International Journal of Computer Vision
https://doi.org/10.1007/s11263-021-01564-7

RePCD-Net: Feature-Aware Recurrent Point Cloud Denoising Network

Honghua Chen[1] · Zeyong Wei[1] · Xianzhi Li[3] · Yabin Xu[1] · Mingqiang Wei[1,2] · Jun Wang[1]

Received: 18 February 2021 / Accepted: 25 November 2021

Abstract
The captured 3D point clouds by depth cameras and 3D scanners are often corrupted by noise, so point cloud denoising is typically required for downstream applications. We observe that: (i) the scale of the local neighborhood has a significant effect on the denoising performance against different noise levels, point intensities, as well as various kinds of local details; (ii) non-iteratively evolving a noisy input to its noise-free version is non-trivial; (iii) both traditional geometric methods and learning-based methods often lose geometric features with denoising iterations, and (iv) most objects can be regarded as piece-wise smooth surfaces with a small number of features. Motivated by these observations, we propose a novel and task-specific point cloud denoising network, named RePCD-Net, which consists of four key modules: (i) a recurrent network architecture to effectively remove noise; (ii) an RNN-based multi-scale feature aggregation module to extract adaptive features in different denoising stage; (iii) a recurrent propagation layer to enhance the geometric feature perception across stages; and (iv) a feature-aware CD loss to regularize the predictions towards multi-scale geometric details. Extensive qualitative and quantitative evaluations demonstrate the effectiveness and superiority of our method over state-of-the-arts, in terms of noise removal and feature preservation.

Keywords Point cloud · 3D deep learning · RNN · Multi-scale feature learning · Geometric feature preservation

1 Introduction

Depth cameras and 3D optical and laser scanners have been widely used to digitize real-world objects, which are initially represented by point clouds. A variety of advanced technologies currently benefit from the acquired point clouds, such as 3D inspection (Wang et al. 2020), robotics (Guo et al. 2014), remote sensing (Kong et al. 2013), autonomous driving, and smart manufacturing (Xie et al. 2020). Regrettably, noise inevitably creeps in, during both the capture and 3D reconstruction procedures. This makes point cloud denoising an essential pre-processing step for downstream applications.

The problem of point cloud denoising has been extensively explored, yet not well-solved, since evolving a noisy point cloud to its unknown noise-free counterpart is an ill-posed problem. Existing techniques can be roughly divided into two categories: traditional geometric-based methods and learning-based methods. Traditional ones usually design a filter to iteratively remove noise, or build an optimization model based on specific priors, e.g., noise model (Lu et al. 2017), feature sparsity (Sun et al. 2015), and non-local self-similarity (Chen et al. 2019). However, a noisy point cloud may contain unknown type of noise, which depends on the actual point cloud acquisition mechanism (Hu et al. 2020), and the irregular surface structure, which consists of different scales of geometric features. Hence, making use of a specific filter or a prior assumption to remove noise of measurement surface may not always produce satisfactory results. Also,

Communicated by Zuzana Kukelova.

This work was supported in part by the National Key Research and Development Program of China (No. 2020YFB2010702, No. 2019YFB1707504), National Natural Science Foundation of China (No. 61772267, No. 62172218), Natural Science Foundation of Jiangsu Province (No. BK20190016), and the Free Exploration of Basic Research Project, Local Science and Technology Development Fund Guided by the Central Government of China (No. 2021Szvup060). The corresponding author for this paper is Jun Wang.

✉ Jun Wang
wjun@nuaa.edu.cn

Honghua Chen
chenhonghuacn@gmail.com

[1] Nanjing University of Aeronautics and Astronautics, Nanjing, China

[2] Shenzhen Research Institute, Shenzhen, China

[3] Huazhong University of Science and Technology, Wuhan, China

Published online: 17 January 2022

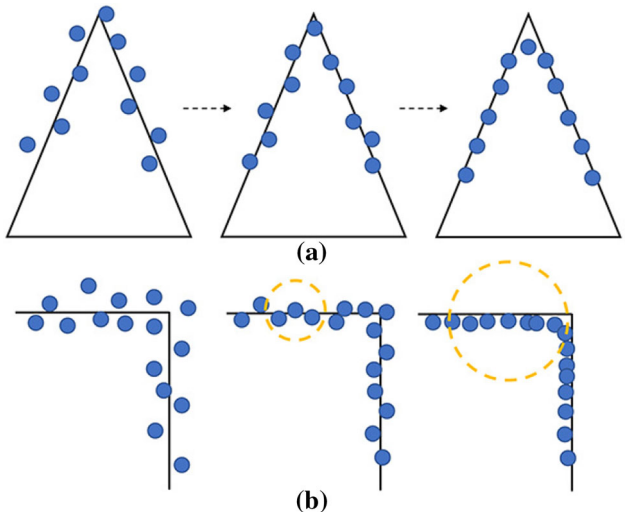

Fig. 1 Toy examples of common point cloud denoising problem. **a** A shape of sharp geometric feature is over-smoothed with iterations. **b** When the input contains strong noise, a small-scale neighborhood (light yellow circle) may cause noise residual, while a large-scale neighborhood leads to shape collapse and geometric feature over-smoothing

traditional methods always heavily rely on parameters tuning (Fleishman et al. 2005; Lipman et al. 2007).

More recently, to resolve the above limitations, learning-based methods have attracted many researchers' attention (Roveri et al. 2018; Yu et al. 2018a; Hermosilla et al. 2019; Rakotosaona et al. 2020; Huang et al. 2020). These learning-based methods attempt to regress a general function that directly maps noisy inputs to the ground truths. Albeit the improved denoising results, their performance is limited by the ability of feature representation for distinguishing the noise and the geometric features.

Moreover, we have observed from a large number of experimental results that: (1) the chosen scale of the neighborhood around a point for denoising itself, introduces an unavoidable trade-off between noise removal and preservation of fine geometric details (see from Fig. 1b). A large neighborhood over-smoothes sharp features and small details but can effectively eliminate heavy noise. A small neighborhood, on the other hand, may reproduce small geometric features but is more sensitive to noise; (2) it is easy to retain excessive noise or bring in extra artifacts by using an optimization model or a single-stage network without iterations; (3) noise and geometric details are commonly removed simultaneously during iterative execution, due to the fact that the magnitudes of geometric details are usually similar to that of noise (see from Fig. 1a); (4) existing deep learning-based methods seek to minimize the overall loss, while ignoring the geometric features which are sparse but meaningful on the 3D shapes. The main characteristics of the existing techniques and our method are concluded in Table 1.

Based on the above observations, in this work, we propose a feature-aware recurrent point cloud denoising network (i.e., RePCD-Net). Our network first builds multi-scale neighborhoods for each point, followed by using a PointNet-like network to extract features for each neighborhood. We then feed the extracted multi-scale features of each point to a bi-directional RNN (BRNN) module, for producing an adaptive feature for the current denoise task (contributing more on denoising, or recovering feature, or both). To avoid the geometric feature over-smoothing problem, we embed a recurrent propagation layer to exploit the deep features across denoising stages, so that the detailed geometry information can be still retained or recovered during iterations. Furthermore, we design a feature-aware chamfer distance (CD) loss to drive the predictions to closely locate on the geometric feature regions and to enable denoising in a feature-aware manner. We performed various experiments to evaluate our method using both synthetic and real-scanned data. Extensive experiments and comparisons reported in Sect. 5 demonstrate that our method outperforms the state-of-the-art methods, in terms of visual quality and quantitative accuracy. The main contributions of our work are as follows:

- We design a novel feature-aware recurrent point cloud denoising network, which is able to effectively remove noise, while well preserving various geometric features.
- To learn a more representative feature for each point, we introduce a bi-directional RNN based multi-scale feature aggregation module, which is capable of extracting adaptive features for different denoising stages.
- We introduce a recurrent propagation layer, which is used to exploit the deep features across denoising recursion stages, for recovering smoothed fine geometric features.
- We explicitly incorporate the feature smoothness of each point into the loss function, to further enable the denoised point cloud to be faithfully located on the underlying surface, while preserving more geometric features.

2 Related Work

We will review previous researches from traditional geometric methods to recent prevalent learning-based techniques. Geometric denoising approaches can be further classified as: projection-based, sparsity-based, and non-local methods.

