Non-Local Part-Aware Point Cloud Denoising

Chao Huang*, Ruihui Li*, Xianzhi Li, and Chi-Wing Fu

The Chinese University of Hong Kong
chaohuang1997@gmail.com {lirh,xzli,cwfu}@cse.cuhk.edu.hk

Abstract. This paper presents a novel non-local part-aware deep neural network to denoise point clouds by exploring the inherent non-local self-similarity in 3D objects and scenes. Different from existing works that explore small local patches, we design the non-local learning unit (NLU) customized with a graph attention module to adaptively capture non-local semantically-related features over the entire point cloud. To enhance the denoising performance, we cascade a series of NLUs to progressively distill the noise features from the noisy inputs. Further, besides the conventional surface reconstruction loss, we formulate a semantic part loss to regularize the predictions towards the relevant parts and enable denoising in a part-aware manner. Lastly, we performed extensive experiments to evaluate our method, both quantitatively and qualitatively, and demonstrate its superiority over the state-of-the-arts on both synthetic and real-scanned noisy inputs.

Keywords: non-local; part-aware; point cloud denoising.

1 Introduction

The ability to remove noise in captured 3D point clouds is an indispensable tool for many applications. With the growing availability of 3D scanning devices, *e.g.*, depth cameras and LiDAR sensors, 3D point clouds are becoming more common today. However, the captured point clouds are often corrupted by noise, so point cloud denoising is essential, and typically required, as a post-processing step.

The goal of point cloud denoising is to recover a clean point set from a noisy input, while preserving the geometric details on the underlying object surface. The main technical challenges to meet the goal are two-fold: (i) how to adapt to various kinds of noise without assuming any prior on the noise distribution, and (ii) how to maintain the data fidelity by retaining the geometric details, *e.g.*, sharp edges and smooth surface, in the denoised point set. Pioneering works, such as locally optimal projection (LOP) [16] and its variants [7,8,18], achieve notable successes by introducing various shape priors on the surface types and noise models to constrain the denoising process. However, these optimization-based methods strongly rely on priors, so they may not generalize well to handle diverse inputs. Hence, these methods tend to oversmooth or oversharpen the results, while removing the noise, particularly when the noise level is high.

* Joint first authors

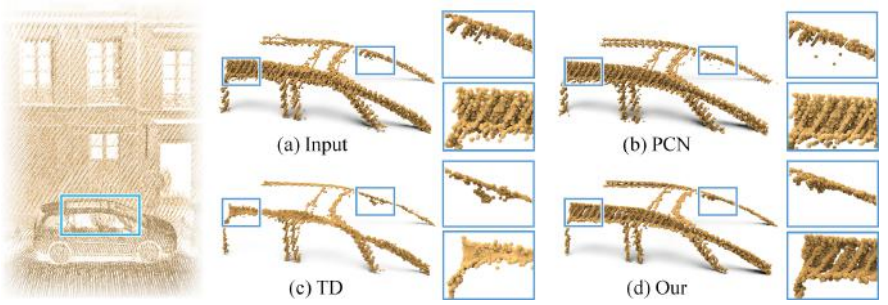

Fig. 1. Denoising (a) a noisy point cloud cropped from the challenging real-scanned Paris-rue-Madame dataset [27] using (b) PointCleanNet (PCN) [24], (c) TotalDenoising (TD) [6], and (d) our method. Compared with others, our method does not produce obvious scattered points away from the object and is more faithful to the input.

Recently, many deep neural networks [22,23,15,38,14,37,28] are designed for analyzing point clouds. Among them, a few recent ones aim at point cloud denoising: PointProNet [25], PointCleanNet [24], TotalDenoising [6], and DSS [33]. These models perform denoising in 2D domain by projecting the noisy inputs into various regular representations, or directly learn a mapping from noisy inputs to noise-free outputs in a small local region. Albeit the improved denoising results, their performance is typically limited by the accuracy of the projection estimation or the local receptive field, so they may move points away from the underlying surface and fail to handle scattered points, particularly in large real scans; see Figures 1 (b) & (c) for results of two very recent methods. Also, these methods are built with general network architectures. So, they may not fully exploit problem-specific knowledge, and their models are *local* without considering the inherent non-local self-similarity in 3D objects and scenes, which has been shown to be very effective for various denoising problems [3,26,10,39,5,41].

In this work, motivated by the observation that point clouds typically consist of self-similar local structures in different and possibly distant locations, we present the first attempt to formulate non-local operations in deep neural network for point cloud denoising. Our non-local part-aware network explores points with *non-local semantically-related neighborhoods* in the input point clouds. Specifically, we devise the *non-local learning unit* (NLU) customized with a new graph attention module, by which we enable a *non-local* receptive field around each input point to effectively consider its *non-local semantically-related features* and their relations over the entire point cloud. To enhance the denoising performance, we cascade multiple NLUs to progressively distill the noise features from the noisy inputs, while implicitly preserving the clean surface. Lastly, to drive the network learning, besides the regular surface reconstruction loss, we further formulate a *semantic part loss* to encourage the predictions to closely locate on the relevant parts and to enable denoising in a *part-aware* manner.

We performed various experiments to evaluate our method using synthetic and real-scanned data, and compared its performance with the state-of-the-arts, both qualitatively and quantitatively; see Figure 1 and Section 4. Experimental results show that our method is robust to handle point sets of various struc-

tures and noises, and facilitates better mesh reconstructions with finer details, as compared with others for both synthetic and real-scanned noisy inputs.

2 Related Work

Optimization-based denoising. Early methods employ various priors to reduce the error of projecting noisy inputs onto the estimated local surface. By assuming a smooth underlying surface, Alexa *et al.* [1,2] introduced the moving least squares and its robust variants [4,20] to preserve the features in the projection. Later, Lipman *et al.* [16] formulated the locally optimal projection (LOP) operator, and Huang *et al.* [7,8], and Lu *et al.* [18] further advanced this technique, *e.g.*, with an anisotropic projection optimization. These pioneering methods, however, rely on fitting local geometry, *e.g.*, normal estimation and smooth surface, so they may not generalize well to diverse inputs and tend to oversmooth or oversharpen the results, while removing the noise.

