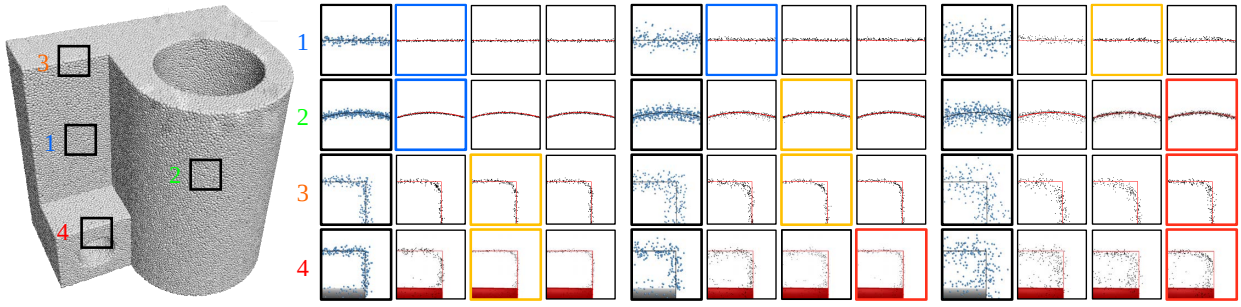
This article has been accepted for publication in
IEEE Transactions on Pattern Analysis and Machine Intelligence, 2024

PathNet: Path-selective Point Cloud Denoising

Zeyong Wei, Honghua Chen, Liangliang Nan, Jun Wang, Jing Qin, and Mingqiang Wei



(a) Ground truth (b) Denoising results (0.5% noise) (c) Denoising results (1% noise) (d) Denoising results (1.5% noise)

Fig. 1: PathNet to handle a typical 3D model with sharp edges and increased levels of noise. The testing validates that different levels of noise and geometries require a network with different depths. (b)-(d) are the denoising results of some representative regions, where the first column is the input fragment data visualized from a specific view, and the 2nd, 3rd, and 4th columns are the denoising results of the network with 1, 3, and 5 denoising blocks, respectively. The best results but with the fewest blocks are framed by colored rectangles. The best results for 1, 3, and 5 denoising block networks are marked by the blue, yellow, and red rectangles, respectively. Thus, an 'optimal' network path should be selected for each point based on its geometric characteristics and noise levels.

Abstract—Current point cloud denoising (PCD) models optimize single networks, trying to make their parameters adaptive to each point in a large pool of point clouds. Such a denoising network paradigm neglects that different points are often corrupted by different levels of noise and they may convey different geometric structures. Thus, the intricacy of both noise and geometry poses side effects including remnant noise, wrongly-smoothed edges, and distorted shape after denoising. We propose *PathNet*, a path-selective PCD paradigm based on reinforcement learning (RL). Unlike existing efforts, PathNet enables dynamic selection of the most appropriate denoising path for each point, best moving it onto its underlying surface. We have two more contributions besides the proposed framework of path-selective PCD *for the first time*. First, to leverage geometry expertise and benefit from training data, we propose a noise- and geometry-aware reward function to train the routing agent in RL. Second, the routing agent and the denoising network are trained jointly to avoid under- and over-smoothing. Extensive experiments show promising improvements of PathNet over its competitors, in terms of the effectiveness for removing different levels of noise and preserving multi-scale surface geometries. Furthermore, PathNet generalizes itself more smoothly to real scans than cutting-edge models. The source code is publicly available at: https://github.com/ZeyongWei/PathNet.git.

Index Terms—PathNet, point cloud denoising, path selection, reinforcement learning, geometry preservation

1 Introduction

Point clouds captured by 3D scanners or depth cameras are often corrupted by noise, due to both the measurement and reconstruction errors. Point cloud denoising (PCD) aims at recovering a clean point cloud to represent the underlying surface from its noisy scan(s). As deep learning goes mainstream in many research fields, impressive progress has been made in learning-based PCD methods, e.g., PointCleanNet [1], Pointfilter [2], and RePCD-Net [3]. However, these cutting-edge models usually denoise all points by a single deep or iterative model, neglecting the fact that different points are corrupted with different levels of noise and possess varying geometric structures. Such a PCD paradigm often leads to side effects including remnant noise, wrongly-smoothed edges, and distorted shape after denoising.

We raise an intriguing question: *Is it a good way to process all points of noisy point clouds with the same network?* The answer may be "NOT". For example, large variations of surface structures and noise intensities always exist in a complex and noisy point cloud. It is easy to deduce a simple fact that, the points that are very close to their ground-truth surface have already conveyed an object's structures well and they just need slight denoising or even avoid such a denoising operation; the points in sharp regions (i.e., the regions around edges and corners) need to be elaborately

- *Z. Wei, J. Wang and M. Wei are with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China (e-mail: weizeyong1@gmail.com; wjun@nuaa.edu.cn; mingqiang.wei@gmail.com).*
- *H. Chen is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore (e-mail: honghua.chen@ntu.edu.sg).*
- *L. Nan is with the Urban Data Science section, Delft University of Technology, Delft, Netherlands (e-mail: liangliang.nan@gmail.com).*
- *J. Qin is with the School of Nursing, Hong Kong Polytechnic University, Hong Kong, China (e-mail: harry.qin@polyu.edu.hk).*

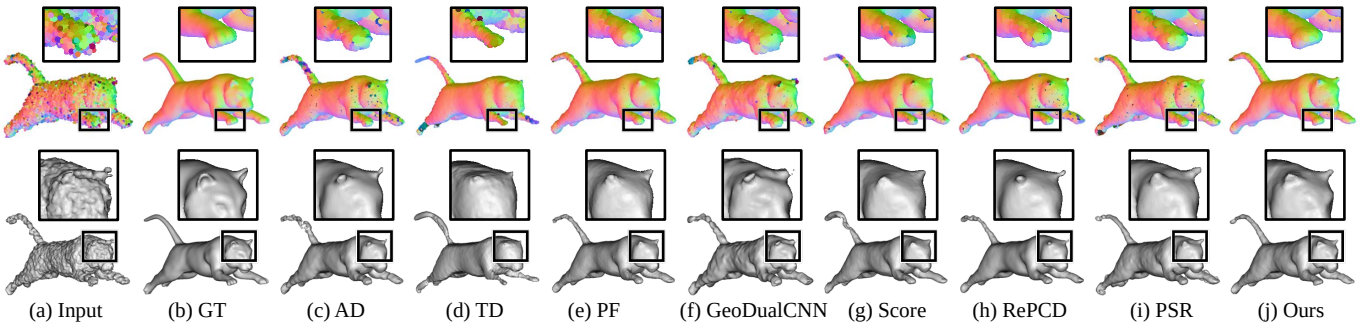


Fig. 2: Visual comparisons of the denoising results of our method (j) against state-of-the-art methods (c-i) including AD [4], TD [5], PF [2], GeoDualCNN [6], Score [7], RePCD [3] and PSR [8]. The input data is corrupted by 1.0% Gaussian noise. Our method outperforms the others in terms of both noise removal and feature preservation (see particularly the close-up views). The colorful points are colored by their normals calculated using the same algorithm. The discontinuity in color indicates noise residuals (legs) that cause errors in the normals. The gray models are the meshes reconstructed by Poisson surface reconstruction.

denoised to preserve the sharp features; while the points within piece-wisely smooth regions are inherently less difficult to denoise and can be achieved in a cheaper way. In general, sharp features of a point cloud are sensitive to denoise, since they fall under the category of geometric details and are considered as signals with high frequency [9]. Therefore, sharp features are easily smoothed and removed along with the noise. To this end, we need deeper networks to differentiate sharp features and noise, while shallow networks are enough to deal with the smooth (planar) regions.

If we adhere to the prevailing wisdom in PCD, a network with a single denoising path is typically employed to denoise an entire point cloud, without taking into account the potential presence of complex noise patterns and various structural elements. Utilizing a network with a single denoising path to process all points within a noisy point cloud indiscriminately is conceptually less effective. Consequently, this approach can lead to the presence of noise residuals in the denoised point clouds of heavily-noised datasets and result in geometric over-smoothing when applied to low-noise point clouds.

A new PCD paradigm that can select a suitable network path for each noisy point is promising to avoid the aforementioned shortcomings. We show a typical CAD-like model that contains planar regions, edges, and corners, as well as different levels of noise in Fig. 1: In (b)-(d), the first column is the input data, and the 2nd, 3rd, and 4th columns are the denoising results of the network with 1, 3, and 5 denoising blocks respectively. As observed, points within the blue rectangles are located in smooth regions and thus can be easily denoised with one denoising block. Increasing the network depth or the number of iterations will increase computation and running time but without clear improvements in denoising quality. In the yellow rectangles, points with moderate noise or geometry features are recovered to the ground-truth surface with three denoising blocks. The points in the red rectangle located in geometry sharp regions or contain intensive noise, require five denoising blocks or more iterations to effectively remove the noise.

Inspired by image restoration [10], we formulate PCD as a decision-making process by which an agent dynamically selects a denoising path for each point. We call the path-selective PCD network PathNet. To the best of our knowledge, we are the first to consider PCD as a reinforcement learning problem. In our work, the agent is formulated in a reinforcement learning (RL) framework. It learns to select the suitable path by analyzing the state of a denoised point in each step driven by a reward that improves the quality of the input point cloud while improving efficiency. Extensive experiments show that our approach achieves state-of-the-art performance compared to the previous methods.

Our main contributions are three-fold:

- We propose PathNet, a novel point cloud denoising network based on the *path-selective* strategy. PathNet selects the most suitable path for denoising each point.
- We provide the first implementation of PathNet based on reinforcement learning, which jointly trains the PCD network and agent for path selection. PathNet shows promising improvements over its competitors in terms of effectiveness and efficiency.
- We introduce a noise- and geometry-aware reward for PCD. This reward is devised to gain more attention to the performance for geometry feature regions and challenging scenarios.

As demonstrated in Fig. 2, the proposed PathNet exhibits better performance over both the traditional and learning-based methods in terms of normal rendering and Poisson surface reconstruction. More visual and numerical results can be found in Sec. 4.

2 Related Work

In this section, we review previous research from optimization-based methods to recent prevalent learning-based methods for PCD, followed by deep reinforcement learning.

2.1 Optimization-based Methods

Optimization-based methods that formulate PCD as an optimization problem can be classified into four categories, i.e., moving least-squares (MLS)-based, locally optimal projection (LOP)-based, sparsity-based, nonlocal-based, and graph-based methods.

MLS-based methods approximate a smooth surface from the input point cloud and achieve denoised results by projecting the noisy point cloud onto the estimated underlying surface [4], [11]–[16]. Alexa et al. [12] define a smooth manifold surface from input points [11] and address PCD based on MLS [17]. Guennebaud et al. [13] define algebraic point set surfaces using local MLS fitting of algebraic spheres. Meanwhile, Cazals et al. [14] propose jet fitting for n-order polynomial surfaces to calculate normals and curvatures. Fleishman et al. [15] and Oztireli et al. [16] extend [12] to preserve features and introduce robust statistics to reduce the sensitivity to outliers. Xu et al. [4] introduce the anisotropic denoising (AD) technique based on a dense aggregation of MLS

estimates characterized by asymmetric directional neighborhoods. This approach can adapt to edges and discontinuities using much larger supports than classical MLS based on symmetric weights. However, smooth surfaces reconstructed by these methods are always prone to over-smooth features together with noise.

LOP-based methods represent the underlying surface with a set of points instead of implicit surface parameters and enforce a uniform distribution of denoised points [18]. The original LOP [18] projects noisy points onto the underlying surface while employing a repulsion term to maintain the point uniformity. However, it is not suitable for point clouds with uneven density distribution and sharp features. Therefore, some improved methods are proposed, such as WLOP [19], EAR [20], FLOP [21], CLOP [22], GPF [23] and TUPD [24]. However, these methods require tedious parameter tuning and are sensitive to outliers.

Sparsity-based methods are based on the theory of sparse representations [25]. The theory assumes that common objects can be defined as piece-wise smooth surfaces with sparse features. Based on it, Avron et al. [26] and Sun et al. [27] adopt l_1 regularization and l_0 minimization respectively, to deal with noisy point clouds. They both retain sharp features well but suffer from undesired staircase effects in smoothly curved areas [28]. Mattei et al. [29] propose Moving Robust Principal Component Analysis (MRPCA). This approach models the point cloud as multiple overlapping 2D subspaces and computes estimated point locations through local average estimation. Digne et al. [30] define the local probing field (LPF) as a local frame and employ dictionary learning for sparse shape description. In our tests, when the noise is large, sparsity-based methods may lead to inaccurate denoising results due to poor normal estimation.

