Pointfilter: Point Cloud Filtering via Encoder-Decoder Modeling

Dongbo Zhang
Beihang University
Beijing, China
zhangdongbo9212@163.com

Xuequan Lu
Deakin University
Australia
xuequan.lu@deakin.edu.au

Hong Qin
Stony Brook University
New York, USA
qin@cs.stonybrook.edu

Ying He
Nanyang Technological University
Singapore
YHe@ntu.edu.sg

ABSTRACT

Point cloud filtering is a fundamental problem in geometry modeling and processing. Despite of advancement in recent years, the existing methods still suffer from two issues: they are either designed without preserving sharp features or less robust in preserving geometric features; they usually have many parameters and require tedious parameter tuning. In this paper, we propose a novel deep learning approach that automatically and robustly filters point clouds with removing noise and preserving sharp features and geometric details. Our point-wise learning architecture consists of an encoder and a decoder. The encoder directly takes points (a point and its neighbors) as input, and learns a latent representation vector which is gone through the decoder and related to the ground-truth position. Our trained network can automatically infer a corresponding quality point set to a noisy point cloud input. Extensive evaluations show that our approach outperforms the state-of-the-art deep learning techniques in terms of visual quality and error metrics. We will make our code and dataset publicly available.

CCS CONCEPTS

• **Theory of computation → Computational geometry**; • **Computing methodologies → Shape modeling**; **Point-based models**;

KEYWORDS

Automatic point cloud filtering, deep learning, autoEncoder, feature-preserving.

1 INTRODUCTION

Point cloud data is inevitably corrupted with noise, with the increasing access to scanning facilities especially the consumer-level depth sensors. Point cloud filtering is to reconstruct a point set by removing noise from its corresponding noisy input. The filtered point clouds can be used in various graphics applications, for example, point set resampling, surface reconstruction, point-based rendering and computer animation. Removing noise while preserving geometric features still remains a challenge in point cloud filtering.

State-of-the-art techniques have made remarkable progress in point cloud filtering. Specifically, the LOP-based methods (LOP [Lipman et al. 2007], WLOP [Huang et al. 2009], CLOP [Preiner et al. 2014]) are robust to noise and outliers. The RIMLS [Öztireli et al. 2009] and GPF [Lu et al. 2018] consider to preserve geometric features. Nevertheless, these techniques still suffer from either feature smearing or less robustness in filtering. More precisely, the LOP-based techniques [Huang et al. 2009; Lipman et al. 2007; Preiner et al. 2014] are not designed for preserving sharp features, because of their inherent isotropic property. RIMLS [Öztireli et al. 2009] and GPF [Lu et al. 2018] can preserve geometric features to some extent; however, they depend greatly on the ability of normal filters which become less robust when meeting either large noise or irregular sampling. Furthermore, GPF [Lu et al. 2018] is not easy to find a radius that balances noise removal and gaps near edges, and it is slow due to the considerable amount of computation of the EM optimization. Finally, they all need trial-and-error parameter tuning which is boring and time-consuming, to achieve desired visual results. Point cloud filtering with resorting to deep learning has been rarely studied so far [Rakotosaona et al. 2019; Yu et al. 2018a]. Nevertheless, they still generate limited results with either smoothing out sharp features or poor generalization. As an ending result, the above limitations substantially restrict the robustness and applicability of these state-of-the-art point cloud filtering techniques. We show the comparison of the main characteristics between these techniques and our method in Table 1.

Table 1: The comparison of the main characteristics (CHs) between the state of the art point set filtering techniques and our method. ✓ and × denote YES and NO, respectively. ✓̂ indicates a "medium" level between YES and NO.

CHs / Methods	normal independency	noise& outlier	sharp features	fixed parameters	fast speed
LOP & WLOP	✓	✓	×	×	✓̂
CLOP	✓	✓	×	×	✓
RIMLS	×	✓̂	✓	×	✓̂
GPF	×	✓̂	✓	×	×
EC-Net	✓	✓̂	✓̂	✓	✓
PCN	✓	✓	×	✓	✓̂
Ours	✓	✓	✓	✓	✓

In this paper, we propose a novel point cloud filtering approach to settle the above issues. Motivated by the noticeable successes of AutoEncoder and PointNet [Charles et al. 2017; Schmidhuber 2015] in vision tasks, we design a learning framework referred to as Pointfilter for point cloud filtering. In particular, our framework

is an encoder-decoder based architecture which straightforwardly takes the raw neighboring points of each noisy point as input, and regresses a displacement vector to push this noisy point back to its ground truth position. In designing the loss function, we also take geometric features into account so that features can be preserved by this network. Given a noisy point cloud as input, our trained model can automatically and robustly predict a corresponding quality point cloud, by removing noise and preserving geometric features. Various experiments demonstrate that our method achieves better performance than the state-of-the-art techniques (or comparable to optimization based methods like RIMLS and GPF which require trial-and-error parameter tuning), in terms of visual quality and error metrics. Our method is fast and avoids manual parameter tuning. The main contributions of this work are:

- a novel framework that achieves point cloud filtering by encoder-decoder modeling;
- a powerful loss function that takes geometric features into account;
- extensive experiments and analysis in terms of visual quality and quantity;
- the source code and involved dataset which will be made publicly available.

2 RELATED WORK

We first review point cloud filtering techniques, and then look back to some previous works that conducted deep learning on point clouds.

2.1 Point Cloud Filtering

Point cloud filtering can be generally classified into two types: two-step based techniques and projection-based methods. The two-step based framework consists of at least two steps: normal smoothing and point position update under the guidance of the filtered normals. Avron et al. [Avron et al. 2010] and Sun et al. [Sun et al. 2015] introduced L_1 and L_0 optimization for point set filtering, respectively. Recently, with sharp feature skeletons, a point cloud smoothing technique was presented based on the guided filter [Zheng et al. 2017]. Zheng et al. [Zheng et al. 2018] extended the rolling guidance filter to point set filtering and designed a new point position updating strategy to overcome sharp edge shrinkage. Lu et al. [Lu et al. 2018] proposed a two-step geometry filtering approach for both meshes and point clouds. Most point set filtering methods achieve filtered results through projecting the input point set onto the underlying point set surface. One popular category of this type is moving least squares and its variants [Alexa et al. 2003, 2001; Amenta and Kil 2004; Fleishman et al. 2005; Levin 1998; Levin 2004; Öztireli et al. 2009]. The moving least squares (MLS) has been seminally formulated by Levin [Levin 1998; Levin 2004]. Some works defined moving least squares (MLS) and extremal surfaces [Alexa et al. 2003, 2001; Amenta and Kil 2004]. Later, two different variants have been presented for projection: statistics-based and robust implicit moving least squares (RIMLS) [Fleishman et al. 2005; Öztireli et al. 2009]. Lange et al. [Lange and Polthier 2005] developed a method for anisotropic fairing of a point sampled surface using an anisotropic geometric mean curvature flow. Recently, the LOP (locally optimal projection) based methods have become increasingly popular. For example, Lipman et al. [Lipman et al. 2007] proposed the locally optimal projection operator (LOP) which is parameterization free. Later, Huang et al. [Huang et al. 2009] presented a weighted LOP (WLOP), which enhances the uniform distribution of the input points. A kernel LOP has also been proposed to speed up the computation of LOP [Liao et al. 2013]. More recently, a continuous LOP (CLOP) has been presented to reformulate the data term to be a continuous representation of the input point set and arrives at a fast speed [Preiner et al. 2014]. Note that a few projection-based methods utilize smoothed normals as prior to preserve geometric features, such as EAR [Huang et al. 2013] and GPF [Lu et al. 2018].

2.2 Deep Learning on Point Clouds

Qi et al. [Charles et al. 2017] proposed the pioneering network architecture, named PointNet, that can consume raw point clouds directly. The key ingredient of PointNet is that using fully connected layers (MLPs) to extract features instead of convolution operators which are not suitable in irregular domain. Although PointNet achieves remarkable success in shape classification and segmentation, point-wise features fail to characterize local structures which is crucial for high-level semantic understanding. To do this, an improved version, PointNet++ [Qi et al. 2017], was proposed to aggregate local structures in a hierarchical way. Following PointNet and PointNet++, lots of network architectures applied on raw point clouds emerged. For instance, based on dynamic local neighborhood graph structure, Wang et al. [Wang et al. 2019] designed an EdgeConv block to capture the relationships both in spatial and feature space. At the same time, an alternative convolutional framework, SpiderCNN [Xu et al. 2018], was proposed to aggregate neighboring features by a special family of parameterized weighted functions instead of MLPs. Inspired by the Scale Invariance Feature Transform [Lowe 2004] (SIFT) which is a robust 2D representation, the SIFT-like module [Jiang et al. 2018] was developed to encode information of different orientations and scales and could be flexibly incorporated into PointNet-style networks. Besides shape classification and segmentation tasks, there are few point input network architectures applied on upsampling [Yifan et al. 2019; Yu et al. 2018b], local shape properties estimation [Guerrero et al. 2018] and so on. As for point cloud filtering, Roveri et al. [Roveri et al. 2018] proposed a filtering network, PointProNet, designed for consolidating raw point clouds corrupted with noise. Benefiting from powerful 2D convolution, PointProNet transfers 3D point clouds consolidation into 2D height map filtering. To preserve sharp edges while filtering, Yu et al. [Yu et al. 2018a] introduced an novel edge-aware network architecture, EC-Net, by incorporating a joint loss function. Recently, a two-stage network architecture, PointCleanNet (PCN) [Rakotosaona et al. 2019], was developed for removing outliers and denoisnig separately.