Deep learning-based denoising. A straightforward approach to adopt deep learning to the problem is to project the noisy inputs onto a 2D regular domain, then perform denoising in 2D using image-based convolutional neural networks (CNN). Roveri *et al.* [25] designed PointProNets, a deep neural network to denoise point sets by converting them into height maps via a learned local frame. Wang *et al.* [33] formulated a differentiable surface splatting method to render noisy inputs into images and reconstruct geometric structures guided by the denoised images using the Pix2Pix [9] framework. Their performance is, however, limited typically by the accuracy of the projection estimation.

Inspired by networks that directly process points [22,23], Rakotosaona *et al.* [24] designed PCN to predict noise-free central points from noisy patches using a point processing network. However, since point features are extracted independently, some denoised points tend to scatter; see Figure 1 (b). Yu *et al.* [34] learned to detect sharp edges in small point patches and preserved edges when consolidating a point cloud; the method, however, considers mainly with small patches and requires manual edge annotations. Zhou *et al.* [40] adopted a statistical and non-differentiable method for removing outliers in 3D adversarial point cloud defense. Very recently, extending Noise2Void [13], Hermosilla *et al.* [6] designed TotalDenoising (TD) to denoise point clouds in an unsupervised manner. The method performs very well for flat and smooth surfaces, but for thin and fine geometric details, the results could be distorted; see Figure 1 (c). Different from existing works that operate locally on individual points or small patches, we design a novel framework to capture non-local semantically-related features over the entire point cloud and progressively denoise it.

3 Method

3.1 Network Architecture Overview

To start, we first introduce the basic notations and elaborate on the overall network architecture. Given a noisy point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$ with N points, each

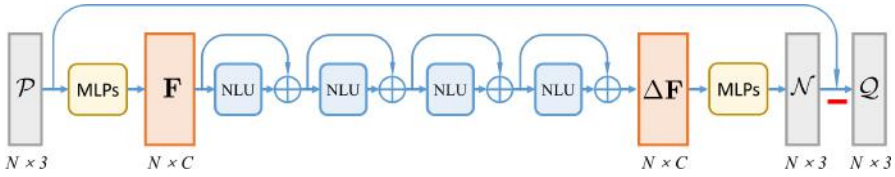

Fig. 2. The overall architecture of our non-local part-aware neural network. N is the number of points in noisy input point cloud $\mathcal{P}$, C is the number of channels in feature $\mathbf{F}$, and NLU denotes our non-local learning unit (see Figure 3). We regress noise component $\mathcal{N}$ and remove it from $\mathcal{P}$ to produce the noisy-free output $\mathcal{Q}$.

represented by 3D Cartesian coordinates, we aim to recover the underlying noise-free point set $\mathcal{Q} \in \mathbb{R}^{N\times 3}$ by regressing the noise component $\mathcal{N} \in \mathbb{R}^{N\times 3}$ of an unknown distribution, then removing $\mathcal{N}$ from input $\mathcal{P}$.

In this work, we formulate a non-local part-aware neural network to regress $\mathcal{N}$ from $\mathcal{P}$. Overall, we have the following considerations in our approach: (i) rather than working on small local patches [24,6], we aim to distill the diverse noise in a *non-local* manner by exploring points with *semantically-similar features* but possibly far apart in space; (ii) a direct shape-wise regularization tends to cause the network to focus mainly on large parts with more points and ignore small parts in shapes that are intrinsically related to the fine geometric details; so, we regularize the denoised points with respect to *semantically-relevant parts* to enable part-aware denoising; and (iii) noise component $\mathcal{N}$ generally has small magnitude and simple patterns, so we extract the latent noise map *progressively* from the noisy inputs, instead of directly estimating the noise-free underlying geometric structures, similar to routines in image denoising [35,36].

Figure 2 shows the overall network architecture. Specifically, we first encode the noisy input into C-dimensional point-wise features $\mathbf{F} \in \mathbb{R}^{N\times C}$, which include the features of both the noise and the underlying clean surface, using a series of multi-layer perceptrons (MLPs). Since we may not always remove noise in one shot, we thus cascade multiple residual blocks, composed by our non-local learning units (NLU) (see Section 3.2) and skip to progressively distill the noise feature map $\Delta\mathbf{F}$ from $\mathbf{F}$. In such process, an intermediate noise feature map $\Delta\mathbf{F}_i$ produced by a block is fed as input to the next block. The initial feature $\mathbf{F}$ will go through all the blocks to produce the final noise feature $\Delta\mathbf{F}$, from which we regress noise displacement $\mathcal{N} \in \mathbb{R}^{N\times 3}$ using another set of shared MLPs. In the end, we subtract noise displacement $\mathcal{N}$ from $\mathcal{P}$ to obtain the denoised point set $\mathcal{Q}$, and formulate a joint loss function on $\mathcal{Q}$ (see Section 3.3) to end-to-end train the whole network. Particularly, we introduce a semantic part regularization in the loss to guide the denoising in a part-aware manner.

3.2 Non-local Learning Unit (NLU)

The purpose of the non-local learning units in the network is to distill the noise feature $\Delta\mathbf{F} \in \mathbb{R}^{N\times C}$ from the input feature $\mathbf{F} \in \mathbb{R}^{N\times C}$ by aggregating non-local

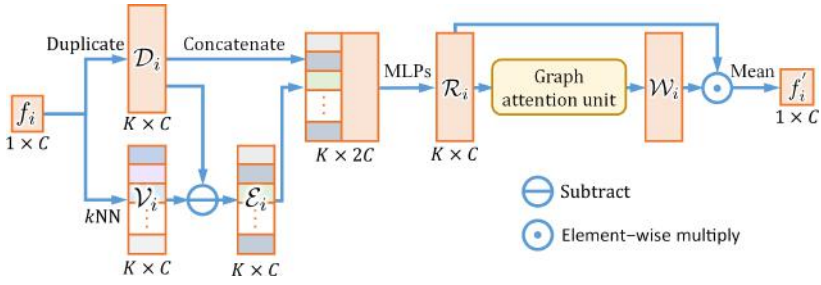

Fig. 3. Illustration of our non-local learning unit (NLU). Note that we illustrate the procedure of NLU by considering the i-th feature vector $\mathbf{f}_i$ (for the i-th point) in $\mathbf{F}$.