2.1 Geometric Methods

Projection-Based Methods This kind of approaches assume a smooth underlying surface, and project the noisy data to the estimated local surface. The projection schemes consist of two main categories: the moving least squares based methods (Alexa et al. 2001, 2003; Fleishman et al. 2005; Öztireli

Table 1 Comparison of the main characteristics between the state-of-the-art point cloud denoising techniques and our method, in terms of number of iterations, neighborhood scale, the sensitivity to scale and the ability of feature preservation

Characteristics	Iterations	Neighborhood scale	Scale-sensitive	Feature-aware
Methods				
RIMLS (Öztireli et al. 2009)	8–15	Single-scale	Yes	Yes
GPF (Lu et al. 2017)	15–20	Single-scale	Yes	Yes
WLOP (Huang et al. 2009)	15–30	Single-scale	Yes	No
ECN (Yu et al. 2018a)	1	Multi-scale	No	Yes
PCN (Rakotosaona et al. 2020)	3	Single-scale	Yes	No
GPD (Pistilli et al. 2020)	1	Single-scale	Yes	No
TD (Hermosilla et al. 2019)	3–10	Two-scale	No	No
Ours	1	Multi-scale	No	Yes

et al. 2009) and the locally optimal projection based operator (Lipman et al. 2007; Huang et al. 2009, 2013; Preiner et al. 2014; Lu et al. 2017). However, these methods rely on fitting local geometry, e.g., normal estimation and smooth surface, so they may not generalize well to diverse inputs and tend to over-smooth or over-sharpen the results.

Sparsity-Based Methods This class of methods are based on the observation that common objects can be defined as piecewise smooth surfaces with sparse features. L_0 minimization (Sun et al. 2015), L_1-norm (Avron et al. 2010), and sparse dictionary learning (Digne et al. 2017) are common techniques to be applied for denoising task. From another point of view, Remil et al. (2017) explored the sparsity of local shapes in the library of 3D objects. This method was later successfully generalized to recover the clean surface of 3D objects with complex structures.

Non-local Methods Non-local based approaches (Rosman et al. 2013; Lu et al. 2018; Chen et al. 2019) are inspired by the geometric statistics which indicate that a number of surface patches sharing approximate geometric properties always exist within a 3D model. So, these methods collect multiple neighborhoods with similar geometry to collaboratively denoise a local structure. The main challenges in this kind of methods are the definition of the similarity metric and the regular representation of local structures.

2.2 Learning-Based Methods

Recently, learning-based methods gradually show its power for point cloud noise removal. Roveri et al. (2018) introduced PointProNets, a fully differentiable, CNN-based deep learning architecture to process point clouds. An application of point cloud desnoising demonstrates the effectiveness of this network. Wang et al. (2019) formulated a differentiable surface splatting method to render noisy inputs into images and reconstruct geometric structures guided by the denoised images. Inspired by networks that directly process points Qi et al. (2017a, b) and Rakotosaona et al. (2020) designed the PointCleanNet (PCN) to first remove outliers and then predict noise-free central points from noisy patches using a point processing network. Yu et al. (2018a) presented an edge-aware point cloud consolidation network, which employs ground-truth edge points to obtain edge-aware predictions. Although this method is robust to neighborhood scale and is able to preserve prominent edges (see from Table 1), it cannot well handle surfaces with tiny features. Different from Yu et al. (2018a) and Huang et al. (2020) designed a denoising network to progressively denoise point clouds, which emphasizes smaller but meaningful local parts. Zhou et al. (2019) adopted a statistical and non-differentiable method for removing outliers in 3D adversarial point cloud defense. Lu et al. (2020) proposed to first learn the ground-truth point normals, followed by the point position updating. Very recently, Hermosilla et al. (2019) designed the Total Denoising (TD) to denoise point clouds in an unsupervised manner. The method is able to completely eliminate noises. Pistilli et al. (2020) presented a graph-convolutional point cloud denoising neural network (GPD). Luo and Hu (2020) presented a novel paradigm of learning the underlying manifold of a noisy point cloud from differentiably sub-sampled points. This network can also be trained in an unsupervised fashion. The proposed learning network is different from existing ones, which is more like to mimic a human being cleaning a noisy point cloud, with supervisions from ground truths, as well as multi-scale geometric feature information extracted from the training data. In addition, among all the above approaches, PCN (Rakotosaona et al. 2020) is the most similar one. However, our method has at least three main differences: (1) PointCleanNet learns the point residual from a single patch of a fixed scale, which leads to the produced results being sensitive to the noise intensity and surface details. Our method benefits from multi-scale deep features extraction and BRNN-based feature fusion module, and thus it can better remove surface noise and preserve geometric details. (2) To handle different noise magnitudes, PointCleanNet enables iterative denoising, yet without feature information transfer

across different denoising stages. Our network is in a recurrent structure. Besides, the main network modules (MLP, RNN with Attention, BRNN, and FCN) share the parameters during recursion, which leads to learn more adaptive and more robust features. We also propose to embed a recurrent feature propagation layer in our network to further exploit the deep features across denoising stages. (3) Lastly, our method utilizes a feature-aware loss function, which can enhance the effect of multi-scale geometric feature preservation.

3 Method

Given a noisy 3D point cloud $\mathbf{P}$ with only point coordinates, the goal of point cloud denoising is to obtain its clean version $\hat{\mathbf{P}}$, while maintaining fine details. We start this section by first presenting how we model the point cloud denoising problem in Sect. 3.1. Then, in Sect. 3.2, we describe the architecture of our network. Lastly, we introduce our end-to-end joint loss function in Sect. 3.3.

3.1 Denoising Model

There exist a wide range of point cloud acquisition devices from consumer-level depth sensors (e.g., Kinect) to high-end outdoor scanners (e.g., Leica ScanStation P20). By using these 3D devices for data acquisition, both the type and scale of real-world noise that creep into point clouds are unknown, and hence modeling real noise is ill-posed. Here, we use a very intuitive but commonly-used way to formulate the noise removal model:

$$\mathbf{P} = \hat{\mathbf{P}} + \varepsilon, \tag{1}$$

By eliminating the noise ε from the noisy input $\mathbf{P}$, we can obtain the noise-free version $\hat{\mathbf{P}}$. Inspired by Rakotosaona et al. (2020) and Li et al. (2020), we consider learning the residual ε, rather than directly learning $\hat{\mathbf{P}}$, since the underlying noise-free surface is usually more complicated compared with the noise patterns. However, directly learning the noise ε by a single-stage network is non-trivial. Iteration operation is often required for effectively reducing noise, whether the employed technique is a traditional method or a learning-based one. We hence re-model the denoising process in a recurrent way. Besides, the deep features from previous stages are also propagated to the current stage, for keeping geometric details. The final model is as follows:

$$\begin{aligned} \hat{\mathbf{P}}^1 &= \mathbf{P}, \\ \varepsilon^i &= f^i(\hat{\mathbf{P}}^i, \mathbf{F}^1, \ldots, \mathbf{F}^{i-1}), \\ \hat{\mathbf{P}}^{i+1} &= \hat{\mathbf{P}}^i + \varepsilon^i \end{aligned} \tag{2}$$

where $\mathbf{F}^i$ indicates the features from the i-th stage, and $\hat{\mathbf{P}}^{i+1}$ is the intermediate denoised point cloud after the i-th stage. Our target is to learn the function f^i that estimates residuals ε^i from the noisy input and then to gradually move noisy points closer to the underlying surface.

3.2 Network Architecture

Based on the above denoising model (Eq. 2), we propose the recurrent feature-aware point cloud denoising network, i.e., RePCD-Net. This subsection outlines the general architecture of our network; see its pipeline in Fig. 2. In general, we remove the noise of point cloud stage by stage. At each recursion stage (each row in Fig. 2), our network consists of four parts: multi-scale neighbor sampling, multi-scale feature extraction, bi-directional RNN (BRNN) based feature fusion, and the denoised point cloud output. To avoid losing geometric features during recursions, we embed a recurrent layer to exploit dependencies of deep features across denoising stages. This is due to the fact that the upper-level features contain more geometric details. Next, we would like to introduce these main modules in our RePCD-Net.

Multi-scale Neighbor Sampling Given a noisy point cloud, we first normalize it into a unit sphere centered at the origin, and then build multi-scale neighborhoods for each point. Specifically, we employ ball query to find K scales of neighborhoods, and we empirically set $K = 4$. Within each scale, the number of neighbor points is 32, 48, 64, and 128, respectively. The corresponding searching radius are 0.2, 0.4, 0.6, and 0.8, respectively. We pad neighborhoods with too few points with zeros and take a random subset from neighborhoods with too many points.

Multi-scale Feature Extraction For each neighborhood, we extract its corresponding feature vector, by using multi-layer perceptrons (MLPs, three layers: 32, 64, 128) followed by a max-pooling operation, like several other works (Qi et al. 2017a; Rakotosaona et al. 2020). In such a way, both the features from small-scale neighborhoods, that capture more detailed information, and the features from large-scale neighborhoods, that are more shape-aware, can be encoded simultaneously.