Nonlocal-based methods are based on the geometric statistics that many surface patches sharing similar geometric properties always exist within a 3D model. This kind of method exploits the non-local similarities among surface patches to collaboratively filter a noisy point cloud [31]–[36]. The main challenge of these methods is the regular representation of irregular local point cloud structures. Lu et al. [33] use the matrix organized from the non-local isotropic neighbors. Different from [33], Chen et al. [34] build a height-map patch for each point and pack similar patches into a height-map patch-group matrix, which is further used to denoise the target local structure. Inspired by [33] and [34], Zhou et al. [35] capture non-local similarities by normal height projection. They project the neighboring points of each point onto its normal to build a projective height vector. In general, non-local-based methods usually work better than other competitors, but with two drawbacks. First, the lack of similar patches within the point cloud may degrade their performance. Secondly, these methods usually suffer from high computational complexity.

Graph-based methods first construct a graph for a point cloud and then filter the graph for PCD [37]–[45]. Schoenenberger et al. [37] construct a k-NN graph from the points and denoise the graph by a convex optimization method. Gao et al. [38] treat the distance of each point to the approximated surface as the graph signal and remove noise through convex optimization with a graph-signal smoothness prior. Instead of constructing a graph in Euclidean space, Dinesh et al. [43] present a reweighted graph Laplacian regularizer as the signal prior for point clouds, offering two characteristics: rotation invariance and the enhancement of piecewise smoothness. Zeng et al. [42] and represent point clouds by a low-dimensional manifold model and denoise the patches by minimizing the manifold dimension. Hu et al. [45] represent dynamic point clouds on spatial-temporal graphs and exploit the temporal consistency via a manifold-to-manifold distance. Graph-based methods could achieve satisfactory performance on the point clouds with low-intensity noise. However, heavy noise may affect the stability of graph construction and denoising performance [46].

2.2 Learning-based PCD Methods

Learning-based PCD methods can be categorized into four categories including point-based, normal-based, multi-task, and unsupervised methods.

Point-based methods. Most PCD methods use neural networks to predict the displacement (or additive noise) of each noisy point and restore each point by adding the displacement (or subtracting the additive noise) [3], [7], [47]–[53]. Huang et al. [47] present a non-local part-aware PCD method to explore semantically-relevant features in point clouds, which considers the inherent non-local self-similarity in 3D objects and scenes. Pistilli et al. [48] present GPD based on graph convolutional networks that enhance the robustness of the neural denoiser. Besides, Luo et al. [49] note that displacement prediction methods typically suffer from two types of artifacts: shrinkage and outliers, which are due to inaccurate estimation of noisy displacements. As a result, they propose to learn the underlying manifold (surface) of a noisy point cloud for reconstruction in a downsample-upsample architecture. Later, they further present a score-based method that models a noisy point cloud as samples from a noise-convolved distribution [7]. This method estimates the score of the distribution and leverages the score to denoise point clouds via gradient ascent. The extended work Point Set Resampling (PSR) [8] presents a continuous and global gradient field model and reconstructs degraded point clouds using gradient ascent with Graph Laplacian Regularizer. Mao et al. [52] propose PD-Flow to uncover latent representations of noise-free data at higher dimensions by utilizing normalizing flows, and then obtain the filtered displacements. Chen et al. [3] present a feature-aware recurrent network (RePCD), which uses multi-scale features and trains feature-aware loss iteratively to predict multi-scale geometric details. This method iterates to recover better geometric features at the expense of efficiency. Considering that RePCD ignores the iterative reduction of noise and still requires iteration during testing, Edirimuni et al. [53] propose an iterative point cloud filtering network (IterativePFN) to explicitly model the iterative filtering process internally, which corresponds to a testing iteration of other methods. Nevertheless, because 3D coordinates typically provide limited information, point-based denoising methods often struggle to preserve sharp features effectively. To enhance denoising results, it can be advantageous to incorporate joint learning with other closely related tasks, such as normal estimation.

Normal-based methods. The first-order normal variations can better describe surface variations than point position variations. Therefore, normal-based point cloud denoising approaches also attacked many researchers' attention [2], [6], [54]–[58]. This kind of method first filters the point normals, and then the point positions are then adjusted to well match the filtered normals. [56] projects 3D patches to regular 2D height maps, then uses a CNN architecture to iteratively estimate normals and update point positions based on the estimated normals. PF [2] integrates prior normal information into training loss to preserve sharp features. GeoDualCNN [6] leverages a dual convolutional neural network architecture with geometry support to effectively filter point cloud

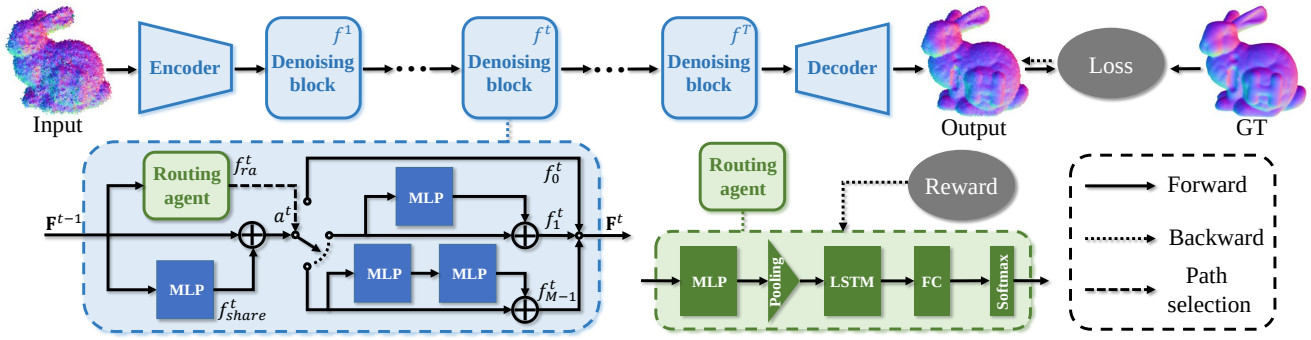


Fig. 3: Overview of PathNet. Following [10] for image restoration, PathNet consists of an encoder and a decoder at the start- and end-point, and T denoising blocks between the encoder and decoder. The encoder extracts patch features as input to the first denoising block, and the decoder predicts a displacement vector as the noise for each point. In each denoising block, there is a routing agent for generating a probabilistic distribution of plausible paths for selection. The routing agent and the denoising network are jointly trained for path-selective denoising based on reinforcement learning. The reward encourages the routing agent to focus more on regions with rich geometric features.

normals and shows the potential performance. However, its end-to-end applicability is limited by the necessity to compute an extra homogeneous neighborhood for each point.

Multi-task methods. Some PCD methods combine denoising with other tasks, such as upsampling, edge extraction, and outlier removing [1], [59]–[62]. Yu et al. [59] introduce an edge-aware point cloud consolidation network (EC-Net) that can generate high-density edge-aware denoising results. Zhou et al. [60] propose DUP-Net that combines denoising and upsampling together, which can simultaneously enhance point resolutions and remove outliers in an adversarial manner. Rakotosaona et al. [1] design a two-stage network architecture, PointCleanNet (PCN), that first removes outliers followed by denoising the central points of local patches. PCDNF [61] jointly learns the tasks of denoising and filtering normals of point clouds. However, it is subject to constraints imposed by the quality of initial normal estimations and the sparsity of the point cloud.

Unsupervised methods. Unsupervised methods strive to denoise point clouds directly without the supervision of clean point clouds since it is difficult to acquire ground-truth data in various contexts. Hermosilla et al. [5] propose the first unsupervised PCD method, called Total Denoising (TD), based on the assumption that points with denser surroundings are closer to the underlying surface. However, due to the lack of clear feature information during the training phase, this method cannot preserve geometric features. To explore geometric structures of point clouds, Chen et al. [63] propose a graph-based auto-encoder with folding, graph-topology inference, and graph filtering to achieve compact representations of unorganized 3D point clouds in an unsupervised manner. The decoder leverages a learnable graph topology to push the codeword to preserve representative features.

In this work, we propose a new framework that distinguishes from the aforementioned methods. We formulate PCD as a decision-making process that dynamically selects a denoising path for each point. Our method is shown to preserve features while removing noise and achieves better denoising performance.

2.3 Deep Reinforcement Learning

Reinforcement learning is a powerful tool for training an agent to make decisions and maximize accumulative rewards. Following Mnih et al. [64], several deep reinforcement learning networks have been successfully applied to 3D point cloud processing tasks such as semantic parsing [65], completion [66], registration [67], classification [68], and segmentation [69], [70]. In this work, we investigate path selection for point cloud denoising by proposing a reinforcement learning framework. This is the first time that deep reinforcement learning is applied to PCD.

3 Method

3.1 Overview

Given a noisy point cloud $\hat{\mathbf{P}}$, PCD is to recover a clean point cloud $\mathbf{P}$ from $\hat{\mathbf{P}}$ to truly convey the geometry of a 3D object. We formulate a noisy point cloud as

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

where ε is the additive noise between $\hat{\mathbf{P}}$ and $\mathbf{P}$. Existing deep learning-based methods use the same model to remove ε for all points indiscriminately. Since ε may vary largely in a point cloud and the point cloud possesses multi-scale geometric features, the same model poorly gives attention to different scales of noise and geometric features. Thus, the side-effects are inevitably introduced including remnant noise, wrongly-smoothed edges, and distorted shape after denoising. Differently, we treat PCD as a decision-making process and propose PathNet to adaptively select the most suitable denoising path for each point based on the extracted noise features and geometric features. Fig. 3 demonstrates the network architecture of PathNet.

3.2 Network Architecture

Based on the network architecture in Fig. 3, we propose PathNet that offers a dynamic network for the denoising task without knowing the noise prior. PathNet consists of three parts: an encoder, T denoising blocks, and a decoder. The encoder extracts patch features as input to the first denoising block, and the decoder predicts a displacement vector as the noise for each point. In the t-th denoising block, there is a shared sub-network f^t_{share}, a denoising path sub-network, and a routing agent f^t_{ra}. The denoising path sub-network contains M selectable denoising paths $f^t_0, f^t_1, ..., f^t_{M-1}$, and the routing agent provides a probabilistic distribution of plausible paths for selection.

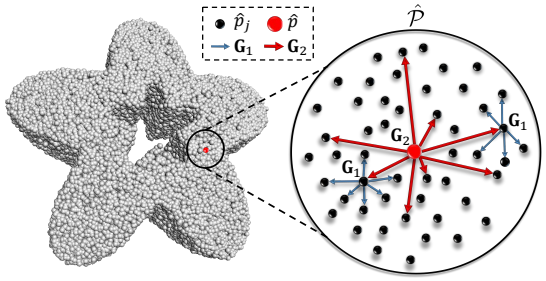


Fig. 4: The patch-graph module consists of the first-layer graphs $\mathbf{G}_1$ built at all the neighbors and the second-layer graph $\mathbf{G}_2$. In the patch $\hat{\mathcal{P}}$, the black dots denote the neighbors $\hat{p}_j$ of the red point $\hat{p}$. $\hat{p}$ is the point to be denoised and all the points in the patch serve to denoise $\hat{p}$. The first-layer graphs connect each neighbor point $\hat{p}_j$ with its local K-Nearest neighbors (in blue). The second-layer graph is a unique graph with its edges (in red) connecting $\hat{p}$ with all its neighbors $\hat{p}_j \in \hat{\mathcal{P}}$.

Encoder. We first normalize $\hat{\mathbf{P}}$ into a unit sphere centered at the origin and then build a local patch $\hat{\mathcal{P}}$ for each input point $\hat{p} \in \hat{\mathbf{P}}$ to extract features. Specifically, we employ a range query to generate patches with a specified radius r. To keep all patches with the same number of points, we either pad a patch (if it has too few points) with the queried point $\hat{p}$ or randomly downsample the patch (if it has too many points) to the desired number of points. For rotation invariance, we follow PF [2] to align these patches via the PCA technique. To enhance the feature representation, we employ two-layer graph convolutions [71], named the patch-graph module (see Fig. 4), to learn more local contextual information for robust noise residual estimation. Different from the general graph convolution, in the second layer, only the global graph for the patch center point (red point in Fig. 4) is constructed. The patch-graph module is formulated as

$$\mathbf{F}^0 = \mathrm{MLP}\left(\mathrm{cat}\left(\max\left(\mathrm{MLP}(\mathbf{G}_1)\right), \mathrm{MLP}(\mathbf{G}_2)\right)\right), \quad (2)$$

where $\mathbf{G}_1$ and $\mathbf{G}_2$ are the first-layer graphs and second-layer graph, respectively; $\mathrm{MLP}(\cdot)$ denotes a fully connected network; $\mathrm{cat}(\cdot,\cdot)$ is the concatenation operation; $\max(\cdot)$ means the max pooling operation. Empirically, the patch point number is set to 128, and the patch radius r defaults to 0.05 times the diagonal length of the bounding box of the point cloud. The number of points in $\mathbf{G}_1$ is set to 10.