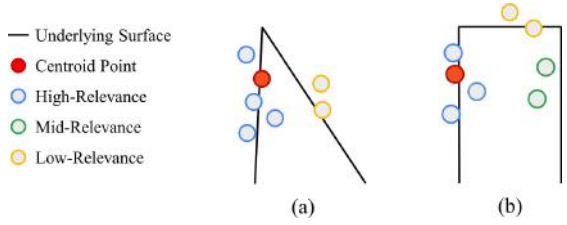

Fig. 4. Degrees of relevance (similarity) between the reference point (red) and others.

semantically similar features. Particularly, to achieve a *more efficient non-local reception field*, we customize a novel graph attention unit inside the NLU. By this unit, we enable adaptive aggregation of the non-local context based on the relevance from each neighboring feature to the central point being considered, rather than simply treating all the non-local similar point features equally.

Figure 3 illustrates the detailed architecture of the NLU. Note that to facilitate the understanding of the processing procedure in NLU, we present its processing procedure and architecture (see again Figure 3) by considering the input feature vector of the i-th point in the input cloud, *i.e.*, $f_i \in \mathbb{R}^{1\times C}$ of $\mathbf{F}$, instead of considering the whole $\mathbf{F}$ altogether. In the NLU, the same processing procedure (as described below) is used for all the feature vectors in $\mathbf{F}$.

Specifically, given f_i in $\mathbf{F}$, we first duplicate f_i K times to obtain a duplicated feature map (see $\mathcal{D}_i$ in Figure 3) of size $K \times C$. On the other hand, we locate f_i's K-most similar point feature vectors by using the KNN search in $\mathbf{F}$ based on the feature similarity, and pack them together into the neighbor feature map $\mathcal{V}_i = \{f_{i1}, f_{i2}, \cdots, f_{iK}\}$ of size $K \times C$; see Figure 3. Then, a subtraction is conducted between $\mathcal{D}_i$ and $\mathcal{V}_i$ to produce a set of edge features $\mathcal{E}_i = \{e_{i1}, e_{i2}, \cdots, e_{iK}\}$, where $e_{ij}{=}f_i{-}f_{ij}$. We concatenate $\mathcal{E}_i$ and $\mathcal{D}_i$, followed by a series of MLPs to obtain the non-local context $\mathcal{R}_i \in \mathbb{R}^{K\times C}$ for f_i. Note that, $\mathcal{R}_i$ encodes the *non-local* similar features of f_i, since the KNN search is based on the feature similarity rather than the spatial distance between points.

Given $\mathcal{R}_i$, we may follow previous works such as [30] to directly use a max-pooling or a mean operation along K to produce the final extracted feature $f'_i \in \mathbb{R}^{1\times C}$. However, such operation equally treats all the non-local similar features, which may include dissimilar features and even outliers. Note, for any $\mathbf{f}_i$,

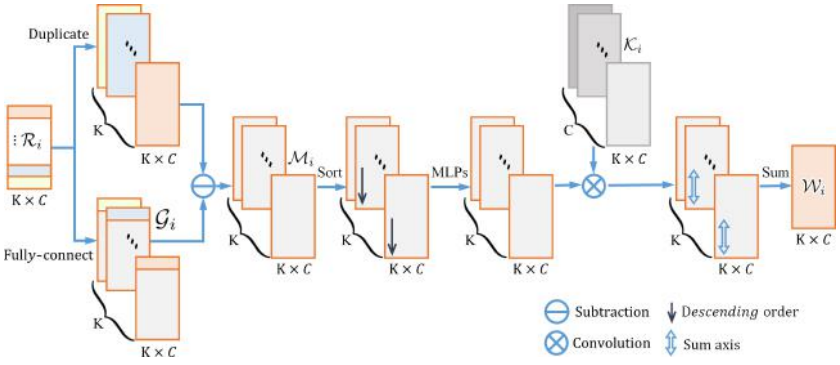

Fig. 5. Illustration of the graph attention unit inside the non-local learning unit (NLU). We first calculate the reference map $\mathcal{M}_i$ from the input local context $\mathcal{R}_i$. Then we enhance it by a sorting operator and MLPs, followed by a convolution with a learnable shared kernel $\mathcal{K}_i$. C is the number of feature channels and K is the number of neighboring features.

we may not always find K semantically-relevant (similar) features in $\mathbf{F}$. Figure 4 gives two illustrations, where the red points represent the reference point being considered and the other points shown in each illustration represent points with features similar (found by KNN) to the red one. We use different colors on these points to indicate their similarity with the red point. Clearly, not all points should contribute equally to the reference point. In this work, we further propose a new graph attention unit to introduce a weight $\mathcal{W}_i$ to each located similar feature to enable an adaptive feature aggregation. As shown in Figure 3, we obtain the attention-guided f_i' by $\text{mean}(\mathcal{W}_i \odot \mathcal{R}_i)$, where $\odot$ denotes a dot product.

Graph attention unit. Figure 5 gives the details of our graph attention unit to regress weights for each located non-local similar feature. Specifically, we first create K copies of each row in $\mathcal{R}_i$ and pack them into a duplicated feature volume of size $K \times K \times C$. On the other hand, for each row (a feature vector) in $\mathcal{R}_i \in \mathbb{R}^{K \times C}$, we treat all the other $K-1$ rows as its neighbor features, and pack them (including the row itself on the top) into one of the planes of $\mathcal{G}_i$ in Figure 5. Hence, $\mathcal{G}_i$ is a fully-connected feature volume of size $K \times K \times C$, which is essentially like a fully-connected graph that interconnects all pairs of features in $\mathcal{R}_i$, and each plane of $\mathcal{G}_i$ in Figure 5 corresponds to a particular row in $\mathcal{R}_i$.

Then, we subtract $\mathcal{G}_i$ from the duplicated feature volume to obtain the reference feature volume $\mathcal{M}_i \in \mathbb{R}^{K \times K \times C}$, as shown in Figure 5. Next, for each $K \times C$ feature plane of $\mathcal{M}_i$ (see Figure 5), we calculate the norm of each feature vector (row) and sort the K rows on each plane in descending order based on the norm value. Hence, we can arrange the more relevant neighbor features on top. After that, we feed the reorganized $\mathcal{M}_i$ into a set of shared MLPs to obtain an enhanced difference map, then perform a convolution between it and a learnable shared kernel $\mathcal{K}_i \in \mathbb{R}^{C \times K \times C}$, followed by a sum aggregation (along the "sum

axis" marked in Figure 5), to produce the final attention weight $\mathcal{W}_i$:

$$\mathcal{W}_i = \text{sum}(\mathcal{K}_i \otimes \mathcal{M}_i), \tag{1}$$

where $\otimes$ is the convolution operation.