3 METHOD

3.1 Overview

Given a noisy point cloud, we aim to restore its clean version by our Pointfilter in a manner of supervised learning. Before introducing details of our Pointfilter framework, we first formulate noisy point cloud as follows:

$$\hat{P} = P + N, \tag{1}$$

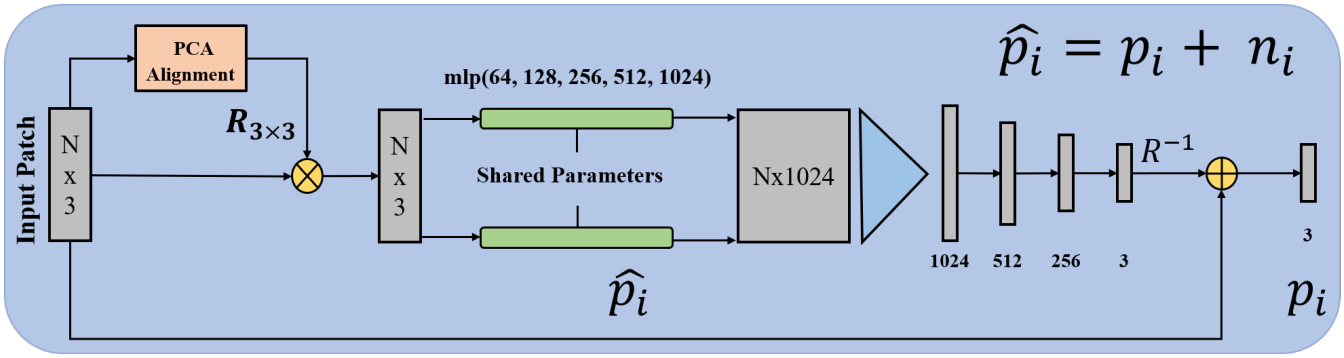


Figure 1: Pointfilter Framework:Given a noisy patch with N points, a pre-processing is applied before feeding into our proposed network. Normalized input passes through the shared multi-layer perceptrons (MLPs) to extract local features and then aggregates each points features by a max pooling layer. The MLPs consists of 5 hidden layers with neuron sizes 64, 128, 256, 512, 1024. Following the aggregated features, three fully connected layers, with neron sizes 512, 256, 3, are used to regress displacement vectors between noisy point and underlying surface. In addition, BatchNorm and ReLU are used for all layers except last layer. For last layer, only activation function TanH is employed to constrain displacement vectors space.

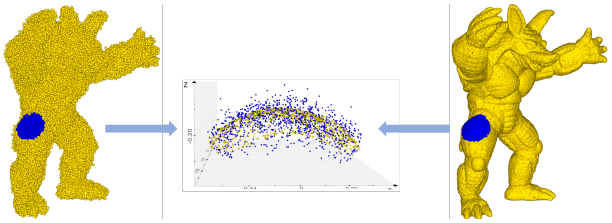


Figure 2: Illustrating our pre-processing: Given noisy point cloud (left) and clean point cloud (right), a pair of noisy patch and clean patch colored in blue are created. The results of pre-processing are shown in the middle (clean patch is colored in yellow to distinguish noisy patch).

where $\hat{P} = \{\hat{p_1}, \ldots, \hat{p_n} \mid \hat{p_i} \in R^3, i = 1 \ldots n\}$ is an observed point cloud corrupted with noise, P is the corresponding clean point cloud (underlying surface) and N is the additive noise. In this work, we address the filtering problem in a local way, which means the filtered result of a noisy point only depends on its neighboring structure. As we know, point cloud filtering is an ill-posed problem and it is difficult to straightforwardly regress the additive noise for each noisy point like image filtering. As an alternative, we handle point cloud filtering by projecting each noisy point onto the underlying surface. More specifically, we treat the additive noise N as displacement vectors between the noisy point cloud $\hat{P}$ and the clean point cloud P, and learn the displacement vector for each noisy point. To achieve this, we propose an encoder-decoder architecture network, named Pointfilter, to regress the additive noise N, shown in Fig. 1. We briefly introduce a pre-processing step for the input data in Section 3.2, and then show how to model our Pointfilter in Section 3.3. We finally explain how we train our network in Section 3.4 and how we make inference with the trained network in Section 3.5.

3.2 Preprocessing

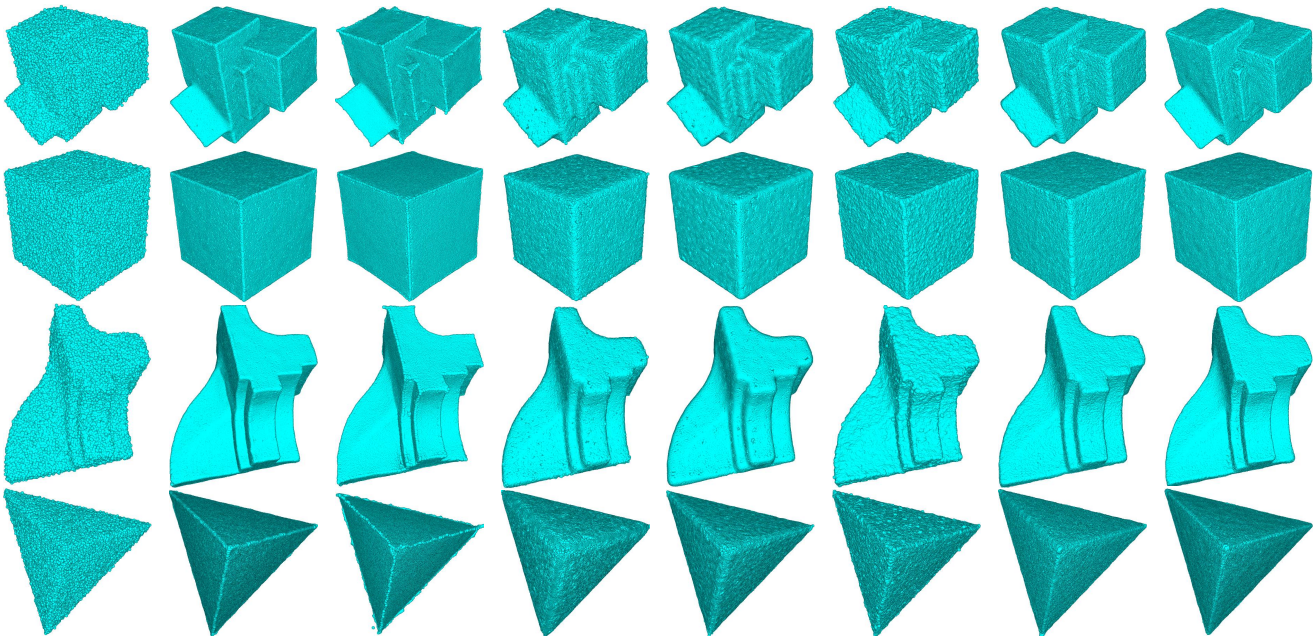
Given a pair of point clouds P and $\hat{P}$, the noisy patch $\hat{\mathcal{P}}_i$ and its corresponding ground truth patch $\mathcal{P}_i$ are defined as follows

$$\hat{\mathcal{P}}_i = \{\hat{p}_j \mid \|\hat{p}_j - \hat{p}_i\| < r\}, \quad \mathcal{P}_i = \{p_j \mid \|p_j - \hat{p}_i\| < r\}, \tag{2}$$

where $\hat{p}_i, \hat{p}_j \in \hat{P}$, $p_j \in P$ and r is the patch radius. Once patches are generated, two issues need to be addressed in point cloud filtering: (1) how to avoid unnecessary degrees of freedom from observed space? (2) how to guarantee our Pointfilter is insensitive to certain geometric transformations (e.g. rigid transformations)? For the first issue, an immediate remedy is to translate patches into origin and then scale them into unit length, i.e., $\hat{\mathcal{P}}_i = (\hat{\mathcal{P}}_i - \hat{p}_i)/r$. Similarly, the ground truth patch $\mathcal{P}_i$ does the same thing, i.e., $\mathcal{P}_i = (\mathcal{P}_i - \hat{p}_i)/r$. To be invariant to rigid transformations (e.g., rotations), a few methods [Charles et al. 2017; Qi et al. 2017] attempted to predict rotation matrix $R \in SO(3)$ via an additive spatial transformer network, while it has been proven to be fragile to rotations without massive data augmentation [You et al. 2018]. In this work, we align the input patches by means of aligning their principle axis of the PCA axes with the Z-axis. The alignment process is illustrated in Fig. 2. To effectively tune network parameters with batch processing, the number of points in each input patch should be the same. In experiments, we empirically set $|\hat{\mathcal{P}}_i| = 500$ by default. We pad the origin for patches with insufficient points (< 500) and do random sampling for patches with sufficient points (> 500). As for patch generation, the patch radius r is default to 5% of the model's bounding box diagonal.