Denoising block. The denoising block aims to offer different denoising paths of complexity for each point. We design the denoising block as a structure containing parallel paths, as shown in Fig. 3. Specifically, the t-th denoising block consists of a shared sub-network f^t_{share}, a denoising path sub-network containing M selectable denoising paths, and a routing agent f^t_{ra}. We formulate the t-th denoising block f^t as

$$\mathbf{F}^t = f^t_{a^t}\left(f^t_{share}(\mathbf{F}^{t-1})\right), \quad 1 \le t \le T, \quad (3)$$

where $\mathbf{F}^{t-1}$ and $\mathbf{F}^t$ are the input features and output features of the t-th denoising block f^t. The sub-network f^t_{share} and f_{a^t} are both the residual MLP, and $f^t_{a^t}$ denotes the selected path from the denoising path sub-network. a^t is the path selection action provided by the routing agent f^t_{ra}. The denoising path sub-network consists of two or more different paths: a bypass path for the simple tasks (a^t is 0) and the others for solving the complex tasks (a^t is $1, 2, ..., M-1$). We empirically set M to 2.

Decoder. Once we obtain the output features F^t of the t-th denoising block ($1 \le t \le T$), the decoder regresses the displacement vectors ε^t by applying max pooling and two fully-connected layers. The network then outputs the denoised point cloud P^t by adding the regressed displacement vectors ε^t to the original point cloud $\hat{P}$.

Routing agent. To achieve an adaptive dynamic network, we attempt to employ a routing network. It learns to select the denoising paths. Although the routing network can be trained by the denoising loss, it does not achieve satisfactory denoising results (see Sec. 4.4 and Sec. 4.5). Thus, we adopt a routing agent trained by reinforcement learning strategy to select suitable denoising paths for each point.

We first clarify some terminologies like state, action, and reward, followed by the details of the network. Given the input state s^t, the routing agent $f^t_{ra}(\cdot)$ outputs the policy $\pi^t(a|s^t)$ in t-th denoising block. The procedure is expressed as

$$\pi^t(a|s^t) = f^t_{ra}(s^t), \quad (4)$$

where s^t consists of the patch features and the hidden state of Long Short-Term Memory (LSTM), and it contains information that could be observed by the agent. $\pi^t(a|s^t)$ is the probabilistic distribution of the action space (i.e., all possible actions that could be selected by the routing agent). During training, the routing agent samples a^t from $\pi^t(a|s^t)$ for more exploration. In the test, the routing agent selects an action corresponding to the highest probability $a^t = \mathrm{argmax}_a\, \pi^t(a|s^t)$ for a higher reward.

The agent obtains the reward r^t after selecting the action a^t. The reward is an evaluation of the selected action and drives the training of the agent. The agent aims to maximize the cumulative reward $R = \sum_{i=1}^{T} r^t$ for all selected actions. Thus, the reward is critical for the agent. To select proper denoising paths for points in different regions, we employ a noise- and geometry-aware reward to guide the agent to pay more attention to noisy and feature-rich regions. The detailed reward is explained in Sec. 3.3.

Agent structure. The routing agent is a connection of the following modules (see Fig. 3): an MLP followed by a max pooling layer, an LSTM module, a fully-connected (FC) layer, and an activation function (Softmax). LSTM is used to capture the correlations of path selection in different dynamic denoising blocks from the contextual information of historical features and actions. Since the routing agent is non-differentiable, we cast the sequential path decision as a Markov Decision Process (MDP) and employ reinforcement learning to train the agent.

Finally, our denoising model is formulated as

$$\begin{cases} \mathbf{F}^0 = E(\hat{\mathbf{P}}) \\ \mathbf{F}^t = f^t(\mathbf{F}^{t-1}), \ \ 1 \le t \le T \\ \varepsilon^t = D(\mathbf{F}^t) \\ \mathbf{P}^t = \hat{\mathbf{P}} + \varepsilon^t \end{cases}, \quad (5)$$

where $E(\cdot)$ and $D(\cdot)$ are the encoder and decoder, respectively. $\mathbf{F}^t$ indicates the output features from the t-th denoising block f^t. ε^t and $\mathbf{P}^t$ are the displacement vectors regressed from $\mathbf{F}^t$ and denoising results, respectively. T is the number of denoising blocks, i.e., the maximum length of the denoising path.

3.3 Loss and Reward

We propose a joint loss function to train PathNet. The joint loss consists of two components, i.e., a denoising loss and a repulsion

loss. Moreover, a noise- and geometry-aware reward is designed to train the routing agent using the REINFORCE algorithm [72].

Denoising loss. To encourage the denoised points to be as close as possible to the ground-truth surface, we calculate the denoising loss as the squared distance between the denoised point and its nearest neighbor in the ground-truth [1]. The denoising loss function of p^t is formulated as

$$L_d^t = \|p^t - p_n\|_2^2, \tag{6}$$

where $p^t \in \mathbf{P}^t$ is the denoising result of $\hat{p}$ in the t-th denoising block. $p_n = NN(p^t, \mathcal{P})$, with $NN(p^t, \mathcal{P})$ denoting the nearest point to p^t in $\mathcal{P}$. $\mathcal{P}$ refers to the local ground-truth patch centered at the noisy point $\hat{p}$ in the ground-truth point cloud. The radius of $\mathcal{P}$ is the same as that of the noisy input patch $\hat{\mathcal{P}}$.

Repulsion loss. Minimizing the loss defined in Eq. 6 helps regress noisy points toward the underlying surface. However, it often leads to the side effect, i.e., the denoised points are clustered together and distributed non-uniformly, especially when multiple denoised points share the same nearest point in the ground truth. To mitigate the side effect, we introduce a repulsion loss [1]. This repulsion loss measures the squared distance between the denoised point and its farthest point in the local ground-truth patch. The loss function is formulated as

$$L_r^t = \|p^t - p_f\|_2^2, \tag{7}$$

where $p_f = FN(p^t, \mathcal{P})$, and $FN(p^t, \mathcal{P})$ is the farthest point to p^t in the local ground-truth patch $\mathcal{P}$. Note that the farthest point search needs to be updated in every training epoch. By minimizing Eq. 7, we ensure that the denoised point p^t remains centered in the local ground-truth patch, thereby promoting a regular distribution of denoised points and preventing excessive clustering.

Overall, the joint loss function L is formulated as

$$\begin{aligned} L &= \frac{1}{N}\sum_{\hat{p}\in\hat{\mathbf{P}}} L_{\hat{p}}, \\ L_{\hat{p}} &= \frac{1}{T}L_b^T + \gamma\sum_{i=1}^{T-1} L_b^t, \\ L_b^t &= (1-\alpha)L_d^t + \alpha L_r^t, \end{aligned} \tag{8}$$

where $L_{\hat{p}}$ is the final loss of noisy point $\hat{p}$. N is the number of points in noisy point cloud $\hat{\mathbf{P}}$. T is the number of denoising blocks. L_b^t is the joint loss that is the weighted average of L_d^t and L_r^t. α is a trade-off factor to balance the importance of each loss, and we empirically set α to 0.01. γ is set to 0.1 to control the intermediate denoising blocks.

Reward. It is challenging to remove noise for points with complex geometric patterns and/or high levels of noise. The reward function is crafted to steer the routing agent towards choosing deeper networks with a more robust denoising capability for points with more complex geometry and higher levels of noise. When designing the reward function, we devise a penalty component to avoid choosing complex paths for points that are easy to denoise, thereby reducing the waste of computing resources. The reward function counteracts this penalty by obtaining a higher positive reward for noise removal in regions characterized by higher levels of noise and more complex geometric features. It leads to the selection of complex paths for points that are difficult to denoise.

Based on the above design principle, and inspired by the image restoration methods [10], [73], to encourage the routing agent to select a proper denoising path for different points that have varying noise and shape features, we propose the noise- and geometry-aware reward r^t at the t-th denoising block formulated as

$$r^t = \begin{cases} -pa^t, & 1 \le t \le T-1 \\ -pa^t - (v + w_n + \lambda w_g)(L_b^T - L_b^0), & t = T \end{cases}, \tag{9}$$

where a^t is the action decided by the routing agent (0 or 1). p is a constant, which denotes the penalty for choosing a complex path. L_b^0 is the loss between the original noisy point and the ground-truth point. L_b^T is the loss between the final denoising result and the ground-truth point. $-(L_b^T - L_b^0)$ is a general positive reward. v is the fixed base weight. $w_n = L_b^T / \max(L_b^T)$ is the noise-aware weight, in which $\max(L_b^T)$ represents the maximum loss value of all points in the training dataset. $w_g = e^{-\lfloor \frac{l}{\sigma} \rfloor}$ is the geometry-aware weight, in which l is the geometric feature level classified by a feature extraction algorithm (see from Sec. 4.1) and σ is the scaling parameter of the geometric feature level. The lower the level l, the sharper the geometric features. λ is a trade-off factor to balance the noise weight and geometry-aware weight, and we empirically set λ to 0.1. $v + w_n + \lambda w_g$ can increase the positive reward for noise removal in regions characterized by high levels of noise and geometric features.

4 Experiments

4.1 Datasets and Implementation Details

Training dataset. We select 32 mesh models from the training dataset of PU-GAN [74], with varying complexities classified into three categories: *Simple* (10 models), *Medium* (10 models), and *Complex* (12 models). For each object, we uniformly sample 20K points from its surface mesh as the ground-truth point cloud. To create the noisy input, each point cloud is added with Gaussian noise with standard deviations of 0.0%, 0.5%, 1.0%, and 1.5% of the diagonal length of the bounding box of the object. For each point cloud, we generate 500 patches for training. Hence, there are in total $32 \times 4 \times 500 = 64$K training patches.

Testing dataset. We construct four datasets for the experiments, including two synthetic datasets (called **Synth-A** and **Synth-B**) and two real-world datasets (called **Real-A** and **Real-B**). In the following, we provide a detailed description of these datasets.

Synth-A. We employ all mesh models from the test dataset of PU-GAN [74], which contains 27 mesh models. For each testing mesh, we sample 10K, 20K, and 50K points, and each point cloud is added by Gaussian noise with standard deviations of 0.5%, 1.0%, and 1.5% of the diagonal length of the object bounding box. This dataset is intended to evaluate the robustness of the PCD methods against varying point densities, noise levels, and geometric complexity of the models.

Synth-B. Through this dataset, we aim to investigate the effectiveness of our methodology under various scenarios, such as different noise intensities, fluctuating densities, and distinct noise types. The models within this dataset are the same as those in **Synth-A**, but they are further enhanced and categorized into four sub-datasets. In the first sub-dataset, each point cloud is divided into left and right parts, containing 5K and 25K points, respectively. In the second sub-dataset, the left and right parts of each point cloud have noise levels of 0.5% and 1%, respectively. In the third sub-dataset, the noise level in each point cloud increases from 0.5% to 2% from left to right. In the fourth sub-dataset, the left and right parts of each point cloud contain 1% Gaussian

noise and 1% impulse noise, respectively. Analyzing these sub-datasets allows us to assess the effectiveness of our proposed model under a variety of conditions, which will help validate the model's robustness and flexibility.

Real-A. It comprises all real-scanned data from [75] to validate the generalization capability of denoising methods. It is divided into three sub-datasets, containing point clouds acquired via different scanning methods[1]. Specifically, the first two sub-datasets are captured from single views by Kinect v1 and Kinect v2, respectively, with each containing 144 point clouds, while the third sub-dataset includes 7 point clouds that are obtained by fusing scans captured by Kinect v1 from multiple views. For all models in this dataset, their ground truths are obtained from high-precision 3D mesh models of the scanned objects.

Real-B. It consists of 5 real-scanned LiDAR point clouds, among which 4 LiDAR point clouds were collected from [76], and an entire airplane point cloud scanned by ourselves. The models in this dataset have no ground truths and are exclusively used for visual evaluation.

Geometric feature labeling. Geometric feature extraction is a well-studied topic in the field of geometric processing. To better restore geometric features, we fuse features and rewards to train the agent. Inspired by the selective geometry texture filtering [9], we calculate the geometric feature level as the label of each ground-truth point cloud and the corresponding noisy point cloud (i.e., 0-5, the values of l in Eq. 9). Fig. 5 demonstrates the visualization of the geometric feature levels in a continuous scalar field l of our training dataset. Lower geometric feature levels represent sharper geometric regions. Notably, in Eq. 9, instead of converting the geometric feature level l directly into weights, we balance the proportion of each level by merging the levels with low proportion but high similarity. Specifically, we merge 6 levels into 3 levels by the scaling parameter σ in Eq. 9. Furthermore, it will ignore geometric features in smooth regions if we directly classify all points into 3 levels as shown in Fig. 6. In Fig. 6, (b) classifies all points into 3 levels directly, while (c) classifies all points into 6 levels. (d) merges 6 levels of (c) into 3 levels to balance the proportion of each level. It is obvious that (d) gives a higher level to the geometric feature area in the smooth region compared with (b), which makes our network preserve geometric features better while denoising.