In previous attention methods [31,32,17], they directly generalize the self-attention model [29] from machine translation to handle 3D point clouds, where the attention weights are obtained by transforming the features for relation capturing. We design the above attention unit with the goal of highlighting more relevant (or semantically more similar) neighbor features. Hence, the attention weights are obtained based on the convolution between the sorted reference map and a learnable kernel, which is customized for the NLU and critical to improving the denoising performance; see Section 4.4 for a related analysis.

3.3 End-to-end Training

Next, we present the loss function to train our network in an end-to-end fashion. There are two considerations when we design the loss function.

- *Proximity-to-surface* encourages the points in the denoised point clouds to lie on the underlying surface. To achieve this, besides a conventional (i) shape-wise reconstruction loss to globally regularize the outputs, we further formulate a (ii) part-wise reconstruction loss to encourage the predictions towards relevant local parts to enable a part-aware denoising.
- *Distribution regularity* encourages the denoised points to distribute more evenly on the underlying surface, instead of being cluttered together.

Below, we present the three component terms in our loss function:

(i) Shape-wise reconstruction loss. Given our network output $\mathcal{Q}$ and the target point cloud $\hat{\mathcal{Q}}$, we employ the Chamfer Distance (CD) to measure the average closest point distance between them ($\|\cdot\|$ is the $L2$-norm):

$$\begin{aligned}\mathcal{L}_{shape} &= CD(\mathcal{Q}, \hat{\mathcal{Q}}) \\ &= \sum_{q_i \in \mathcal{Q}} \min_{\hat{q}_i \in \hat{\mathcal{Q}}} \|q_i - \hat{q}_i\| + \sum_{\hat{q}_i \in \hat{\mathcal{Q}}} \min_{q_i \in \mathcal{Q}} \|\hat{q}_i - q_i\|.\end{aligned} \tag{2}$$

(ii) Part-wise reconstruction loss. Shape-wise measurement alone may put more emphasis on parts with larger number of points and dismiss the smaller local parts. As a result, denoised points may cluster in larger areas, *e.g.*, a table panel. However, small parts in 3D shapes are usually semantically-significant, since they distinctively represent the fine geometric details in the shapes, *e.g.*, the ears of a cat or a lamp cable. To avoid the issue, we formulate a part-wise measurement to further consider parts in the given shapes.

Specifically, benefited from the part-level annotations provided by [19], we propose to calculate the CD distance (see Eq. (2)) between parts $\mathbf{q}_i \subseteq \mathcal{Q}$ and

$\hat{\mathbf{q}}_i \subseteq \hat{\mathcal{Q}}$ to help recover the fine details in the denoised point cloud:

$$\mathcal{L}_{part} = \sum_{i=1}^{M} CD(\mathbf{q}_i, \hat{\mathbf{q}}_i), \tag{3}$$

where M is the total number of parts in the object (which varies from objects to objects). Note that, we only employ part-level annotations but not the object-level labels from the data set [19]. This part-wise loss helps to emphasize the semantic parts and encourages the network predictions towards the relevant small local parts, enabling the denoising in a part-aware manner. Hence, more geometric details can be preserved in our denoised points; see an experiment in the supplemental material.

(iii) Repulsion loss. Lastly, we adopt the repulsion loss in [34] to encourage the denoised points to move away from one another on the object surface:

$$\mathcal{L}_{repu} = \frac{1}{N \cdot k} \sum_{i=0}^{N} \sum_{i' \in K(i)} \eta \|q_{i'} - q_i\|, \tag{4}$$

where $K(i)$ is the set of indices for the k-nearest neighbors of point $q_i \in \mathcal{Q}$, $\eta(r)=\max(0, h^2-r^2)$ indicates the penalization weights, and $h=0.03$ empirically.

To sum up, our joint loss function is formulated as

$$\mathcal{L} = \lambda_s \mathcal{L}_{shape} + \lambda_p \mathcal{L}_{part} + \lambda_r L_{repu}, \tag{5}$$

where λ_s, λ_p, and λ_r are weights to balance each loss term, and we empirically set them as 0.1, 1.0, and 0.5, respectively.

3.4 Implementation Details

We implemented our network on PyTorch [21] and trained it on a single NVidia Titan Xp GPU for 150 epochs with the Adam optimizer [12]. The learning rate is initialized as 10^{-3} and decays by a factor of 0.2 every 10 epochs until 10^{-6}. In our NLU, we set $K=16$ and $C=32$. Commonly, given a noisy point set with 20,000 points, our inference time is only around 1.5s. Please see the supplemental material for the detailed network configuration. We will *publicly release our code* on GitHub upon the publication of this work.

4 Experiments

4.1 Datasets

Training dataset. We trained our method on the entire 3D point clouds instead of on small local patches, aiming to take advantage of the semantically-similar

local regions in complete 3D shapes for point cloud denoising. Specifically, we used the 3D benchmark dataset PartNet [19], which contains not only a large amount of 3D objects, but also the semantic part annotations. In details, we randomly selected 1,600 models from the PartNet dataset [19] as our training dataset, covering a wide variety of 3D shapes, with simple to complex underlying geometric structures. For each model, we directly used the 10,000 points provided in the dataset as the target noise-free output $\hat{Q}$. We then prepared the noisy inputs by first normalizing $\hat{Q}$ to fit in a unit sphere, then corrupting the points with Gaussian noise to obtain the corresponding noisy input $\mathcal{P}$ on-the-fly. To enrich the training samples and avoid network overfitting, we used three levels of Gaussian noise with a standard deviation of 1.0%, 1.5%, and 2.0% of the bounding box diagonal to corrupt the inputs. In this way, we totally prepared 4,800 training pairs. Also, we employed the provided part annotations for calculating the part-wise reconstruction loss; see Eq. (3). Note that we do not use the object-level labels, since our network does not need to know which class the point set belongs to. Hence, our network can generalize well to other kinds of testing shapes without additional training.