Bi-Directional RNN (BRNN) Based Feature Fusion As detailed before, in our work, we extract four different scales of local neighborhoods. Commonly, if the noise intensity is high, the feature extracted from large scale (receptive field) is more important for effectively suppressing noise, and vice versa. To make full use of our extracted different scales of point features, inspired by the Point2Sequence model (Liu et al. 2019), we design an RNN sequence model to exploit the correlation among different neighborhood scales and deduce an adaptive feature for the noise reduction of the target point. Naturally, there are two input orders when feeding these extracted features into the RNN based fusion model: from

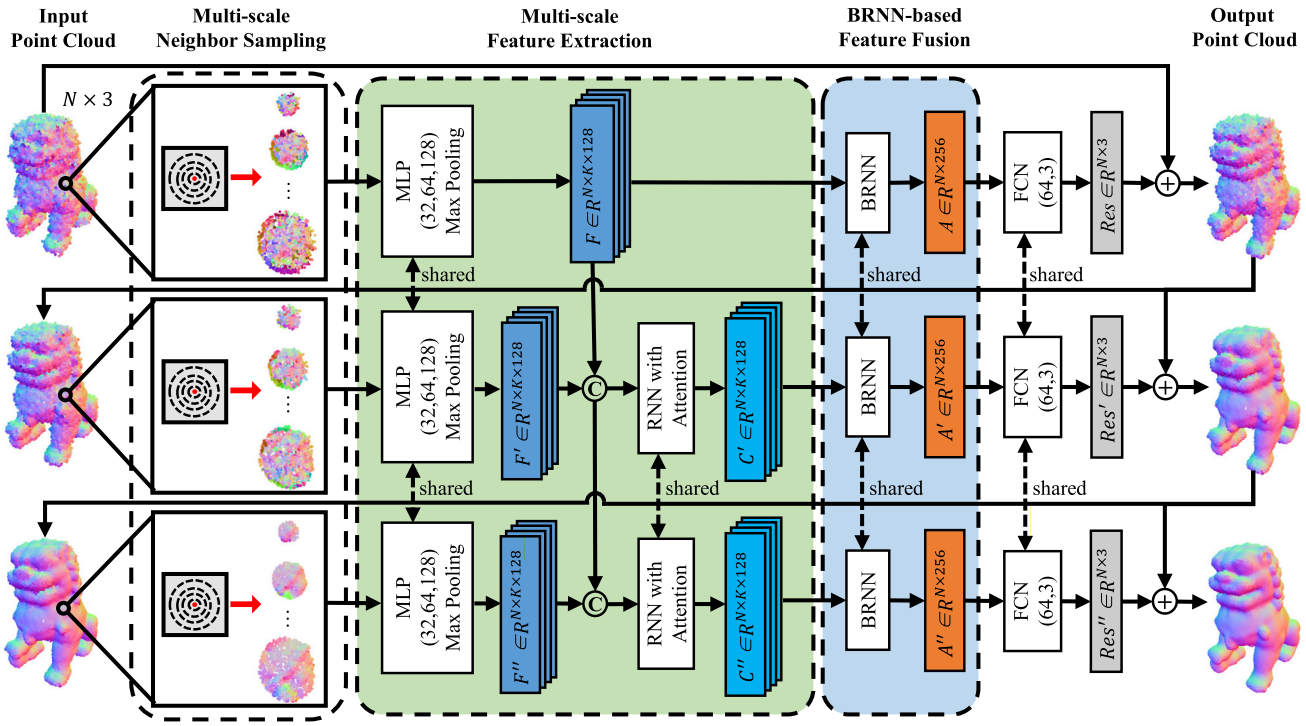

Fig. 2 Our network architecture. The full network is in a recurrent structure. In each recursion, we first extract multi-scale features from each point's multi-scale neighborhoods. Then, a BRNN-based feature fusion module is introduced to determine an adaptive feature which results in the denoised point in the current recursion stage. Besides, a recurrent feature propagation layer is embedded before feature fusion, to enhance the geometric feature perception across stages. Lastly, the fused deep feature is used to regress the coordinate residual and obtain the denoised result. Notice that the MLP, RNN with Attention, BRNN, and FCN share the same paramters across stages

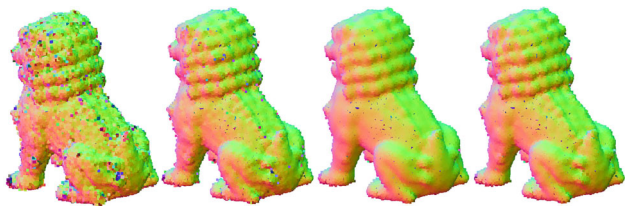

Fig. 3 A test of different input feature orders. From left to right: Noisy input, the denoised result by the order from large scale to small scale, the denoised result by the order from small scale to large scale, and the denoised result by combing both two input orders together. We can observe that combing both two input orders achieves a better balance between noise removal and feature preservation

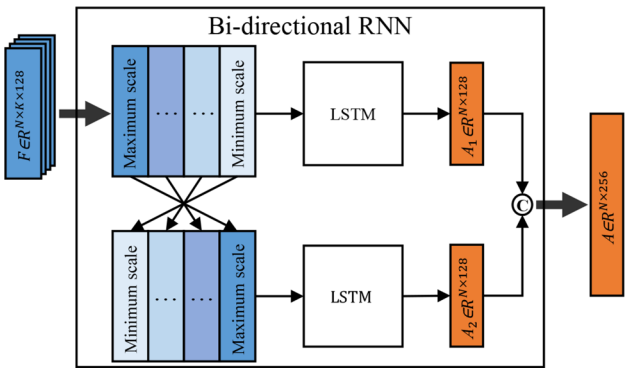

Fig. 4 Illustration of the BRNN module

large scale to small scale, or the opposite. Experimentally, we find that the former order retains more noise, while the latter order often produces a feature-smoothed surface; see Fig. 3 for an example. The reason behind is that, RNN tends to pay more attention to the recently entered features, while gradually ignores/forgets previously entered features. Hence, for the input order of large-to-small scale, the network is inclined to the small scale features, thus retaining excessive noise. On the contrary, for the input order of small-to-large scale, the network is inclined to the large scale features, thus leading to the surface over-smoothing.

To resolve this problem, we design a BRNN module to produce more adaptive features, by combining both two input orders. Figure 4 shows the detailed structure of the BRNN module. Specifically, we sequentially feed the extracted four features into two LSTM (Hochreiter and Schmidhuber 1997) units from large-to-small scale, and small-to-large scale, respectively. The final output of BRNN is computed by con-

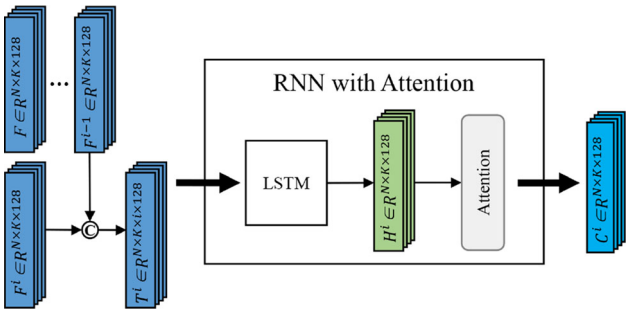

Fig. 5 Illustration of the recurrent feature propagation layer

catenating the outputs of two LSTM units, and it is further used to regress the coordinate residual ε.

• *Remark* Obviously, we can replace the BRNN module with the self-attention based transformer module (Vaswani et al. 2017). In our experiment, we observe that the transformer module does not perform better than the BRNN module, but with nearly twice the training time consumption. This is because the transformer module is designed for capturing long-range dependent correlation, while the number of scales ($K = 4$) is not large in our case. Detailed quantitative evaluation is reported in the ablation study.

Denoised Point Cloud Output Once we obtain per-point feature, we regress the residual 3D coordinates by applying two fully-connected layers. The network then outputs the denoised points by adding the original 3D coordinates of the input points to the regressed residual 3D coordinates.

Recurrent Feature Propagation Layer To overcome the drawback of feature over-smoothing during recurrent denoising, we introduce a recurrent feature propagation layer to implicitly help recover geometric features. Specifically, we delivery the extracted multi-scale features from all previous stages to the feature extraction part of the current stage. An RNN with attention encoder (Liu et al. 2019) is embedded to generate a refined feature for each scale. Figure 5 shows the detailed structure. Compared with direct concatenating features of the same scale, the RNN with attention encoder is able to correlatively learn a more robust feature for the same neighborhood with different levels of noise and detail. The visual comparisons with and without this propagation layer is shown in Fig. 6c vs. a.