3.3 The Pointfilter Framework

The architecture of our point cloud filtering framework is demonstrated in Fig. 1. The key idea of our Pointfilter is to project each noisy point onto the underlying surface according to its neighboring structure. To achieve this, we design our Pointfilter network as an encoder-decoder network. Specifically, the encoder consists of two main parts: (1) feature extractors (i.e., MLPs) that are used to extract different scales of features; (2) a collector that is used to aggregate the features ($N \times 1024$) as a latent vector $z \in R^{1024}$. The



(a) Noisy (b) RIMLS (c) GPF (d) WLOP (e) CLOP (f) EC-Net (g) PCN (h) Ours

Figure 3: Visual comparison of point cloud filtering with 0.5% noise. Our results are better than other state of art methods in terms of filtering and sharp feature preservation

encoder module attempts to obtain a compact representation for an input patch. In the decoder module, a regressor is employed to evaluate the displacement vectors with the latent representation vector z as input. In our paper, we adopt the recent PointNet [Charles et al. 2017] as the backbone in our Pointfilter. In practice, the extractors and collector are realised by the shared MLPs and max pooling layer, respectively, and the regressor is constructed by three fully connected layers. Details of our Pointfilter network are shown in Fig. 1. At the beginning of our Pointfiler, a PCA-induced rotation matrix R is applied to transform the input patch to a canonical space. Therefore, at the end of our Pointfiler, an inverse matrix R^{-1} should be multiplied by the evaluated displacement vector to get final displacement vector.

Loss function. To enable the filtered point cloud approximating the underlying surface while preserving sharp features, the loss function should be elaborately defined. A simple option for measuring the filtered point cloud would be the L_2 distance, which has been used in [Rakotosaona et al. 2019]. As shown in Fig. 4, compared with the $L2$-based distance (4 (b)) which is sampling dependent, a more general alternative is to project noisy points onto the underlying surface (4 (c), (d)). Moreover, the L_2-based distance can hardly retain sharp features in point cloud filtering. Thus, the loss should be capable of measuring the projection distance. Inspired by [Kolluri 2008], our projection loss is defined as

$$L^a_{proj} = \frac{\sum_{p_j \in \mathcal{P}_i} |(\bar{p_i} - p_j) \cdot n^T_{p_j}| \cdot \phi(\|\bar{p_i} - p_j\|)}{\sum_{p_j \in \mathcal{P}_i} \phi(\|\bar{p_i} - p_j\|)}, \tag{3}$$

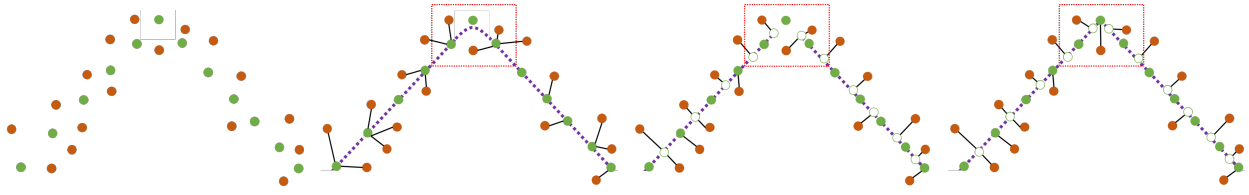
where $\bar{p_i}$ is the filtered point of the noisy point $\hat{p}_i$, and n_{p_j} is the ground-truth normal of the point p_j. And $\phi(\|\bar{p_i} - p_j\|)$ is a Gaussian function giving larger weights to points near $\bar{p}_i$, which is defined as

$$\phi(\|\bar{p_i} - p_j\|) = \exp -(\frac{\|\bar{p_i} - p_j\|}{\sigma_p})^2, \tag{4}$$

where σ_p is the support radius which is normally defined as $\sigma_p = 4\sqrt{diag/m}$. Here, $diag$ is the diagonal length of the bounding box of $\mathcal{P}_i$ and ~~$m = |\hat{\mathcal{P}}_i|$~~ [Huang et al. 2009]. Besides approximating the underlying surface, we also assume that the filtered point cloud should have a relatively regular distribution. To do this, a repulsion term is added to mitigate points aggregation. Therefore, the whole loss function is formulated as:

$$L = \eta L^a_{proj} + (1 - \eta) L_{rep}, \quad L_{rep} = \max_{p_j \in \mathcal{P}_i} |\bar{p_i} - p_j|, \tag{5}$$

where η is a trade-off factor to control the repulsion force in the filtering process and we empirically set $\eta = 0.97$ in our training stage. However, we observed that the above project loss L^a_{proj} would generate gaps near sharp features, since the normal-based distance would increase in the geometric dissimilarity by definition (see Fig. 4 (c)). Although the regular loss function L_{rep} can alleviate gaps to some extent, it still fails to preserve sharp feature duing filtering process (see Fig. 5) (b)). We address this issue by considering the normal similarity in our loss function, in which we introduce a bilateral mechanism to construct the projection distance formula (Eq. (3)). Specifically, the function is defined as the normal similarity between the current point and its neighboring points in the patch. For



(a) Noisy (b) L_2 (c) L^a_{proj} (d) L^b_{proj}

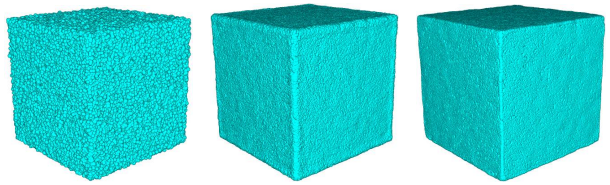
Figure 4: An illustration of different loss functions. Solid green points denote ground-truth points and solid dark-red points denote the noisy points. The dotted purple line represents the underlying surface induced by the corresponding loss function. Compared with directly mapping noisy points back to sampled points (b), our key idea is to project each noisy point onto the underlying surface (hollow green points), which is more reasonable for the ill-posed point cloud filtering (c-d). See red dashed boxes for the differences.

simplicity, the function is $\theta(n_{\bar{p}_i}, n_{p_j}) = \exp -(\frac{1 - n^T_{\bar{p}_i} n_{p_j}}{1 - cos(\sigma_n)})$ [Huang et al. 2013], where $n_{\bar{p}_i}$ is the normal of filtered point $\bar{p}_i$ and σ_n is support angle that the default value is 15°. Thus, our final projection function is defined as

$$L^b_{proj} = \frac{\sum_{p_j \in \mathcal{P}_i} |(\bar{p}_i - p_j) \cdot n^T_{p_j}| \cdot \phi(\|\bar{p}_i - p_j\|)\theta(n_{\bar{p}_i}, n_{p_j})}{\sum_{p_j \in \mathcal{P}_i} \phi(\|\bar{p}_i - p_j\|)\theta(n_{\bar{p}_i}, n_{p_j})}. \quad (6)$$

For efficiency and simplicity, the normal of the filtered point $n_{\bar{p}_i}$ is assigned by the normal of the ground truth point which is nearest to the filtered point. *It should be noted that our Pointfilter only requires ground-truth point normals in the training stage.*

We chose the encoder-decoder structure because: (1) it is stable and mature; (2) it can learn complex and compact representations of point cloud; (3) the learned latent representations are helpful to regress the displacement vector according to the input noisy patch.



(a) Noisy (b) L^a_{proj} (c) L^b_{proj}

Figure 5: Comparison of our two projection loss L^a_{proj} and L^b_{proj}. Our final projection loss L^b_{proj} has obvious superiority in terms of sharp features preservation.

3.4 Network Training

Our Pointfilter is implemented in PyTorch on a desktop machine with an Intel Core I7-8750H CPU (2.20 GHz, 16GB memory) and a GeForce GTX 1060 GPU (6GB memory, CUDA 9.0). The epoch number of the training stage is 50. SGD is set as our optimizer and the learning rate is decreased from 1e-4 to 1e-8. The mini-batch size is 64 and batch-normalization [Ioffe and Szegedy 2015], ReLU [Nair and Hinton 2010] and TanH are used in our Pointfilter framework.