Testing dataset. We employed two kinds of testing datasets in our experiments, *i.e.*, synthetic and real-scanned datasets. For the synthetic dataset, we adopted the 145 models kindly provided by [14], which contain not only smooth structured models, but also highly-detailed models. We uniformly sampled 20,000 points on each mesh surface as the noise-free ground truths. Again, noise of three different levels, *i.e.*, 1.0%, 1.5%, and 2.0%, is employed to corrupt the sampled noise-free points as the testing noisy inputs. Note that, we directly fed the whole noisy inputs into our network to get the denoised outputs. For the real-scanned dataset, we employed the large-scale Parisrue-Madame dataset [27], kindly provided by the authors. This dataset has totally 20 millions of points. We followed the patch-based strategies in [6,24] to first select seeds using farthest sampling, then grow a small patch of 10,000 points per seed. Note that, these patches should cover the whole scene. Then, we fed these patches to the network to produce the denoised outputs and combined them into the final results.

4.2 Comparisons on Synthetic Noisy Data

To evaluate the denoising performance of our method on synthetic noisy data, we compare our method both qualitatively and quantitatively with three state-of-the-arts, including GPF [18], PointCleanNet (PCN) [24], and a recent work TotalDenoising (TD) [6]. For GPF, we used the released executable program to generate the optimal results with carefully fine-tuned parameters. For PCN and TD, we re-trained their networks on the same synthetic dataset as ours.

Quantitative comparisons. We first conducted a quantitative comparison on the synthetic dataset. Since it contains ground truths, we can quantitatively measure the difference between the denoised outputs produced by various methods and the corresponding ground truths. Here, we used four evaluation metrics: (i) Chamfer distance (CD), (ii) Hausdorff distance (HD), (iii) the mean point-to-

Table 1. Comparing our method with GPF [18], PCN [24], and TD [6] on the synthetic testing dataset. The best results are highlighted in bold. The values shown here are averaged over all the testing samples.

Noise level	Metric	GPF	PCN	TD	Our	Improve on TD
1.0%	CD ($\times 10^{-4}$)	3.99	2.81	2.22	**1.72**	22.5%
	HD ($\times 10^{-3}$)	9.76	3.29	2.45	**1.63**	33.5%
	P2F Avg ($\times 10^{-3}$)	10.26	5.48	5.51	**3.77**	31.6%
	P2F Std ($\times 10^{-3}$)	7.84	4.19	4.41	**3.12**	29.3%
1.5%	CD ($\times 10^{-4}$)	4.16	3.20	3.13	**2.31**	26.2%
	HD ($\times 10^{-3}$)	11.67	5.94	3.89	**2.74**	29.6%
	P2F Avg ($\times 10^{-3}$)	11.64	6.41	6.23	**5.70**	8.5%
	P2F Std ($\times 10^{-3}$)	8.65	5.49	5.15	**4.56**	11.5%
2.0%	CD ($\times 10^{-4}$)	6.43	8.92	5.11	**2.65**	48.1%
	HD ($\times 10^{-3}$)	13.71	13.95	6.80	**3.80**	44.1%
	P2F Avg ($\times 10^{-3}$)	12.90	9.04	8.49	**5.91**	31.2%
	P2F Std ($\times 10^{-3}$)	9.69	8.54	8.07	**4.98**	38.3%

surface distance (P2F Avg), and (iv) the corresponding standard variance (P2F Std). The lower the metric values are, the better the denoised results are.

The evaluation results are summarized in Table 1. We tested each method on three levels of noise, *i.e.*, 1.0%, 1.5%, and 2.0%. From the results we can see that, our method clearly outperforms all the other approaches with the lowest values on all the metrics across all the noise levels. See also the last column in Table 1, presenting the amount of improvements achieved by our method over the recent competitive method TD. Particularly, given inputs corrupted by a large noise level, *e.g.*, 2.0%, other methods suffer from an obvious performance drop, while our method can still produce more stable denoising performance.

Qualitative comparisons. Besides quantitative comparisons, we further present the visual comparisons. Figure 6 shows three sets of results produced by different methods (c-f) given three different inputs (a), as well as the ground truths (b) for evaluation; see the CD and HD values below each denoised point clouds. Note that, for each result, we present both the denoised point cloud and the corresponding surface reconstruction result row by row. Here, we employ the Poisson surface reconstruction algorithm [11] to reconstruct the surfaces from the denoised points. From the results, we can see that GPF [18], which requires accurate normal estimation to guide the denoising process, is able to clean the noisy point sets for reconstructing plausible surfaces, but it may inflate the entire shape. PCN [24] can better preserve the geometric details but may generate scattered points out of the surface; see the blown-up views. TD [6] is able to recover noisy point sets and works particularly well for smooth surfaces. However, it may oversmooth the whole surface and take away details in the results, *e.g.*, the nose of the dog (top) and the faces of the other models. Compared with these methods, our method not only effectively removes noise and produces smooth surfaces, but also better preserves the geometric details with the lowest errors compared to the ground truths. More comparison results can be found in the supplemental material.