Analysis of Different Stages The recurrent design (Eq. 2) of our network can well address the limitations of existing methods, i.e., retaining excessive noise or over-smoothing fine details. Intuitively, an interpretation behind our design is that, the early denoising stages are analogous to a low-pass filter that removes high-frequency components (i.e., noise), while the latter denoising stages are similar to a high-pass filter that recovers details to avoid over-smoothing. We also visualize

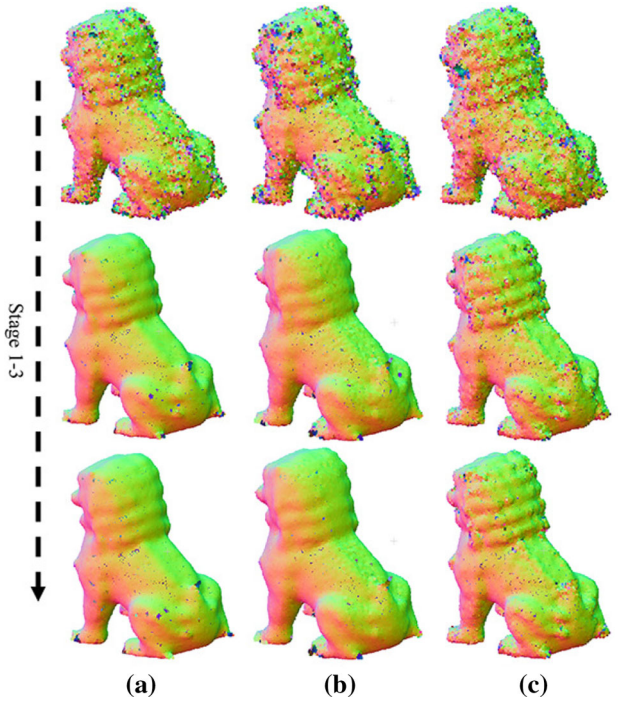

Fig. 6 **a** Intermediate denoising results without the feature propagation layer. **b** Intermediate denoising results without the feature weight in the loss. **c** Intermediate denoising results with both the feature propagation layer and the feature-aware loss

Fig. 7 The shapes labelled with feature scale used for the RePCD-Net training. We cluster 6 kinds of features. Different colors mean different feature scales. The redder the color, the more smooth the surface

the intermediate denoising results; see Fig. 6c. These results are consistent with our intuition.

3.3 Loss Function

To train our network in an end-to-end fashion, we propose a joint loss function to supervise the denoised results from all stages. Our loss consists of two terms: feature-aware CD loss and repulsion loss.

Feature-Aware CD Loss To encourage our denoised point cloud $\hat{\mathbf{P}}$ to be consistent with the ground-truth point cloud $\mathbf{P}^\star$, a widely-used loss function is the chamfer distance. However, the fact is that the points located on the fine geometric structures are often sparse on a 3D shape compared with the points located on smooth regions, thus leading to the network focusing more on those smooth regions and losing fine details, when minimizing the conventional CD loss. To further encourage our network to pay more attention to the geometric details, we thus propose a feature-aware CD loss by introducing a per-point feature-aware weight g_j:

$$\mathcal{L}_{\text{fea}} = \sum_i \sum_{p_j^\star \in \mathbf{P}^\star} g_j \min_{\hat{p}_j^{i+1} \in \hat{\mathbf{P}}^{i+1}} \left\| p_j^\star - \hat{p}_j^{i+1} \right\| + \sum_{\hat{p}_j^{i+1} \in \hat{\mathbf{P}}^{i+1}} g_j \min_{p_j^\star \in \mathbf{P}^\star} \left\| \hat{p}_j^{i+1} - p_j^\star \right\|, \tag{3}$$

where $p_j^\star$ is the j-th point in ground-truth point cloud $\mathbf{P}^\star$, and $\hat{p}_j^{i+1}$ is the j-th denoised point in the i-th stage. Notice that the Earth Mover's distance (EMD) (Fan et al. 2017) is another candidate for evaluating the similarity between two point sets.

The per-point feature-aware weight g_j is obtained according to the smoothness of the associated noise-free point $p_j^\star$. Specifically, in our work, we roughly divide all points in $\mathbf{P}^\star$ into six categories. We then assign a label ID to each point based on the smoothness of the features, with smaller IDs for points on smoother surfaces and larger IDs for sharper ones; see Sect. 4 for the detailed smoothness calculation procedure. The weight g_i is then computed by the corresponding feature label multiplied by 100. Note that, 100 is an empirical value, which is used to balance the loss magnitude. Note also that the proposed RePCD-Net only needs g_j in the training stage. The visual comparisons with and without g_j are shown in Fig. 6c vs. b.

Repulsion Loss To encourage the denoised points distributed uniformly, we follow (Lipman et al. 2007; Yu et al. 2018b) to adopt the repulsion loss:

$$\mathcal{L}_{\text{rep}} = \sum_j \sum_{j' \in K(j)} \eta\left(\left\| \hat{p}_{j'}^{i+1} - \hat{p}_j^{i+1} \right\|\right) w\left(\left\| \hat{p}_{j'}^{i+1} - \hat{p}_j^{i+1} \right\|\right), \tag{4}$$

where $K(j)$ is the index set of the k-nearest neighbors of point $\hat{p}_j^{i+1}$, $w(\cdot)$ is the Gaussian weight function, and $\eta(r) = -r$ is the decreasing function to penalize two points located too close to each other.

Overall, our joint loss function is formulated as:

$$\mathcal{L} = \mathcal{L}_{\text{fea}} + \lambda \mathcal{L}_{\text{rep}}, \tag{5}$$

Table 2 Quantitative comparisons by using various methods on synthetic noisy models with different noise levels and different point intensities

	0.5%-10,000	0.5%-20,000	0.5%-50,000	1%-10,000	1%-20,000	1%-50,000	1.5%-10,000	1.5%-20,000	1.5%-50,000	Average
WLOP	8.68	6.33	4.19	13.2	7.26	7.04	19.0	10.7	12.55	9.88
ECN	5.67	2.13	1.48	6.99	2.78	2.37	8.54	3.90	4.41	4.25
PCN	3.62	1.89	1.39	7.88	2.98	1.88	12.47	4.48	2.48	4.34
GPD	3.32	1.98	1.80	**5.06**	2.14	1.82	6.43	2.67	2.63	3.09
TD	3.56	1.64	1.23	8.35	3.60	3.13	14.01	7.40	7.27	5.58
Ours	**3.21**	**1.49**	**1.19**	5.14	**2.01**	**1.68**	**6.37**	**2.42**	**2.24**	**2.86**

Best performance are shown in bold

where λ is a weight to balance the importance of each loss term and we empirically set it as 0.01.

4 Experimental Settings

Noise Assumption Considering that the type and scale of real-world noise are unknown and irregular, we thus follow most existing methods to assume noise as Gaussian distribution. Moreover, Gaussian noise has also been shown to be reasonably accurate for popular depth cameras like Kinect (Nguyen et al. 2012).

Training Data Preparation Considering that point cloud denoising is a low-level task, we thus crop patches from an input point cloud as network inputs, although we present a whole point cloud in Fig. 2. To build our training data, we collected a synthetic dataset with 17 mesh models, including 9 CAD-like models and 8 smooth models. Given a mesh model, we first produce a point cloud by uniformly sampling 100,000 points on its surface. Then, 100 seed points are randomly selected as the patch centroids. For each seed, we generate a geodesic-aware local patch by using the Dijkstra algorithm. The point number in each patch is fixed as 500. We regard this patch as a clean patch. To obtain its noisy counterpart, we add Gaussian noise with the standard deviation of 0.25%, 0.5%, 1.0%, 2.0%, 2.5% of the original shape's bounding box diagonal. Hence, we generate in total $17 \times 100 \times 5 = 8,500$ synthetic patch pairs for training.

Multi-scale Geometric Feature Labelling Multi-scale geometric feature extraction has been well studied in geometry processing area. Inspired by the selective geometry texture filtering in Wei et al. (2020), we propose to use normal tensor voting to label each ground-truth point as belonging to a certain feature scale. Specifically, we first compute the normal voting tensor of each point, and the three eigenvalues of this voting tensor, which are used to determine the feature scale. We then perform clustering with the K-means clustering algorithm (Gersho and Gray 2012) using the voting results to determine the feature scale of each point. We refer the reader to Wei et al. (2020) for a detailed explanation. Note that the clustering is conducted on the all points, for making the feature scale consistent over the whole dataset, namely the feature label of the same kind of geometric features on different training model is consistent. We empirically cluster six kinds of geometric features. Figure 7 demonstrates the shape set with feature labels used for training our network. Different colors indicate different feature scales.

Network Inference During the inference stage, we try two schemes to obtain the final denoised result. The first one crops the whole point cloud into small patches, and obtains the denoised results by synthesizing all denoised patches. We use farthest point sampling to generate seed points as the centers of small patches. The second one directly feeds the whole point cloud to the network. We treat the entire point cloud as a patch, and normalize it so that the size of the four neighborhoods of each point in the normalized point cloud is similar to the training data. Comparing these two schemes, for each local point, they both provide similar multi-scale neighborhood information which is consistent with the training data. We perform both two schemes, and find that the two schemes produce similar results. Detailed visual results are shown in Sect. 5.5.