Comparison with state of the art. To evaluate the denoising and geometric feature preservation capabilities of our method, we compare it with the state-of-the-art point cloud denoising methods, including WLOP [19], PCN [1], GPD [48], TD [5], PF [2], Score [7], RePCD [3], AD [4], GeoDualCNN [6], PCDNF [61], and PSR [8]. Among these methods, WLOP and AD are traditional denoising methods, TD is an unsupervised method, and the other methods are learning-based denoising methods. To ensure a fair comparison, we re-train the competing models using their publicly released source code on our dataset. Particularly, for PSR [8], we strictly adhere to its training strategy, supplying 10K, 20K, and 50K noise-free Poisson-sampled data points and perturbing points with Gaussian noise with the standard deviation from 0.5% to 3%. Also, following the original work and its default implementation, we train PSR on patches and validate it on complete point clouds to select the optimal model.

1. The noise characteristics of Kinect v1 and v2 are different because of their different scanning principles: Kinect v1 is based on structured light, and Kinect v2 is based on time-of-flight.

Fig. 5: The geometric feature levels (0-5) of each point in the training dataset. The lower geometric feature levels (in blue) represent sharper regions.

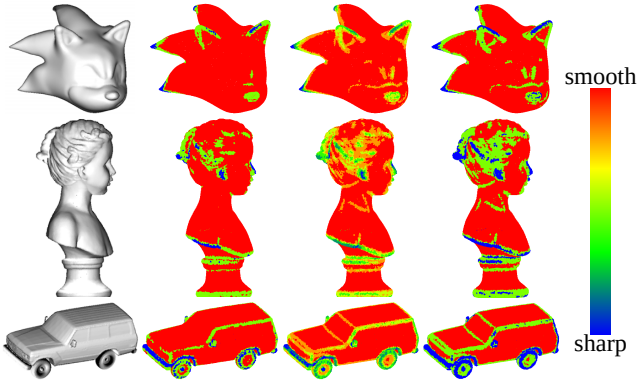


Fig. 6: The geometric feature levels (3 levels and 6 levels) of three 3 point clouds in the training dataset. (b) classifies all points into 3 levels directly, while (c) classifies all points into 6 levels. (d) merges 6 levels into 3 levels to balance the proportion of each level by merging the levels with low proportion but high similarity. The lower geometric feature levels (in blue) represent sharper regions. Compared with (b), (d) gives a higher level to the geometric feature area in the smooth region compared with (b), which avoids geometric smoothing in the smooth region while denoising.

Network training. We divide the whole training process into two stages. In the first stage, we train the denoising block with the joint loss function L (in Eq. 8) to initialize the denoising paths, in which each path is randomly selected by the routing agent. In the second stage, we jointly train the denoising block and routing agent and alternately optimize the parameters of the denoising block and the routing agent. The training process of the denoising block is similar to that in the first stage, with one difference that γ in Eq. 8 is set as 0 to de-control the intermediate denoising blocks and enhance the adaptiveness of the network. The routing agent is trained with the reward r^t in Eq. 9 by the REINFORCE algorithm [72]. Also, we reveal the default parameter settings of our network, i.e., the block number $T = 6$, the scale parameter $\sigma = 2.3$, the penalty value $p = 0.002$, and $\max(L_b^t) = 0.4$.

We implement our PathNet in Pytorch and test it on a PC with an Intel Core I7-8700K CPU (3.70 GHz, 16GB memory) and a GeForce GTX 1080 GPU (8GB memory, CUDA 10.0). PathNet is trained with the Adam optimizer using a learning rate of $1e^{-6}$. We train PathNet with a batch size of 64, and the epoch number of the first and second stages is set to 150 and 100, respectively.

Patch-based and whole point cloud inference schemes. Our method involves the procedure of denoising each point in a point cloud based on its local patch. When handling the point cloud of an entire scene, we search for a local patch around each point

TABLE 1: Quantitative comparison on **Synth-A** and **Real-A**. The best results are **bolded**. The metrics are the Chamfer Distance (CD) and Mean Square Error (MSE). The units of CD and MSE are 10^{-5} and 10^{-3}, respectively.

Metric	Method	Synth-A										Real-A			
		10K			20K			50K			AVE	fusion	v1	v2	AVE
		0.5%	1%	1.5%	0.5%	1%	1.5%	0.5%	1%	1.5%					
CD	WLOP	84.78	121.01	176.30	64.34	73.88	107.52	39.84	67.44	103.24	93.15	15.24	21.67	36.15	24.35
	PCN	29.14	60.13	95.41	19.45	30.76	45.40	10.34	13.35	16.65	35.63	13.25	20.58	35.56	23.13
	GPD	29.93	39.19	51.81	20.89	23.01	29.25	11.93	12.15	20.65	26.53	17.65	23.25	32.61	24.50
	TD	34.86	64.61	82.42	19.55	32.04	43.08	13.64	24.01	38.37	39.17	38.28	50.55	70.20	53.01
	PF	28.88	42.20	50.38	15.27	22.76	31.79	13.20	17.18	22.25	27.10	14.36	**18.07**	32.45	21.63
	Score	24.43	36.73	45.98	**15.11**	21.13	28.59	7.25	10.69	17.28	23.02	14.14	19.71	32.07	21.97
	RePCD	26.87	38.38	47.01	15.79	21.41	26.23	7.34	10.10	**13.75**	22.99	13.50	20.42	**31.28**	21.73
	AD	32.88	46.21	58.77	17.36	26.14	37.78	7.56	12.04	20.73	28.83	13.13	21.45	33.62	22.73
	GeoDualCNN	31.23	43.37	54.92	17.05	24.60	35.33	7.56	13.79	26.09	28.21	12.92	20.61	33.34	22.29
	PCDNF	74.25	85.33	96.19	22.75	31.21	40.75	7.30	10.72	15.88	42.71	13.21	18.94	38.76	23.64
	PSR	24.13	**35.74**	45.68	15.78	21.03	26.25	8.66	11.81	17.83	22.99	17.02	22.13	34.64	24.60
	Ours	**22.72**	36.53	**45.45**	15.24	**20.72**	**25.84**	**6.92**	**9.72**	14.03	**21.91**	**12.84**	18.82	31.59	**21.08**
MSE	WLOP	31.81	33.82	37.02	22.86	23.48	25.91	15.44	15.86	17.12	24.81	15.42	17.28	25.93	19.54
	PCN	30.00	32.26	35.45	21.52	22.62	24.14	13.97	14.45	15.00	23.27	14.89	16.18	25.56	18.88
	GPD	29.50	29.76	31.03	21.43	21.56	22.25	14.20	14.30	16.11	22.24	15.10	15.47	**24.17**	18.25
	TD	30.21	32.67	34.09	21.61	22.92	24.25	14.76	16.70	18.50	23.97	17.52	19.70	28.19	21.80
	PF	29.47	29.97	30.51	**20.81**	21.29	22.06	14.39	15.11	15.94	22.17	14.94	**15.44**	24.81	18.40
	Score	29.18	29.75	30.61	20.90	21.51	22.55	13.46	14.14	15.57	21.96	14.91	15.75	24.79	18.48
	RePCD	29.12	29.60	30.06	20.85	**21.20**	21.77	13.36	13.78	**14.39**	21.57	14.74	15.72	24.46	18.31
	AD	29.23	29.53	30.13	**20.81**	21.32	22.16	**13.27**	13.81	14.84	21.68	14.87	15.95	25.00	18.60
	GeoDualCNN	29.72	30.17	31.10	21.09	21.80	23.10	13.47	14.63	16.91	22.44	14.77	15.96	25.08	18.60
	PCDNF	32.62	33.25	34.21	21.50	22.23	23.22	13.37	14.03	15.04	23.28	14.85	15.61	24.91	18.46
	PSR	29.37	**29.41**	30.07	21.08	21.31	22.03	13.87	14.31	15.62	21.90	14.99	15.88	24.78	18.55
	Ours	**29.03**	29.44	**30.03**	20.83	**21.20**	**21.76**	13.29	**13.76**	14.49	**21.54**	**14.74**	15.63	24.32	**18.23**

of interest. Denoising is performed independently for each patch, allowing to efficiently process multiple patches in batches, and the denoising result of the entire scene is naturally obtained after denoising all the patches. Importantly, whether the test data is complete or not, the denoising results remain consistent, since the local context remains unchanged for both an individual patch and the entire model.

4.2 Quantitative Comparisons

To quantitatively compare our PathNet against its competitors, we follow existing works and use the Chamfer Distance (CD) and Mean Square Error (MSE) between the denoised point clouds and their ground-truth counterparts as the evaluation metrics. Lower CD and MSE values indicate better denoising performance. CD and MSE are formulated as

$$\mathrm{CD}(\overline{\mathbf{P}}, \mathbf{P}) = \frac{1}{N}\sum_{\overline{p}_i \in \overline{\mathbf{P}}} \min_{p_j \in \mathbf{P}} ||\overline{p}_i - p_j||_2^2 + \frac{1}{M}\sum_{p_j \in \mathbf{P}} \min_{\overline{p}_i \in \overline{\mathbf{P}}} ||p_j - \overline{p}_i||_2^2, \tag{10}$$

$$\mathrm{MSE}(\overline{\mathbf{P}}, \mathbf{P}) = \frac{1}{kN}\sum_{\overline{p}_i \in \overline{\mathbf{P}}} \sum_{p_j \in K(\overline{p}_i)} ||\overline{p}_i - p_j||_2, \tag{11}$$

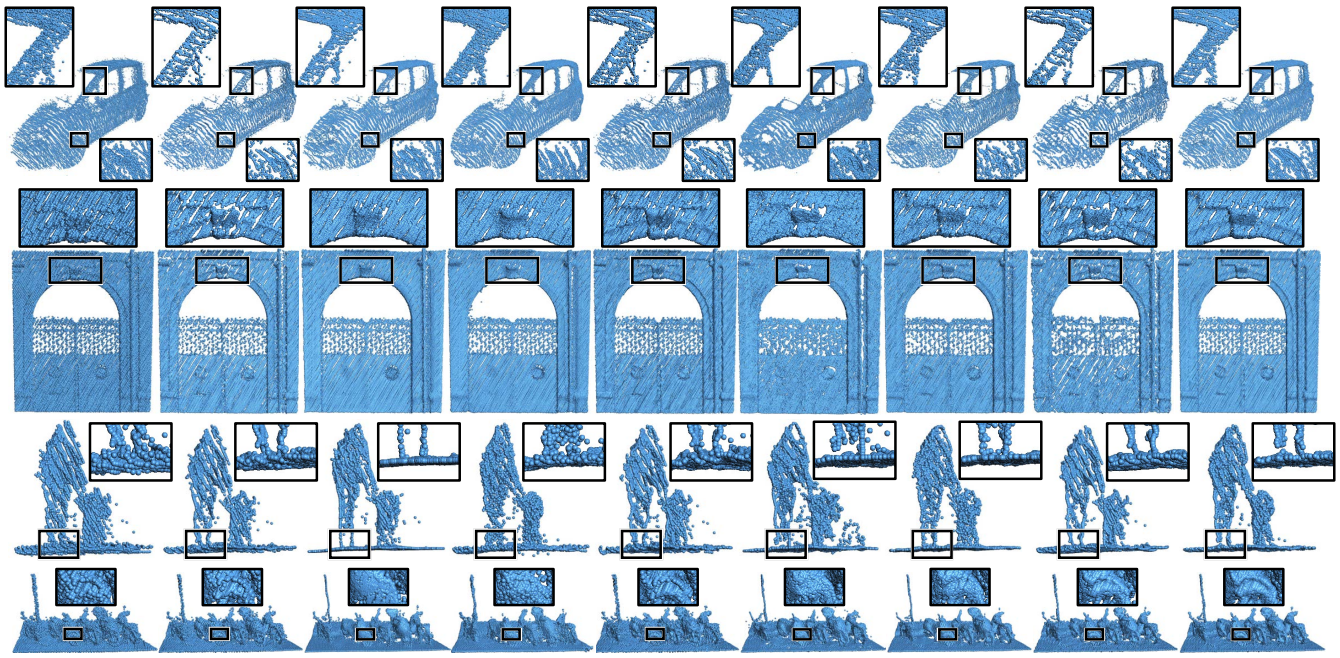
where $\overline{\mathbf{P}}$ is the denoised result of the noisy input $\hat{\mathbf{P}}$. $\mathbf{P}$ is the ground truth. N and M are the point numbers of $\overline{\mathbf{P}}$ and $\mathbf{P}$, respectively. $K(\overline{p}_i)$ is the k-nearest neighbors of $\overline{p}_i$ in $\mathbf{P}$. k is set to 10. The units of CD and MSE are 10^{-5} and 10^{-3}, respectively.