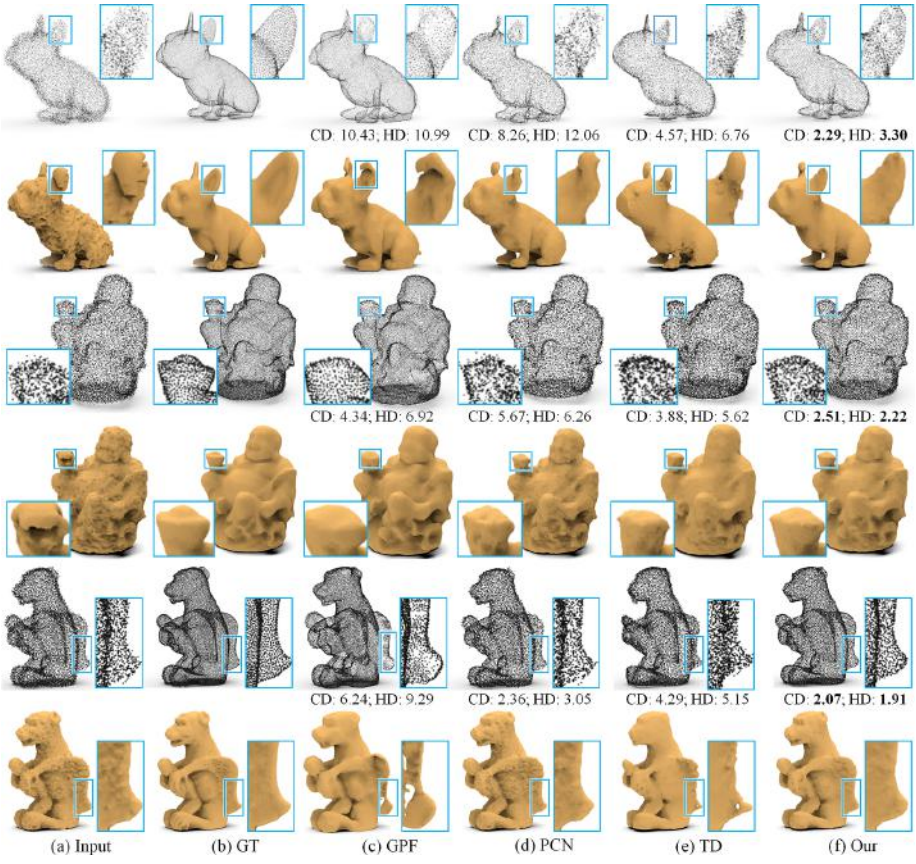

Fig. 6. Comparing the denoised results produced by different methods (c-f) from the noisy inputs (a), where (b) shows the corresponding ground truths (GT). The three noisy inputs carry different noise levels: 2%, 1.5%, and 1% (from top to bottom). The units of CD and HD are 10^{-4} and 10^{-3}, respectively. More results are shown in supp.

4.3 Comparisons on Real-scanned Noisy Data

Next, we evaluate the denoising performance of our method against the recent TD [6] on real-scanned noisy data. The comparison results are shown in Figure 7, where we employed two challenging large-scale scenes in Parisrue-Madame dataset [27] as the testing inputs. In this experiment, TD is trained on the real-scanned dataset (but without supervision), while our network is directly applied to denoise this unseen real-scanned noise pattern, after training on the synthetic dataset with Gaussian noise (see Section 4.1). Clearly, our method can better reveal the object's underlying surface and preserve the sharp edges and details, while TD tends to cluster the points together. Please see the supplementary material for more comparison results with both TD and PCN.

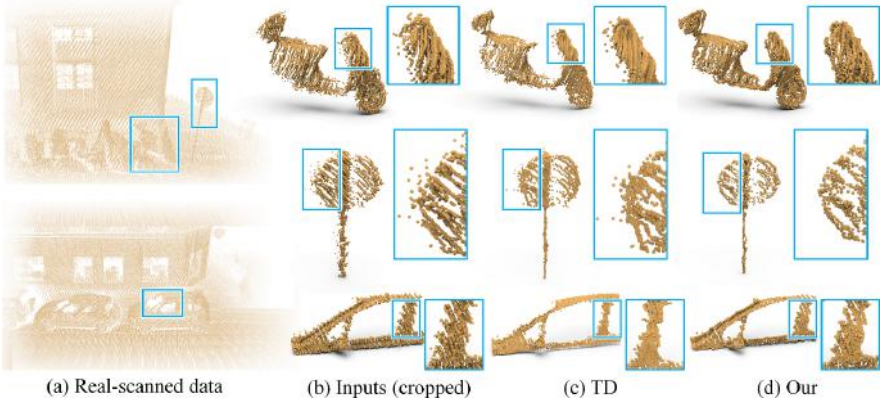

Fig. 7. Denoising (b) noisy patches cropped from (a) the real-scanned Parisrue-Madame dataset [27] by using (c) TotalDenoising (TD) [6] and (d) our method. More comparison results are shown in supplementary material.

Table 2. Comparing the denoising results in terms of CD ($\times 10^{-4}$) between our method and the state-of-the-arts on a synthetic testing dataset with unseen noise levels.

	singe noise level			mixed noise level		
	2.5%	3.0%	3.5%	1.0%&2.0%	1.0%&3.0%	2.0%&3.0%
PCN	7.46	8.59	10.82	6.78	7.47	9.28
TD	6.82	7.06	9.96	6.21	6.99	7.11
our	**4.39**	**5.96**	**8.08**	**1.90**	**2.75**	**4.28**

4.4 Model Analysis

Noise Tolerance Test. To analyze the noise tolerance of our network, we employed our previously-trained network on a synthetic dataset, which is corrupted by different levels of Gaussian noise: 1.0%, 1.5%, and 2.0%, as well as test noisy models with unseen large noise levels and even mixed noise levels. Table 2 reports the comparison results in terms of the CD metric. Note that, the shown values are averaged over all the testing models of the same category (noise level). Clearly, our method consistently outperforms others across all the noise settings. Also, we provide the visual comparisons in Figure 8. Compared with our results (bottom row), other methods tend to retain excessive noise. Please see the supplementary material for more comparison results.

Ablation study. To analyze the contribution of major components in our method, including the shape-wise reconstruction loss, part-wise reconstruction loss, repulsion loss, NLU module and graph attention unit, we conducted an incremental ablation study. Specifically, we first prepared a baseline model (denoted as A) by only keeping the shape-wise reconstruction loss. Based on model A, we then added back the remaining components back one at a time to obtain a new model, which is denoted as B to E, respectively; see Table 3 for the detailed configuration of each model. Naturally, model E is our full pipeline with all the

