge-aware point set consolidation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 386–402.

[38] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2790–2799.

[39] Jin Zeng, Gene Cheung, Michael Ng, Jiahao Pang, and Yang Cheng. 2019. 3D Point Cloud Denoising using Graph Laplacian Regularization of a Low Dimensional Manifold Model. *IEEE Transactions on Image Processing* 29 (December 2019), 3474–3489.