Implementation Details We implemented our network on TensorFlow and trained it on a single NVIDIA GTX 1080 GPU for 100 epochs with the Adam optimizer using a learning rate of 10^{-6}. Generally, the training took about 10 h. *We will release our code, model, and data for both training and evaluation on GitHub upon publication.*

5 Results and Discussion

5.1 Quantitative Comparisons

Competitors To validate the effectiveness of both noise removal and geometric feature preservation, we compare our RePCD-Net against state-of-the-art point cloud denoising methods, including the weighted local optimal projection (WLOP) (Huang et al. 2009), PCN (Rakotosaona et al. 2020), ECN (Yu et al. 2018a), GPD (Pistilli et al. 2020), and the unsupervised TD (Hermosilla et al. 2019). WLOP is a traditional projection-based denoising method, and the latter three methods are state-of-the-art learning-based denoising approaches. PCN, ECN, and GPD are supervised methods, while TD is an unsupervised method. For PCN and TD, we re-trained their released codes using our collected shapes. For ECN and GPD, we directly employed their released trained network model for testing.

Synthetic Test Dataset We collected a set of mesh models mainly from the dataset provided by Li et al. (2019) as the synthetic test dataset. These shapes can be divided into three categories: Simple, Medium, and Complicated category, in which there are 12, 10, and 10 models respectively. To validate the robustness of each method on handling point clouds with different point intensities, instead of sampling a fixed number of points, we sample 10,000, 20,000, and 50,000 points on each mesh surface as the ground-truth point cloud. To prepare noisy input, each sampled clean point cloud is then perturbed by Gaussian noise with a standard deviation of 0.5%, 1.0%, and 1.5% of the diagonal length of the bounding box. Hence, there are totally $32 \times 3 \times 3 = 288$ test point clouds. Note that, there is no mesh overlap between our prepared synthetic training dataset and test dataset. By default, we directly feed the whole point cloud into our trained network for inference, rather than the patch-based scheme.

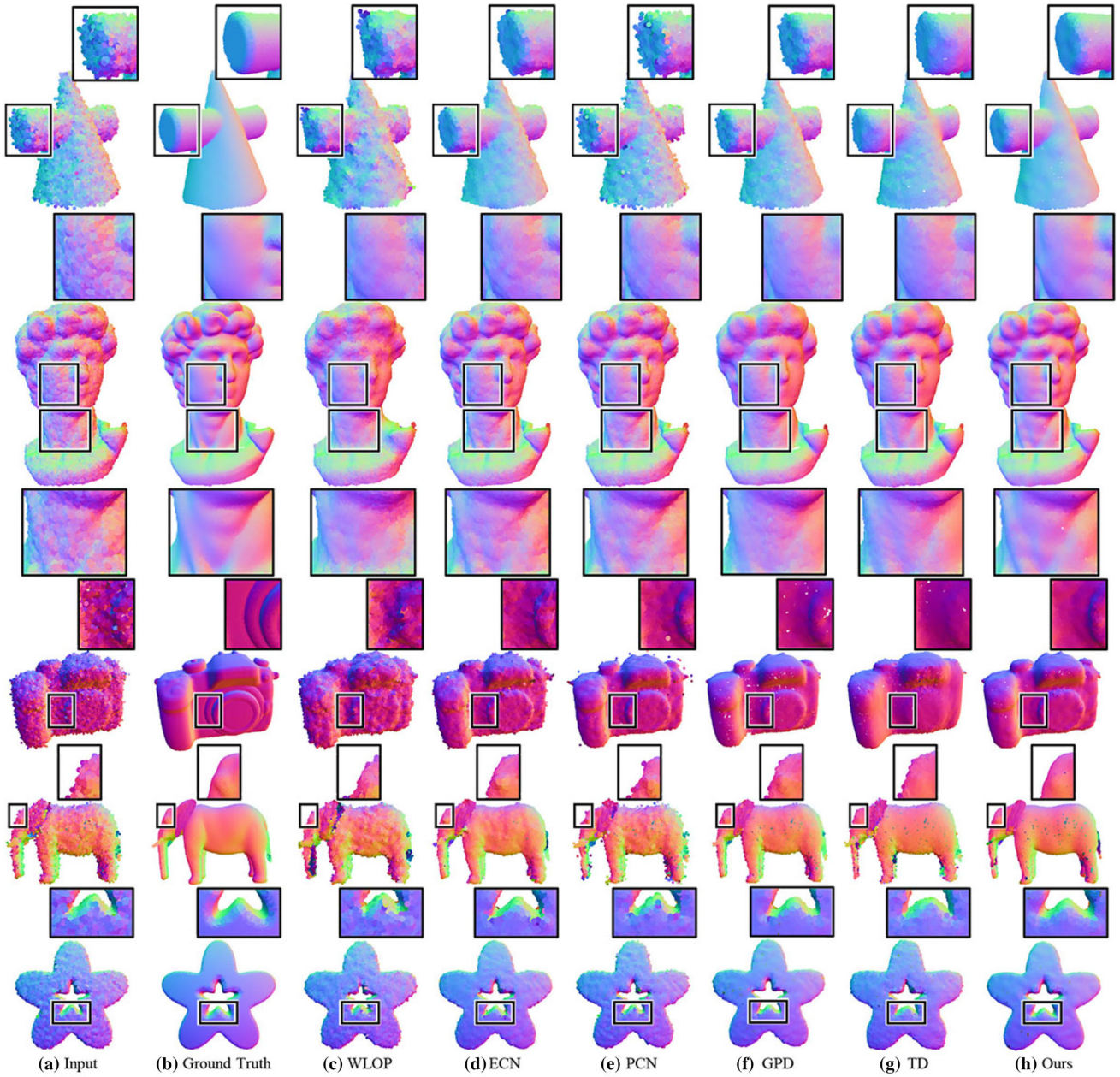

Fig. 8 Visual comparisons of the denoising results by using our method (**h**) against state-of-the-arts (**c**–**g**). **a** shows the noisy inputs **b** is the ground-truth models. The input models in the top two rows are real-scanned data by Kinect, and the input models in the bottom three rows are corrupted by Gaussian noise with the level of 1.5%. Clearly, our method outperforms others in terms of both noise removal and feature preservation; see particularly the blown-up views

Quantitative Evaluation Results To quantitatively compare our method against state-of-the-arts, we follow existing works to use the Chamfer Distance (CD) between denoised results and their ground-truth counterparts as the evaluation metric. A lower CD value indicates a better denoising result. Table 2 shows the comparison results in terms of different noise levels and different point intensities. Clearly, our method yields the best denoising performance with the lowest CD values across almost all the noise levels and point intensities, compared with the state-of-the-art competitors. Also, when the noise level is fixed, a lower point intensity (e.g., 5000 points) usually indicates limited geometric details. Even under such case, our method still achieves the best denoising results.

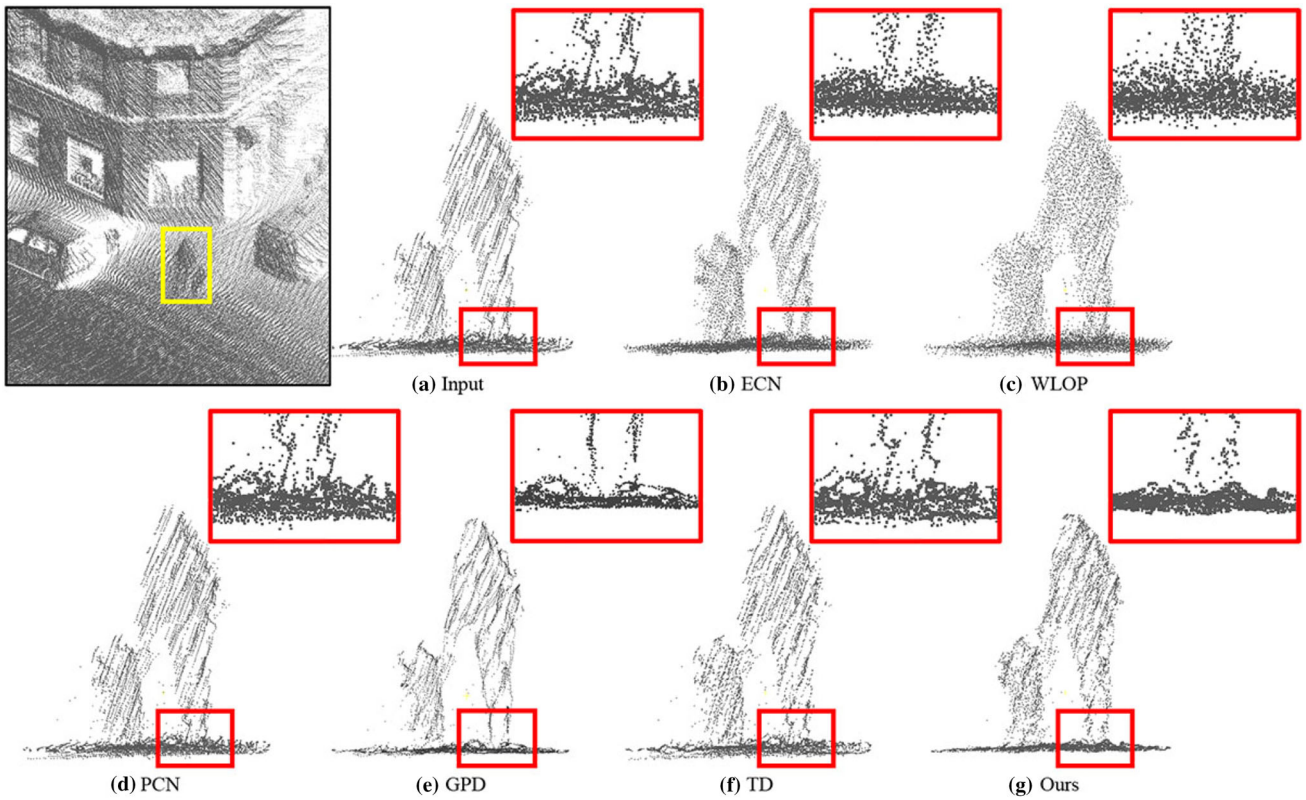

Fig. 9 Denoising a real-scanned point cloud scene. Our method clearly removes heavy noise better, and avoids introducing additional artifacts