3.5 Network Inference

Given a trained Pointfilter, our approach filters noisy point cloud in a point-wise way. Firstly, we build a patch structure for each noisy point and transform each patch to a canonical space by following Section 3.2. Secondly, each pre-processed patch is fed into the trained Pointfilter to infer a displacement vector. Finally, the displacement vector evaluated by our Pointfilter should be mapped back to the original space. The inference can be formulated as follows:

$$\bar{p}_i = R^{-1} * r * f(R * (\hat{\mathcal{P}}_i - \hat{p}_i)/r) + \hat{p}_i, \quad (7)$$

where $\bar{p}_i$ and $\hat{p}_i$ are the filtered point and noisy point, respectively. f represents our Pointfilter. R is the PCA-induced rotation matrix, and r is the patch radius. To get better filtered results, we adopt multiple iterations of inference to progressively filter the noisy point cloud, especially for point clouds corrupted with larger noise.

Figure 6: Our training dataset.

4 EXPERIMENTAL RESULTS

4.1 Dataset

As a supervised learning method, we prepare a training dataset consisting of 22 3D clean models (11 CAD models and 11 non-CAD models) which are shown in Fig. 6. Each model is generated by randomly sampling 100k points from its original surface. Given a clean model, its corresponding noisy models are synthesized by adding Gaussian noise with the standard deviations of 0.0%, 0.25%,

0.5%, 1%, 1.5% and 2.5% of the clean model's bounding box diagonal. In sum, our training dataset contains 132 (22×6) models. *Notice that these models are our final training dataset, and we do not augment any data on-the-fly in training.* Besides, the normal information for clean models are required for training, as indicated in Eq. (6).

To demonstrate the generalization of the proposed Pointfilter, our test dataset includes both synthesized noisy models and raw-scan models, which will be explained in the following experiments (Section 4.4 and 4.5).

4.2 Compared Techniques

We compare our approach with the state-of-the-art point cloud filtering techniques, namely WLOP [Huang et al. 2009], CLOP [Preiner et al. 2014], RIMLS [Öztireli et al. 2009], GPF [Lu et al. 2018], EC-Net [Yu et al. 2018a] and PointCleanNet (PCN) [Rakotosaona et al. 2019]. Specifically, RIMLS and GPF are designed to preserve sharp features by incorporating smoothed normals. For fair comparisons and visualization purposes, we (i) tune the main parameters of each state of the art technique to achieve as good visual results as possible (EC-Net, PCN and our method have fixed parameters); (ii) employ the same surface reconstruction parameters for the same model. Notice that surface reconstruction is straightforwardly applied to the filtered point sets. As for PCN, we use the source code released by the authors to train a new model over our training dataset. Since EC-Net requires manually labelling polylines of sharp edges for training, which is infeasible for training a new model on our training dataset, we simply utilize the trained model released by the authors instead. We compare our method with these methods, in terms of both visual quality and quantity.

4.3 Evaluation Metric

For the sake of analysing performance of our Pointfilter quantitatively, the evaluation metrics should be defined. The Pointfilter aims to project noisy points onto its underlying surface. It is intuitive to evaluate the distance errors by averaging the distances between a point in the ground truth and its closest points in the filtered point cloud [Lu et al. 2018]. The distance error between two models can be defined as

$$D(p_i) = \frac{1}{M} \sum_{\bar{p}_j \in NN(p_i)} \|p_i - \bar{p}_j\|_2^2, \tag{8}$$

where p_i is the ground truth point and $\bar{p}_j$ is one of its neighboring point in the filtered point cloud. ~~$M = |NN(p_i)|$~~ and NN represents the nearest neighbors. In our paper, we set M as 10. Inspired by [Fan et al. 2017], we also introduce the Chamfer Distance (CD) to evaluate the error between the filtered point cloud and its corresponding ground truth (clean point cloud). CD is defined as

$$C(P, \bar{P}) = \frac{1}{N_1} \sum_{p_i \in P} \min_{\bar{p}_j \in \bar{P}} (\|p_i - \bar{p}_j\|_2^2) + \frac{1}{N_2} \sum_{p_j \in P} \min_{\bar{p}_i \in \bar{P}} (\|p_j - \bar{p}_i\|_2^2), \tag{9}$$

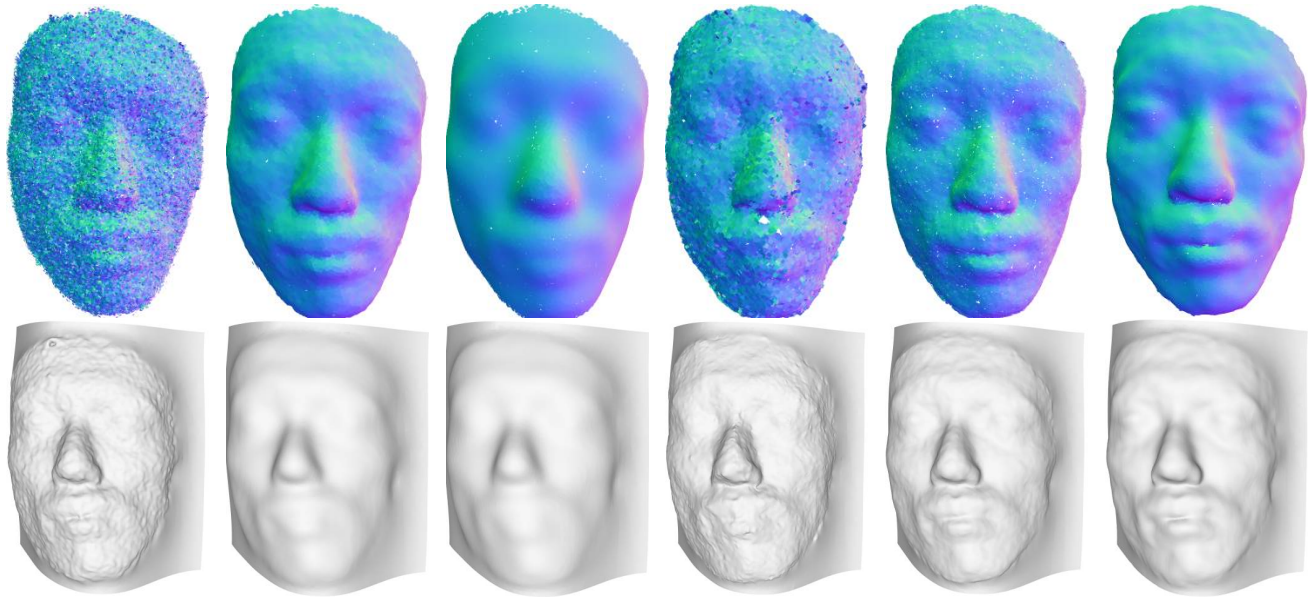
where N_1 and N_2 represent the cardinalities of the clean point cloud P and the filtered point cloud $\bar{P}$, respectively. The CD metric finds the nearest neighbor in the other set and sums the squared distances up. It can be viewed as an indicator function which measures the "similarity" between two point sets. Also, it can be easily implemented in parallelism.

4.4 Visual Comparisons

Point clouds with synthetic noise. The synthetic noise level is estimated by the diagonal length of the bounding box. For example, 0.5% noise denotes 0.5% of the diagonal length. As shown in Fig. 3, we test four CAD models (Boxunion, Cube, Fandisk and Tetrahedron) with 0.5% noise. Compared with the state-of-the-art point cloud filtering techniques, we observe that results by our Pointfilter generates visually better results, in terms of noise removal and features preservation. Note that RIMLS and GPF can also preserve sharp features to some extent; however, they depend greatly on the capability of normal filters which become less robust when meeting large noise. Compared to RIMLS and GPF, we elegantly detour the normal filtering issue since our framework requires the easily obtained ground-truth normals for training only. As shown in Fig. 10, RIMLS and GPF produce less desired results when handling 1.0% noise, for example, obvious gaps in sharp edges (10 (b)) and striking outliers (10 (c)). By contrast, our Pointfilter can also preserve sharp features. Since the outliers exist in RIMLS and GPF results, their filtered point clouds involve shrinkage to some extent. Despite that WLOP and CLOP are good at generating smooth results, they still fail to retain sharp features. Regarding EC-Net, it generates less pleasing results, in terms of removing noise. Besides CAD models, we also test some non-CAD models corrupted with synthetic noise. As shown in Fig. 7, our proposed Pointfilter can also output visually decent results while preserving the geometric features. The reconstruction results of WLOP and CLOP has slight shrinkage in arms and legs (Fig. 8 (b) and 8 (c)).