Tab. 1 presents the comparison in terms of CD and MSE, evaluated on the synthetic point clouds with varying noise levels in **Synth-A** and real-scanned Kinect data in **Real-A**. The comparison demonstrates that our method achieves state-of-the-art results on both the synthetic data and the Kinect data. This provides evidence that our methodology exhibits powerful denoising and generalization performance when applied to real-scanned data.

Our method may exhibit slightly inferior performance compared to several methods in the case of 0.5% noise. There are two main reasons: i) The path selection mechanism guides points with varying levels of noise to choose different paths, with the aim of achieving higher overall accuracy. PathNet is based on the idea that selecting longer paths for points with higher noise levels can yield greater rewards. Consequently, points with lower noise levels are compelled to opt for shorter paths, which may, unfortunately, reduce their denoising accuracy. This hypothesis can be validated by comparing the results of variants $V3$ and $V6$ in Sec. 4.4. ii) The CD metric computes bidirectional shape similarity, while MSE is unidirectional. The method, Score [7], computes CD values based on uniformly sampled denoising results, which can result in higher CD values compared to our approach. The method, AD [4], outperforms our approach in terms of MSE but lags behind in the CD metric. This discrepancy can be attributed to AD producing clustered points, resulting in a lower MSE value but a higher CD value. To further promote performance on 0.5% noise, it could be advantageous to explore increasing the complexity of the chosen paths and incorporating uniform resampling techniques for the final results.

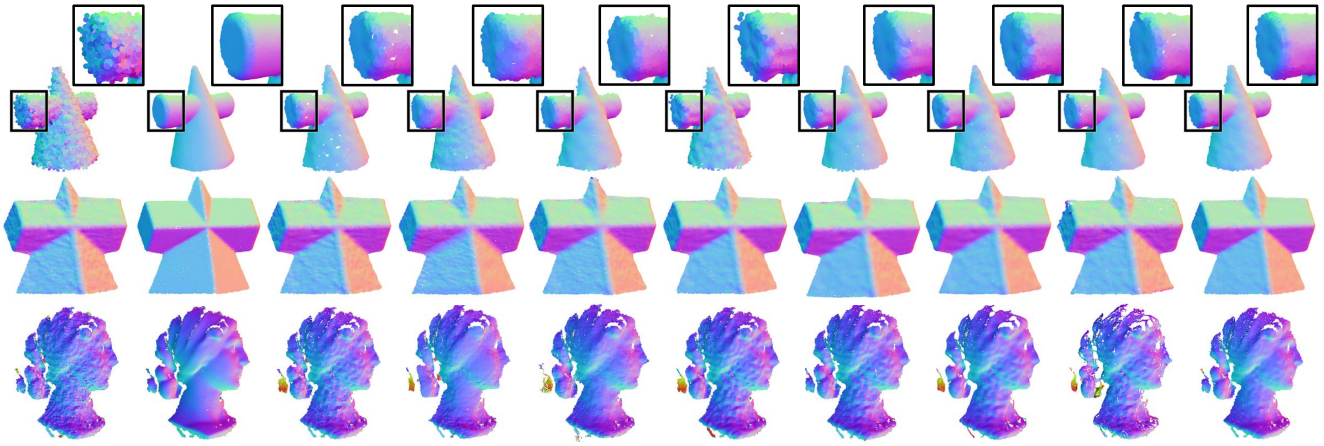
4.3 Visual Comparisons

Apart from the quantitative comparisons, we conduct visual comparisons on both the synthetic and real-scanned point clouds. Fig. 7 shows the results of real-scanned LiDAR data in **Real-B** from [76]. Fig. 8 presents the denoised results of three point clouds from the fusion sub-dataset and Kinect v1 sub-dataset in **Real-A**, respectively. Fig. 9 shows the denoising results of a real-scanned airplane with 246K points in **Real-B**, which is scanned by a Leica RTC360. Fig. 10 demonstrates the results of an animal



(a) Input (b) AD (c) TD (d) PF (e) GeoDualCNN (f) Score (g) RePCD (h) PSR (i) Ours

Fig. 7: Denoising real-scanned LiDAR data in **Real-B**. PathNet removes heavy noise and preserves geometry better (see the magnified parts) than its competitors.



(a) Input (b) GT (c) AD (d) TD (e) PF (f) GeoDualCNN (g) Score (h) RePCD (i) PSR (g) Ours

Fig. 8: Denoising real-scanned Kinect data in **Real-A**. The models in the first two rows are obtained by fusing scans from multiple views, and the one in the bottom row is a single-view point cloud. PathNet outperforms the others in terms of both noise removal and feature preservation. The points are colored by their normals calculated using the same algorithm.

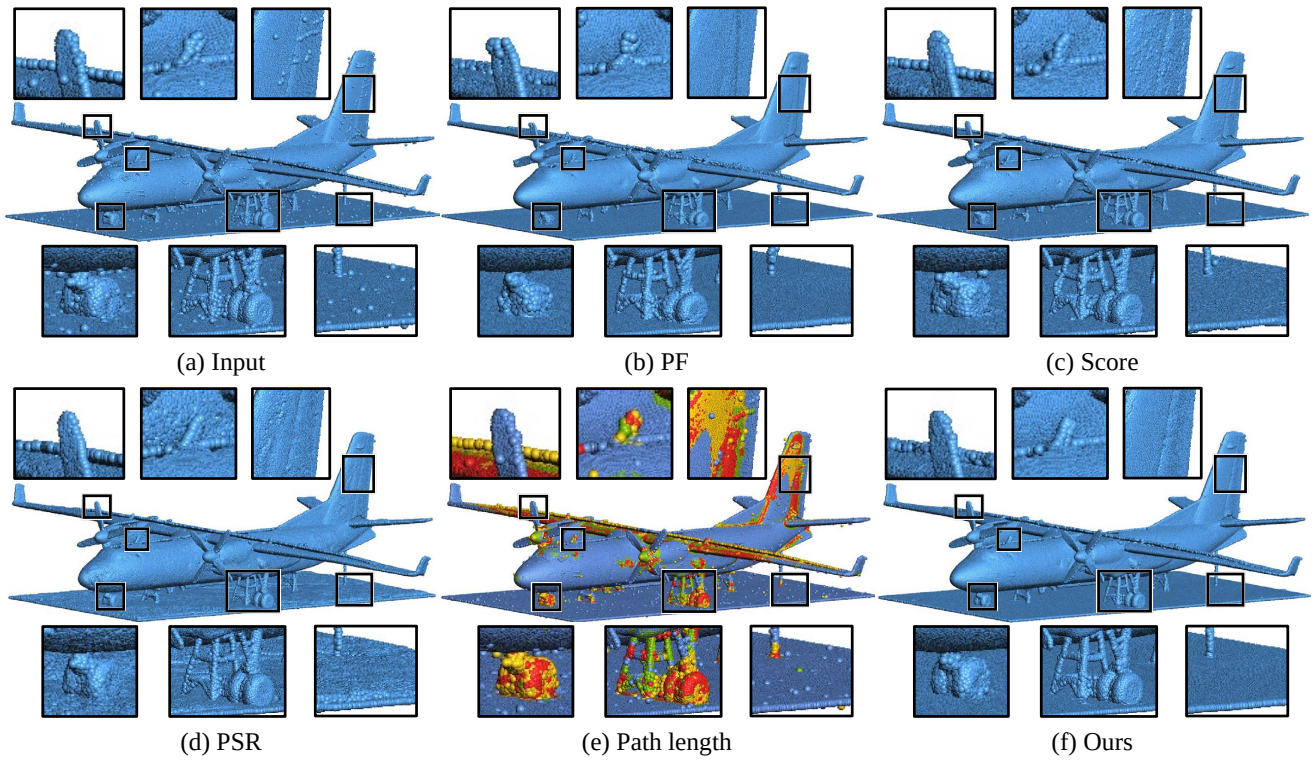
model and a CAD model with rich or sharp features in **Synth-A**, respectively. The gray models are the meshes reconstructed by the Poisson surface reconstruction algorithm.

In general, GeoDualCNN [6] is prone to retain excessive noise. AD [4], TD [5], PF [2], Score [7], and PSR [8] tend to over-smooth the sharp geometric features of the objects. In contrast, with the path routing agent and the noise- and geometry-aware reward, our method offers adaptive denoising paths for the points in varying noisy and geometric regions, thereby robustly removing heavy noise and restoring finer geometric details.

4.4 Ablation Study

We analyze the contribution of the major modules of our method by comparing its several variants.

- $V1$ "no patch-graph": the patch-graph module is replaced by MLPs to extract features. All denoising blocks choose the bypass path without training the routing agent.
- $V2$ "all path 0": all denoising blocks choose the bypass path without training the routing agent.
- $V3$ "all path 1": all denoising blocks choose the complex path without training the routing agent.
- $V4$ "one stage without RL": all denoising blocks select paths by a routing network. The routing network is trained with the denoising blocks during the first training stage simultaneously. No second stage and reinforcement learning are available.
- $V5$ "no RL": all denoising blocks select paths by a routing



(a) Input (b) PF (c) Score

(d) PSR (e) Path length (f) Ours

Fig. 9: Visualization of the real-scanned airplane data (246K points) in **Real-B** and its denoising results. The airplane is scanned from multiple stations by a Leica RTC360. PathNet removes noise and preserves geometry better (see the magnified parts) than three state-of-the-art methods, namely PF [2], Score [7], and PSR [8].

network. The routing network is trained with the denoising blocks during the second training stage simultaneously. No reinforcement learning is available.

- $V7$ "no aware": in the second training stage, we train the routing agent without the noise- and geometry-aware reward by setting the two weights to 0.
- $V8$ "no geometry-aware": in the second training stage, we train the path routing agent without the geometry-aware reward by setting the geometry-aware weight to 0.
- "Full": our complete scheme.
- "ES-1": following [77], we employ ensemble learning to train a variant based on three frozen pre-trained denoising sub-networks. These sub-networks are trained based on different loss functions, including the L_2 losses (Eqs. 6 and 7), noise-aware L_2 loss, and geometry-aware L_2 loss.
- "ES-2": an ensemble variant is trained concurrently with three denoising sub-networks.

We perform quantitative comparisons between these variants on **Synth-A**, and the denoising results are summarized in Tab. 2. By comparing the performance of these variants, it is obvious that each module plays a positive role in denoising.

Particularly, comparing the variant $V1$ "no patch-graph" and $V2$ "all path 0", the patch-graph module improves the denoising results. The reason is that the patch-graph module extracts the local geometric features of each point. The module enables learning the correlation between the point $\hat{p}$ and its neighbor points in the patch $\hat{\mathcal{P}}$, and more consistent local structure information for robust residual estimation. Besides, compared to the variant $V2$ "all path 0", the variant $V3$ "all path 1" improves denoising accuracy using a deeper network by selecting all complex paths.

TABLE 2: Ablative results of the variants on **Synth-A**. The shown data are delta values relative to $V3$. The metrics are Chamfer Distance (CD) and Mean Square Error (MSE). The units of CD and MSE are 10^{-5} and 10^{-3}, respectively. "↓" indicates improved performance.