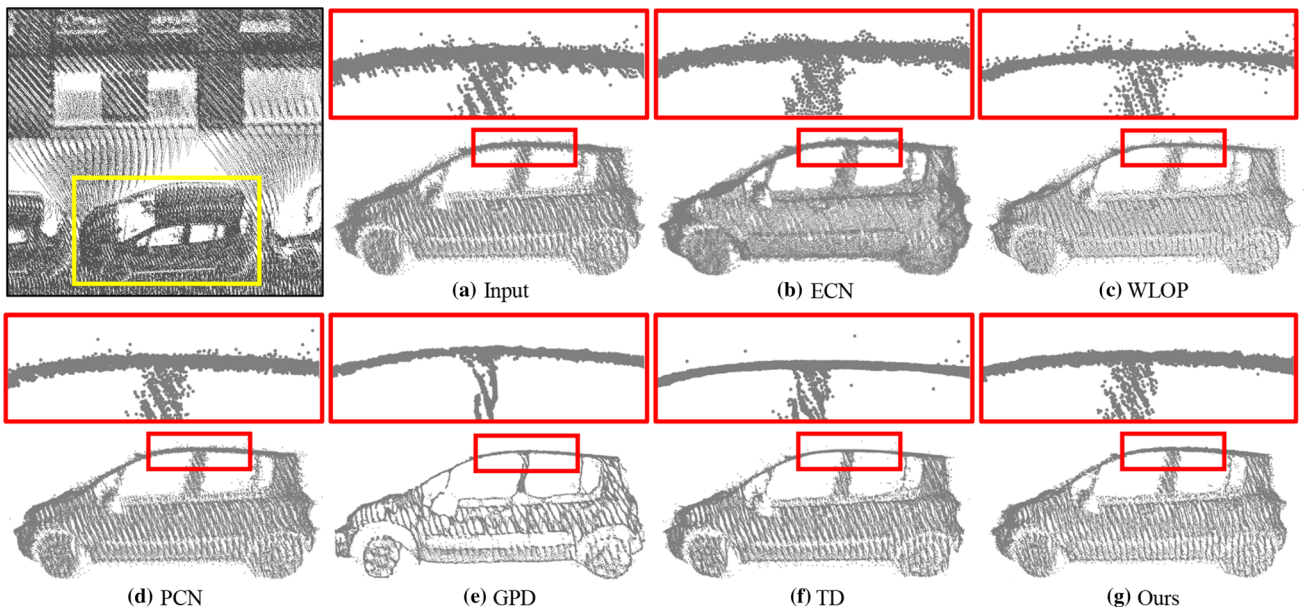

Fig. 10 Denoising a real-scanned point cloud scene. Our method clearly removes heavy noise better, and avoids introducing additional artifacts

5.2 Visual Comparisons

Apart from the quantitative comparisons on synthetic dataset, we also show the visual comparisons on both synthetic and real-scanned noisy point clouds. The first two rows in Fig. 8 present the denoised results of two real-scanned point clouds, which are scanned by Kinect (Wang et al. 2016). The bottom three rows are the synthetic noisy models. In general, WLOP (Huang et al. 2009) (c) is hard to balance the noise removal and the feature preservation. ECN (Yu et al. 2018a) (d) and PCN (Rakotosaona et al. 2020) (e) are prone to retain excessive noise in the results. GPD (Pistilli et al. 2020) (f) and TD (Hermosilla et al. 2019) (g) tends to over-smooth the geometric features contained in the point clouds. As a contrast, thanks to the recurrent structure, the feature propagation layer, and the geometric-aware loss of our RePCD-Net, the denoised results produced by our method are much cleaner, while still preserving tiny details; see particularly the blown-up views.

Furthermore, we show the qualitative results of two real scans of outdoor scenes provided by Serna et al. (2014) in Figs. 9 and 10. As observed from these results, our approach produces generally more desirable results with few outliers, in terms of heavy noise removal. It is noteworthy that we do not re-train all the methods including ours.

Notably, the noise type and point resolution of the fisrt two inputs in Fig. 8 and the inputs in Figs. 9 and 10 are different from our training data. That is to say, there exits certain domain gap between them. However, it is observed that our method generalizes well to these kind of data.

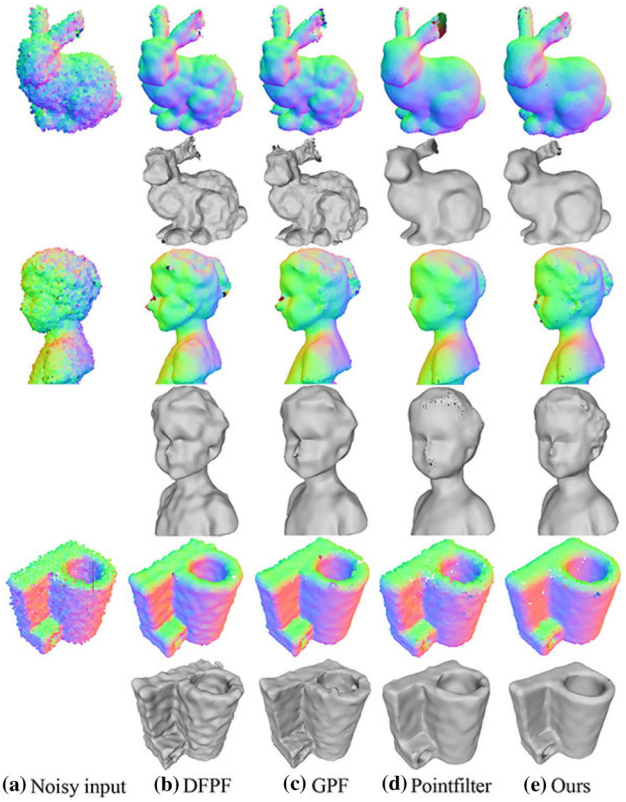

Fig. 11 Comparisons of denoising results by using normal-based methods (**b**–**d**) and our method (**e**). **a** shows the noisy inputs. For each noisy model, the first row shows the denoised results, and the second row shows the corresponding surface reconstruction results. Note that we re-compute the point normals by the PCA technique for each denoised model, for fairly comparing the reconstruction results

5.3 Comparison with Normal-Based Denoising Methods

Normal-based point cloud denoising approaches also attacked many researchers' attention. This is because the first-order normal variations can better describe surface variations than point position variations. Normal-based methods usually first filter the point normals. The point positions are then adjusted to well match the filtered normals.

Here, we further compare our RePCD-Net with three advanced normal-based denoising methods, i.e., Pointfilter (Zhang et al. 2020), DFPF (Lu et al. 2020), and GPF (Lu et al. 2017). Pointfilter and DFPF (Lu et al. 2020) are the learning-based method, while GPF (Lu et al. 2017) is an optimized-based method. Figure 11 shows the visual comparisons in terms of both denoised point clouds and the associated reconstructed meshes by using the marching cubes algorithm. By comparing (b–d) against (e), we can observe that the denoising results of both DFPF (Lu et al. 2020) and GPF (Lu et al. 2017) contain unsmoothed surfaces or over-sharpened features, while the results of Pointfilter tend to be over-smoothed. However, the results produced by our method are cleaner and more smooth. Let us take the Child's face (middle row) as an example. For these normal-based methods, the face region is hard to be denoised, due to the inaccurate normal estimation. Hence, the results of DFPF and GPF either retain excessive noise or over-sharpen features. For Pointfilter, it incorporates the additional normal information in the loss function to preserve geometric sharp features. However, when computing the residual coordinates, their method sets a fixed kernel size to weight the neighboring points' importance to the target denoised point, which hinders the maintenance of multi-scale features, thus leading to the over-smoothing effect. Table 3 further reports the quantitative comparisons (i.e. CD metric). The lowest CD values also confirm the effectiveness and superiority of our method against these normal-based denoising approaches.