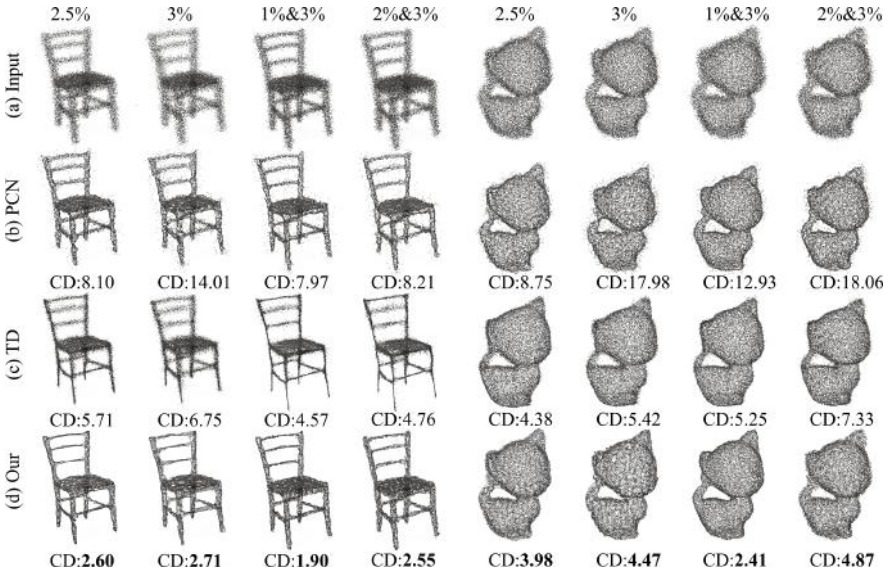

Fig. 8. Comparing the denoising results by applying PCN (b), TD (c), and our method (d) to inputs (a) with unseen noise levels. More comparison results are shown in supp.

Table 3. Ablation study on major components in our method. NLU* represents the NLU without the graph attention unit.

Model	$\mathcal{L}_{shape}$	$\mathcal{L}_{part}$	$\mathcal{L}_{repu}$	NLU*	NLU	CD ($\times 10^{-4}$)
A	✓					2.76
B	✓	✓				2.62
C	✓	✓	✓			2.60
D	✓	✓	✓	✓		2.48
E (full)	✓	✓	✓		✓	**2.31**

major components. We then trained and tested each model on the synthetic dataset with a noise level of 1.5%.

Table 3 summarizes the evaluation results. Note that, for these models without NLU, *i.e.*, models A, B, and C, we employ MLPs for feature extraction. Clearly, model E (our full pipeline) achieves the best denoising performance with the lowest CD value, showing that each component contributes to improve the denoising performance. Particularly, comparing the CD values achieved by model B and model A we can see that, adding the part-wise reconstruction loss back leads to a notable performance improvement. Similarly, adding our NLU back also improves the performance obviously; see the comparisons between model C and model D. The two observations clearly demonstrate the effectiveness of the non-local part-aware context learning approach for point cloud denoising.

Analysis on the number of NLUs. In our default settings, we cascade four NLUs; see Figure 2. To explore the effect of using different number of NLUs, we further re-implemented our network with three and five NLUs, respectively,

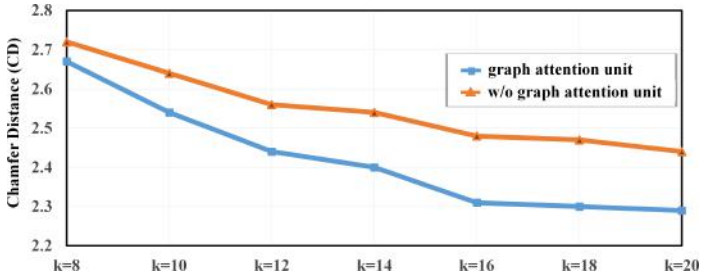

Fig. 9. Denoised performance by increasing the hyper-parameter K in our NLU with or without the graph attention unit. Note that the unit of CD is 10^{-4}.

and trained each network on the same synthetic dataset with a noise level of 1.5%. The testing results show that, the CD values ($\times 10^{-4}$) of using three, four (default), and five NLUs are 2.40, 2.31, and 2.30, respectively. Although the performance of using five NLUs is slightly better than our default setting, the network has 55.7K parameters as compared to 44.6K parameters with four NLUs. Hence, we choose to use four NLUs to balance the performance and efficiency.

Analysis on the hyper-parameter K in NLU. Generally, a smaller K in NLU means that we take fewer neighboring features into the local context, resulting in a smaller receptive field. However, a larger K may involve some unrelated features of different underlying geometric structures. Our graph attention unit is to learn the weights among the neighbor point features, therefore enabling our method to attend to more relevant features. Hence, our method benefits more from a larger K and achieves a better denoising performance. To verify this, we test the denoising performance for different choices of K with/without our graph attention unit, and the results are summarized in Figure 9. Clearly, when we remove the graph attention unit, the CD values are much higher across all the K values; see the orange plot vs. blue plot. Further, as shown in the blue plot, the denoising performance keeps improving as K increases, and finally converges at around K=20. In our work, we choose K=16 by considering both the denoising performance and computational efficiency.

5 Conclusion

In this paper, we presented a non-local part-aware deep neural network to progressively denoise point clouds. Different from existing works that perform denoising in small local regions, we consider the inherent non-local self-similarity in 3D objects and scenes, by designing the NLU customized with a graph attention module to explore semantically-relevant features in point clouds non-locally. Also, to preserve more geometric details in the denoised point clouds, besides the regular surface reconstruction loss, we empower our network with part-awareness by formulating a semantic part loss to encourage the predictions to closely locate on the relevant parts. Extensive experiments demonstrated the effectiveness of

our method, showing that it outperforms the state-of-the-arts in various configurations, from synthetic data set to large real-scanned point clouds.

Since our method is trained in a supervised manner, our approach typically requires the provision of ground truth information in the training. In the future, we plan to explore a weakly-supervised framework to simultaneously learn from both the synthetic and real-scanned inputs, by means of various domain adaptation strategies. We will also consider the potential of multi-task learning, by performing denoising with upsampling together, to let the network learn to produce dense and cleaned outputs, to address the issues of both sparsity and noise, which are particularly common in real-scanned data.