Model	CD				MSE			
	0.5%	1%	1.5%	AVE	0.5%	1%	1.5%	AVE
$V1$	1.50↑	1.49↑	4.15↑	2.38↑	0.13↑	0.17↑	0.59↑	0.30↑
$V2$	0.77↑	1.05↑	3.38↑	1.74↑	0.04↑	0.12↑	0.45↑	0.20↑
$V3$	-	-	-	-	-	-	-	-
$V4$	0.07↑	0.61↓	1.12↓	0.55↓	0.01↓	0.10↓	0.18↓	0.18↓
$V5$	0.16↑	0.55↓	1.40↓	0.60↓	0.01↓	0.12↓	0.26↓	0.13↓
$V6$	0.03↑	0.45↓	2.30↓	0.91↓	-	0.20↓	0.72↓	0.31↓
$V7$	0.04↑	0.83↓	3.44↓	1.41↓	0.05↓	0.23↓	0.76↓	0.35↓
"Full"	**0.17**↓	**1.01**↓	**3.65**↓	**1.61**↓	**0.06**↓	**0.27**↓	**0.82**↓	**0.38**↓
"ES-1"	0.80↑	0.13↓	1.58↓	0.31↓	-	0.12↓	0.61↓	0.25↓
"ES-2"	1.37↑	0.05↓	2.89↓	0.52↓	0.12↑	0.13↓	0.72↓	0.25↓

In the variants $V4$ "one stage without RL", $V5$ "no RL", and $V6$ "no aware", the dynamic denoising path is more suitable for the denoising tasks than a fixed network. Note that, the CD accuracy of 0.5% noise is not effectively improved, which means that the deeper network of $V3$ "all path 1" has exceeded the requirement of low noise and increased unnecessary computation. By comparing the variants $V5$ "no RL" and $V6$ "no aware", the routing agent trained by reinforcement learning selects more suitable paths for better results. In addition, the noise-aware reward of $V7$ "no geometry-aware" encourages the network to pay more attention to the heavily noisy points that are difficult to denoise, which effectively decreases the metric values. Finally, our complete pipeline "Full" proves that the geometry-aware reward

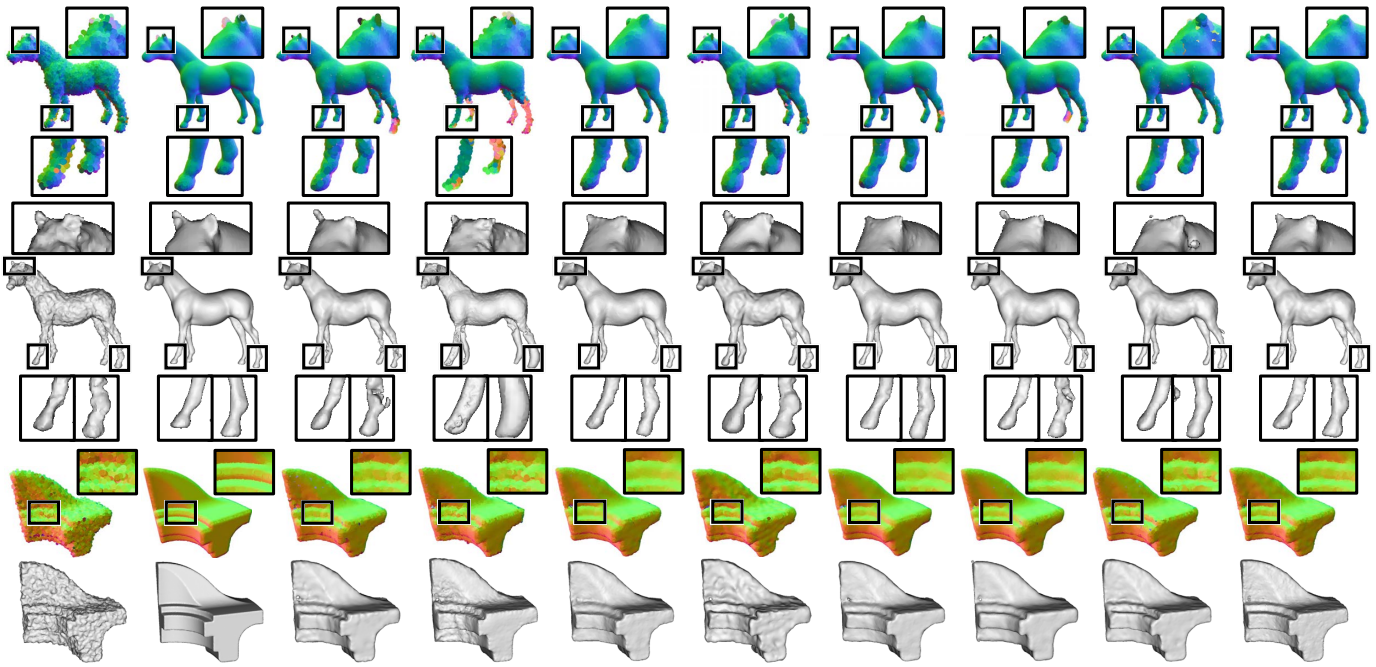


Fig. 10: Denoising synthetic point clouds in **Synth-A**. The noisy models are an animal and a CAD model corrupted by 1.0% and 1.5% Gaussian noise, respectively. PathNet outperforms the others in terms of both noise removal and feature preservation (see the close-up views). The colorful points are colored by their normals. The discontinuity in color indicates noise residuals (horse legs) that cause errors in the normals. The gray models are the meshes reconstructed by Poisson surface reconstruction.

is effective for point clouds with levels of noise.

More importantly, by comparing $V3$ "all path 1" and "Full", our full method "Full" achieves better results on all metrics. However, the network of $V3$ is more powerful since it chooses all complex paths for each point and uses more network parameters than "Full". This strongly proves the effectiveness of our strategy.

Besides the above RL-based path selection, we also design two ensemble learning-based variants, "ES-1" and "ES-2", inspired by the image denoising wisdom [77]. The ensemble learning network consists of two main components: denoising sub-networks and an ensemble module. The ensemble module adaptively assigns a weight to each denoising sub-network, functioning as an alternative form of path selection. From the comparison results of the variants "ES-1", "ES-2", and "Full" in Tab. 2, we can see that our method achieves superior results in terms of both metrics. It shows the advantages of RL's reward function for dynamic denoising tasks involving path selection.

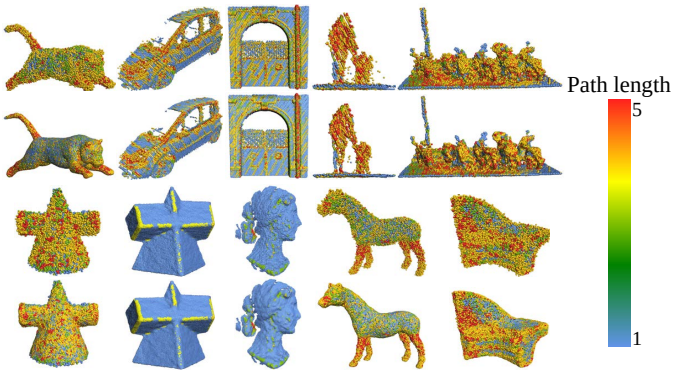


Fig. 11: Visualization of path lengths of ten models from **Synth-A**, **Real-A**, and **Real-B** datasets. The 1st and 3rd rows demonstrate the path lengths on the inputs. The 2nd and 4th rows are on the denoised outputs.

4.5 Path Selection

In Fig. 11, we visualize the path lengths of ten models, where the path length represents the number of times that a complex path is selected. These models are drawn from **Synth-A**, **Real-A**, and **Real-B**, and they include synthetic data, real-scanned LiDAR data, and real-scanned Kinect data. It is noteworthy that complex paths are frequently chosen by points located within regions characterized by high-intensity noise or sharp features. Building on this observation, in this section, we delve deeper into the analysis of path selection under various scenarios, e.g., different noise intensities and types, and varying densities.

Varying point densities, noise intensities, and noise patterns within a single point cloud. Handling varying point densities, noise intensities, and noise patterns within a single point cloud is a challenging task for denoising algorithms. To further investigate the efficiency of our method under various scenarios, we test our approach on **Synth-B**. Table 3 presents the quantitative comparison results. By leveraging the path-routing agent and the noise- and geometry-aware reward, our method flexibly removes noise and restores geometric details, achieving the best results across all scenarios and metrics. Fig. 12 displays a visualization result of our method in each scenario. For sub-figure (a), we use the same reception field for all points. It is obvious that our method is less influenced by the changes in point cloud density. This is because ball query and sampling techniques are employed to obtain neighboring contextual information. For sub-figure (b) and

TABLE 3: Quantitative comparison on **Synth-B**. The metrics are the Chamfer Distance (CD) and Mean Square Error (MSE). The units of CD and MSE are 10^{-5} and 10^{-3}, respectively. The left and right parts of each point cloud in this dataset are differently contaminated by densities, noise, and noise types. Our proposed method demonstrates superior performance in all scenarios and across both metrics.

Method	CD					MSE				
	5K + 25K	0.5% + 1%	0.5% to 2%	Type	AVE	5K + 25K	0.5% + 1%	0.5% to 2%	Type	AVE
WLOP	76.20	53.48	96.88	57.57	71.03	18.85	15.63	25.11	17.19	19.19
PCN	21.28	11.72	39.06	12.68	21.18	17.42	14.19	23.47	15.64	17.68
GPD	16.62	12.09	27.22	13.10	17.26	16.84	14.24	22.05	15.64	17.19
TD	30.58	18.96	37.25	20.08	26.72	19.34	15.78	23.56	17.16	18.96
PF	21.25	15.01	27.79	16.54	20.15	17.63	14.73	21.71	16.17	17.56
Score	14.88	8.97	25.29	9.65	14.70	16.75	13.84	22.11	15.15	16.96
RePCD	14.96	8.60	23.63	9.59	14.19	16.42	13.61	21.59	14.93	16.64
AD	17.92	9.93	32.52	10.45	17.70	16.40	13.58	21.83	14.88	16.67
GeoDualCNN	18.67	10.86	30.62	11.48	17.91	17.24	14.08	22.64	15.40	17.34
PCDNF	23.19	8.91	36.16	9.82	19.52	17.22	13.68	22.79	15.07	17.19
PSR	15.65	10.31	23.68	11.12	15.19	16.87	14.08	21.84	15.46	17.06
Ours	**14.37**	**8.25**	**23.38**	**8.91**	**13.73**	**16.35**	**13.53**	**21.53**	**14.81**	**16.56**

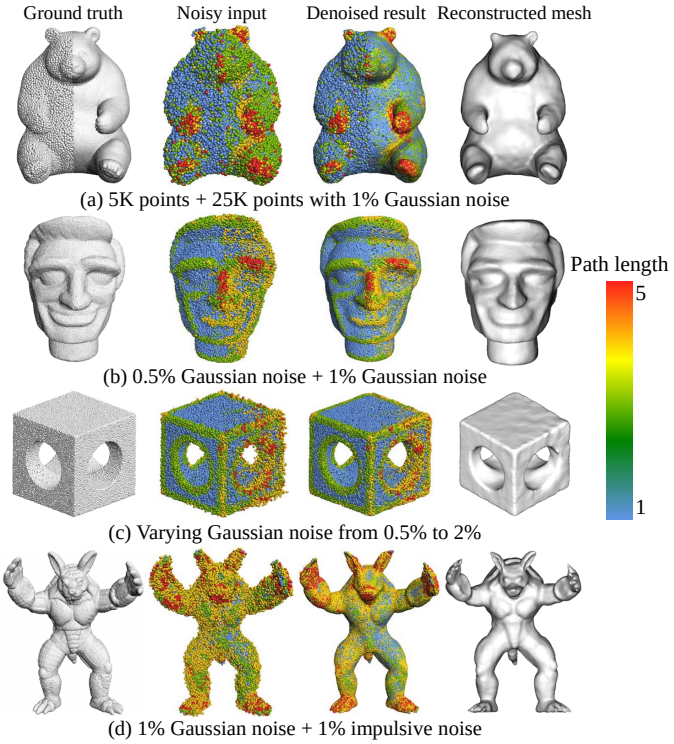


Fig. 12: Path selection visualization of four point clouds with different densities, noise levels, and noise types. Each point cloud is separated into two parts (left and right) in (a), (b), and (d). (c) is a point cloud with increasing noise from 0.5% to 2%. Obviously, our path selection mechanism works in all four cases.

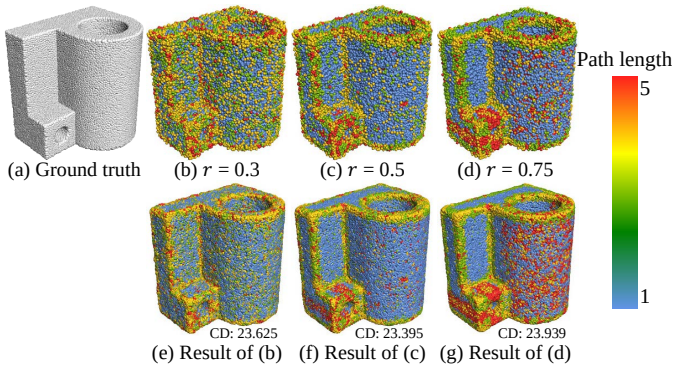


Fig. 13: Path selection visualization for a point cloud (1% noise and 50K points) with different receptive fields. The two rows respectively demonstrate the input and output colored by the path length. The receptive fields from (b) to (d) are 0.3, 0.5, and 0.75, respectively. PathNet operates smoothly with these different receptive fields.

(c), our model adaptively selects more paths for better suppressing the noise. For sub-figure (d), since the Armadillo model is of rich geometry details, 1% noise is still heavy relative to the scale of details, leading to a more complex denoising path. Moreover, impulsive noise is not present in the training dataset. We can observe that our approach can still effectively process it and select the appropriate path.

Different receptive fields. The receptive field (namely neighborhood scale) is an important factor affecting access to neighborhood contextual information [3]. The receptive field is relative to the scale of geometric features and noise intensities. Intuitively, for one noise point, a smaller receptive field will make the noise appear larger, thus choosing a more complex path. As shown in Fig. 13 (b) and (c), it is obvious that more points are denoised by a more complex path. On the contrary, the noise level present within the larger receptive field is proportionally reduced, making the removal of high-level noise more feasible (see Fig. 13 (c)). Besides, we find that more red points are selected in the small cubic region of Fig. 13 (d). This is because a larger receptive field makes the local structure of this region more complex, thus requiring a longer denoising path. Finally, in sub-figures (e-g), our method produces similar CD quantitative results across three different receptive fields. This indicates that our model can flexibly and adaptively select the appropriate path to handle different situations while achieving good denoising performance.