In general, by comparing with the normal-based method, we find that: (1) Point-based method does not need to prepare the ground-truth normals as the supervision information. (2) First-order normal variation is feature-sensitive, which also means that these normal-based methods are less robust to noise. Thus, they cannot well handle the data with a high-level

Table 3 Quantitative comparisons by using normal-based methods and our method in terms of the CD ($\times 10^{-4}$) values on several typical noisy models

Models	Fig. 11 Bunny	Fig. 11 Boy	Fig. 11 CAD model
Pointfilter	2.92	3.94	3.53
DFPF	6.89	4.77	3.62
GPF	6.02	5.36	3.94
Ours	**2.17**	**2.90**	**3.20**

Best performance are shown in bold

noise. (3) The orientation of the normal heavily affects the normal estimation, especially for the learning-based method, like DFPF (Li et al. 2020). (4) Since normals are computed from points, it would be a promising direction to collaboratively learning normals and point positions altogether to further promote the performance of point cloud denoising.

5.4 Ablation Study

In this subsection, we analyze the contribution of major modules in our method, by designing several variants of RePCD-Net, as follows:

- Variant_1: replacing the multi-scale neighborhoods with a single-scale neighborhood. We re-trained two different single-scale networks (Radius = 0.4 and 1.0, respectively).
- Variant_2: directly concatenating the multi-scale features together (Qi et al. 2017b), rather than employing the BRNN based feature fusion module.
- Variant_3: using different recursion stages to train our network. We re-trained three new models with 1, 4, and 5 recursion stages, respectively.
- Variant_4: iteratively performing our three-stage full network. We set the iteration number as 3.
- Variant_5: removing the feature propagation layer from our full pipeline.
- Variant_6: replacing the RNN-based feature propagation layer with directly concatenating all previous features.
- Variant_7: keeping the feature delivery, but removing the attention part from the RNN encoder.
- Variant_8: replacing the BRNN based fusion module with the self-attention based transformer module.
- Variant_9: using conventional CD loss to replace our proposed feature-aware CD loss (see Eq. 3).

We performed quantitative evaluations for these variants on the synthetic test dataset, and summarized the denoising results in Table 4. By comparing each variant with our full pipeline, we can observe that each module contributes to a better denoising performance. Particularly, in Variant_3, we

Table 4 Ablation analysis: quantitative comparisons of different network variants

Variants	Variant_1		Variant_2	Variant_3			Variant_4	Variant_5	Variant_6	Variant_7	Variant_8	Variant_9	Full
	Radius = 0.4	Radius = 1.0		Stage = 1	Stage = 4	Stage = 5							
CD (10^{-4})	4.01	4.20	3.80	3.47	2.95	2.95	**2.86**	2.93	2.98	2.96	3.27	2.95	**2.86**

Best performance are shown in bold

Table 5 The statistics of different scales' significances

Stages	Scale 1		Scale 2		Scale 3		Scale 4	
	Cosine	Percentage	Cosine	Percentage	Cosine	Percentage	*Cosine*	*Percentage*
Stage 1	0.455665	23.14	0.472500	23.99	0.471475	23.94	0.569772	**28.93**
Stage 2	0.387723	21.08	0.523266	**28.44**	0.472503	25.68	0.456171	24.80
Stage 3	0.456273	21.31	0.644369	**30.09**	0.644369	24.41	0.517966	24.19

Most important scale are shown in bold
We measure the significance by the *Cosine* value between the input feature vector and the output feature vector of the BRNN based feature fusion module. *Percentage* denotes the each input feature's proportion

can see that a recurrent structure is helpful for effectively removing noise than the single-stage network (Variant_3, stage = 1). However, too many stages (> 4) cannot produce better results. Besides, we also iteratively perform our three-stage full network three times (Variant_4). The quantitative result does not change, as shown in the Table 4.

5.5 Discussion and Analysis

Scale Selection in Different Denoising Stages Note that one of the basic observations in our method is that the neighborhood scale significantly affects the denoising results. The large-scale neighborhood is more suitable for removing noise, while small-scale provides more detail information. This observation motivates us to design a multi-scale feature encoder (BRNN based feature fusion) to learn an adaptive feature, for well handling different noise levels or geometric structures.

To explore which scale contributes more to the final extracted deep feature, i.e., the output of the BRNN based feature fusion module, we here employ the *Cosine* similarity between the input feature vector and the output feature vector as a metric to measure the scale contribution. A larger *Cosine* value closer to 1 indicates a larger contribution of a specific scale. For a fair evaluation, we select 80 testing point clouds in our benchmark, and compute the average *Cosine* value, as summarized in Table 5. We also report each input feature's proportion, for better observing its significance. From Table 5, it is easy to find that the feature of the largest neighborhood scale has the largest proportion in the first stage, which means this neighborhood contributes more to reduce noise. In the latter two stages, small-scale neighborhoods are more important, as the noise level is low. Besides, we visualize the most important scale for each point, as shown in Fig. 12. This provides some insights regarding each neighborhood's selection. In Stage 1, due to the high-level noise, the largest neighborhood scale contributes most. In the latter two stages, the points in the feature regions need more small neighborhoods. This is consistent with our design intuition.

Inference Scheme: Patch-Based vs. Whole Point Cloud Though our network is trained in a patch-based manner, it could be tested using either the patch-based scheme or the whole point cloud. Fig. 13 shows the comparisons on two examples. We can observe that the two schemes produce similar denoising results. However, considering that the patch-based scheme usually consumes more computation time, since it needs to feed the small patches to the network multiple times, while feeding the whole point cloud only executes the network once, which consumes less time. We thus recommend to directly feed the whole point cloud.

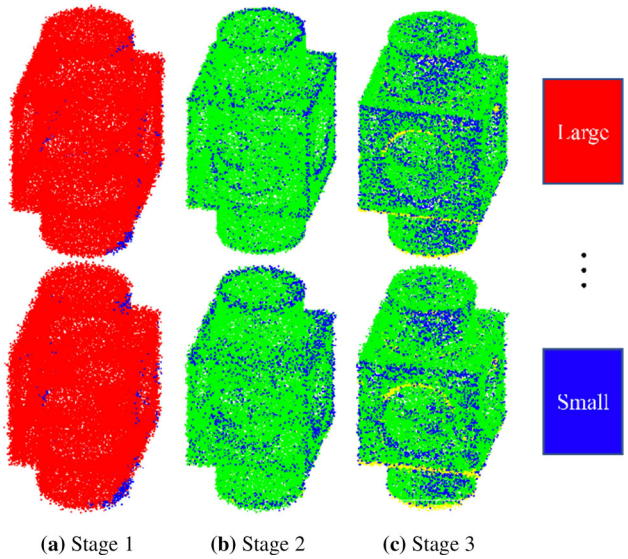

Fig. 12 Visualization of the most important neighborhood scale for each point in different denoising stages. Color coding is given at the right-most side. The first row shows the denoising results of the noisy Block model with 1.0% noise. The bottom row shows the denoising results of the noisy Block model with 1.5% noise

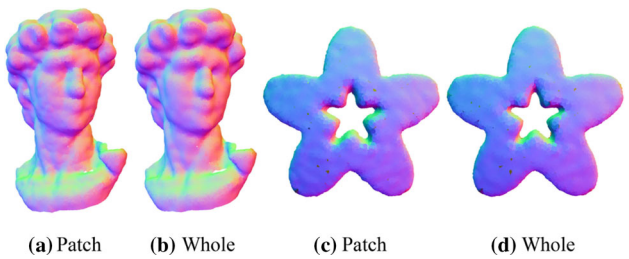

Fig. 13 Comparisons of using different inference schemes. We do not observe obvious differences between using patch-based manner and feeding whole point clouds

Table 6 Timing statistics (in seconds) for different approaches performed on the models in Fig. 8. R1, R2, R3, R4, and R5 represent the first, second, third, fourth, and fifth row in Fig. 8. N is the number of input points

Models	Fig. 8 R1 N: 10,000	Fig. 8 R2 N: 51,789	Fig. 8 R3 N: 50,000	Fig. 8 R4 N: 20,000	Fig. 8 R5 N: 24,440
WLOP	5.67	48.84	15.77	6.34	7.86
ECN	2.96	13.69	13.26	7.05	6.81
PCN	69.41	360.11	347.54	138.86	171.37
GPD	29.29	140.25	142.13	55.34	67.10
TD	3.06	242.30	43.22	7.23	8.10
Ours	19.01	396.16	186.35	119.01	103.65

5.6 Limitations

Despite the promising performance of our method, it still has some limitations. First, the running time of our method is not very fast. Table 6 reports the running time of different methods on several typical inputs from Fig. 8. We can see that our method is only faster than PCN (Rakotosaona et al. 2020). This is because: (1) our network is recurrent; and (2) our method needs to search multi-scale neighborhoods for each point. Second, the neighborhood scale is a hyper-parameter that needs to be determined (4 fixed scales in current version). The mechanism of selecting a more soft neighborhood maybe an interesting direction.