References

1. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Point set surfaces. In: IEEE Visualization. pp. 21–28 (2001)
2. Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., Silva, C.T.: Computing and rendering point set surfaces. IEEE Trans. Vis. & Comp. Graphics. **9**(1), 3–15 (2003)
3. Buades, A., Coll, B., Morel, J.M.: A non-local algorithm for image denoising. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 60–65 (2005)
4. Fleishman, S., Cohen-Or, D., Silva, C.T.: Robust moving least-squares fitting with sharp features. ACM Trans. on Graphics. **24**(3), 544–552 (2005)
5. Gu, S., Zhang, L., Zuo, W., Feng, X.: Weighted nuclear norm minimization with application to image denoising. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) pp. 2862–2869 (2014)
6. Hermosilla, P., Ritschel, T., Ropinski, T.: Total Denoising: Unsupervised learning of 3D point cloud cleaning. In: IEEE Int. Conf. on Computer Vision (ICCV). pp. 52–60 (2019)
7. Huang, H., Li, D., Zhang, H., Ascher, U., Cohen-Or, D.: Consolidation of unorganized point clouds for surface reconstruction. ACM Trans. on Graphics (SIGGRAPH Asia). **28**(5), 176:1–7 (2009)
8. Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U., Zhang, H.: Edge-aware point set resampling. ACM Trans. on Graphics. **32**(1), 9:1–12 (2013)
9. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 1125–1134 (2017)
10. Ji, H., Liu, C., Shen, Z., Xu, Y.: Robust video denoising using low rank matrix completion. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) pp. 1791–1798 (2010)
11. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Eurographics Symposium on Geometry Processing (SGP). vol. 7 (2006)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Krull, A., Buchholz, T.O., Jug, F.: Noise2Void-learning denoising from single noisy images. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 2129–2137 (2019)

14. Li, R., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: PU-GAN: a point cloud upsampling adversarial network. In: IEEE Int. Conf. on Computer Vision (ICCV). pp. 7203–7212 (2019)
15. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: PointCNN: Convolution on $\mathcal{X}$-transformed points. In: Conference and Workshop on Neural Information Processing Systems (NeurIPS). pp. 828–838 (2018)
16. Lipman, Y., Cohen-Or, D., Levin, D., Tal-Ezer, H.: Parameterization-free projection for geometry reconstruction. ACM Trans. on Graphics (SIGGRAPH). **26**(3), 22:1–5 (2007)
17. Liu, X., Han, Z., Wen, X., Liu, Y.S., Zwicker, M.: L2G auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention. In: ACM Int. Conf. on Multimedia (ACM MM). pp. 989–997 (2019)
18. Lu, X., Wu, S., Chen, H., Yeung, S.K., Chen, W., Zwicker, M.: GPF: GMM-inspired feature-preserving point set filtering. IEEE Trans. Vis. & Comp. Graphics. **24**(8), 2315–2326 (2017)
19. Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 909–918 (2019)
20. Öztireli, A.C., Guennebaud, G., Gross, M.: Feature preserving point set surfaces based on non-linear kernel regression. Computer Graphics Forum. **28**(2), 493–501 (2009)
21. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. In: NeurIPS Workshop (2017)
22. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: Deep learning on point sets for 3D classification and segmentation. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 652–660 (2017)
23. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Conference and Workshop on Neural Information Processing Systems (NeurIPS). pp. 5099–5108 (2017)
24. Rakotosaona, M.J., La Barbera, V., Guerrero, P., Mitra, N.J., Ovsjanikov, M.: PointCleanNet: Learning to denoise and remove outliers from dense point clouds. Computer Graphics Forum. (2019), to appear
25. Roveri, R., Öztireli, A.C., Pandele, I., Gross, M.: PointProNets: Consolidation of point clouds with convolutional neural networks. Computer Graphics Forum. **37**(2), 87–99 (2018)
26. Schall, O., Belyaev, A., Seidel, H.P.: Feature-preserving non-local denoising of static and time-varying range data. In: ACM Symposium on Solid and Physical Modeling. pp. 217–222 (2007)
27. Serna, A., Marcotegui, B., Goulette, F., Deschaud, J.E.: Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In: ICPRAM (2014)
28. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: KPConv: Flexible and deformable convolution for point clouds. In: IEEE Int. Conf. on Computer Vision (ICCV). pp. 6411–6420 (2019)
29. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Conference and Workshop on Neural Information Processing Systems (NeurIPS). pp. 5998–6008 (2017)

30. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Trans. on Graphics. **38**(5), 146:1–12 (2019)
31. Xie, S., Liu, S., Chen, Z., Tu, Z.: Attentional shapecontextnet for point cloud recognition. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 4606–4615 (2018)
32. Yang, J., Zhang, Q., Ni, B., Li, L., Liu, J., Zhou, M., Tian, Q.: Modeling point clouds with self-attention and gumbel subset sampling. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 3323–3332 (2019)
33. Yifan, W., Serena, F., Wu, S., Öztireli, A.C., Sorkine-Hornung, O.: Differentiable surface splatting for point-based geometry processing. ACM Trans. on Graphics (SIGGRAPH Asia). **38**(6), 230:1–14 (2019)
34. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: EC-Net: An edge-aware point set consolidation network. In: European Conf. on Computer Vision (ECCV). pp. 386–402 (2018)
35. Zhang, K., Zuo, W., Chen, Y., Meng, D., Zhang, L.: Beyond a Gaussian Denoiser: Residual learning of deep CNN for image denoising. IEEE Trans. Image Proc (TIP). **26**(7), 3142–3155 (2017)
36. Zhang, K., Zuo, W., Gu, S., Zhang, L.: Learning deep CNN denoiser prior for image restoration. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 3929–3938 (2017)
37. Zhang, Z., Hua, B.S., Yeung, S.K.: ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics. In: IEEE Int. Conf. on Computer Vision (ICCV). pp. 1607–1616 (2019)
38. Zhao, H., Jiang, L., Fu, C.W., Jia, J.: PointWeb: Enhancing local neighborhood features for point cloud processing. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 5565–5573 (2019)
39. Zheng, Q., Sharf, A., Wan, G., Li, Y., Mitra, N.J., Cohen-Or, D., Chen, B.: Non-local scan consolidation for 3D urban scenes. ACM Trans. on Graphics (SIGGRAPH). **29**(4), 94:1–9 (2010)
40. Zhou, H., Chen, K., Zhang, W., Fang, H., Zhou, W., Yu, N.: DUP-Net: Denoiser and upsampler network for 3D adversarial point clouds defense. In: IEEE Int. Conf. on Computer Vision (ICCV). pp. 1961–1970 (2019)
41. Zhu, L., Fu, C.W., Brown, M.S., Heng, P.A.: A non-local low-rank framework for ultrasound speckle reduction. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 5650–5658 (2017)