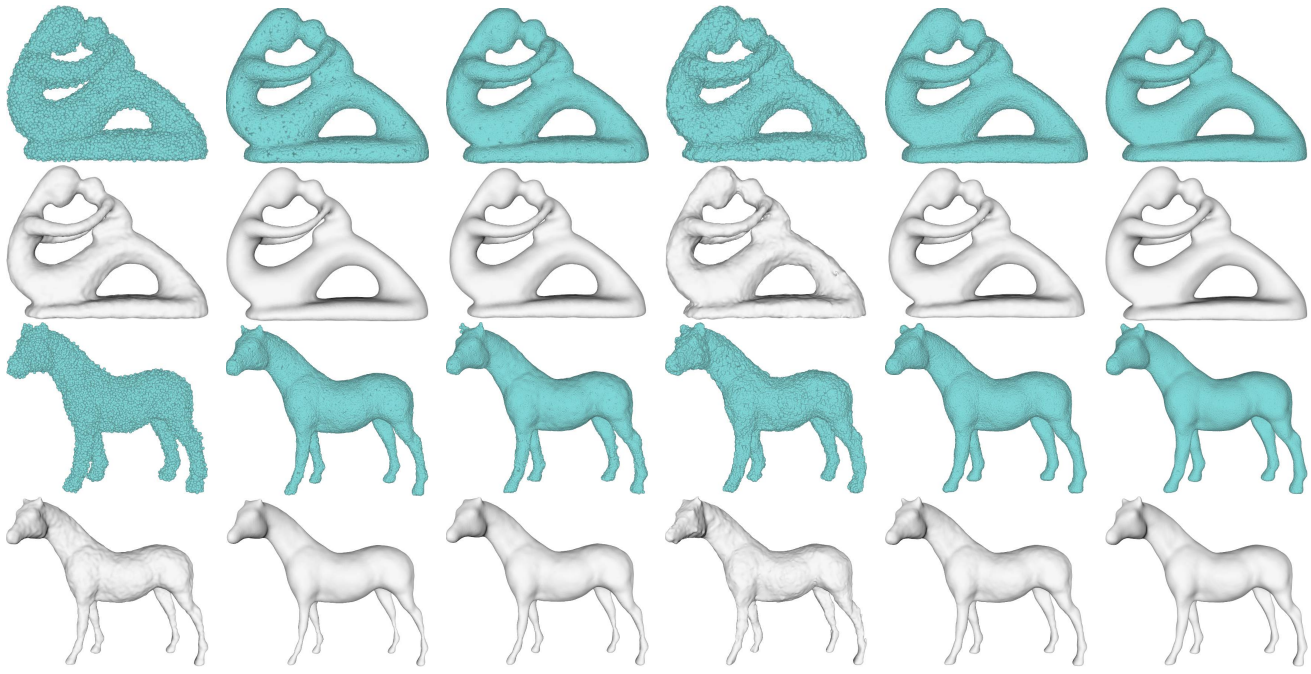
Point clouds with raw noise. We also evaluate our Pointfilter on raw scanned point clouds corrupted with raw noise. Since the ground truth models of these raw scanned point sets are not available, we demonstrate the visual comparisons with other methods, as suggested by previous techniques [Lu et al. 2018]. Notice that we do not re-train our Pointfilter for the type of raw noise. From Fig. 11, we see that the result by our method is nicer than the state-of-the-art techniques. Besides noise removal, our Pointfilter is capable of retaining geometric features which are marked by yellow box and black arrows in Fig. 11. Fig. 13 shows that our approach induces a better enhancement on the surface reconstruction quality, in terms of preserving geometric features. Fig. 12 shows a virtually scanned point cloud model. Compared with other filtering methods, our Pointfilter still produces higher quality results, in terms of preserving sharp edges. In addition, we also test our Pointfilter on scene-level models from the Paris-rue-Madame Database [Serna et al. 2014] (see Fig. 18 and Fig. 19). As we can see from these figures, our Pointfilter is still able to produce better results.

Point clouds with strong outliers. Although the Pointfilter is not particularly designed for outliers removal, we still produce competitive results in point clouds with large outliers. As suggested by [Lu et al. 2018], we also conduct an experiment by comparing our method with LOP-based methods (WLOP, CLOP) which are robust to outliers due to the $L1$-norm term. This experiment demonstrates the capability of our Pointfiler in dealing with strong outliers. From



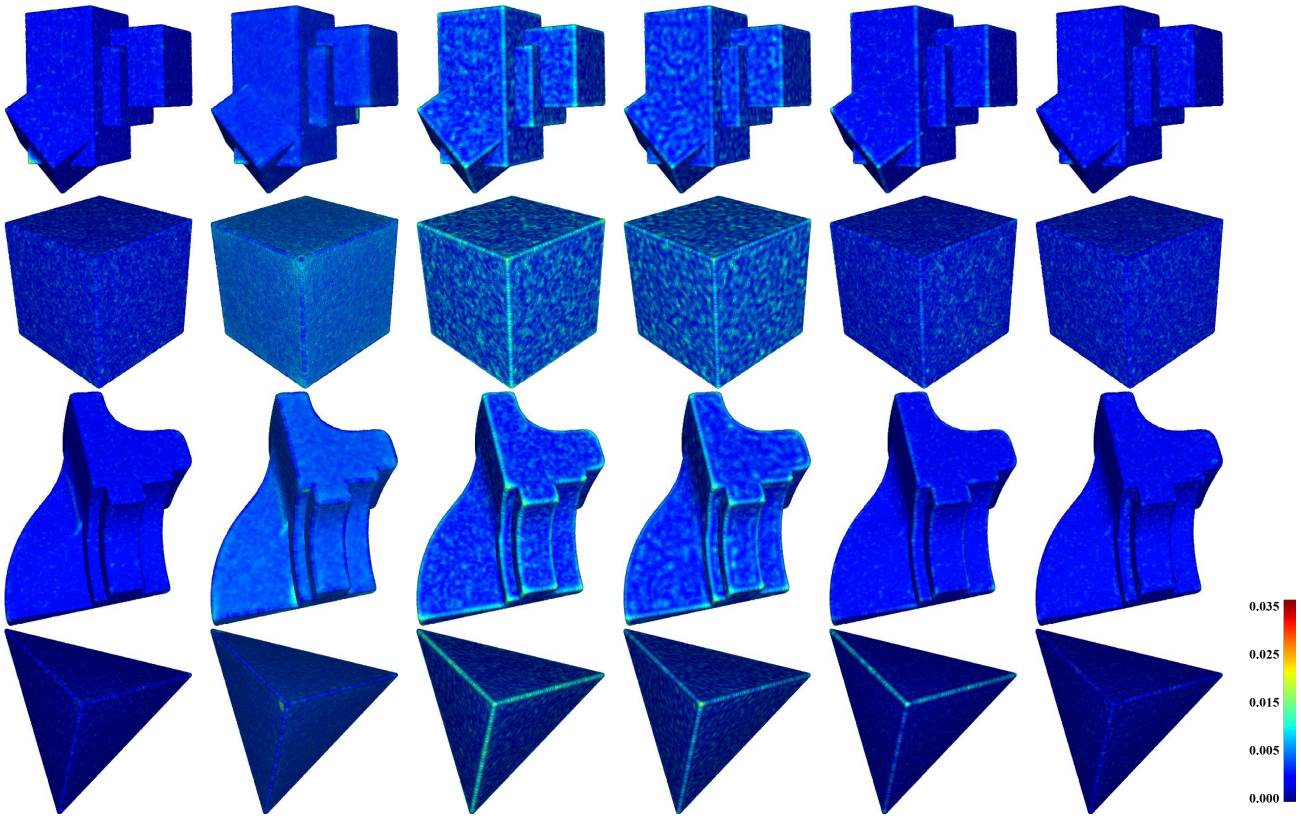
(a) Noisy (b) WLOP (c) CLOP (d) EC-Net (e) PCN (f) Ours

Figure 7: Visual comparison of point clouds filtering with 0.5% synthetic noise. The corresponding normals and Poisson reconstruction results are shown in the middle and bottom, respectively.



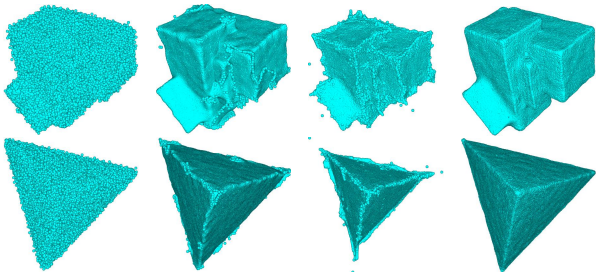
(a) Noisy (b) WLOP (c) CLOP (d) EC-Net (e) PCN (f) Ours

Figure 8: Visual comparison of point clouds filtering with 0.5% synthetic noise. The corresponding Poisson reconstruction results are shown in second and fourth row.