6 Conclusion

In this paper, we presented a feature-aware recurrent network (RePCD-Net) to denoise unstructured point clouds. To tackle the main two challenges (noise removal and feature recovery), we elaborately design our RePCD-Net from four aspects: (1) design a BRNN based encoder to learn an adaptive feature from multi-scale neighborhoods for different denoising task; (2) propose to use a recurrent denoising architecture to effectively reduce noise; (3) design a recurrent feature propagation layer to exploit multi-scale features across stages, for recovering lost geometric details. (4) a feature-aware loss to regularize the predictions towards multi-scale geometric details. Extensive experiments demonstrated the effectiveness of our method, showing that it outperforms the state-of-the-arts in various configurations, from synthetic benchmark to large real-scanned point clouds.

Since we find that the task of point cloud denoising benefits from the prior of the geometric feature or the point normal information, we would like to explore the possibility of designing a more general network that can effectively leverage these prior information to co-support multiple point cloud processing tasks, even in an unsupervised manner. Besides, attention mechanism is an important direction for aggregating more robust patch-level or point-wise information. We will try to use it in our future work.

Acknowledgements The work was supported by the Hong Kong Centre for Logistics Robotics.

References

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., & Silva, C. T. (2001). Point set surfaces. In *Proceedings visualization, 2001. VIS'01* (pp. 21–29). IEEE.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D., & Silva, C. T. (2003). Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics, 9*(1), 3–15.

Avron, H., Sharf, A., Greif, C., & Cohen-Or, D. (2010). L1-sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics (TOG), 29*(5), 1–12.

Chen, H., Wei, M., Sun, Y., Xie, X., & Wang, J. (2019). Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint. *IEEE Transactions on Visualization and Computer Graphics, 26*(11), 3255–3270.

Digne, J., Valette, S., & Chaine, R. (2017). Sparse geometric representation through local shape probing. *IEEE Transactions on Visualization and Computer Graphics, 24*(7), 2238–2250.

Fan, H., Su, H., & Guibas, L. J. (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*, pp. 605–613.

Fleishman, S., Cohen-Or, D., & Silva, C. T. (2005). Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics (TOG), 24*(3), 544–552.

Gersho, A., & Gray, R. M. (2012). *Vector quantization and signal compression* (Vol. 159). Berlin.

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., & Wan, J. (2014). An integrated framework for 3-d modeling, object detection, and pose estimation from point-clouds. *IEEE Transactions on Instrumentation and Measurement, 64*(3), 683–693.

Hermosilla, P., Ritschel, T., & Ropinski, T. (2019). Total denoising: Unsupervised learning of 3d point cloud cleaning. In *Proceedings of the IEEE international conference on computer vision*, pp. 52–60.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Hu, W., Gao, X., Cheung, G., & Guo, Z. (2020). Feature graph learning for 3d point cloud denoising. *IEEE Transactions on Signal Processing, 68*, 2841–2856.

Huang, C., Li, R., Li, X., & Fu, C. W. (2020). Non-local part-aware point cloud denoising. arXiv:200306631

Huang, H., Li, D., Zhang, H., Ascher, U., & Cohen-Or, D. (2009). Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics (TOG), 28*(5), 1–7.

Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U., & Zhang, H. (2013). Edge-aware point set resampling. *ACM Transactions on Graphics (TOG), 32*(1), 1–12.

Kong, D., Xu, L., Li, X., & Li, S. (2013). K-plane-based classification of airborne lidar data for accurate building roof measurement. *IEEE Transactions on Instrumentation and Measurement, 63*(5), 1200–1214.

Li, R., Li, X., Fu, C. W., Cohen-Or, D., & Heng, P. A. (2019). Pu-gan: a point cloud upsampling adversarial network. In *Proceedings of the IEEE international conference on computer vision*, pp. 7203–7212.

Li, X., Li, R., Zhu, L., Fu, C. W., & Heng, P. A. (2020). Dnf-net: a deep normal filtering network for mesh denoising. *IEEE Transactions on Visualization and Computer Graphics*, *27*(10), 4060–4072.

Lipman, Y., Cohen-Or, D., Levin, D., & Tal-Ezer, H. (2007). Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics (TOG), 26*(3), 22-es.

Liu, X., Han, Z., Liu, Y. S., & Zwicker, M. (2019). Point2sequence: Learning the shape representation of 3d point clouds with an attention-based sequence to sequence network. In *Proceedings of the AAAI conference on artificial intelligence,* (Vol. 33, 8778–8785).

Lu, D., Lu, X., Sun, Y., & Wang, J. (2020). Deep feature-preserving normal estimation for point cloud filtering. *Computer-Aided Design, 125*, 102860.

Lu, X., Schaefer, S., Luo, J., Ma, L., & He, Y. (2018). Low rank matrix approximation for geometry filtering. CoRR arXiv:abs/1803.06783.

Lu, X., Wu, S., Chen, H., Yeung, S. K., Chen, W., & Zwicker, M. (2017). Gpf: Gmm-inspired feature-preserving point set filtering. *IEEE Transactions on Visualization and Computer Graphics, 24*(8), 2315–2326.

Luo, S., & Hu, W. (2020). Differentiable manifold reconstruction for point cloud denoising. In *Proceedings of the 28th ACM international conference on multimedia*, pp. 1330–1338.

Nguyen, C. V., Izadi, S., & Lovell, D. (2012). Modeling kinect sensor noise for improved 3d reconstruction and tracking. In *2012 second international conference on 3D imaging, modeling, processing, visualization & transmission* (pp. 524–530). IEEE.

Öztireli, A. C., Guennebaud, G., & Gross, M. (2009). Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum, 28*(2), 493–501.

Pistilli, F., Fracastoro, G., Valsesia, D., & Magli, E. (2020). Learning graph-convolutional representations for point cloud denoising. In *European conference on computer vision* (pp. 103–118). Springer.

Preiner, R., Mattausch, O., Arikan, M., Pajarola, R., & Wimmer, M. (2014). Continuous projection for fast l1 reconstruction. *ACM Transactions on Graphics (TOG), 33*(4), 47-1.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 5099–5108). Red Hook, NY: Curran Associates Inc.

Rakotosaona, M., Barbera, V. L., Guerrero, P., Mitra, N. J., & Ovsjanikov, M. (2020). Pointcleannet: Learning to denoise and remove outliers from dense point clouds. *Computer Graphics Forum, 39*(1), 185–203.

Remil, O., Xie, Q., Xie, X., Xu, K., & Wang, J. (2017). Surface reconstruction with data-driven exemplar priors. *Computer-Aided Design, 88*, 31–41.

Rosman, G., Dubrovina, A., & Kimmel, R. (2013). Patch-collaborative spectral point-cloud denoising. *Computer Graphics Forum, 32*(8), 1–12.

Roveri, R., Öztireli, A. C., Pandele, I., & Gross, M. (2018). Pointpronets: Consolidation of point clouds with convolutional neural networks. *Computer Graphics Forum, 37*, 87–99.

Serna, A., Marcotegui, B., Goulette. F., & Deschaud, J. E. (2014). Paris-rue-madame database: A 3d mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In *4th International Conference on Pattern Recognition, Applications and Methods ICPRAM 2014*.

Sun, Y., Schaefer, S., & Wang, W. (2015). Denoising point sets via l0 minimization. *Computer Aided Geometric Design, 35*, 2–15.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 5998–6008). Red Hook, NY: Curran Associates Inc.

Wang, P. S., Liu, Y., & Tong, X. (2016). Mesh denoising via cascaded normal regression. *ACM Transactions on Graphics, 35*(6), 232–1.

Wang, Y., Liu, Y., Xie, Q., Wu, Q., Guo, X., Yu, Z., & Wang, J. (2020). Density-invariant registration of multiple scans for aircraft measurement. *IEEE Transactions on Instrumentation and Measurement, 70*, 1–15.

Wang, Y., Serena, F., Wu, S., Öztireli, C., & Sorkine-Hornung, O. (2019). Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG), 38*(6), 1–14.

Wei, M., Feng, Y., & Chen, H. (2020). Selective guidance normal filter for geometric texture removal. *IEEE Transactions on Visualization and Computer Graphics*, *27*(12), 4469–4482.

Xie, Q., Lu, D., Huang, A., Yang, J., Li, D., Zhang, Y., & Wang, J. (2020). Rrcnet: Rivet region classification network for rivet flush measurement based on 3-d point cloud. *IEEE Transactions on Instrumentation and Measurement, 70*, 1–12.

Yu, L., Li, X., Fu, C. W., Cohen-Or, D., & Heng, P. A. (2018a). Ec-net: An edge-aware point set consolidation network. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 386–402.

Yu, L., Li, X., Fu, C. W., Cohen-Or, D., & Heng, P. A. (2018b). Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp 2790–2799.

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

Zhou, H., Chen, K., Zhang, W., Fang, H., Zhou, W., & Yu, N. (2019). Dup-net: Denoiser and upsampler network for 3d adversarial point clouds defense. In *Proceedings of the IEEE international conference on computer vision*, pp. 1961–1970.

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com