In the above experiment, the routing agent selects a suitable denoising path for each point. Specifically, the routing agent arranges longer denoising paths for points with heavier noise or more complex features. This complies with the expectation that PathNet focuses on processing noisy and feature points to remove noise and restore geometric details. In addition, the routing agent could identify a large number of low-noise points close to the ground-truth surface and heavy-noise points in smooth regions, which do not require a complex denoising network. Our network

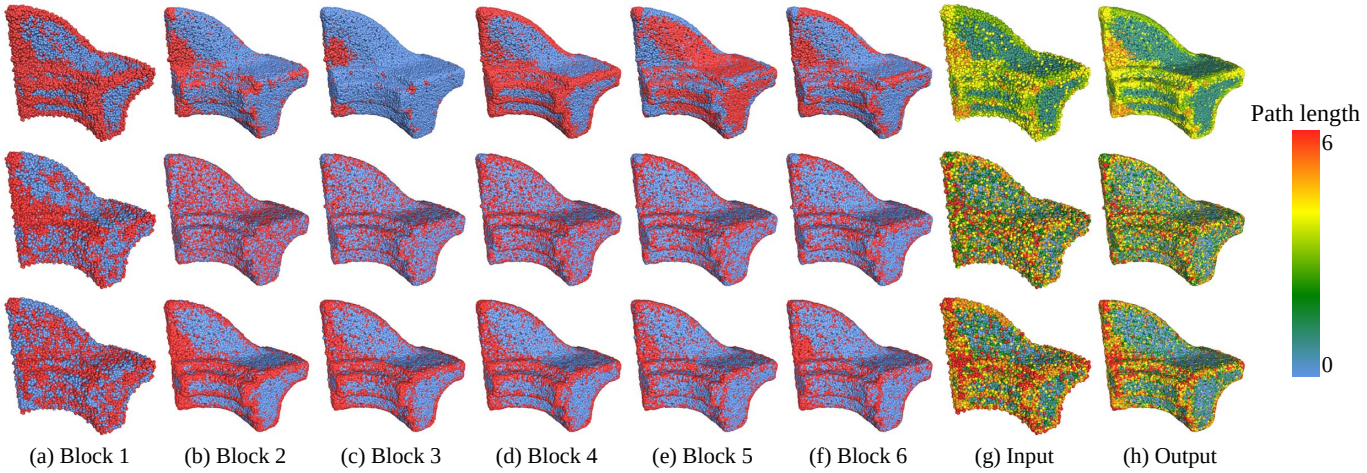


Fig. 14: Path selection visualization in each denoising block. The three rows show the path selection of the variants "Full", $V4$ "one stage w/o RL", and $V5$ "no RL", respectively. In (a-f), the red and blue points respectively represent the complex and bypass paths. (g) and (f) visualize the path lengths on the input and output, respectively. The CAD model is corrupted by 1.0% Gaussian noise.

processes them with suitable paths to maintain the overall structure of the point clouds and prevent over-smoothing. For these reasons, our method achieves satisfactory results with a good balance between denoising and feature preservation.

Path selection in each denoising block. Also, we visualize the path selection results of different learning strategies on a CAD model corrupted by 1.0% Gaussian noise in Fig. 14. In Fig. 14 (a-f), the red and blue points separately represent the complex and bypass paths. (g) and (f) show the path length for each point on the input and output, respectively. From the 1st row in Fig. 14, we can see that the complex paths in Blocks 1 to 4 are mainly used to restore points with heavy noise or in complex geometric regions. The points in the smooth regions select a complex path in Block 5, while the points in the points in geometric regions select a complex path in Block 6. This way, our network can effectively preserve structures while reducing noise.

In addition, we observe that the longest path length in all denoising results shown in Fig. 11 is 5. Since we designed a total of 6 denoising blocks, the longest path length can be up to 6. From the 1st row of Fig. 14, by comparing Block 5 (e) and Block 6 (f), we can find that their path selection results are completely opposite, which indicates that Block 5 (e) and Block 6 (f) are denoising blocks that can handle points with opposing properties. Since each point can only choose one of their complex paths, this explains the observation that the longest path length is 5. The above observation proves that there are low-consistency tasks in the denoising process, which are not suitable for being processed by a single model. The routing agent could arrange different denoising paths for these low-consistency tasks to obtain better denoising results. Moreover, we have tried to omit the step of aligning the patch to the Z-axis with PCA to improve efficiency. It turns out that the routing agent will arrange different paths for points with different orientations, which indicates that points with different orientations have opposite properties. This also proves the existence of low-consistency tasks in the denoising process, and it is necessary to handle them by using different networks or by enhancing their consistency.

Path-selective denoising without reinforcement learning. In the 2nd and 3rd rows of Fig. 14, we also visualize the path selection results of the variants $V4$ "one stage without RL" and $V5$ "no RL". For the variant $V4$ "one stage without RL" in the 2nd row, the path selection results are not clear, and only the first block selects complex paths for some geometric feature regions. The reason is that the denoising block cannot effectively remove noise in the early stage of training, which leads to wrong path selection results. The wrong path selection results further affect the training of the denoising paths, resulting in the network falling into the local optimum. In addition, the variant $V5$ "no RL" in the 3rd row tends to choose complex paths for regions with rich geometric features. Compared with our full scheme, the variant $V5$ "no RL" is less sensitive to noise and geometric features. When we try to improve them by applying the awareness of the reward function to the loss function, the accuracy of denoising is reduced, which means that the awareness weights directly applied to the loss function affect the overall denoising performance. On the contrary, our method applies awareness weights to the reward function of reinforcement learning, which further improves denoising performance. Therefore, it is necessary to use the reinforcement learning strategy.

TABLE 4: Running times (in seconds) of different methods on several point clouds shown in Fig. 11. N is the point number.

Models	Cone N: 10K	David N: 51K	Horse N: 20K	Car N: 43K
WLOP	5.71	49.83	9.22	39.24
PCN	68.36	360.21	151.10	310.53
GPD	28.03	147.82	65.44	116.24
TD	4.86	238.38	63.45	165.77
PF	15.06	80.43	27.23	106.30
Score	7.96	67.08	20.84	49.22
RePCD	18.43	197.69	61.25	148.75
AD	3.21	7.82	4.31	6.44
GeoDualCNN	4.12	24.23	10.24	18.68
PSR	3.24	61.54	4.84	55.43
Ours	17.38	86.01	28.34	69.09

4.6 Running Time

Although our network has multiple denoising blocks, it is still competitive in efficiency. Tab. 4 demonstrates the running time of our network for the models shown in Fig. 11. Besides, we

separately measure the average processing time for synthetic data with 20,000 points at different noise levels in **Synth-A**. It requires 25.78s, 28.33s, and 31.27s for denoising the data with noise levels of 0.5%, 1%, and 1.5%, respectively. Notably, our method achieves a 21.3% faster denoising speed for the 0.5% noise data compared to that of the 1.5% noise data.

5 Limitations and Future Work

Despite the promising performance, our method still has some limitations. First, the results of path selection are sensitive to hyper-parameters of the reward. Thus, the hyper-parameters must be tuned by trial-and-error experiments to obtain the expected results of path selection. Second, PathNet is patch-based, which does not fully exploit the correlation between points. Thus, PathNet can only denoise an entire point cloud after multiple passes, which makes it impossible to rebuild patch features after each denoising block. In the future, we will explore an RL-based neighborhood selection mechanism to further enhance the denoising quality.

6 Conclusion

We introduce a new deep model designed for path-selective point cloud denoising, called PathNet. PathNet is the deep reinforcement learning effort for point cloud denoising. We formulate the noise-aware and geometry-aware rewards to train the routing agent. Our path-selective denoising strategy selects the most suitable paths to adaptively denoise points with different properties. Our method achieves state-of-the-art performance in terms of effectiveness and robustness in noise removal and geometric detail preservation, and it generalizes well to real scans. In the future, we will adjust our network architecture to improve efficiency and expand it to other 3D tasks, such as point cloud completion and segmentation.

References

[1] M. Rakotosaona, V. L. Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov, "Pointcleannet: Learning to denoise and remove outliers from dense point clouds," *CGF*, vol. 39, no. 1, pp. 185–203, 2020.

[2] D. Zhang, X. Lu, H. Qin, and Y. He, "Pointfilter: Point cloud filtering via encoder-decoder modeling," *TVCG*, vol. 27, no. 3, pp. 2015–2027, 2021.

[3] H. Chen, Z. Wei, X. Li, Y. Xu, M. Wei, and J. Wang, "Repcd-net: Feature-aware recurrent point cloud denoising network," *IJCV*, vol. 130, no. 3, pp. 615–629, 2022.

[4] Z. Xu and A. Foi, "Anisotropic denoising of 3d point clouds by aggregation of multiple surface-adaptive estimates," *TVCG*, vol. 27, no. 6, pp. 2851–2868, 2021.

[5] P. Hermosilla, T. Ritschel, and T. Ropinski, "Total denoising: Unsupervised learning of 3d point cloud cleaning," in *ICCV*, 2019, pp. 52–60.

[6] M. Wei, H. Chen, Y. Zhang, H. Xie, Y. Guo, and J. Wang, "Geodualcnn: Geometry-supporting dual convolutional neural network for noisy point clouds," *TVCG*, vol. 29, no. 2, pp. 1357–1370, 2023.

[7] S. Luo and W. Hu, "Score-based point cloud denoising," in *ICCV*, 2021, pp. 4583–4592.

[8] H. Chen, B. Du, S. Luo, and W. Hu, "Deep point set resampling via gradient fields," *TPAMI*, vol. 45, no. 3, pp. 2913–2930, 2023.

[9] M. Wei, Y. Feng, and H. Chen, "Selective guidance normal filter for geometric texture removal," *TVCG*, vol. 27, no. 12, pp. 4469–4482, 2021.

[10] K. Yu, X. Wang, C. Dong, X. Tang, and C. C. Loy, "Path-restore: Learning network path selection for image restoration," *TPAMI*, 2021.

[11] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Point set surfaces," in *Proceedings Visualization, 2001. VIS'01.*, 2001, pp. 21–29.

[12] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva, "Computing and rendering point set surfaces," *TVCG*, vol. 9, no. 1, pp. 3–15, 2003.

[13] G. Guennebaud and M. H. Gross, "Algebraic point set surfaces," *ACM Trans. Graph.*, vol. 26, no. 3, p. 23, 2007.

[14] F. Cazals and M. Pouget, "Estimating differential quantities using polynomial fitting of osculating jets," *CAD*, vol. 22, pp. 121–146, 2005.

[15] S. Fleishman, D. Cohen-Or, and C. T. Silva, "Robust moving least-squares fitting with sharp features," *TOG*, vol. 24, no. 3, pp. 544–552, 2005.

[16] A. C. Öztireli, G. Guennebaud, and M. H. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," *CGF*, vol. 28, no. 2, pp. 493–501, 2009.

[17] D. Levin, "The approximation power of moving least-squares," *Math. Comput.*, vol. 67, no. 224, pp. 1517–1531, 1998.

[18] Y. Lipman, D. Cohen-Or, D. Levin, and H. Tal-Ezer, "Parameterization-free projection for geometry reconstruction," *TOG*, vol. 26, no. 3, pp. 22–es, 2007.

[19] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or, "Consolidation of unorganized point clouds for surface reconstruction," *TOG*, vol. 28, no. 5, pp. 1–7, 2009.

[20] H. Huang, S. Wu, M. Gong, D. Cohen-Or, U. Ascher, and H. Zhang, "Edge-aware point set resampling," *TOG*, vol. 32, no. 1, pp. 1–12, 2013.

[21] B. Liao, C. Xiao, L. Jin, and H. Fu, "Efficient feature-preserving local projection operator for geometry reconstruction," *CAD*, vol. 45, no. 5, pp. 861–874, 2013.

[22] R. Preiner, O. Mattausch, M. Arikan, R. Pajarola, and M. Wimmer, "Continuous projection for fast l1 reconstruction." *TOG*, vol. 33, no. 4, pp. 47–1, 2014.

[23] X. Lu, S. Wu, H. Chen, S.-K. Yeung, W. Chen, and M. Zwicker, "Gpf: Gmm-inspired feature-preserving point set filtering," *TVCG*, vol. 24, no. 8, pp. 2315–2326, 2017.

[24] S. Chen, J. Wang, W. Pan, S. Gao, M. Wang, and X. Lu, "Towards uniform point distribution in feature-preserving point cloud filtering," *CVM*, vol. 9, no. 2, pp. 249–263, 2023.