(a) RIMLS (b) GPF (c) WLOP (d) CLOP (e) PCN (f) Ours

Figure 9: Quantitative comparison of point clouds.



(a) Noisy (b) RIMLS (c) GPF (d) Ours

Figure 10: The filtered results corrupted with larger noise: [a] Point clouds with 1.0% noise. [b] The results of RIMLS. [c] The results of GPF. [d] The results of our.

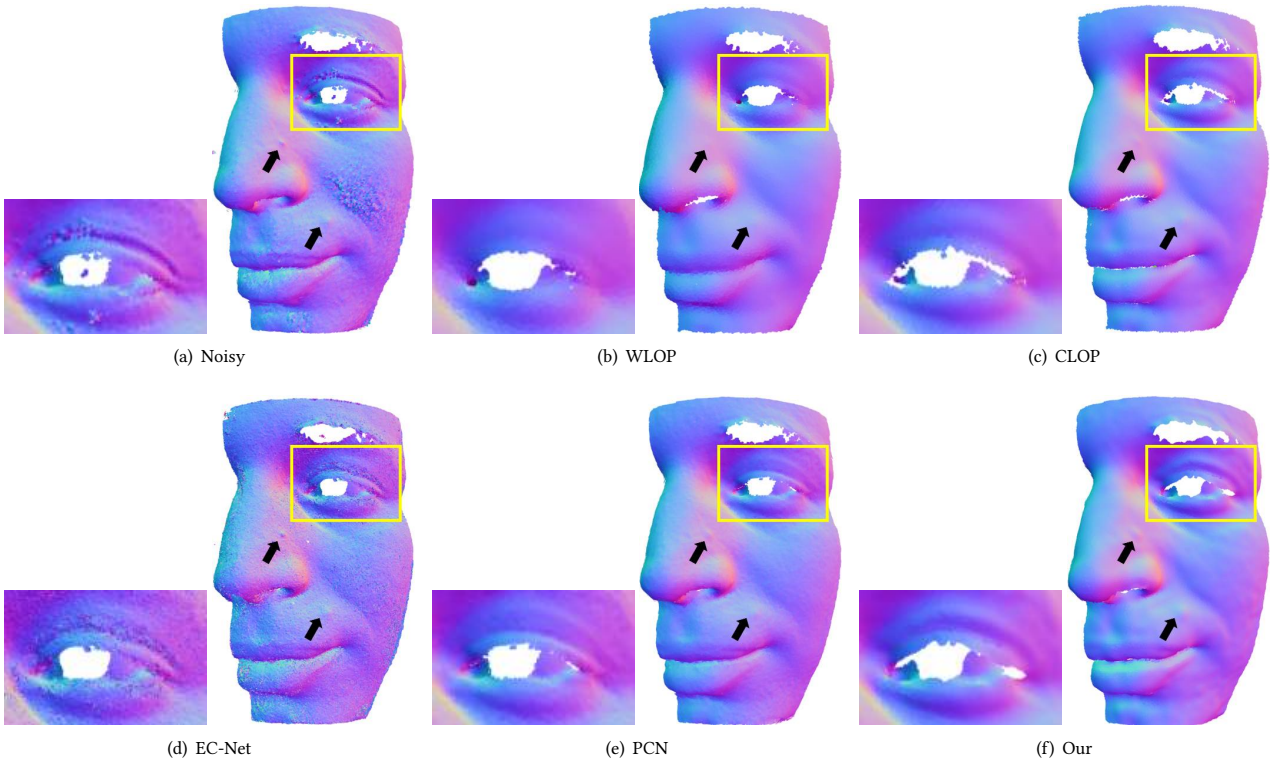
the results in Fig. 14, we can observe that the Pointfilter can also generate a comparable result to WLOP and CLOP.

4.5 Quantitative Comparisons

Besides the above visual comparisons, we also contrast all the methods quantitatively. Specifically, we compare the above defined metrics, as well as the runtime for all methods.

Errors. We calculate the above metrics of all compared methods on some point clouds. These point sets are achieved by adding synthetic noise to the ground truth. To depict the distance errors, we calculate the mean square error (MSE) for each ground truth point via Eq. (8) and visualize the results in Fig. 9. From these visualization results, it is seen that the Pointfiler generates comparable or surpassing results, especially on the sharp features. To comprehensively evaluate our Pointfiler, we also introduce the chamfer distance (CD) to compute the overall error between a filtered point cloud and its ground truth. As illustrated in Fig. 15, we calculate the average errors of the models appeared in Fig. 9 in terms of MSE and CD, respectively. Despite that the results of RIMLS are comparable to our results, it requires trial-and-error parameter tuning to obtain satisfactory results. Moreover, such parameter tuning is boring and time-consuming, and even becomes difficult for users who do not have any background knowledge. By contrast, our method is automatic and simple for users to use, and is the most accurate one among all compared approaches.

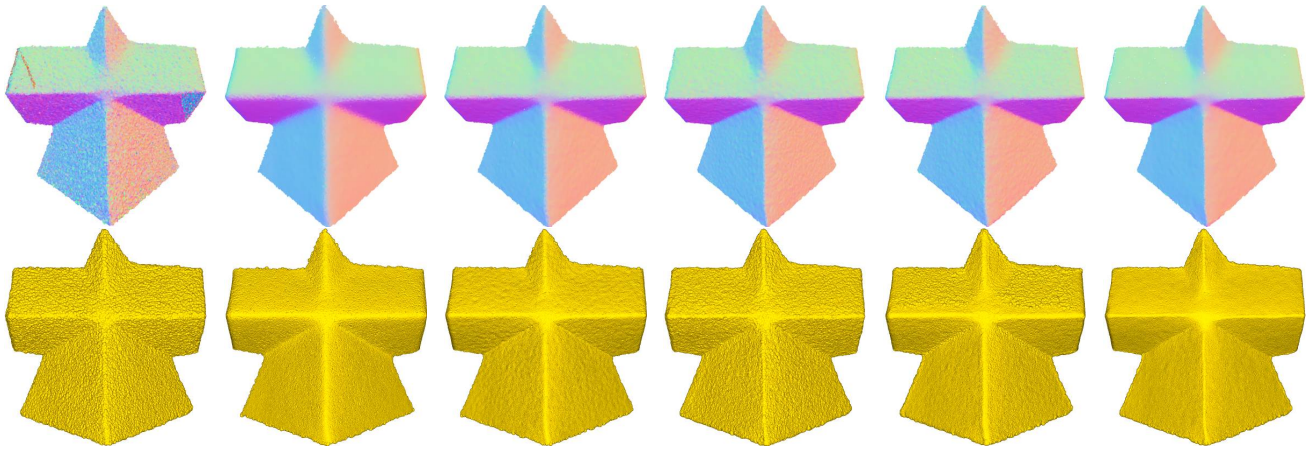
Runtime. Because surface reconstruction is an application over point clouds, we only calculate the runtime of each point set filtering method. In particular, optimized-based methods (RIMLS, GPF, WLOP and CLOP) involve multiple steps and require trial-and-error efforts to tune parameters to produce decent visual results, which