[25] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A survey of sparse representation: Algorithms and applications," *IEEE Access*, vol. 3, pp. 490–530, 2015.

[26] H. Avron, A. Sharf, C. Greif, and D. Cohen-Or, "l_1-sparse reconstruction of sharp point set surfaces," *ACM Trans. Graph.*, vol. 29, no. 5, pp. 135:1–135:12, 2010.

[27] Y. Sun, S. Schaefer, and W. Wang, "Denoising point sets via $_{10}$ minimization," *CAD*, vol. 35-36, pp. 2–15, 2015.

[28] Z. Liu, X. Xiao, S. Zhong, W. Wang, Y. Li, L. Zhang, and Z. Xie, "A feature-preserving framework for point cloud denoising," *CAD*, vol. 127, p. 102857, 2020.

[29] E. Mattei and A. Castrodad, "Point cloud denoising via moving RPCA," *CGF*, vol. 36, no. 8, pp. 123–137, 2017.

[30] J. Digne, S. Valette, and R. Chaine, "Sparse geometric representation through local shape probing," *TVCG*, vol. 24, no. 7, pp. 2238–2250, 2018.

[31] J. Digne, "Similarity based filtering of point clouds," in *CVPR Workshops*, 2012, pp. 73–79.

[32] G. Rosman, A. Dubrovina, and R. Kimmel, "Patch-collaborative spectral point-cloud denoising," *CGF*, vol. 32, no. 8, pp. 1–12, 2013.

[33] X. Lu, S. Schaefer, J. Luo, L. Ma, and Y. He, "Low rank matrix approximation for 3d geometry filtering," *TVCG*, vol. 28, no. 4, pp. 1835–1847, 2022.

[34] H. Chen, M. Wei, Y. Sun, X. Xie, and J. Wang, "Multi-patch collaborative point cloud denoising via low-rank recovery with graph constraint," *TVCG*, vol. 26, no. 11, pp. 3255–3270, 2020.

[35] Y. Zhou, R. Chen, Y. Zhao, X. Ai, and G. Zhou, "Point cloud denoising using non-local collaborative projections," *Pattern Recognit.*, vol. 120, p. 108128, 2021.

[36] J. Wang, J. Jiang, X. Lu, and M. Wang, "Rethinking point cloud filtering: A non-local position based approach," *CAD*, vol. 144, p. 103162, 2022.

[37] Y. Schoenenberger, J. Paratte, and P. Vandergheynst, "Graph-based denoising for time-varying point clouds," in *3DTV-CON*, 2015, pp. 1–4.

[38] X. Gao, W. Hu, and Z. Guo, "Graph-based point cloud denoising," in *BigMM*, 2018, pp. 1–6.

[39] C. Dinesh, G. Cheung, I. V. Bajic, and C. Yang, "Local 3d point cloud denoising via bipartite graph approximation & total variation," in *MMSP*, 2018, pp. 1–6.

[40] W. Hu, X. Gao, G. Cheung, and Z. Guo, "Feature graph learning for 3d point cloud denoising," *IEEE Trans. Signal Process.*, vol. 68, pp. 2841–2856, 2020.

[41] W. Hu, J. Pang, X. Liu, D. Tian, C. Lin, and A. Vetro, "Graph signal processing for geometric data and beyond: Theory and applications," *IEEE Trans. Multim.*, vol. 24, pp. 3961–3977, 2022.

[42] J. Zeng, G. Cheung, M. Ng, J. Pang, and C. Yang, "3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model," *TIP*, vol. 29, pp. 3474–3489, 2020.

[43] C. Dinesh, G. Cheung, and I. V. Bajic, "3d point cloud denoising via bipartite graph approximation and reweighted graph laplacian," *arXiv preprint arXiv:1812.07711*, 2018.

[44] M. A. Irfan and E. Magli, "3d point cloud denoising using a joint geometry and color k-nn graph," in *EUSIPCO*, 2020, pp. 585–589.

[45] W. Hu, Q. Hu, Z. Wang, and X. Gao, "Dynamic point cloud denoising via manifold-to-manifold distance," *TIP*, vol. 30, pp. 6168–6183, 2021.

[46] F. Pistilli, G. Fracastoro, D. Valsesia, and E. Magli, "Learning robust graph-convolutional representations for point cloud denoising," *IEEE J. Sel. Top. Signal Process.*, vol. 15, no. 2, pp. 402–414, 2021.

[47] C. Huang, R. Li, X. Li, and C.-W. Fu, "Non-local part-aware point cloud denoising," *CoRR*, vol. abs/2003.06631, 2020.

[48] F. Pistilli, G. Fracastoro, D. Valsesia, and E. Magli, "Learning graph-convolutional representations for point cloud denoising," in *ECCV*, 2020, pp. 103–118.

[49] S. Luo and W. Hu, "Differentiable manifold reconstruction for point cloud denoising," in *ACM MM*, 2020, pp. 1330–1338.

[50] A. Huang, Q. Xie, Z. Wang, D. Lu, M. Wei, and J. Wang, "Modnet: Multi-offset point cloud denoising network customized for multi-scale patches," *CGF*, vol. 41, no. 7, pp. 109–119, 2022.

[51] D. Zhu, H. Chen, W. Wang, H. Xie, G. Cheng, M. Wei, J. Wang, and F. L. Wang, "Nonlocal low-rank point cloud denoising for 3-d measurement surfaces," *TIM*, vol. 71, pp. 1–14, 2022.

[52] A. Mao, Z. Du, Y. Wen, J. Xuan, and Y. Liu, "Pd-flow: A point cloud denoising framework with normalizing flows," in *ECCV*, vol. 13663, 2022, pp. 398–415.

[53] D. de Silva Edirimuni, X. Lu, Z. Shao, G. Li, A. Robles-Kelly, and Y. He, "Iterativepfn: True iterative point cloud filtering," in *CVPR*, 2023, pp. 13 530–13 539.

[54] P. Guerrero, Y. Kleiman, M. Ovsjanikov, and N. J. Mitra, "Pcpnet learning local shape properties from raw point clouds," *CGF*, vol. 37, no. 2, pp. 75–85, 2018.

[55] R. Roveri, A. C. Öztireli, I. Pandele, and M. Gross, "Pointpronets: Consolidation of point clouds with convolutional neural networks," in *CGF*, vol. 37, no. 2, 2018, pp. 87–99.

[56] D. Lu, X. Lu, Y. Sun, and J. Wang, "Deep feature-preserving normal estimation for point cloud filtering," *CAD*, p. 102860, 2020.

[57] J. Zhou, W. Jin, M. Wang, X. Liu, Z. Li, and Z. Liu, "Fast and accurate normal estimation for point clouds via patch stitching," *Comput. Aided Des.*, vol. 142, p. 103121, 2022.

[58] H. Zhou, H. Chen, Y. Zhang, M. Wei, H. Xie, J. Wang, T. Lu, J. Qin, and X. Zhang, "Refine-net: Normal refinement neural network for noisy point clouds," *TPAMI*, vol. 45, no. 1, pp. 946–963, 2023.

[59] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Ec-net: an edge-aware point set consolidation network," in *ECCV*, 2018, pp. 386–402.

[60] H. Zhou, K. Chen, W. Zhang, H. Fang, W. Zhou, and N. Yu, "Dup-net: Denoiser and upsampler network for 3d adversarial point clouds defense," in *ICCV*, 2019, pp. 1961–1970.

[61] Z. Liu, Y. Zhao, S. Zhan, Y. Liu, R. Chen, and Y. He, "Pcdnf: Revisiting learning-based point cloud denoising via joint normal filtering," *TVCG*, 2023.

[62] D. de Silva Edirimuni, X. Lu, G. Li, and A. Robles-Kelly, "Contrastive learning for joint normal estimation and point cloud filtering," *TVCG*, no. 01, pp. 1–15, 2023.

[63] S. Chen, C. Duan, Y. Yang, D. Li, C. Feng, and D. Tian, "Deep unsupervised learning of 3d point clouds via graph topology inference and filtering," *TIP*, vol. 29, pp. 3183–3198, 2020.

[64] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[65] F. Liu, S. Li, L. Zhang, C. Zhou, R. Ye, Y. Wang, and J. Lu, "3dcnn-dqn-rnn: A deep reinforcement learning framework for semantic parsing of large-scale 3d point clouds," in *ICCV*, 2017, pp. 5678–5687.

[66] M. Sarmad, H. J. Lee, and Y. M. Kim, "Rl-gan-net: A reinforcement learning agent controlled gan network for real-time point cloud shape completion," in *CVPR*, 2019, pp. 5898–5907.

[67] D. Bauer, T. Patten, and M. Vincze, "Reagent: Point cloud registration using imitation and reinforcement learning," in *CVPR*, 2021, pp. 14 586–14 594.

[68] L. Zhang and L. Zhang, "Deep learning-based classification and reconstruction of residential scenes from large-scale point clouds," *TGRS*, vol. 56, no. 4, pp. 1887–1897, 2017.

[69] M. Tiator, C. Geiger, and P. Grimm, "Point cloud segmentation with deep reinforcement learning," in *ECAI*, 2020, pp. 2768–2775.

[70] M. Tiator, C. Geiger, and P. Grimm, "Point cloud segmentation: Solving a perceptual grouping task with deep reinforcement learning," in *GI VR/AR Workshop*, 2020.

[71] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *TOG*, vol. 38, no. 5, pp. 1–12, 2019.

[72] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

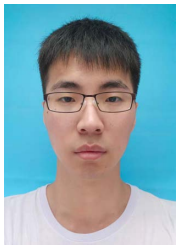
[73] K. Yu, C. Dong, L. Lin, and C. C. Loy, "Crafting a toolchain for image restoration by deep reinforcement learning," in *CVPR*, 2018, pp. 2443–2452.

[74] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: a point cloud upsampling adversarial network," in *ICCV*, 2019, pp. 7203–7212.

[75] P.-S. Wang, Y. Liu, and X. Tong, "Mesh denoising via cascaded normal regression." *ACM Trans. Graph.*, vol. 35, no. 6, pp. 232–1, 2016.

[76] A. Serna, B. Marcotegui, F. Goulette, and J. Deschaud, "Paris-rue-madame database - A 3d mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods," in *ICPRAM*. SciTePress, 2014, pp. 819–824.

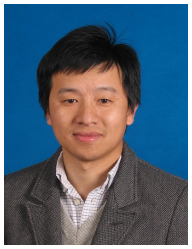
[77] P. Liu, H. Zhang, J. Wang, Y. Wang, D. Ren, and W. Zuo, "Robust deep ensemble method for real-world image denoising," *Neurocomputing*, vol. 512, pp. 1–14, 2022.



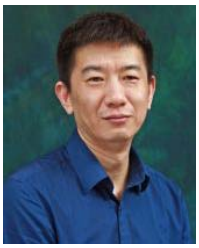
Zeyong Wei is a Ph.D. candidate at Nanjing University of Aeronautics and Astronautics (NUAA). He received his M.Sc. degree from NUAA. He has published several papers on SIGGRAPH, IJCV, IEEE TASE, CAD, etc. His research interests include computer vision and learning-based geometry processing.

Honghua Chen received the Ph.D. degree from the Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2022. He is currently a Research Fellow with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interest is 3D Measurement/Vision and point cloud processing.

Liangliang Nan received his Ph.D. degree in mechatronics engineering from the Graduate University of the Chinese Academy of Sciences, China, in 2009. He is currently an Associate Professor at the Delft University of Technology (TU Delft), The Netherlands, where he is leading the AI Laboratory on 3D Urban Understanding (3DUU). His research interests include computer graphics, computer vision, 3D geoinformation, and machine learning.



Jun Wang is a Professor at Nanjing University of Aeronautics and Astronautics (NUAA). He received his Bachelor's and Ph.D. degrees in Computer-Aided Design from NUAA in 2002 and 2007, respectively. He worked as a senior research engineer at Leica Geosystems Inc., USA. His research interests include geometry processing and geometric modeling.



Jing Qin is a professor at The Hong Kong Polytechnic University. His research focuses on creatively leveraging advanced virtual reality (VR) and artificial intelligence (AI) techniques in healthcare and medicine applications. He won 3 best paper awards for his research on AI-driven medical image analysis and computer-assisted surgery, including one of the most prestigious awards in this field: MIA-MICCAI best paper award in 2017.

Mingqiang Wei (Senior Member, IEEE) received his Ph.D. degree (2014) in Computer Science and Engineering from the Chinese University of Hong Kong (CUHK). He is a professor at the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA). He was a recipient of the CUHK Young Scholar Thesis Awards in 2014. He is now an Associate Editor for ACM TOMM, The Visual Computer, and the Guest Editor for IEEE TMM. His research interests focus on 3D vision, computer graphics, and deep learning.