(a) Noisy (b) WLOP (c) CLOP

(d) EC-Net (e) PCN (f) Our

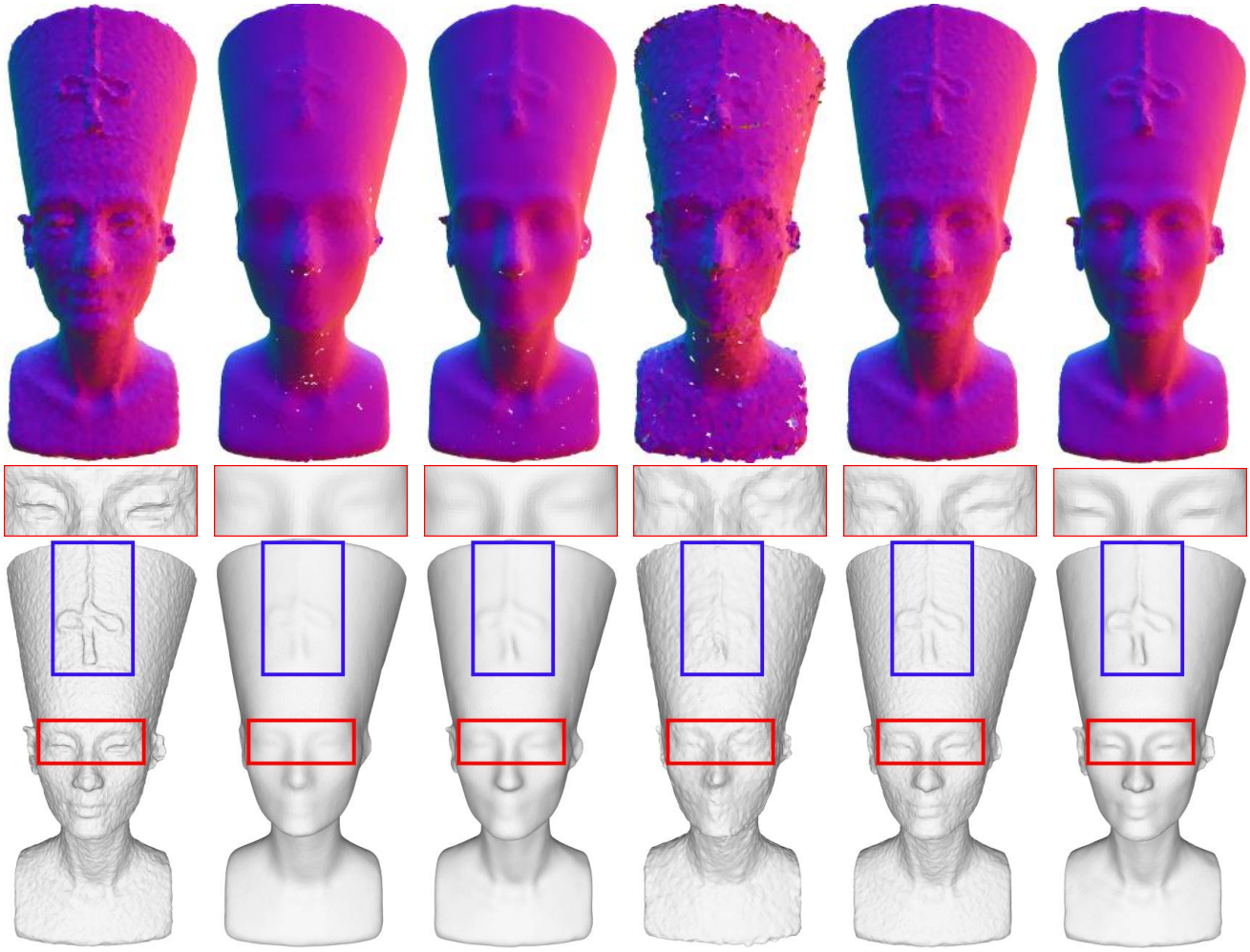
Figure 11: Filtered normal results of raw Face point clouds. Please refer to the zoomed regions.



(a) Noisy (b) WLOP (c) CLOP (d) EC-Net (e) PCN (f) Ours

Figure 12: Filtered results of raw Pyramid point clouds.

means these methods generally require a much longer "runtime" in practice. Thus, we only consider learning-based methods (EC-Net, PCN and Our), in terms of time consumption in the test stage. Table 2 summaries the runtime of each learning-based method on some point clouds using the same configuration. Table 2 sees that EC-Net is the fastest method among the learning-based methods, as it is an upsampling method and only requires a few patches evenly distributed on the noisy input in the test phase. For fair comparisons, we only consider the runtime of the noise removal module in PCN, because extra time consumption would be introduced for the outliers removal module. In spite of this, PCN is still the slowest



(a) Noisy (b) WLOP (c) CLOP (d) EC-Net (e) PCN (f) Ours

Figure 13: Filtered results of raw Nefertiti point clouds. The corresponding Poisson reconstruction results are shown in the bottom. Please refer to the zoomed regions.

one. Our approach ranks the second in speed, and we suspect that it is due to the point-wise manner.

5 CONCLUSION

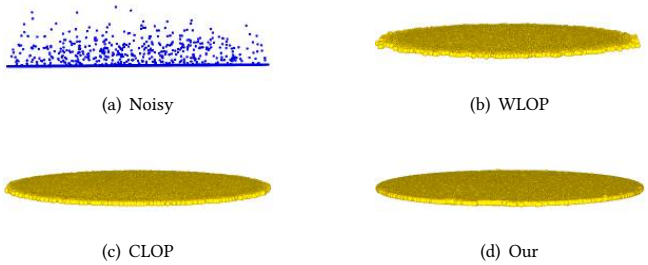
In this paper, we proposed a Pointfilter framework for feature-preserving point cloud filtering. Our architecture can be easily trained. Given an input noisy point cloud, our method can automatically infer the involved displacement vectors and further the filtered point cloud with preserved sharp features. Extensive experiments and comparisons showed that our method outperforms the state-of-the-art point set filtering techniques (or comparable to optimization based methods like RIMLS and GPF which need trial-and-error parameter tuning), in terms of both visual quality and evaluation errors. Our approach is automatic and also achieves impressive performance on test time. Compared to PCN [Rakotosaona et al. 2019], it should be noted that our Pointfilter is not designed for larger-scale outliers removing, and our Pointfiler and PCN are thus complementary in terms of sharp features preservation and heavy outliers removal.

Our method involves a few limitations. First, our Pointfilter becomes hard to retain the sharp features when handling excessive noise (see Fig. 16). Secondly, our method fails to handle significant holes in point clouds (see Fig. 17). In future, we would like to incorporate global shape information into our framework to help guide point cloud filtering.

REFERENCES

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

Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. 2001. Point Set Surfaces. In *Proceedings of the Conference on Visualization '01 (VIS '01)*. IEEE Computer Society, Washington, DC, USA, 21–28. http://dl.acm.org/citation.cfm?id=601671.601673



(a) Noisy (b) WLOP

(c) CLOP (d) Our

Figure 14: Robust to strong outliers of LOP-based methods (e.g., WLOP and CLOP) and our Pointfilter. The noisy point clouds with strong outliers and the corresponding filtered results are colored in blue and yellow, respectively. For all these methods, big radii are employed to deal with strong outliers.

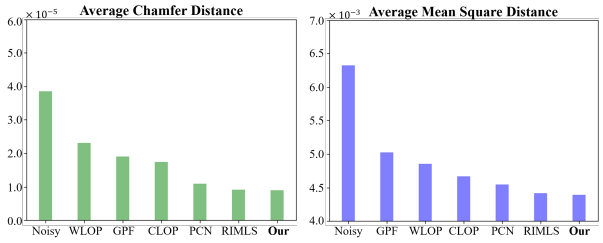
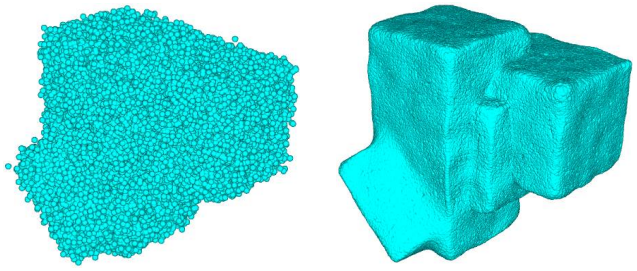


Figure 15: The average metric errors of filtered point clouds (Figure 9) for all compared methods in terms of MSE and CD, respectively.

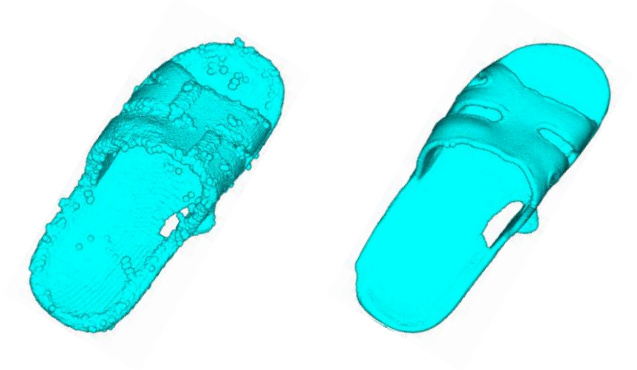
Table 2: Runtime Summaries (in seconds) for three learning-based methods in test stage. All examples were run on the same computer configurations (Section .3.4).

Methods / Models	EC-Net	PCN	**Our**(s)
Cube	27.73	360.64	62.34
Fandisk	26.92	369.67	62.45
Boxunion	26.15	365.09	65.41
Tetrahedron	28.64	326.11	63.21
Horse	26.91	365.62	63.55
Face-Yo	27.27	362.26	63.08
Fertility-tri	27.43	370.21	63.52
Face-Raw	22.63	306.39	55.71
Pyramid-Raw	44.74	618.98	105.85
Nefertiti-Raw	27.00	353.44	62.38



(a) Noisy (b) Our

Figure 16: A failure case of our Pointfilter when handling excessive noise.



(a) Noisy (b) Our

Figure 17: Our Pointfilter fails to handle significant holes.

Nina Amenta and Yong Joo Kil. 2004. Defining Point-set Surfaces. *ACM Transactions on Graphics*. 23, 3 (Aug. 2004), 264–270. https://doi.org/10.1145/1015706.1015713

Haim Avron, Andrei Sharf, Chen Greif, and Daniel Cohen-Or. 2010. ℓ1-Sparse reconstruction of sharp point set surfaces. *ACM Transactions on Graphics* 29, 5 (Oct. 2010), 1–12. https://doi.org/10.1145/1857907.1857911

R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. https://doi.org/10.1109/cvpr.2017.16

Haoqiang Fan, Hao Su, and Leonidas J Guibas. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*. IEEE, 605–613.

Shachar Fleishman, Daniel Cohen-Or, and Cláudio T. Silva. 2005. Robust Moving Least-squares Fitting with Sharp Features. *ACM Transactions on Graphics*. 24, 3 (July 2005), 544–552. https://doi.org/10.1145/1073204.1073227

Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J. Mitra. 2018. PCPNet: Learning Local Shape Properties from Raw Point Clouds. *Computer Graphics Forum* 37, 2 (May 2018), 75–85. https://doi.org/10.1111/cgf.13343

Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. 2009. Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics* 28, 5 (Dec. 2009), 1. https://doi.org/10.1145/1618452.1618522

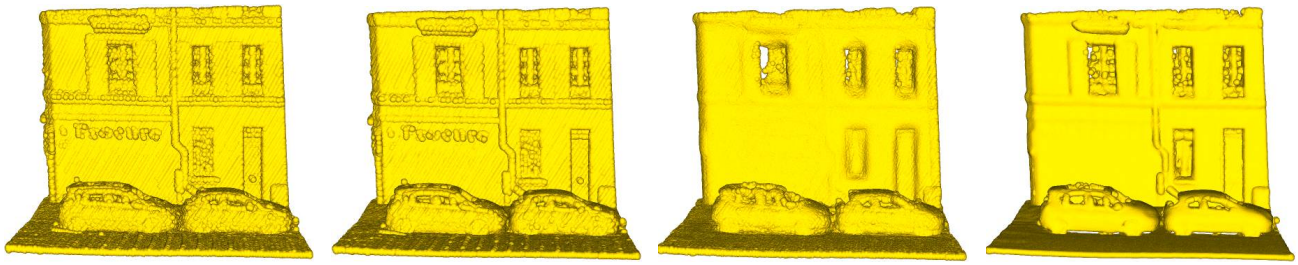
Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao (Richard) Zhang. 2013. Edge-aware point set resampling. *ACM Transactions on Graphics* 32, 1 (Jan. 2013), 1–12. https://doi.org/10.1145/2421636.2421645

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. 2018. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652* (2018).

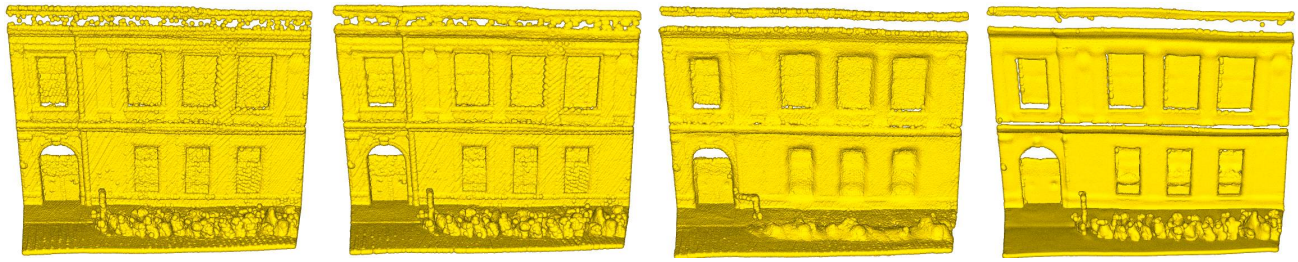
Ravikrishna Kolluri. 2008. Provably good moving least squares. *ACM Transactions on Algorithms* 4, 2 (May 2008), 1–25. https://doi.org/10.1145/1361192.1361195

Carsten Lange and Konrad Polthier. 2005. Anisotropic smoothing of point sets. *Computer Aided Geometric Design* 22, 7 (Oct. 2005), 680–692. https://doi.org/10.1016/j.cagd.2005.06.010



(a) Noisy (b) EC-Net (c) PCN (d) Our

Figure 18: Filtering results of the noisy scanned data ($600K$) sampled from the Paris-rue-Madame Database [Serna et al. 2014].



(a) Noisy (b) EC-Net (c) PCN (d) Our

Figure 19: Filtering results of the noisy scanned data ($1000K$) sampled from the Paris-rue-Madame Database [Serna et al. 2014].

David Levin. 1998. The approximation power of moving least-squares. *Math. Comp.* 67, 224 (Oct. 1998), 1517–1532. https://doi.org/10.1090/s0025-5718-98-00974-0

David Levin. 2004. *Mesh-Independent Surface Interpolation.* Springer Berlin Heidelberg, 37–49. https://doi.org/10.1007/978-3-662-07443-5_3

Bin Liao, Chunxia Xiao, Liqiang Jin, and Hongbo Fu. 2013. Efficient feature-preserving local projection operator for geometry reconstruction. *Computer-Aided Design* 45, 5 (May 2013), 861–874. https://doi.org/10.1016/j.cad.2013.02.003

Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. 2007. Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics* 26, 3 (July 2007), 22. https://doi.org/10.1145/1276377.1276405

David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60, 2 (Nov. 2004), 91–110. https://doi.org/10.1023/b:visi.0000029664.99615.94

Xuequan Lu, Scott Schaefer, Jun Luo, Lizhuang Ma, and Ying He. 2018. Low rank matrix approximation for geometry filtering. *arXiv preprint arXiv:1803.06783* (2018).

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

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML)*. 807–814.

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

Reinhold Preiner, Oliver Mattausch, Murat Arikan, Renato Pajarola, and Michael Wimmer. 2014. Continuous projection for fastL1reconstruction. *ACM Transactions on Graphics* 33, 4 (July 2014), 1–13. https://doi.org/10.1145/2601097.2601172

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Proceedings of the Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5099–5108. http://papers.nips.cc/paper/7095-pointnet-deep-hierarchical-feature-learning-on-point-sets-in-a-metric-space.pdf

Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J. Mitra, and Maks Ovsjanikov. 2019. PointCleanNet : Learning to Denoise and Remove Outliers from Dense Point Clouds. *Computer Graphics Forum* (June 2019). https://doi.org/10.1111/cgf.13753

Riccardo Roveri, A. Cengiz Öztireli, Ioana Pandele, and Markus Gross. 2018. PointProNets: Consolidation of Point Clouds with Convolutional Neural Networks. *Computer Graphics Forum* 37, 2 (May 2018), 87–99. https://doi.org/10.1111/cgf.13344

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (Jan. 2015), 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

Andrés Serna, Beatriz Marcotegui, FranÃğois Goulette, and Jean-Emmanuel Deschaud. 2014. Paris-rue-Madame Database - A 3D Mobile Laser Scanner Dataset for Benchmarking Urban Detection, Segmentation and Classification Methods. In *ICPRAM*.

Yujing Sun, Scott Schaefer, and Wenping Wang. 2015. Denoising point sets via $\ell 0$ minimization. *Computer Aided Geometric Design* 35-36 (May 2015), 2–15. https://doi.org/10.1016/j.cagd.2015.03.011

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Transactions on Graphics* 38, 5 (Oct. 2019), 1–12. https://doi.org/10.1145/3326362

Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. 2018. SpiderCNN: Deep Learning on Point Sets with Parameterized Convolutional Filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer International Publishing, 90–105. https://doi.org/10.1007/978-3-030-01237-3_6

Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. 2019. Patch-Based Progressive 3D Point Set Upsampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE.

Yang You, Yujing Lou, Qi Liu, Lizhuang Ma, Weiming Wang, Yuwing Tai, and Cewu Lu. 2018. PRIN: Pointwise Rotation-Invariant Network. *arXiv preprint arXiv:1811.09361* (2018).

Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018a. EC-Net: an Edge-aware Point set Consolidation Network. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 386–402.

Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. 2018b. PU-Net: Point Cloud Upsampling Network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. https://doi.org/10.1109/cvpr.2018.00295

Yinglong Zheng, Guiqing Li, Shihao Wu, Yuxin Liu, and Yuefang Gao. 2017. Guided point cloud denoising via sharp feature skeletons. *The Visual Computer* 33, 6-8 (May 2017), 857–867. https://doi.org/10.1007/s00371-017-1391-8

Yinglong Zheng, Guiqing Li, Xuemiao Xu, Shihao Wu, and Yongwei Nie. 2018. Rolling normal filtering for point clouds. *Computer Aided Geometric Design* 62 (May 2018), 16–28. https://doi.org/10.1016/j.cagd.2018.03.004